DYNAMIC MODEL BUFFER FOR CONTINUAL FEDERATED DEEP REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Federated Deep Reinforcement Learning (FDRL) enables decentralized DRL training across distributed clients while preserving data privacy, yet it faces significant challenges in maintaining generalization across sequential tasks. Existing Continual FDRL (CFDRL) approaches primarily rely on regularization or replay mechanisms, both of which are constrained by fixed network capacity and thus inevitably suffer from catastrophic forgetting as tasks accumulate. Although expansion-based strategies have been explored in the Continual Learning (CL) community, their substantial hardware overhead renders them impractical for resource-constrained FDRL clients. To address this challenge, we propose a generic expansion-based CFDRL framework that can be seamlessly integrated into existing FDRL methods to equip them with enhanced CL capabilities. Our approach introduces a server-side model buffer that stores models from completed tasks, alongside a model selection strategy that identifies and reuses the most relevant prior model for each incoming task. This design effectively alleviates catastrophic forgetting by retaining the trained models from previous tasks, while improving knowledge utilization efficiency through model selection, without imposing a significant computational burden on clients. Extensive experimental results demonstrate that our method consistently enhances ten state-of-the-art (SOTA) FDRL methods and surpasses five SOTA CDRL and five CFDRL methods across multiple metrics.

1 INTRODUCTION

Federated Deep Reinforcement Learning (FDRL) enables privacy-preserving decentralized training across edge nodes by leveraging localized expertise, but it often faces multiple distinct tasks in practice and thus demands effective cross-task generalization. To this end, Continual FDRL (CFDRL) has been introduced and is gaining traction in domains such as autonomous driving Lu et al. (2026) and robot control Liu et al. (2019).

CFDRL seeks to retain previously acquired knowledge while adapting to new tasks, primarily through regularization and replay strategies Liu et al. (2019)Nath et al. (2023)Liebe et al. (2025)Yuan et al. (2026)Lu et al. (2026). Regularization techniques discourage substantial modifications to weights or policies deemed critical for prior tasks, whereas replay-based approaches preserve past experiences by storing and reusing samples from earlier tasks. Nevertheless, both categories depend on a predetermined network capacity, which inevitably precipitates catastrophic forgetting as novel knowledge accumulates over time. The Continual Learning (CL) community, including CDRL and Continual Federated Learning (CFL), has explored expansion-based methods that dynamically increase model parameterization to accommodate additional knowledge Wang et al. (2023b)Zhang et al. (2024a)Xu et al. (2025c)De Caro et al. (2023)Mori et al. (2022). However, continuous network expansion introduces substantial hardware overhead that resource-constrained FDRL clients cannot support. Furthermore, even when expansion is conducted on the server, transmitting the enlarged models to clients consumes excessive communication bandwidth and imposes high on-device computational and memory costs. Consequently, existing expansion-based CL approaches are not viable for practical FDRL deployments. Furthermore, while certain CFL methods reduce computational overhead via model pruning Li et al. (2023)Yoon et al. (2021) or employ broad learning systems (BLS) for efficiency Le et al. (2021), pruning inevitably discards previously acquired knowledge, whereas BLS relies on randomly initialized enhancement nodes, which essentially scatter knowledge into the feature space in an uncontrolled manner, leading to suboptimal preservation.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Motivated by these observations, this work proposes a generic extension-based CFDRL framework that can be seamlessly incorporated into existing FDRL methods to endow them with enhanced CL capabilities. Our framework equips the server, which is typically assumed to possess abundant hardware resources, with a model buffer implemented as a queue. This buffer retains the model learned from each completed task, mitigating catastrophic forgetting without requiring architectural changes on the clients. Meanwhile, a model selection strategy is introduced to identify and reuse the most relevant prior model for each incoming task, thereby improving the utilization of experience. Specifically, on the server side, we first interact with the environment using a random policy to collect a mini-batch of transitions. For each model stored in the buffer, we compute the Temporal Difference (TD) error on each transition in the mini-batch, yielding a set of TD errors per model. We then analyze this set to determine the selection probability of the corresponding model, and the one with the highest likelihood is chosen as the best match for the current task dynamics. This selection process is generic and can be instantiated by any off-the-shelf sequence analysis method, such as averaging or Gaussian distribution modeling; four variants are compared in this paper. We adopt TD error as the selection criterion because it quantifies the prediction discrepancy of each model against the current task dynamics, where a smaller error indicates superior adaptation. Despite prior efforts on model preservation and reuse in deep learning, current strategies are confined to ensemble-based decision fusion Ramesh & Chaudhari (2021)Marouf et al. (2024)Mao et al. (2025) or policy guidance via historical models Malagon et al. (2025)Schor et al. (2019). Nevertheless, their applicability to FDRL is precluded by the prohibitively large model scale encountered during training.

The key contributions are summarized as follows: 1) We propose a generic CFDRL framework that maintains a model buffer on the server to store models learned from previous tasks, thereby mitigating catastrophic forgetting, and incorporates a model selection strategy to identify and reuse the most suitable prior model for each incoming task; 2) We provide a theoretical analysis that elucidates the performance advantages of our approach over existing FDRL baselines; 3) Extensive experimental results demonstrate that our method consistently improves ten state-of-the-art (SOTA) FDRL methods and outperforms five SOTA CDRL and five CFDRL counterparts.

2 Related work

FL. FL enables collaborative training across distributed clients without sharing raw data, with FedAvg as a foundational aggregation algorithm McMahan et al. (2017)Lee et al. (2024). A key challenge is data heterogeneity across participants, which often degrades performance Wang et al. (2024a)Ahmed et al. (2024). To address this, various weighting schemes have been proposed to evaluate client contributions, including dataset size-based ratios (e.g., FedProx Karimireddy et al. (2020)), reputation scores from local test performance Wang & Kantarci (2021), and Shapley value-based importance measures Tang et al. (2021). The FL community has also adopted CL techniques via regularization, replay Yu et al. (2024)Wuerkaixi et al. (2024)Chen et al. (2025)Li et al. (2024)Zhong et al. (2025)Wang et al. (2024b), or model expansion Li et al. (2023)De Caro et al. (2023)Mori et al. (2022)Zhu et al. (2022). Regularization and replay methods are limited by fixed model capacity and catastrophic forgetting, while expansion approaches incur rising hardware overhead. Conventional FL assumes offline datasets, whereas FDRL relies on online data from agent environment interactions, making standard FL methods inapplicable without adaptation to the online FDRL setting.

CDRL. Current CDRL approaches fall into three primary categories: regularization, replay, and architectural expansion. Regularization techniques, such as Elastic Weight Consolidation Kirkpatrick et al. (2018) and Learning without Forgetting Li & Hoiem (2017), safeguard prior knowledge by restricting updates on parameters vital to earlier objectives Kessler et al. (2020)Muppidi et al. (2024)Zhang et al. (2022)Xu et al. (2025a)Fang et al. (2023a). Replay methods alleviate forgetting by storing real past experiences Li et al. (2021)Buzzega et al. (2021)Zhang et al. (2024b)**?** or generating fictitious ones through generative models Atkinson et al. (2021)Xie et al. (2021)Daniels et al. (2022)Mendez-Mendez et al. (2023). However, regularization and replay are constrained by fixed network capacity, which inevitably leads to the overwriting of old knowledge as new information accumulates. Expansion-based approaches address this limitation by reusing existing policies and options Wang et al. (2023b)Agarwal et al. (2022), applying progressive learning frameworks Cao et al. (2022), or dynamically growing network architectures Huo et al. (2023)Zhang et al. (2024a)Xu et al. (2025c). Nevertheless, the continual growth of network scale in expansion-based methods poses a prohibitive burden for FDRL clients with limited hardware resources.

FDRL. FDRL is generally divided into two primary paradigms: horizontal (HFDRL) and vertical (VFDRL). In HFDRL, each agent functions autonomously within its private environment, without inter-agent interactions Cha et al. (2020)Jiang et al. (2025)Wang et al. (2023a). Representative contributions in this category include a hybrid FDRL framework for personalized gaming by Bodas et al. Bodas et al. (2018), a Byzantine-resilient design by Fan et al. Fan et al. (2021), and a decentralized formulation reported in Wang et al. (2023a). VFDRL, in contrast, places multiple agents in a shared environment where each observes only a limited subset of the global state, as demonstrated by Zhuo et al. Zhuo et al. (2019). Recent advances have increasingly intersected with transfer learning, multi-task learning, and policy distillation, as exemplified by Liang et al. Liang et al. (2022), Liu et al. Liu et al. (2019), and Mai et al. Mai et al. (2023). Conventional FDRL methods are designed for a single task and fail to maintain performance over a sequence of incoming tasks, primarily due to catastrophic forgetting of earlier acquired knowledge. To address this limitation, emerging CFDRL solutions incorporate regularization and replay mechanisms Liu et al. (2019)Nath et al. (2023)Liebe et al. (2025)Yuan et al. (2026)Lu et al. (2026). Nevertheless, these strategies remain confined by fixed network capacity, making them prone to gradual overwriting of prior knowledge as new tasks emerge.

3 PRELIMINARIES

FDRL. In FDRL, a central server collects local policy updates from distributed clients and constructs a global policy parameter vector θ, while all client interaction data remain locally stored to ensure privacy. After aggregation, the updated global model is redistributed to each client for the next round of local policy training. For N_a clients, each client i maintains a local policy θ_i, computes a loss $\mathcal{L}_i$ based on its locally collected samples, and is assigned a weighting scalar w_i. The global optimization objective of FDRL is to minimize the weighted sum of local policy losses, formulated as $\min_\theta \sum_{i=1}^{N_a} w_i \mathcal{L}_i(\theta_i)$. Despite recent advances in model aggregation, i.e., computing w_i, conventional FDRL methods are typically designed for a single task and fail to maintain generalization across sequentially arriving tasks due to catastrophic forgetting of prior knowledge.

CDRL. In the CDRL setting, an agent encounters a sequence of tasks in an unpredictable order, denoted as $\mathcal{T} = [\mathcal{T}_1, \ldots, \mathcal{T}_{N_{ta}}]$. Each task originates from a changing distribution, and the agent lacks explicit knowledge of which task is active at any given time. The primary aim is to derive a policy π that generalizes well over the entire task sequence, necessitating the retention and reuse of knowledge acquired from earlier tasks. Training proceeds incrementally, such that prior knowledge from earlier tasks is transferred to and utilized in subsequent tasks. CDRL performance is quantified through the average cumulative reward across all N_{ta} tasks, where a larger value signifies superior outcomes. More precisely, if $J_i(\pi)$ denotes the total reward collected on task i, then the objective is to maximize $\bar{J}(\pi) = \frac{1}{N_{ta}} \sum_{i=1}^{N_{ta}} J_i(\pi)$. Conventional CDRL is limited to training on data derived from a single environment, which intrinsically hampers the model's ability to learn beyond that data.

CFDRL. The CFDRL paradigm comprises a central server and multiple distributed edge clients that collaboratively learn from an unordered task set $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_{N_{ta}}\}$, where each task $\mathcal{T}_i$ is governed by an independent nonstationary environmental distribution $\mathcal{B}_i$ and clients do not have access to explicit task identity indicators $\mathbb{I}_i$. The primary goal is to derive an optimal global policy $\pi_g \in \Pi$ that generalizes across these tasks. To achieve this, CFDRL employs a uniform federated incremental learning procedure for every task: each edge client updates its local policy π_l using private samples $\tau \sim \mathcal{B}_i$, while the server periodically aggregates local parameters to iteratively refine π_g. Task performance is measured by the expected cumulative reward $J(\pi_g; \mathcal{T}_i) = \mathbb{E}_{\tau \sim \mathcal{B}_i} \left[\sum_{t=0}^{T} r_t(\boldsymbol{s}_t, \boldsymbol{a}_t) \right]$, and the global optimization objective is accordingly formulated as $\max_{\pi_g} \bar{J}(\pi_g) = \frac{1}{N_{ta}} \sum_{i=1}^{N_{ta}} J(\pi_g; \mathcal{T}_i)$. Nevertheless, a critical obstacle remains. Individual clients in FDRL are constrained by limited hardware resources and thus cannot afford the substantial computational cost of network expansion. Moreover, both regularization-based and replay-based methods are restricted to fixed network capacity, which inherently prevents them from mitigating catastrophic forgetting in this setting.

4 METHODOLOGY

This section details our method. **The supplementary materials provide the pseudocode (Section A.2), the four variants of our method (Section A.1), and the theoretical analysis (Section A.3).**

4.1 THE FRAMEWORK FOR OUR METHOD

Fig. 1 illustrates the framework of our method. For each task in CFDRL, the learning process follows the same procedure consisting of **server-side operations** and **client-side training**. During the server-side phase, when a new task commences, a random policy interacts with the environment to collect a mini-batch of interaction samples. The server computes the TD errors for each buffered model using the collected samples, from which the importance of each model is derived. Here, we provide four methods for computing the model importance. A softmax normalization is then applied to these importance scores to obtain selection probabilities, and the model with the highest probability is chosen for the current task. The selected model is then distributed to all clients for local training. After a specified number of local training rounds, clients upload their updated models to the server for global aggregation. The aggregated model is redistributed to the clients, and this process repeats iteratively. Upon task completion, the final global model is enqueued into the model buffer to alleviate catastrophic forgetting, after which the next task commences. During the client-side phase, the local training performed by each client follows the same procedure as in the baseline FDRL method.

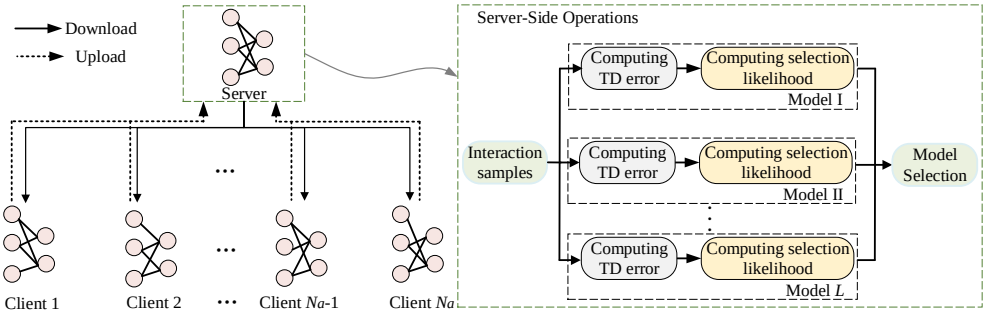


Figure 1: Framework of our approach. In the sequence of tasks to be learned for CFDRL, execution starts from the second task, whereas the first task is trained directly with the initial model. Our approach extends only the model buffer during the CFDRL process, leaving the size of each individual model unchanged.

4.2 SERVER-SIDE OPERATIONS

The server-side operations comprise four steps. The first three steps are designed to select the most suitable model from the model buffer upon the initiation of a new task and to distribute it to clients for training. The fourth step performs global aggregation and global model distribution. Let L denote the number of existing models in the model buffer, θ^{μ} denote the current actor, θ^{Q_i} denote the current i-th critic, $\theta^{\mu'}$ denote the target actor, and $\theta^{Q_i\prime}$ denote the i-th target critic.

(1) Interaction Data Collection: When a new task begins, a random policy interacts with the current task environment to obtain a mini-batch of interaction samples, denoted as $\mathcal{D}_0 = \{(s, a, r, s')\}$. These interaction data are then used to evaluate the most suitable model for distribution to the clients.

(2) TD Error Calculation: For each policy in the model buffer, e.g., the k-th policy π_k, we compute its TD error on the interaction data $\mathcal{D}_0$, which is calculated as follows. First, we compute the target Q-value $\hat{Q}(s, a)$ using $\hat{Q}(s, a) = r + \gamma \min_{i=1,2} Q_{\theta^{Q_i\prime}}(s', \tilde{a})$, where $\tilde{a}$ is generated from $\pi_{\theta^{\mu\prime}}(s') + \epsilon$, with ϵ sampled from a normal distribution $\mathcal{N}(0, 0.2)$. Then, using the target Q-value $\hat{Q}(s, a)$ and the i-th current Q-value $Q_{\theta^{Q_i}}(s, a)$, the TD error of policy π on a sample (s, a) is computed as follows:

$$\varepsilon(s, a) = \sum_{i=1}^{N_{cr}} \left(\hat{Q}(s, a) - Q_{\theta^{Q_i}}(s, a) \right)^2, \tag{1}$$

where N_{cr} denotes the number of critics. Accordingly, the TD errors for policy π over all samples in $\mathcal{D}_0$ are obtained as $\varepsilon_1(s, a), \varepsilon_2(s, a), \ldots, \varepsilon_{\mathcal{D}_0}(s, a)$.

(3) Model Selection: Our framework supports flexible integration of any method for processing the sequential TD errors $\varepsilon_1(s, a), \varepsilon_2(s, a), \ldots, \varepsilon_{\mathcal{D}_0}(s, a)$ to compute model importance. Different methods essentially determine how each individual sample contributes to the final importance

evaluation. **Here we present two representative implementations, denoted as Method I and Method II, with two additional variants detailed in Section A.1 of the supplementary material.**

(3.1) Method I: The first approach directly computes the negative average of the TD errors as the importance score, i.e., $\ell = -\frac{1}{|\mathcal{D}_0|}\sum_{j=1}^{|\mathcal{D}_0|}\varepsilon_j(s,a)$. Applying this to each model in the buffer yields the importance vector $\boldsymbol{\ell} = [\ell_1, \ell_2, \ldots, \ell_L]$. While computationally simple, this method is sensitive to outliers, as extremely large or small values can disproportionately affect the final score.

(3.2) Method II: Given that Gaussian distributions are well-suited for modeling TD error distributions in high-dimensional DRL environments Williams (1992); Todorov et al. (2012); Nasiriany et al. (2021); Hollenstein et al. (2022), we assume the TD error of a policy follows a zero-mean Gaussian distribution, i.e., $\varepsilon \sim \mathcal{N}(0, \sigma^2)$, with variance fixed at a commonly used value $\sigma = 0.25$ Xu et al. (2025d). The selection likelihood for model π_k based on interaction data $\mathcal{D}_0$ is defined as:

$$p(\mathcal{D}_0|\pi_k) = \prod_{(s,a,s')\in\mathcal{D}_0} \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{\varepsilon_k^2}{2\sigma^2}\right). \tag{2}$$

However, multiplying many small likelihood values may lead to floating-point underflow Xu et al. (2024; 2025b). To mitigate this, we adopt logarithmic mean smoothing with exponential restoration, following prior work Xu et al. (2024; 2025b):

$$\ell_k = \exp\left(\frac{1}{|\mathcal{D}_0|}\sum_{(s,a)\in\mathcal{D}_0}\ln p(\mathcal{D}_0|\pi_k)\right). \tag{3}$$

The importance vector is then constructed as $\boldsymbol{\ell} = [\ell_1, \ell_2, \ldots, \ell_L]$. Although slightly more involved, this approach is robust to extreme outliers.

(3.3) Probability Calculation for Model Selection: With the importance vector $\boldsymbol{\ell}$, we apply softmax normalization to reduce large magnitude discrepancies:

$$\boldsymbol{\phi}_k = \text{Softmax}(\boldsymbol{\ell})_k = \frac{\exp(\ell_k)}{\sum_{j=1}^{L}\exp(\ell_j)}. \tag{4}$$

The model with the highest probability is selected as the most suitable for the new task: $l^* = \arg\max(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_L)$. This chosen model l^* is then adopted as the global model θ_{global} and distributed to clients for local training on the current task.

(4) Global Aggregation and Model Distribution: Local models are uploaded to the server after a training period defined by the communication frequency N_c, as specified by the baseline FDRL method. The server then aggregates these local models into a global one using $\theta^{agg} = F(\theta_1, \theta_2, \ldots, \theta_{N_a})$, where N_a is the number of local models and $F(\cdot)$ denotes the aggregation function. The aggregation function varies depending on the baseline method; for instance, FedAvg takes the form $F(\theta_1, \theta_2, \ldots, \theta_{N_a}) = \sum_{j=1}^{N_a}\frac{1}{N_a}\theta_j$. Once aggregation is complete, the resulting global model is sent back to all clients, and step (4) is repeated iteratively. The global model trained on each task is then added to the model buffer to prevent catastrophic forgetting. **Compared with baseline FDRL approaches, our method introduces only one additional step at the beginning of each task, that is, distributing a global model selected from the buffer to the clients. Consequently, our approach incurs only a marginal increase in communication overhead relative to the baselines.**

4.3 Client-Side Operations

These global models θ_{global}, including the global critic θ^Q_{global} and global actor θ^μ_{global}, are distributed to each client by the server, after which each client begins its own training, a process consistent with the baseline FDRL methods. Let $\mathcal{L}(\theta_i^Q)$ represent the critic loss for client i, and let $\mathcal{L}(\theta_i^\mu)$ represent the actor loss for client i. The training for the j-th critic $\theta_i^{Q_j}$ of the i-th client is given by $\theta_i^{Q_j} = \arg\min_{\theta^Q}\mathcal{L}(\theta_i^{Q_j}, \theta^Q_{global})$. The training for the i-th client's actor is given by $\pi_{\theta_i^\mu} = \arg\min_{\pi}\mathcal{L}(\theta_i^\mu, \theta^\mu_{global})$. Since the client-side training procedure remains unchanged, our method incurs no additional hardware overhead on the clients.

5 EXPERIMENT AND ANALYSIS

This section describes the experimental setup, results, and analysis. The software comprises Torch 1.2.0, Gym 0.16.0, and mujoco-py 1.50.0.1, while the hardware includes an Intel Core i7-9700 processor, 64 GB RAM, and an NVIDIA RTX 2080 GPU. Each experiment is run with five random seeds. **We adopt the second implementation of our method in the main experiments. The supplementary material provides additional results from other implementations (Section A.8) and further experiments (Sections A.7, A.9 through A.11), along with implementation details (Section A.4), benchmark descriptions (Section A.5), and episode return curves (Section A.6).**

5.1 EXPERIMENT SETTINGS

(1) Baselines. Ten SOTA FDRL methods are used as baselines to integrate our approach, including **Federated Reinforcement Learning with Linear Speedup (FRLLS)** Khodadadian et al. (2022), **Asynchronous Federated Reinforcement Learning with Policy Gradient Updates (AFedPG)** Lan et al. (2024), **Fast Federated Reinforcement Learning (FFRL)** Pang et al. (2025), **Asynchronous Federated Reinforcement Learning (AFRL)** Lan et al. (2024), **FedHPD** Jiang et al. (2025), **Approximated Behavioral Metric-based State Projection (ABMSP)** Guo et al. (2025), **Regret-Optimal Q-Learning (ROQL)** Zhang et al. (2025), **QAvg** Jin et al. (2022), **Provably Robust FRL (PRFRL)** Fang et al. (2025), and **Self-Regulating Training (SRT)** Xu et al. (2026).

Five SOTA CFDRL methods are respectively integrated into baseline FDRL methods for comparison with our method. These methods include **Lifelong Federated Reinforcement Learning (LFRL)** Liu et al. (2019), **Modulating Masks (MM)** Nath et al. (2023), **Federated Continual Policy Optimization (FCPO)** Liebe et al. (2025), **Federated Continuous Reinforcement Learning (FCRL)** Yuan et al. (2026), and **Federated Reinforcement Learning With High Generalizability (FRLHG)** Lu et al. (2026). In addition, five SOTA CDRL methods are also integrated into baseline FDRL methods for comparison with our method. These methods include **Sparse Training (ST)** Xu et al. (2025b), **Safe Continual Reinforcement Learning (SCRL)** Coursey et al. (2026), **Self-Supervised Continual Reinforcement Learning (SSCRL)** Fang et al. (2023b), **Neuroplastic Expansion (NE)** Liu et al. (2025), and **Few-shot Lifelong Deep Reinforcement Learning (FLDRL)** Xu et al. (2024). Among them, ST, FLDRL, and SCRL are regularization-based approaches, SSCRL is a replay-based approach, and NE is an expansion-based approach.

(2) Benchmarks. Ten benchmark tasks were used, with five clients allocated to each. The task suite includes four robot navigation benchmarks Zhang et al. (2024a)Xu et al. (2025c), namely **Nav-Reacher-v1**, **Nav-Reacher-v2**, **Nav-v1**, and **Nav-v2**; two robotic arm control benchmarks, **RobotArm-v1** and **RobotArm-v2** Ding et al. (2023); and four MuJoCo locomotion benchmarks Zhang et al. (2024a)Xu et al. (2025c), specifically **HalfCheetah-vel**, **Humanoid-vel**, **Walker-vel**, and **Swimmer-vel**. The navigation variants differ in environmental complexity, with v2 introducing obstacle avoidance requirements absent in v1. The locomotion tasks challenge agents to sustain a prescribed forward velocity along the x-axis. The robotic arm scenarios involve a three-degree-of-freedom Dobot Magician, whose end-effector must be steered from an initial configuration to a designated target while circumventing circular barriers. To emulate CDRL conditions, we generate non-stationary task streams by stochastically varying initial states, producing sequences of 20, 50, and 70 tasks, with task repetition permitted within each sequence. To construct FDRL settings, we introduce client heterogeneity by randomly perturbing each client's environmental parameters, such as target positions and movement velocities, within a 10% range of their nominal values.

(3) Metrics. We evaluate the global model performance of each CFDRL method using three performance metrics, all of which are to be maximized. The primary metric is **average return (forward transfer, impact on new tasks)**, defined as the mean cumulative reward averaged over the final 40 episodes of each task in a sequence, computed across multiple seeds. Each task undergoes 200 training episodes in total. The return for a given task is $R = \frac{1}{40}\sum_{i=1}^{40} r_i$, where r_i is the total reward of the i-th episode within the final 40-episode window. Let NoS denote the number of seeds and N the number of tasks. For a sequence of N tasks with returns R_{ij} for the j-th task under the i-th seed, the overall CDRL performance is $\hat{\mathcal{R}} = \frac{1}{NoS}\sum_{i=1}^{NoS}\left(\frac{1}{N}\sum_{j=1}^{N} R_{ij}\right)$, with variability quantified by $\sigma = \sqrt{\frac{1}{NoS}\sum_{i=1}^{NoS}\left(\frac{1}{N}\sum_{j=1}^{N}(R_{ij}-\hat{\mathcal{R}})^2\right)}$. The second metric, **episode return**, illustrates

Table 1: Performance comparison on MuJoCo locomotion tasks (Mean ± Standard deviation).

Methods	HalfCheetah-vel	Walker-vel	Swimmer-vel	Humanoid-vel
AFedPG	-8.788 ± 0.051	-5.682 ± 0.114	-11.722 ± 0.036	86.898 ± 1.091
Ours+AFedPG	$\mathbf{-4.445} \pm 0.028$	$\mathbf{-4.073} \pm 0.053$	$\mathbf{-10.829} \pm 0.043$	$\mathbf{190.490} \pm 1.291$
FFRL	-6.982 ± 0.058	-4.809 ± 0.080	-11.360 ± 0.035	64.531 ± 0.455
Ours+FFRL	$\mathbf{-4.847} \pm 0.031$	-8.144 ± 0.002	$\mathbf{-10.446} \pm 0.036$	$\mathbf{188.635} \pm 1.162$
AFRL	-5.606 ± 0.053	-5.136 ± 0.141	-12.107 ± 0.039	125.495 ± 0.870
Ours+AFRL	$\mathbf{-4.412} \pm 0.039$	$\mathbf{1.635} \pm 0.109$	$\mathbf{-10.584} \pm 0.033$	$\mathbf{173.350} \pm 1.148$
FedHPD	-5.845 ± 0.084	-4.306 ± 0.156	-11.751 ± 0.037	154.331 ± 0.868
Ours+FedHPD	$\mathbf{-3.892} \pm 0.027$	$\mathbf{-2.930} \pm 0.202$	$\mathbf{-11.048} \pm 0.042$	152.050 ± 1.022
ABMSP	-7.546 ± 0.065	-3.938 ± 0.149	-11.188 ± 0.034	162.395 ± 1.019
Ours+ABMSP	$\mathbf{-5.552} \pm 0.044$	-5.033 ± 0.048	$\mathbf{-10.743} \pm 0.049$	$\mathbf{176.644} \pm 0.980$
ROQL	-5.903 ± 0.050	-4.937 ± 0.053	-11.727 ± 0.043	108.362 ± 1.144
Ours+ROQL	$\mathbf{-3.794} \pm 0.038$	$\mathbf{-0.297} \pm 0.110$	$\mathbf{-10.220} \pm 0.049$	$\mathbf{170.990} \pm 0.927$
QAvg	-8.366 ± 0.118	-5.632 ± 0.182	-11.447 ± 0.029	128.798 ± 1.138
Ours+QAvg	$\mathbf{-4.489} \pm 0.057$	$\mathbf{-0.275} \pm 0.148$	$\mathbf{-10.859} \pm 0.032$	$\mathbf{186.119} \pm 1.012$
PRFRL	-6.759 ± 0.063	-1.898 ± 0.239	-11.654 ± 0.041	143.653 ± 0.673
Ours+PRFRL	$\mathbf{-6.509} \pm 0.054$	-8.489 ± 0.026	$\mathbf{-10.412} \pm 0.034$	$\mathbf{166.224} \pm 1.541$
SRT	-8.381 ± 0.056	-10.563 ± 0.082	-11.274 ± 0.052	150.913 ± 0.860
Ours+SRT	$\mathbf{-5.183} \pm 0.060$	$\mathbf{1.819} \pm 0.101$	$\mathbf{-10.576} \pm 0.041$	$\mathbf{168.117} \pm 1.432$

Table 2: Performance comparison under robotic arm tasks (Mean ± Standard deviation).

Methods	Static environment		Dynamic environment	
	RobotArm-v1	RobotArm-v2	RobotArm-v1	RobotArm-v2
FRLLS	-6.182 ± 0.000	-15.573 ± 0.000	-6.986 ± 0.107	-13.890 ± 0.133
Ours+FRLLS	**-0.783** ± 0.005	**-1.011** ± 0.008	**-3.627** ± 0.087	**-4.656** ± 0.129
AFRL	-1.025 ± 0.019	-1.309 ± 0.019	-74.650 ± 1.985	-371.583 ± 4.814
Ours+AFRL	**-0.747** ± 0.003	**-1.016** ± 0.007	**-3.513** ± 0.105	**-5.013** ± 0.109
FedHPD	-77.858 ± 0.919	-331.940 ± 5.635	-3.870 ± 0.079	-4.760 ± 0.084
Ours+FedHPD	**-0.831** ± 0.009	**-1.058** ± 0.010	**-3.768** ± 0.075	**-4.643** ± 0.097
ABMSP	-0.864 ± 0.005	-1.419 ± 0.015	-11.391 ± 0.099	-20.664 ± 0.160
Ours+ABMSP	-3.778 ± 0.042	-6.200 ± 0.127	**-4.444** ± 0.095	**-7.881** ± 0.152
ROQL	-0.882 ± 0.007	-1.237 ± 0.012	-10.505 ± 0.111	-15.969 ± 0.120
Ours+ROQL	**-0.805** ± 0.008	**-1.130** ± 0.012	**-3.682** ± 0.099	**-5.250** ± 0.151
PRFRL	-18.805 ± 0.189	-26.403 ± 0.298	-22.994 ± 0.249	-30.282 ± 0.385
Ours+PRFRL	**-0.750** ± 0.003	**-1.060** ± 0.013	**-3.656** ± 0.087	**-4.909** ± 0.092
AFedPG	-7.210 ± 0.000	-13.539 ± 0.000	-3.706 ± 0.097	-4.880 ± 0.132
Ours+AFedPG	**-0.770** ± 0.005	**-1.041** ± 0.008	**-3.686** ± 0.083	-5.076 ± 0.124
FFRL	-11.347 ± 0.000	-20.292 ± 0.000	-3.523 ± 0.081	-4.941 ± 0.115
Ours+FFRL	**-0.778** ± 0.005	**-1.028** ± 0.007	-3.839 ± 0.074	**-4.858** ± 0.120

performance trends by plotting returns over successive episodes. The third metric, **knowledge retention (backward transfer, impact on old tasks)**, measures how well a task's performance is preserved upon its reappearance relative to its initial evaluation, where higher values indicate better retention of previously acquired knowledge.

5.2 Improving for FDRL methods

This section evaluates the improvements of our method over ten SOTA FDRL approaches under the CFDRL configuration across ten benchmark tasks, each consisting of 20 non-stationary tasks. In addition to static benchmark environments, we also create dynamic environments by randomly perturbing the physical parameters of each client environment and the global model environment within a 50% range of their nominal values. Table 1, Table 2, and Table 3 present the experimental results in terms of average return. From these results, when our method is integrated with baseline FDRL approaches, these methods achieve superior performance under the CFDRL configuration in the vast majority of cases. This indicates that existing FDRL methods fail to perform well under CL settings due to their inability to effectively preserve and leverage prior knowledge. In contrast, our approach endows existing FDRL methods with CL capabilities by leveraging a model buffer that retains previous knowledge and a model selection strategy that better exploits prior knowledge, thereby achieving higher returns.

5.3 Comparison with SOTA CDRL and CFDRL methods

This section validates the superiority of our method over SOTA CDRL and CFDRL approaches in improving FDRL. Specifically, we employ FedHPD and AFRL as baseline FDRL methods. For each

Table 3: Performance comparison for robot navigation tasks (Mean ± Standard deviation).

Methods	Nav-v1	Nav-v2	Nav-Reacher-v1	Nav-Reacher-v2
Static environment				
FRLLS	-4.736 ± 0.000	-5.206 ± 0.000	-27.597 ± 0.000	-27.800 ± 0.000
Ours+FRLLS	$\mathbf{-0.512} \pm 0.003$	$\mathbf{-0.575} \pm 0.007$	$\mathbf{-9.578} \pm 0.024$	$\mathbf{-10.127} \pm 0.032$
AFedPG	-4.561 ± 0.000	-5.476 ± 0.000	-33.804 ± 0.000	-43.371 ± 0.000
Ours+AFedPG	$\mathbf{-0.503} \pm 0.002$	$\mathbf{-0.517} \pm 0.004$	$\mathbf{-9.497} \pm 0.015$	$\mathbf{-9.925} \pm 0.016$
FFRL	-6.548 ± 0.000	-7.744 ± 0.000	-34.715 ± 0.000	-36.697 ± 0.000
Ours+FFRL	$\mathbf{-0.561} \pm 0.008$	$\mathbf{-0.549} \pm 0.006$	$\mathbf{-10.563} \pm 0.029$	$\mathbf{-11.288} \pm 0.019$
AFRL	-0.725 ± 0.015	-0.886 ± 0.028	-13.431 ± 0.056	-14.450 ± 0.064
Ours+AFRL	$\mathbf{-0.572} \pm 0.009$	$\mathbf{-0.512} \pm 0.003$	$\mathbf{-10.575} \pm 0.022$	$\mathbf{-11.295} \pm 0.023$
FedHPD	-30.631 ± 0.185	-89.127 ± 1.225	-81.279 ± 0.535	-74.410 ± 0.321
Ours+FedHPD	$\mathbf{-0.516} \pm 0.004$	$\mathbf{-0.513} \pm 0.004$	$\mathbf{-9.670} \pm 0.015$	$\mathbf{-10.065} \pm 0.017$
ABMSP	-0.535 ± 0.003	-0.609 ± 0.007	-14.936 ± 0.088	-16.143 ± 0.061
Ours+ABMSP	-1.389 ± 0.020	-1.394 ± 0.013	$\mathbf{-13.875} \pm 0.018$	$\mathbf{-15.042} \pm 0.032$
ROQL	-0.613 ± 0.007	-0.665 ± 0.008	-14.758 ± 0.054	-15.258 ± 0.036
Ours+ROQL	$\mathbf{-0.560} \pm 0.007$	$\mathbf{-0.530} \pm 0.005$	$\mathbf{-9.701} \pm 0.025$	$\mathbf{-9.805} \pm 0.020$
QAvg	-0.544 ± 0.005	-0.679 ± 0.011	-14.293 ± 0.069	-16.508 ± 0.121
Ours+QAvg	-1.858 ± 0.016	-1.879 ± 0.023	$\mathbf{-10.630} \pm 0.020$	$\mathbf{-10.742} \pm 0.014$
PRFRL	-12.820 ± 0.102	-11.684 ± 0.152	-24.035 ± 0.132	-23.539 ± 0.034
Ours+PRFRL	$\mathbf{-0.511} \pm 0.003$	$\mathbf{-0.499} \pm 0.002$	$\mathbf{-9.832} \pm 0.011$	$\mathbf{-10.416} \pm 0.021$
SRT	-2.085 ± 0.047	-1.995 ± 0.063	-20.439 ± 0.107	-17.483 ± 0.064
Ours+SRT	$\mathbf{-0.610} \pm 0.009$	$\mathbf{-1.401} \pm 0.024$	$\mathbf{-10.201} \pm 0.029$	$\mathbf{-10.727} \pm 0.020$
Dynamic environment				
FRLLS	-4.430 ± 0.074	-4.336 ± 0.083	-26.740 ± 0.068	-44.635 ± 0.069
Ours+FRLLS	$\mathbf{-2.374} \pm 0.053$	$\mathbf{-2.474} \pm 0.059$	$\mathbf{-10.939} \pm 0.052$	$\mathbf{-12.475} \pm 0.073$
AFedPG	-2.352 ± 0.057	-2.407 ± 0.064	-17.239 ± 0.302	-19.133 ± 0.359
Ours+AFedPG	-2.501 ± 0.070	$\mathbf{-2.339} \pm 0.052$	$\mathbf{-11.542} \pm 0.069$	$\mathbf{-12.572} \pm 0.052$
FFRL	-2.364 ± 0.058	-2.451 ± 0.060	-15.971 ± 0.112	-16.787 ± 0.097
Ours+FFRL	-2.615 ± 0.058	$\mathbf{-2.407} \pm 0.047$	$\mathbf{-12.463} \pm 0.062$	$\mathbf{-14.967} \pm 0.096$
AFRL	-26.293 ± 0.169	-106.080 ± 1.954	-65.685 ± 0.326	-74.958 ± 0.564
Ours+AFRL	$\mathbf{-2.429} \pm 0.057$	$\mathbf{-2.360} \pm 0.070$	$\mathbf{-11.377} \pm 0.056$	$\mathbf{-12.388} \pm 0.061$
FedHPD	-2.517 ± 0.063	-2.384 ± 0.049	-14.996 ± 0.105	-16.906 ± 0.124
Ours+FedHPD	$\mathbf{-2.397} \pm 0.050$	-2.392 ± 0.061	$\mathbf{-11.070} \pm 0.068$	$\mathbf{-12.445} \pm 0.060$
ABMSP	-6.016 ± 0.088	-7.633 ± 0.068	-76.797 ± 0.152	-39.422 ± 0.066
Ours+ABMSP	$\mathbf{-2.840} \pm 0.058$	$\mathbf{-2.820} \pm 0.063$	$\mathbf{-16.851} \pm 0.074$	$\mathbf{-19.325} \pm 0.077$
ROQL	-3.628 ± 0.073	-5.639 ± 0.079	-25.889 ± 0.104	-30.676 ± 0.099
Ours+ROQL	$\mathbf{-2.469} \pm 0.060$	$\mathbf{-2.461} \pm 0.059$	$\mathbf{-10.730} \pm 0.075$	$\mathbf{-12.583} \pm 0.051$
QAvg	-2.468 ± 0.059	-2.461 ± 0.060	-17.498 ± 0.577	-16.915 ± 0.111
Ours+QAvg	-2.639 ± 0.056	-2.681 ± 0.063	$\mathbf{-11.807} \pm 0.064$	$\mathbf{-13.017} \pm 0.062$
PRFRL	-12.973 ± 0.204	-14.043 ± 0.181	-25.988 ± 0.084	-31.007 ± 0.142
Ours+PRFRL	$\mathbf{-2.382} \pm 0.062$	$\mathbf{-2.448} \pm 0.056$	$\mathbf{-11.057} \pm 0.064$	$\mathbf{-12.443} \pm 0.054$
SRT	-3.023 ± 0.094	-3.119 ± 0.089	-16.851 ± 0.094	-19.967 ± 0.100
Ours+SRT	$\mathbf{-2.738} \pm 0.071$	$\mathbf{-2.571} \pm 0.069$	$\mathbf{-11.735} \pm 0.071$	$\mathbf{-13.337} \pm 0.046$

Table 4: Comparison with SOTA CDRL and CFDRL methods (Mean ± Standard deviation).

Methods	Nav-v2	Nav-Reacher-v1	Nav-Reacher-v2	RobotArm-v1	RobotArm-v2
FedHPD+ST	-59.327 ± 0.217	-85.399 ± 0.680	-87.142 ± 0.958	-35.049 ± 1.538	-408.502 ± 4.771
FedHPD+SCRL	-70.768 ± 1.324	-76.929 ± 0.591	-84.389 ± 0.871	-58.971 ± 0.814	-313.225 ± 4.485
FedHPD+FCPO	-30.012 ± 0.223	-81.669 ± 0.556	-73.972 ± 0.801	-35.239 ± 2.023	-445.225 ± 5.568
FedHPD+FCRL	-4.877 ± 0.135	-24.353 ± 0.146	-44.777 ± 0.193	-6.627 ± 0.135	-11.931 ± 0.239
FedHPD+SSCRL	-30.871 ± 1.350	-87.377 ± 0.349	-84.429 ± 0.942	-56.713 ± 2.565	-383.042 ± 6.026
FedHPD+LFRL	-628.535 ± 1.850	-221.076 ± 0.118	-225.428 ± 0.001	-366.384 ± 2.311	-1275.045 ± 0.174
FedHPD+NE	-67.193 ± 1.606	-29.257 ± 0.261	-27.182 ± 0.323	-65.336 ± 1.200	-301.693 ± 5.874
FedHPD+MM	-466.012 ± 0.816	-61.377 ± 0.124	-71.373 ± 0.168	-324.646 ± 0.370	-887.102 ± 0.425
FedHPD+FLDRL	-74.785 ± 0.947	-77.630 ± 0.627	-51.096 ± 0.442	-41.390 ± 0.246	-552.521 ± 5.143
FedHPD+FRLHG	-7.286 ± 0.214	-24.301 ± 0.074	-28.462 ± 0.112	-7.651 ± 0.214	-67.070 ± 1.247
Ours+FedHPD	$\mathbf{-0.513} \pm 0.004$	$\mathbf{-9.670} \pm 0.015$	$\mathbf{-10.065} \pm 0.017$	$\mathbf{-0.831} \pm 0.009$	$\mathbf{-1.058} \pm 0.010$
AFRL+ST	-8.879 ± 0.000	-39.973 ± 0.000	-46.148 ± 0.000	-11.052 ± 0.000	-27.858 ± 0.000
AFRL+SCRL	-8.069 ± 0.000	-45.039 ± 0.000	-48.767 ± 0.000	-11.458 ± 0.000	-23.254 ± 0.000
AFRL+FCPO	-9.353 ± 0.000	-51.179 ± 0.000	-44.304 ± 0.000	-11.541 ± 0.000	-18.537 ± 0.000
AFRL+FCRL	-1.107 ± 0.031	-16.658 ± 0.036	-17.290 ± 0.091	-1.402 ± 0.037	-1.666 ± 0.031
AFRL+SSCRL	-10.381 ± 0.000	-45.849 ± 0.000	-62.235 ± 0.000	-11.137 ± 0.000	-20.059 ± 0.000
AFRL+LFRL	-499.778 ± 0.000	-194.435 ± 0.000	-213.595 ± 0.000	-361.656 ± 0.000	-892.952 ± 0.000
AFRL+NE	-27.472 ± 0.000	-36.951 ± 0.000	-39.578 ± 0.000	-39.792 ± 0.000	-64.651 ± 0.000
AFRL+MM	-560.591 ± 0.000	-177.368 ± 0.000	-181.330 ± 0.000	-352.156 ± 0.000	-1115.484 ± 0.000
AFRL+FLDRL	-12.942 ± 0.000	-34.542 ± 0.000	-41.918 ± 0.000	-13.482 ± 0.000	-24.558 ± 0.000
AFRL+FRLHG	-8.032 ± 0.160	-21.464 ± 0.066	-24.932 ± 0.090	-10.542 ± 0.298	-16.485 ± 0.393
Ours+AFRL	$\mathbf{-0.512} \pm 0.003$	$\mathbf{-10.575} \pm 0.022$	$\mathbf{-11.295} \pm 0.023$	$\mathbf{-0.747} \pm 0.003$	$\mathbf{-1.016} \pm 0.007$

Table 5: Knowledge Retention.

Methods	Nav-v2	Nav-Reacher-v1	Nav-Reacher-v2	RobotArm-v1	RobotArm-v2
FedHPD+ST	-47.8069	-15.3681	-21.1413	-9.2387	-96.0444
FedHPD+SCRL	-23.1005	-25.9069	-39.6517	-19.7566	-299.3442
FedHPD+FCPO	-87.7241	-34.3953	-10.6481	-38.8248	-250.5339
FedHPD+FCRL	0.6015	-8.7373	-36.1230	-4.6029	-43.2188
FedHPD+SSCRL	-47.1724	-42.6263	-39.3998	1.1765	-203.7867
FedHPD+LFRL	-46.3799	-3.3401	-2.6489	-41.2917	-481.1590
FedHPD+NE	-64.3476	3.7177	-9.9744	-25.2458	-88.4537
FedHPD+MM	-14.0423	-1.3740	-10.4086	0.5967	2.4071
FedHPD+FLDRL	-98.3441	-16.4597	-54.0463	-21.5785	-70.1967
FedHPD+FRLHG	-56.7465	4.6325	2.5981	-5.3134	-128.3671
Ours+FedHPD	**1.4319**	**6.6182**	**3.5648**	**1.2985**	**4.1410**

baseline, we incorporate five SOTA CDRL methods, five SOTA CFDRL methods, and our method. Through this comparative experiment, we show that our approach consistently outperforms existing methods in FDRL enhancement. We validate our method on three robot navigation tasks and two robotic arm tasks based on average return, where each benchmark comprises 20 non-stationary tasks. Table 4 presents the experimental results. From these results, neither the SOTA CDRL methods nor the CFDRL methods help the baseline FDRL methods achieve optimal performance. This is due to the following reasons: 1) Existing CFDRL methods mitigate catastrophic forgetting through regularization or experience replay. However, as the amount of acquired knowledge increases, the fixed model capacity becomes insufficient to accommodate new knowledge, leading to catastrophic forgetting and suboptimal policy performance. 2) Among existing CDRL methods, regularization-based and replay-based approaches similarly suffer from catastrophic forgetting due to fixed network capacity, while expansion-based methods cannot be directly integrated into clients in principle, because expanding network capacity causes ever-increasing computational complexity on the client side. Even when a SOTA expansion-based method such as NE is integrated into FDRL approaches, it indiscriminately applies all previously acquired knowledge to new tasks. Among this knowledge, irrelevant information degrades the learning performance on the current task and consequently leads to suboptimal policies. In contrast, our method maintains a model buffer on the server to accommodate new memories. This design not only effectively preserves prior knowledge but also avoids imposing additional computational overhead on the clients. Meanwhile, the server selects the most suitable prior model for the current task and distributes it to all clients to better exploit prior knowledge. **In addition, the supplementary material further validates the effectiveness of our method by showing that it surpasses three SOTA expansion-based CFL approaches, namely model pruning Li et al. (2023)Yoon et al. (2021) and BLS Le et al. (2021), when integrated into FDRL. These experiments are presented in Section A.7.**

Table 5 presents knowledge retention results across five benchmark tasks, comparing our method against five SOTA CDRL methods and five CFDRL methods. All ten baseline methods, along with our method, are integrated into the FedHPD method. Specifically, each benchmark task involves a sequence of 20 tasks, where five distinct tasks appear randomly four times each. Experimental results demonstrate that our method improves knowledge retention relative to the baselines. Our method more effectively mitigates catastrophic forgetting by maintaining a model buffer that preserves models trained on prior tasks and boosts performance on previously encountered tasks through a model selection strategy, thereby demonstrating superior knowledge retention and utilization capabilities.

5.4 Supplementary experiments

This section summarizes the supplementary experimental results, which are presented in Sections A.8 through A.11. We first compare four implementations of our method against several baseline FDRL algorithms on two robotic arm benchmarks. This comparison not only evaluates the performance of all four variants but also confirms their consistent superiority over the baselines. We then examine scalability by varying client counts and the number of sequential tasks, showing that our method effectively mitigates catastrophic forgetting and improves knowledge utilization across different scales. Finally, ablation and variant analyses quantify the individual contributions of the model buffer and model selection strategy, demonstrating that their integrated design is essential for achieving strong performance in CFDRL settings.

6 Conclusion and limitations

This work presents a generic CFDRL framework that can be seamlessly integrated into existing FDRL methods to endow them with enhanced CL capabilities. Our approach maintains a server-side model buffer to retain previously learned models, mitigating catastrophic forgetting without incurring the heavy hardware overhead of client-side expansion, while a model selection strategy further enhances knowledge utilization by ensuring each new task benefits from the most relevant prior knowledge. Extensive experimental results demonstrate the effectiveness of our approach. Future work can explore adaptive buffer management that adjusts buffer size by evaluating the importance of prior CL models and discarding less important ones, and can also adopt sparse training techniques Xu et al. (2025b) to reduce the storage cost of the model buffer.

7 THE USE OF LARGE LANGUAGE MODELS (LLMS)

We used LLMs solely to edit the manuscript for improved readability. We did not use LLMs for any required disclosure tasks, including generating synthetic data, developing theoretical models, formulating mathematical claims, proposing hypotheses, designing experiments, implementing methods, or interpreting results. We reviewed and verified all LLM-edited text and take full responsibility for the final content of this work, including all claims and text.

REFERENCES

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Reincarnating reinforcement learning: Reusing prior computation to accelerate progress. *Advances in Neural Information Processing Systems*, 35:28955–28971, 2022.

Syed Thouheed Ahmed, V Vinoth Kumar, TR Mahesh, LV Narasimha Prasad, AK Velmurugan, V Muthukumaran, and VR Niveditha. Fedopt: federated learning-based heterogeneous resource recommendation and optimization for edge computing. *Soft Computing*, pp. 1–12, 2024.

Craig Atkinson, Brendan McCane, Lech Szymanski, and Anthony Robins. Pseudo-rehearsal: Achieving deep reinforcement learning without catastrophic forgetting. *Neurocomputing*, 428:291–307, 2021.

Dimitri Bertsekas. Convergence of discretization procedures in dynamic programming. *IEEE Transactions on Automatic Control*, 20(3):415–419, 2003.

Anand Bodas, Bhargav Upadhyay, Chetan Nadiger, and Sherine Abdelhak. Reinforcement learning for game personalization on edge devices. In *2018 International Conference on Information and Computer Technologies (ICICT)*, pp. 119–122. IEEE, 2018.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, and Simone Calderara. Rethinking experience replay: a bag of tricks for continual learning. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 2180–2187. IEEE, 2021.

Xinyuan Cao, Weiyang Liu, and Santosh Vempala. Provable lifelong learning of representations. In *International Conference on Artificial Intelligence and Statistics*, pp. 6334–6356. PMLR, 2022.

Han Cha, Jihong Park, Hyesung Kim, Mehdi Bennis, and Seong-Lyun Kim. Proxy experience replay: Federated distillation for distributed reinforcement learning. *IEEE Intelligent Systems*, 35(4): 94–101, 2020.

Jiao Chen, Jiayi He, Jianhua Tang, Weihua Li, and Zihang Yin. Knowledge efficient federated continual learning for industrial edge systems. *IEEE Transactions on Network Science and Engineering*, 12(3):2107–2120, 2025.

CHEE-S Chow and John N Tsitsiklis. An optimal one-way multigrid algorithm for discrete-time stochastic control. *IEEE transactions on automatic control*, 36(8):898–914, 1991.

Austin Coursey, Abel Diaz-Gonzalez, Marcos Quinones-Grueiro, and Gautam Biswas. Safe continual reinforcement learning in non-stationary environments. *arXiv preprint arXiv:2604.19737*, 2026.

Zachary Daniels, Aswin Raghavan, Jesse Hostetler, Abrar Rahman, Indranil Sur, Michael Piacentino, and Ajay Divakaran. Model-free generative replay for lifelong reinforcement learning: Application to starcraft-2. *arXiv preprint arXiv:2208.05056*, 2022.

Valerio De Caro, Claudio Gallicchio, and Davide Bacciu. Continual adaptation of federated reservoirs in pervasive environments. *Neurocomputing*, 556:126638, 2023.

Hongyu Ding, Yuanze Tang, Qing Wu, Bo Wang, Chunlin Chen, and Zhi Wang. Magnetic field-based reward shaping for goal-conditioned reinforcement learning. *IEEE/CAA Journal of Automatica Sinica*, 10(12):2233–2247, 2023.

Francois Dufour and Tomas Prieto-Rumeau. Finite linear programming approximations of constrained discounted markov decision processes. *SIAM Journal on Control and Optimization*, 51(2):1298–1324, 2013.

Francois Dufour and Tomas Prieto-Rumeau. Approximation of average cost markov decision processes using empirical distributions and concentration inequalities. *Stochastics An International Journal of Probability and Stochastic Processes*, 87(2):273–307, 2015.

Xiaofeng Fan, Yining Ma, Zhongxiang Dai, Wei Jing, Cheston Tan, and Bryan Kian Hsiang Low. Fault-tolerant federated reinforcement learning with theoretical guarantee. *Advances in Neural Information Processing Systems*, 34:1007–1021, 2021.

Minghong Fang, Xilong Wang, and Neil Zhenqiang Gong. Provably robust federated reinforcement learning. In *Proceedings of the ACM on Web Conference 2025*, pp. 896–909, 2025.

Yuyang Fang, Bin Guo, Jiaqi Liu, Kaixing Zhao, Yasan Ding, Na Wang, and Zhiwen Yu. Scrl: Self-supervised continual reinforcement learning for domain adaptation. In *2023 International Conference on Artificial Intelligence of Things and Systems (AIoTSys)*, pp. 55–63, 2023a.

Yuyang Fang, Bin Guo, Jiaqi Liu, Kaixing Zhao, Yasan Ding, Na Wang, and Zhiwen Yu. Scrl: Self-supervised continual reinforcement learning for domain adaptation. In *2023 International Conference on Artificial Intelligence of Things and Systems (AIoTSys)*, pp. 55–63. IEEE, 2023b.

Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J Cree. Regularisation of neural networks by enforcing lipschitz continuity. *Machine Learning*, 110(2):393–416, 2021.

Zengxia Guo, Bohui An, and Zhongqi Lu. Approximated behavioral metric-based state projection for federated reinforcement learning. *arXiv preprint arXiv:2505.09959*, 2025.

J. Hollenstein, S. Auddy, M. Saveriano, et al. Action noise in off-policy deep reinforcement learning: Impact on exploration and performance. *arXiv preprint arXiv:2206.03787*, 2022.

Fushuo Huo, Wenchao Xu, Jingcai Guo, Haozhao Wang, Yunfeng Fan, and Song Guo. Offline-online class-incremental continual learning via dual-prototype self-augment and refinement. *arXiv preprint arXiv:2303.10891*, 2023.

Wenzheng Jiang, Ji Wang, Xiongtao Zhang, Weidong Bao, Cheston Tan, and Flint Xiaofeng Fan. Fedhpd: Heterogeneous federated reinforcement learning via policy distillation. *arXiv preprint arXiv:2502.00870*, 2025.

Hao Jin, Yang Peng, Wenhao Yang, Shusen Wang, and Zhihua Zhang. Federated reinforcement learning with environment heterogeneity. In *International Conference on Artificial Intelligence and Statistics*, pp. 18–37. PMLR, 2022.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*, pp. 5132–5143. PMLR, 2020.

Samuel Kessler, Jack Parker-Holder, Philip Ball, Stefan Zohren, and Stephen J Roberts. Unclear: A straightforward method for continual reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

Sajad Khodadadian, Pranay Sharma, Gauri Joshi, and Siva Theja Maguluri. Federated reinforcement learning: Linear speedup under markovian sampling. In *International Conference on Machine Learning*, pp. 10997–11057. PMLR, 2022.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Reply to huszár: The elastic weight consolidation penalty is empirically valid. *Proceedings of the National Academy of Sciences*, 115(11):E2498–E2498, 2018.

Guangchen Lan, Dong-Jun Han, Abolfazl Hashemi, Vaneet Aggarwal, and Christopher G Brinton. Asynchronous federated reinforcement learning with policy gradient updates: Algorithm design and convergence analysis. *arXiv preprint arXiv:2404.08003*, 2024.

Junqing Le, Xinyu Lei, Nankun Mu, Hengrun Zhang, Kai Zeng, and Xiaofeng Liao. Federated continuous learning with broad network architecture. *IEEE Transactions on Cybernetics*, 51(8): 3874–3888, 2021.

Su Hyeong Lee, Sidharth Sharma, Manzil Zaheer, and Tian Li. Efficient adaptive federated optimization. *arXiv preprint arXiv:2410.18117*, 2024.

Beibei Li, Zerui Shao, Ao Liu, and Peiran Wang. Fedclip: Federated learning with client pruning. *arXiv preprint arXiv:2301.06768*, 2023.

Chunmao Li, Yang Li, Yinliang Zhao, Peng Peng, and Xupeng Geng. Sler: Self-generated long-term experience replay for continual reinforcement learning. *Applied Intelligence*, 51(1):185–201, 2021.

Yichen Li, Yuying Wang, Haozhao Wang, Yining Qi, Tianzhe Xiao, and Ruixuan Li. Rehearsal-free continual federated learning with synergistic synaptic intelligence. *arXiv preprint arXiv:2412.13779*, 2024.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

Xinle Liang, Yang Liu, Tianjian Chen, Ming Liu, and Qiang Yang. Federated transfer reinforcement learning for autonomous driving. In *Federated and Transfer Learning*, pp. 357–371. Springer, 2022.

Lucas Liebe, Thanh-Tung Nguyen, and Dongman Lee. Fcpo: federated continual policy optimization for real-time high-throughput edge video analytics. *arXiv preprint arXiv:2507.18047*, 2025.

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

Jiashun Liu, Johan S Obando Ceron, Aaron Courville, and Ling Pan. Neuroplastic expansion in deep reinforcement learning. In *International Conference on Learning Representations*, volume 2025, pp. 90056–90078, 2025.

Sikai Lu, Yingfeng Cai, Hai Wang, Ze Liu, Qingchao Liu, and Xiaoqiang Sun. Scenario-driven federated reinforcement learning autonomous driving system with high generalizability. *IEEE Transactions on Intelligent Transportation Systems*, 27(3):3476–3492, 2026.

Weiming Mai, Jiangchao Yao, Gong Chen, Ya Zhang, Yiu-Ming Cheung, and Bo Han. Server-client collaborative distillation for federated reinforcement learning. *ACM Transactions on Knowledge Discovery from Data*, 18(1):1–22, 2023.

Mikel Malagon, Josu Ceberio, and Jose A Lozano. Self-composing policies for scalable continual reinforcement learning. *arXiv preprint arXiv:2506.14811*, 2025.

Yuchuan Mao, Zhi Gao, Xiaomeng Fan, Yuwei Wu, Yunde Jia, and Chenchen Jing. Adaptive model ensemble for continual learning. *arXiv preprint arXiv:2509.19819*, 2025.

Imad Eddine Marouf, Subhankar Roy, Enzo Tartaglione, and Stéphane Lathuilière. Weighted ensemble models are strong continual learners. In *European Conference on Computer Vision*, pp. 306–324. Springer, 2024.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.

Jorge Mendez-Mendez, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Embodied lifelong learning for task and motion planning. In *Conference on Robot Learning*, pp. 2134–2150. PMLR, 2023.

Junki Mori, Isamu Teranishi, and Ryo Furukawa. Continual horizontal federated learning for heterogeneous data. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2022.

Aneesh Muppidi, Zhiyu Zhang, and Heng Yang. Fast trac: A parameter-free optimizer for lifelong reinforcement learning. *Advances in Neural Information Processing Systems*, 37:51169–51195, 2024.

Soroush Nasiriany, Vitchyr H. Pong, Ashvin Nair, Alexander Khazatsky, Glen Berseth, and Sergey Levine. Disco rl: Distribution-conditioned reinforcement learning for general-purpose policies. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6635–6641, 2021.

Saptarshi Nath, Christos Peridis, Eseoghene Ben-Iwhiwhu, Xinran Liu, Shirin Dora, Cong Liu, Soheil Kolouri, and Andrea Soltoggio. Sharing lifelong reinforcement learning knowledge via modulating masks. In *Conference on Lifelong Learning Agents*, pp. 936–960. PMLR, 2023.

Yiran Pang, Zhen Ni, and Xiangnan Zhong. A fast federated reinforcement learning approach with phased weight-adjustment technique. *Neurocomputing*, 626:129550, 2025.

Rahul Ramesh and Pratik Chaudhari. Model zoo: A growing" brain" that learns continually. *arXiv preprint arXiv:2106.03027*, 2021.

Yuhang Ran, Yi-Chen Li, Fuxiang Zhang, Zongzhang Zhang, and Yang Yu. Policy regularization with dataset constraint for offline reinforcement learning. In *International conference on machine learning*, pp. 28701–28717. PMLR, 2023.

Nadav Schor, Oren Katzir, Hao Zhang, and Daniel Cohen-Or. Componet: Learning to generate the unseen by part synthesis and composition. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8758–8767. IEEE, 2019.

Zuoqi Tang, Feifei Shao, Long Chen, Yunan Ye, Chao Wu, and Jun Xiao. Optimizing federated learning on non-iid data using local shapley value. In *CAAI International Conference on Artificial Intelligence*, pp. 164–175. Springer, 2021.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012.

Jiayi Wang, Shiqiang Wang, Rong-Rong Chen, and Mingyue Ji. A new theoretical perspective on data heterogeneity in federated optimization. *arXiv preprint arXiv:2407.15567*, 2024a.

Jin Wang, Jia Hu, Jed Mills, Geyong Min, Ming Xia, and Nektarios Georgalas. Federated ensemble model-based reinforcement learning in edge computing. *IEEE Transactions on Parallel and Distributed Systems*, 34(6):1848–1859, 2023a.

Qiang Wang, Bingyan Liu, and Yawen Li. Traceable federated continual learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12872–12881, 2024b.

Yuwei Wang and Burak Kantarci. Reputation-enabled federated learning model aggregation in mobile platforms. In *ICC 2021-IEEE international conference on communications*, pp. 1–6. IEEE, 2021.

Zhi Wang, Chunlin Chen, and Daoyi Dong. A dirichlet process mixture of robust task models for scalable lifelong reinforcement learning. *IEEE Transactions on Cybernetics*, 53(12):7509–7520, 2023b. doi: 10.1109/TCYB.2022.3170485.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

Abudukelimu Wuerkaixi, Sen Cui, Jingfeng Zhang, Kunda Yan, Bo Han, Gang Niu, Lei Fang, Changshui Zhang, and Masashi Sugiyama. Accurate forgetting for heterogeneous federated continual learning. In *International Conference on Learning Representations*, volume 2024, pp. 50778–50796, 2024.

Annie Xie, James Harrison, and Chelsea Finn. Deep reinforcement learning amidst continual structured non-stationarity. In *International Conference on Machine Learning*, pp. 11393–11403. PMLR, 2021.

Huaqing Xiong, Tengyu Xu, Lin Zhao, Yingbin Liang, and Wei Zhang. Deterministic policy gradient: Convergence analysis. In *Uncertainty in Artificial Intelligence*, pp. 2159–2169. PMLR, 2022.

Meng Xu, Xinhong Chen, and Jianping Wang. Policy correction and state-conditioned action evaluation for few-shot lifelong deep reinforcement learning. *IEEE transactions on neural networks and learning systems*, 36(4):6843–6857, 2024.

Meng Xu, Xinhong Chen, Weiwei Fu, Yi-Rong Lin, Yung-Hui Li, and Jianping Wang. A unified sparse training framework for lightweight lifelong deep reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 55(4):2998–3012, 2025a.

Meng Xu, Xinhong Chen, Weiwei Fu, Yi-Rong Lin, Yung-Hui Li, and Jianping Wang. A unified sparse training framework for lightweight lifelong deep reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 55(4):2998–3012, 2025b.

Meng Xu, Xinhong Chen, and Jianping Wang. Policy correction and state-conditioned action evaluation for few-shot lifelong deep reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, 36(4):6843–6857, 2025c.

Meng Xu, Xinhong Chen, Zihao Wen, Weiwei Fu, and Jianping Wang. A two-stage selective experience replay for double-actor deep reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–16, 2025d.

Meng Xu, Xinhong Chen, Zhongying Chen, Guanyi Zhao, Yang Jin, and Jianping Wang. A unified self-regulating training framework for federated deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 27296–27304, 2026.

Jaehong Yoon, Wonyong Jeong, Giwoong Lee, Eunho Yang, and Sung Ju Hwang. Federated continual learning with weighted inter-client transfer. In *International conference on machine learning*, pp. 12073–12086. PMLR, 2021.

Lang Yu, Lina Ge, Guanghui Wang, Jianghao Yin, Qin Chen, Jie Zhou, and Liang He. Pi-fed: Continual federated learning with parameter-level importance aggregation. *IEEE Internet of Things Journal*, 11(22):37187–37199, 2024.

Xiaoming Yuan, Kunda Huang, Boxuan Cao, Jiaqi Zhao, Changle Li, and Qingxu Deng. Dynamic offloading strategy in vehicular networks via federated continuous reinforcement learning. *IEEE Transactions on Vehicular Technology*, pp. 1–14, 2026.

Haochen Zhang, Zhong Zheng, and Lingzhou Xue. Regret-optimal q-learning with low cost for single-agent and federated reinforcement learning. *arXiv preprint arXiv:2506.04626*, 2025.

Tiantian Zhang, Xueqian Wang, Bin Liang, and Bo Yuan. Catastrophic interference in reinforcement learning: A solution based on context division and knowledge distillation. *IEEE Transactions on Neural Networks and Learning Systems*, 34(12):9925–9939, 2022.

Tiantian Zhang, Zichuan Lin, Yuxing Wang, Deheng Ye, Qiang Fu, Wei Yang, Xueqian Wang, Bin Liang, Bo Yuan, and Xiu Li. Dynamics-adaptive continual reinforcement learning via progressive contextualization. *IEEE Transactions on Neural Networks and Learning Systems*, 35(10):14588–14602, 2024a.

Zhi Zhang, Chris Chow, Yasi Zhang, Yanchao Sun, Haochen Zhang, Eric Hanchen Jiang, Han Liu, Furong Huang, Yuchen Cui, and Oscar Hernan Madrid Padilla. Statistical guarantees for lifelong reinforcement learning using pac-bayes theory. *arXiv preprint arXiv:2411.00401*, 2024b.

Zhengyi Zhong, Weidong Bao, Ji Wang, Jianguo Chen, Lingjuan Lyu, and Wei Yang Bryan Lim. Sacfl: Self-adaptive federated continual learning for resource-constrained end devices. *IEEE Transactions on Neural Networks and Learning Systems*, 36(9):17169–17183, 2025.

Meng-yuan Zhu, Zhuo Chen, Ke-fan Chen, Na Lv, and Yun Zhong. Attention-based federated incremental learning for traffic classification in the internet of things. *Computer Communications*, 185:168–175, 2022.

Hankz Hankui Zhuo, Wenfeng Feng, Yufeng Lin, Qian Xu, and Qiang Yang. Federated deep reinforcement learning. *arXiv preprint arXiv:1901.08277*, 2019.

A APPENDIX

A.1 THIRD AND FOURTH IMPLEMENTATIONS OF OUR METHOD

We demonstrate four implementations of our method, denoted as Method I through Method IV.

(3) Model Selection: Our framework supports flexible integration of any method for processing the sequential TD errors $\varepsilon_1(s,a), \varepsilon_2(s,a), \ldots, \varepsilon_{\mathcal{D}_0}(s,a)$ to compute model importance. Different methods essentially determine how each individual sample contributes to the final importance evaluation.

(3.1) Method I: The first approach directly computes the negative average of the TD errors as the importance score, i.e., $\ell = -\frac{1}{|\mathcal{D}_0|}\sum_{j=1}^{|\mathcal{D}_0|}\varepsilon_j(s,a)$. Applying this to each model in the buffer yields the importance vector $\boldsymbol{\ell} = [\ell_1, \ell_2, \ldots, \ell_L]$. While computationally simple, this method is sensitive to outliers, as extremely large or small values can disproportionately affect the final score.

(3.2) Method II: Given that Gaussian distributions are well-suited for modeling TD error distributions in high-dimensional DRL environments Williams (1992); Todorov et al. (2012); Nasiriany et al. (2021); Hollenstein et al. (2022), we assume the TD error of a policy follows a zero-mean Gaussian distribution, i.e., $\varepsilon \sim \mathcal{N}(0, \sigma^2)$, with variance fixed at a commonly used value $\sigma = 0.25$ Xu et al. (2025d). The selection likelihood for model π_k based on interaction data $\mathcal{D}_0$ is defined as:

$$p(\mathcal{D}_0|\pi_k) = \prod_{(s,a,s')\in\mathcal{D}_0} \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{\varepsilon_k^2}{2\sigma^2}\right). \tag{5}$$

However, multiplying many small likelihood values may lead to floating-point underflow Xu et al. (2024; 2025b). To mitigate this, we adopt logarithmic mean smoothing with exponential restoration, following prior work Xu et al. (2024; 2025b):

$$\ell_k = \exp\left(\frac{1}{|\mathcal{D}_0|}\sum_{(s,a)\in\mathcal{D}_0}\ln p(\mathcal{D}_0|\pi_k)\right). \tag{6}$$

The importance vector is then constructed as $\boldsymbol{\ell} = [\ell_1, \ell_2, \ldots, \ell_L]$. Although slightly more involved, this approach is robust to extreme outliers.

(3.3) Method III: The third approach adopts a pairwise preference strategy that relies solely on the relative ordering of TD errors rather than their absolute magnitudes. For each pair of models (π_i, π_j), model π_i is said to win on sample k if $\varepsilon_i^{(k)} < \varepsilon_j^{(k)}$, with ties contributing 0.5 points to each. The importance score of model π_k is defined as its total wins across all pairwise comparisons, normalized by the maximum possible score:

$$\ell_k = \frac{1}{|\mathcal{D}_0|(L-1)}\sum_{j\neq k}\sum_{m=1}^{|\mathcal{D}_0|}\Big[\mathbb{I}\big(\varepsilon_k^{(m)} < \varepsilon_j^{(m)}\big) + \tfrac{1}{2}\,\mathbb{I}\big(\varepsilon_k^{(m)} = \varepsilon_j^{(m)}\big)\Big], \tag{7}$$

where $\mathbb{I}(\cdot)$ is the indicator function. This formulation is inherently robust to outliers, as each sample contributes at most one vote regardless of the magnitude of its TD error. The importance vector is then $\boldsymbol{\ell} = [\ell_1, \ell_2, \ldots, \ell_L]$.

(3.4) Method IV: The fourth approach incorporates uncertainty awareness by jointly considering the mean and variance of per-sample TD errors. A model that exhibits low average TD error yet high variance across samples is deemed unreliable for the current task, as the inconsistency suggests epistemic uncertainty. The importance score is defined as:

$$\ell_k = -\big(\alpha\cdot\mu_k + \beta\cdot\sigma_k^2\big), \quad \mu_k = \frac{1}{|\mathcal{D}_0|}\sum_{j=1}^{|\mathcal{D}_0|}\varepsilon_k^{(j)}, \;\; \sigma_k^2 = \frac{1}{|\mathcal{D}_0|}\sum_{j=1}^{|\mathcal{D}_0|}\big(\varepsilon_k^{(j)} - \mu_k\big)^2, \tag{8}$$

where $\alpha, \beta > 0$ control the relative weights of the mean error and disagreement signal. We set both α and β to 0.5 in our experiments. The importance vector is $\boldsymbol{\ell} = [\ell_1, \ell_2, \ldots, \ell_L]$. This method penalizes models whose predictions fluctuate erratically across the interaction batch, favoring those that are both accurate and consistent.

(3.5) Probability Calculation for Model Selection: With the importance vector $\boldsymbol{\ell}$, we apply softmax normalization to reduce large magnitude discrepancies:

$$\boldsymbol{\phi}_k = \text{Softmax}(\boldsymbol{\ell})_k = \frac{\exp(\ell_k)}{\sum_{j=1}^{L} \exp(\ell_j)}. \tag{9}$$

The model with the highest probability is selected as the most suitable for the new task: $l^* = \arg\max(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_L)$. This chosen model l^* is then adopted as the global model θ_{global} and distributed to clients for local training on the current task.

A.2 THE PSEUDOCODE

The client-side training procedure for each task is presented in Algorithm 1.

Algorithm 1 Training procedure of our method (Client)

Initialize local actor θ^{μ}, critics θ^{Q_1} and θ^{Q_2}, and target networks $\theta^{\mu'}$, $\theta^{Q'_1}$, $\theta^{Q'_2}$
Initialize local dataset $\mathcal{D}$, maximum communication rounds E_c, and local training duration T
$round = 1$
repeat
  for $t = 1$ to T **do**
    Sample a random mini-batch $\mathcal{D}_0$ from $\mathcal{D}$
    Update critics: $\theta_i^{Q_j} = \arg\min_{\theta^Q} \mathcal{L}(\theta_i^{Q_j}, \theta_{global}^{Q})$
    Update actor: $\pi_{\theta_i^{\mu}} = \arg\min_{\pi} \mathcal{L}(\theta_i^{\mu}, \theta_{global}^{\mu})$
    Every d steps, update target networks: $\theta^{Q'_j} \leftarrow \tau\theta^{Q_j} + (1-\tau)\theta^{Q'_j}$, $\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'}$, for $j = 1, 2$
  end for
  Upload local models θ^{μ}, θ^{Q_1}, and θ^{Q_2} to the server
  $round \leftarrow round + 1$
until $E_c - round$ =0

The server-side workflow for each task is presented in Algorithm 2, taking the second implementation of our method as an example.

Algorithm 2 Operational procedure of our method (Server)

Initialize maximum communication rounds E_c, number of clients N_a, and number of critics N_{cr}
Set $round = 1$
repeat
  A new task commences
  Collect a mini-batch $\mathcal{D}_0 = \{(s, a, r, s')\}$ by interacting with the environment using a random policy
  for each policy π_k in the model buffer $\mathcal{M}$ **do**
    Compute target Q-value: $\hat{Q}(s,a) = r + \gamma \min_{i=1,2} Q_{\theta^{Q_i\prime}}(s', \tilde{a}), \tilde{a} \sim \pi_{\theta^{\mu\prime}}(s') + \mathcal{N}(0, 0.2)$
    Compute TD error: $\varepsilon_k(s,a) = \frac{1}{|\mathcal{D}_0|} \sum_{i=1}^{N_{cr}} \sum_{(s,a)\in\mathcal{D}_0} (\hat{Q}(s,a) - Q_{\theta^{Q_i}}(s,a))^2$
    Compute likelihood: $p(\mathcal{D}_0|\pi_k) = \prod_{(s,a,s')\in\mathcal{D}_0} \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{\varepsilon_k^2}{2\sigma^2})$ with $\sigma = 0.25$
    Compute selection likelihood: $\ell_k = \exp(\frac{1}{|\mathcal{D}_0|} \sum_{(s,a)\in\mathcal{D}_0} \ln p(\mathcal{D}_0|\pi_k))$
  end for
  Compute selection probabilities: $\boldsymbol{\phi}_k = \text{Softmax}(\boldsymbol{\ell})_k = \frac{\exp(\ell_k)}{\sum_{j=1}^{L} \exp(\ell_j)}$
  Select the best policy: $l^* = \arg\max(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_L)$
  Distribute the selected global model $(\theta^{\mu}_{global}, \theta^{Q}_{global})$ to all clients
  for $k = 1$ to N_a **do**
    Client k performs local training (see Algorithm 1)
    Client k uploads its local models $(\theta^{\mu}_k, \theta^{Q_1}_k, \theta^{Q_2}_k)$ to the server
  end for
  Aggregate models: $\theta^{agg} = F(\theta_1, \theta_2, \ldots, \theta_{N_a})$
  Distribute the updated global model $(\theta^{\mu}_{global}, \theta^{Q}_{global})$ to all clients
  $round \leftarrow round + 1$
until $E_c - round$ =0

A.3 THEORETICAL ANALYSIS

This section provides a theoretical analysis of the proposed method, characterizing its performance relative to the optimal policy in terms of discounted cumulative rewards and action-value functions. We first present a definition, one assumption, and six supporting lemmas that underpin Theorems 1–3. Lemmas 2–4 together with Assumption 1 are used to establish Lemma 6, which is central to the proof of Theorem 1. Theorem 1 demonstrates that incorporating our method can enhance local policy quality beyond that of baseline FDRL approaches. Theorem 2 follows directly from Theorem 1 to establish an upper bound on the performance gap between our method and the optimal policy, while Theorem 3 extends the result to Q-value discrepancies using Lemma 1.

Definition 1: A mapping $f : \mathbb{R}^n \to \mathbb{R}^m$ is Lipschitz continuous if there exists a constant $K \geq 0$ such that for all $x, y \in \mathbb{R}^n$,

$$\|f(x) - f(y)\| \leq K\|x - y\|, \tag{10}$$

which ensures that the function's variation is uniformly bounded globally.

Assumption 1: The reward function $r(s)$ satisfies $|r(s)| \leq r_{\max}$ for all states $s \in \mathcal{S}$, where $r_{\max} > 0$ is a fixed constant.

Lemma 1: Since the critic is realized by a neural network, it naturally fulfills the Lipschitz condition Gouk et al. (2021). Consequently, for any two policies π^1 and π^2, there exists a constant $K_o > 0$ such that

$$\left\|Q\left(s, \pi^1(s)\right) - Q\left(s, \pi^2(s)\right)\right\| \leq K_o \left\|\pi^1(s) - \pi^2(s)\right\|. \tag{11}$$

Lemma 2: Let $d^\pi : \mathcal{S} \to \mathbb{R}$ denote the occupancy measure induced by policy π Ran et al. (2023). Since policies are neural network parameterized, they satisfy Lipschitz continuity Gouk et al. (2021). As shown in Xiong et al. (2022), for any policies π_1 and π_2, there exists a constant $L_c > 0$ such that

$$\int_{\mathcal{S}} \left| d^{\pi_1(s)} - d^{\pi_2(s)} \right| \mathrm{d}s \leq L_c \max_{s \in \mathcal{S}} \|\pi_1(s) - \pi_2(s)\|. \tag{12}$$

The Lipschitz constants K_o and L_c are routinely adopted in reinforcement learning theory to support convergence and optimality analyses Bertsekas (2003)Chow & Tsitsiklis (1991)Dufour & Prieto-Rumeau (2013)Dufour & Prieto-Rumeau (2015). Recent DRL research also frequently relies on these constants together with occupancy measures to establish theoretical guarantees Xiong et al. (2022). Their presence is essential for the derivations that follow.

Lemma 3: For a fixed discount factor $\gamma \in (0, 1)$, the long-term reward objective of DRL, defined as

$$J(\pi(s)) = \max \mathbb{E}_{\pi(s)} \left[\sum_{i=t}^{T} \gamma^{i-t} r_i \mid s_0, a_0 \right],$$

where T is the learning horizon and t starts at 0, can be expressed as follows Xiong et al. (2022):

$$J(\pi(\theta^\mu)) = \frac{1}{1-\gamma} \mathbb{E}_{s \sim d^{\pi(\theta^\mu)}}[r(s)]. \tag{13}$$

Here, γ denotes the discount factor.

Lemma 4: The expectation over the occupancy measure is defined as Ran et al. (2023)

$$\mathbb{E}_{s \sim d^{\pi(s)}}[r(s)] = \int_{\mathcal{S}} r(s)\, d^{\pi(s)}\, \mathrm{d}s. \tag{14}$$

Lemma 5: Let $f : S \subset \mathbb{R}^m \to \mathbb{R}$ be integrable on $[p, q] \subset S$. Then

$$\left| \int_p^q f(x)\, \mathrm{d}x \right| \leq \int_p^q |f(x)|\, \mathrm{d}x. \tag{15}$$

Proof: Define the positive part $f^+(x) = \max\{f(x), 0\}$ and the negative part $f^-(x) = \max\{-f(x), 0\}$. Then $f(x) = f^+(x) - f^-(x)$ and $|f(x)| = f^+(x) + f^-(x)$ for all $x \in [p, q]$. By the linearity of the integral,

$$\int_p^q f(x)\, \mathrm{d}x = \int_p^q f^+(x)\, \mathrm{d}x - \int_p^q f^-(x)\, \mathrm{d}x. \tag{16}$$

Since both $f^+(x)$ and $f^-(x)$ are non-negative, their integrals are non-negative. Therefore,

$$\left|\int_p^q f(x)\,\mathrm{d}x\right| = \left|\int_p^q f^+(x)\,\mathrm{d}x - \int_p^q f^-(x)\,\mathrm{d}x\right| \le \int_p^q f^+(x)\,\mathrm{d}x + \int_p^q f^-(x)\,\mathrm{d}x = \int_p^q |f(x)|\,\mathrm{d}x, \tag{17}$$

where the inequality follows from the elementary fact that for any non-negative real numbers a, b, we have $|a-b| \le a+b$. This completes the proof.

Lemma 6: For two policies π^1 and π^2, the performance difference satisfies

$$\begin{aligned}
&|J(\pi^1(s)) - J(\pi^2(s))| \\
&\overset{(Lemma3)}{=} \frac{1}{1-\gamma}\left|\mathbb{E}_{s\sim d^{\pi^1(s)}}[r(s)] - \mathbb{E}_{s\sim d^{\pi^2(s)}}[r(s)]\right| \\
&\overset{(Lemma4)}{=} \frac{1}{1-\gamma}\left|\int_{\mathcal{S}} r(s)\left(d^{\pi^1(s)} - d^{\pi^2(s)}\right)\mathrm{d}s\right| \\
&\overset{(Lemma5)}{\le} \frac{1}{1-\gamma}\int_{\mathcal{S}} |r(s)|\left|d^{\pi^1(s)} - d^{\pi^2(s)}\right|\mathrm{d}s \\
&\overset{(Assumption1)}{\le} \frac{r_{\max}}{1-\gamma}\int_{\mathcal{S}} \left|d^{\pi^1(s)} - d^{\pi^2(s)}\right|\mathrm{d}s \\
&\overset{(Lemma2)}{\le} \frac{r_{\max}L_c}{1-\gamma}\max_{s\in\mathcal{S}}\|\pi^1(s) - \pi^2(s)\|.
\end{aligned} \tag{18}$$

Since $\frac{r_{\max}L_c}{1-\gamma}$ remains a fixed positive constant, the difference in performance between any two policies is effectively constrained by their divergence.

Theorem 1: Consider a set of N_a local models indexed by $i = 1, \ldots, N_a$. Let $\pi_{l_i}(s)$ denote the policy of the i-th local model, and $\pi^*_{l_i}(s)$ its theoretical optimum. Let $\pi_g(s)$ and $\pi^*_g(s)$ be the global policy and its optimal counterpart, respectively. The performance gap between the optimal global policy and the one derived from different CFDRL methods is bounded as follows:

$$\begin{aligned}
&|J(\pi^*_g(s)) - J(\pi_g(s))| \\
&= |J(\pi^*_g(s)) + J(\pi^*_{l_1}(s)) - J(\pi^*_{l_1}(s)) - J(\pi_g(s))| \\
&\le |J(\pi^*_g(s)) - J(\pi^*_{l_1}(s))| + |J(\pi^*_{l_1}(s)) - J(\pi_g(s))| \\
&= |J(\pi^*_g(s)) - J(\pi^*_{l_1}(s))| + |J(\pi^*_{l_1}(s)) + J(\pi^*_{l_2}(s)) - J(\pi^*_{l_2}(s)) - J(\pi_g(s))| \\
&\le |J(\pi^*_g(s)) - J(\pi^*_{l_1}(s))| + |J(\pi^*_{l_1}(s)) - J(\pi^*_{l_2}(s))| + |J(\pi^*_{l_2}(s)) - J(\pi_g(s))| \\
&\le |J(\pi^*_g(s)) - J(\pi^*_{l_1}(s))| + \sum_{k=1}^{N_a-1} |J(\pi^*_{l_k}(s)) - J(\pi^*_{l_{k+1}}(s))| + |J(\pi^*_{l_{N_a}}(s)) - J(\pi_g(s))| \\
&\le \frac{r_{\max}L_c}{1-\gamma}\left(\max_s \|\pi^*_g(s) - \pi^*_{l_1}(s)\| + \sum_{k=1}^{N_a-1}\max_s \|\pi^*_{l_k}(s) - \pi^*_{l_{k+1}}(s)\| + \max_s \|\pi^*_{l_{N_a}}(s) - \pi_g(s)\|\right).
\end{aligned} \tag{19}$$

This bound holds for any ordering of the N_a local models. Let $\pi^o_g(s)$ denote the global policy learned by our method, and $\pi^b_g(s)$ denote the global policy learned by baseline FDRL methods. Accordingly, the performance gap between the optimal global policy and the one derived from our method is bounded as follows:

$$\begin{aligned}
&|J(\pi^*_g(s)) - J(\pi^o_g(s))| \\
&\le \frac{r_{\max}L_c}{1-\gamma}\left(\max_s \|\pi^*_g(s) - \pi^*_{l_1}(s)\| + \sum_{k=1}^{N_a-1}\max_s \|\pi^*_{l_k}(s) - \pi^*_{l_{k+1}}(s)\| + \max_s \|\pi^*_{l_{N_a}}(s) - \pi^o_g(s)\|\right).
\end{aligned} \tag{20}$$

Similarly, the performance gap between the optimal global policy and the one derived from baseline FDRL methods is bounded as follows:

$$
\begin{aligned}
&|J(\pi_g^*(s)) - J(\pi_g^b(s))| \\
&\leq \frac{r_{\max} L_c}{1-\gamma} \left(\max_s \|\pi_g^*(s) - \pi_{l_1}^*(s)\| + \sum_{k=1}^{N_a-1} \max_s \|\pi_{l_k}^*(s) - \pi_{l_{k+1}}^*(s)\| + \max_s \|\pi_{l_{N_a}}^*(s) - \pi_g^b(s)\| \right).
\end{aligned} \tag{21}
$$

Our method maintains a model buffer that stores models trained on previous tasks to preserve prior knowledge, and employs a model selection strategy that chooses the most suitable model for each client. This selection favors prior models exhibiting smaller TD errors with respect to the current task dynamics, indicating better adaptability. Through this buffer and selection mechanism, each local policy can improve upon its baseline. Aggregating these enhanced local policies yields a more powerful global policy, which in turn facilitates the convergence of each local policy toward its theoretical optimum. Consequently, our approach reduces the term $\max_s \|\pi_{l_{N_a}}^*(s) - \pi_g(s)\|$. In contrast, existing FDRL and CFDRL methods suffer from catastrophic forgetting due to fixed network capacity, which leads to inefficient knowledge preservation and consequently prevents the agent from achieving optimal policies. Therefore, the bound on $\max_s \|\pi_{l_{N_a}}^*(s) - \pi_g(s)\|$ achieved by our method is tighter than that of existing FDRL approaches, i.e.,

$$
\max_s \|\pi_{l_{N_a}}^*(s) - \pi_g^o(s)\| \leq \max_s \|\pi_{l_{N_a}}^*(s) - \pi_g^b(s)\|. \tag{22}
$$

It then follows that

$$
\begin{aligned}
&\frac{r_{\max} L_c}{1-\gamma} \left(\max_s \|\pi_g^*(s) - \pi_{l_1}^*(s)\| + \sum_{k=1}^{N_a-1} \max_s \|\pi_{l_k}^*(s) - \pi_{l_{k+1}}^*(s)\| + \max_s \|\pi_{l_{N_a}}^*(s) - \pi_g^o(s)\| \right) \\
&\leq \frac{r_{\max} L_c}{1-\gamma} \left(\max_s \|\pi_g^*(s) - \pi_{l_1}^*(s)\| + \sum_{k=1}^{N_a-1} \max_s \|\pi_{l_k}^*(s) - \pi_{l_{k+1}}^*(s)\| + \max_s \|\pi_{l_{N_a}}^*(s) - \pi_g^b(s)\| \right).
\end{aligned} \tag{23}
$$

Thus, our method attains a smaller upper bound on the deviation from the optimal global policy compared to baseline methods, indicating that it is potentially closer to the theoretical optimum.

Theorem 2: In practical FDRL settings, the global model communicates with local models only at fixed synchronization intervals, ensuring that their deviations remain bounded. Meanwhile, each local model progressively converges toward its optimal policy during training, so the discrepancy between them is also finite. For each local model $i \in \{1, \ldots, N_a\}$, we obtain the following:

$$
\max_s \|\pi_{l_i}(s) - \pi_g(s)\| \leq \zeta_1^{(i)}, \quad \max_s \|\pi_g^*(s) - \pi_{l_i}^*(s)\| \leq \zeta_2^{(i)}, \quad \max_s \|\pi_{l_i}^*(s) - \pi_{l_i}(s)\| \leq \zeta_3^{(i)}. \tag{24}
$$

Additionally, since the environmental discrepancies across clients are bounded, we can obtain the following for consecutive local optimal policies:

$$
\max_s \|\pi_{l_k}^*(s) - \pi_{l_{k+1}}^*(s)\| \leq \zeta_4^{(k)}, \quad k = 1, \ldots, N_a - 1. \tag{25}
$$

Applying the extended Theorem 1 and taking the minimum over all possible orderings yields

$$
|J(\pi_g^*(s)) - J(\pi_g(s))| \leq \frac{r_{\max} L_c}{1-\gamma} \min_{\text{orderings}} \left(\zeta_2^{(i)} + \sum_{k=1}^{N_a-1} \zeta_4^{(k)} + \zeta_1^{(j)} \right), \tag{26}
$$

where i and j denote the first and last local models in the ordering, respectively. Equivalently, with

$$
\zeta_2 = \min_i \zeta_2^{(i)}, \quad \zeta_4 = \min_k \zeta_4^{(k)}, \quad \zeta_1 = \min_i \zeta_1^{(i)},
$$

we obtain the simplified bound

$$
|J(\pi_g^*(s)) - J(\pi_g(s))| \leq \frac{r_{\max} L_c}{1-\gamma} \left(\zeta_2 + (N_a - 1)\zeta_4 + \zeta_1 \right). \tag{27}
$$

Theorem 3: Let $Q(s,\pi)$ denote the action-value function under policy π. Using Lemma 1 and the bounds from Theorem 2, the discrepancy between the optimal global Q-value and that of the learned global Q-value is bounded as follows:

$$
\begin{aligned}
&\|Q(s,\pi_g^*(s)) - Q(s,\pi_g(s))\| \\
&= \|Q(s,\pi_g^*(s)) + Q(s,\pi_{l_1}^*(s)) - Q(s,\pi_{l_1}^*(s)) - Q(s,\pi_g(s))\| \\
&\le \|Q(s,\pi_g^*(s)) - Q(s,\pi_{l_1}^*(s))\| + \|Q(s,\pi_{l_1}^*(s)) - Q(s,\pi_g(s))\| \\
&= \|Q(s,\pi_g^*(s)) - Q(s,\pi_{l_1}^*(s))\| + \|Q(s,\pi_{l_1}^*(s)) + Q(s,\pi_{l_2}^*(s)) - Q(s,\pi_{l_2}^*(s)) - Q(s,\pi_g(s))\| \\
&\le \|Q(s,\pi_g^*(s)) - Q(s,\pi_{l_1}^*(s))\| + \|Q(s,\pi_{l_1}^*(s)) - Q(s,\pi_{l_2}^*(s))\| + \|Q(s,\pi_{l_2}^*(s)) - Q(s,\pi_g(s))\| \\
&\le \|Q(s,\pi_g^*(s)) - Q(s,\pi_{l_1}^*(s))\| + \sum_{k=1}^{N_a-1} \|Q(s,\pi_{l_k}^*(s)) - Q(s,\pi_{l_{k+1}}^*(s))\| \\
&+ \|Q(s,\pi_{l_{N_a}}^*(s)) - Q(s,\pi_g(s))\| \\
&\le K_o \left(\|\pi_g^*(s) - \pi_{l_1}^*(s)\| + \sum_{k=1}^{N_a-1} \|\pi_{l_k}^*(s) - \pi_{l_{k+1}}^*(s)\| + \|\pi_{l_{N_a}}^*(s) - \pi_g(s)\| \right) \\
&\le K_o \left(\max_s \|\pi_g^*(s) - \pi_{l_1}^*(s)\| + \sum_{k=1}^{N_a-1} \max_s \|\pi_{l_k}^*(s) - \pi_{l_{k+1}}^*(s)\| + \max_s \|\pi_{l_{N_a}}^*(s) - \pi_g(s)\| \right) \\
&\le K_o \left(\zeta_2 + (N_a - 1)\zeta_4 + \zeta_1 \right).
\end{aligned}
\tag{28}
$$

A.4 Implementation Details

Since the implementation details of each baseline method are fully described in their respective references, we only present the shared underlying DRL model configurations adopted by both the baselines and our method here.

All clients share an identical DRL architecture based on a multilayer perceptron with two hidden layers and one output layer. Both the actor and critic networks utilize 256 neurons per hidden layer with ReLU activations, whereas Tanh is applied to the actor's output. A fixed learning rate of 0.001 is applied to both networks via the Adam optimizer, with a batch size of 32 samples used for training and drawn from a replay buffer of 10000 transitions. The discount factor is set to 0.99, and target networks are updated with a soft coefficient of 0.05 every two steps. For SAC-based CFDRL implementations, the temperature coefficient remains constant at 1 without adaptive tuning. In our model selection strategy, a random policy is executed to collect a mini-batch of 1000 samples from the environment, and the resulting TD errors are used for model selection.

A.5 Description of the benchmark tasks

Our experimental validation of the proposed CFDRL approach encompasses ten benchmark tasks, with five clients allocated to each. The task suite includes four robot navigation benchmarks Zhang et al. (2024a)Xu et al. (2025c), namely **Nav-Reacher-v1**, **Nav-Reacher-v2**, **Nav-v1**, and **Nav-v2**; two robotic arm control benchmarks, **RobotArm-v1** and **RobotArm-v2** Ding et al. (2023); and four MuJoCo locomotion benchmarks Zhang et al. (2024a)Xu et al. (2025c), specifically **HalfCheetah-vel**, **Humanoid-vel**, **Walker-vel**, and **Swimmer-vel**. The navigation variants differ in environmental complexity, with v2 introducing obstacle avoidance requirements absent in v1. The locomotion tasks challenge agents to sustain a prescribed forward velocity along the x-axis. The robotic arm scenarios involve a three-degree-of-freedom Dobot Magician, whose end-effector must be steered from an initial configuration to a designated target while circumventing circular barriers. Fig. 2 illustrates these ten benchmark tasks.

To create CDRL experimental settings, we designed a series of non-stationary distribution tasks for each simulation environment, where initial states were randomly varied to permit repeated task executions. In the case of navigation and robotic arm simulations, the randomness was introduced

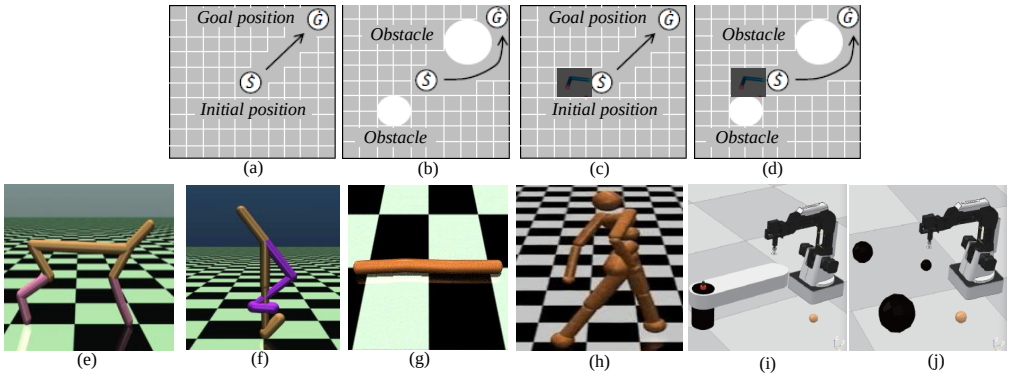


Figure 2: Experimental setups: (a) Nav-v1, (b) Nav-v2, (c) Nav-Reacher-v1, (d) Nav-Reacher-v2, with obstacles shown as white circles, the starting point labeled as S, and the goal position as G. (e) HalfCheetah-vel, (f) Walker-vel, (g) Swimmer-vel, (h) Humanoid-vel, (i) RobotArm-v1, and (j) RobotArm-v2.

through differing start coordinates, goal positions, and barrier configurations. As for the MuJoCo-based locomotion environments, the task variability was achieved by adjusting the preset velocity commands across tasks. To construct FDRL settings, we introduce client heterogeneity by randomly perturbing each client's environmental parameters, such as target positions and movement velocities, within a 10% range of their nominal values.

A.6 Improving for FDRL methods

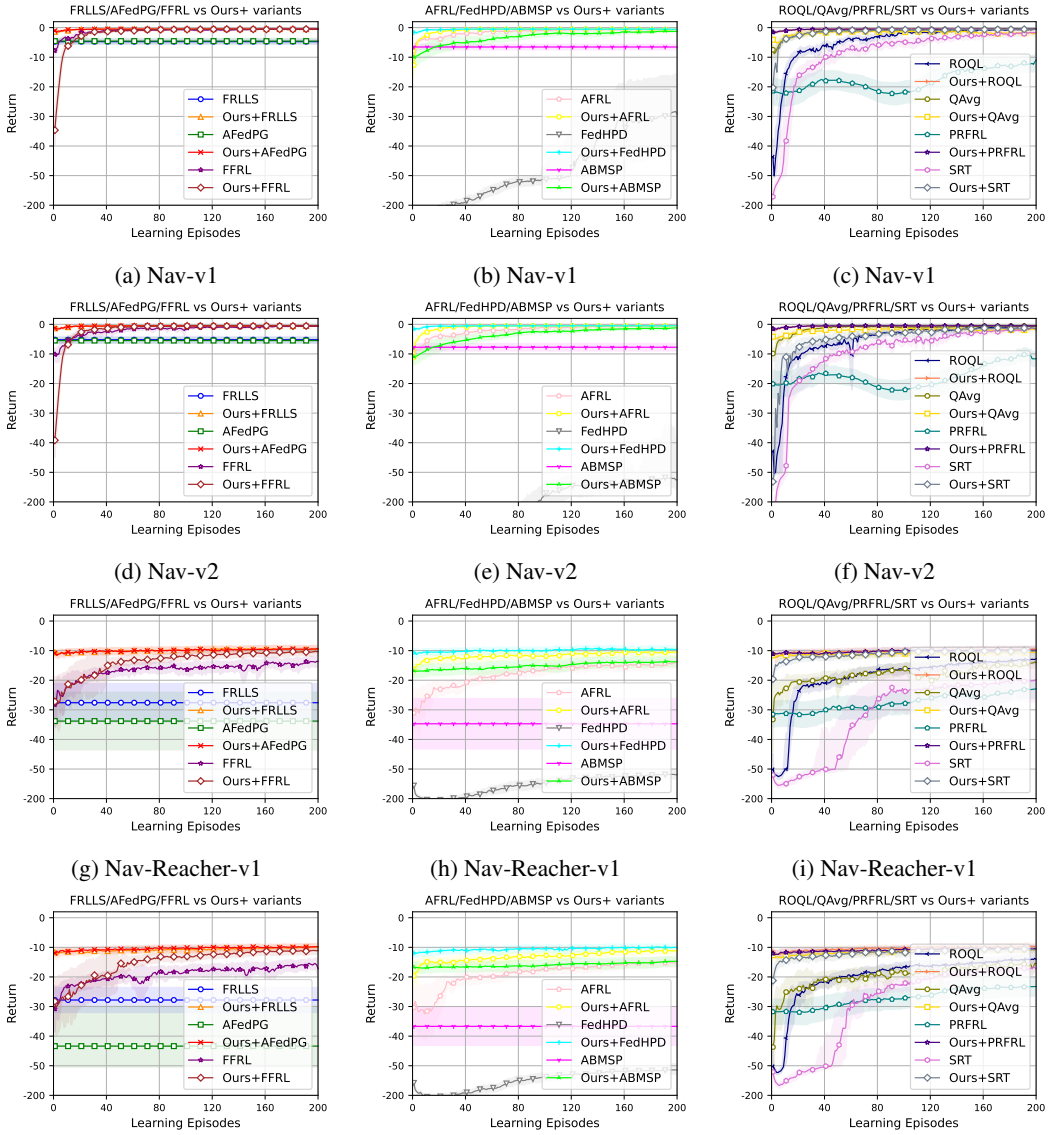
This section evaluates the improvements of our method over ten SOTA FDRL approaches under the CFDRL configuration across ten benchmark tasks, each consisting of 20 non-stationary tasks. In addition to static benchmark environments, we also create dynamic environments by randomly perturbing the physical parameters of each client environment and the global model environment within a 50% range of their nominal values. Figs. 3 to 7 present the experimental results in terms of episode return. From these results, when our method is integrated with baseline FDRL approaches, these methods achieve superior performance under the CFDRL configuration in the vast majority of cases.

A.7 Comparison of SOTA expansion-based CFL approaches

This section evaluates the effectiveness of our method in improving FDRL over state-of-the-art (SOTA) expansion-based continual federated learning (CFL) approaches. We integrate four methods, namely our approach, **FedCliP** Li et al. (2023), **FedWeIT** Yoon et al. (2021), and **FCL-BL** Le et al. (2021), into three FDRL baselines, **FedHPD**, **AFRL**, and **AFedPG**, and compare their performance across five benchmark tasks in terms of average return. Each benchmark task is configured with 20 tasks and five clients. The experimental results, presented in Table 6, show that our method consistently outperforms all three SOTA CFL counterparts across all evaluated FDRL methods. We attribute this superiority to the inherent limitations of existing expansion-based approaches. FedCliP and FedWeIT employ model pruning to reduce the computational cost incurred by expansion, but this inevitably removes previously acquired knowledge and results in suboptimal policies. FCL-BL relies on randomly initialized enhancement nodes, which scatter knowledge across the feature space in an uncontrolled manner, leading to unsatisfactory knowledge preservation and policy quality. In contrast, our method effectively retains prior knowledge through a model buffer and leverages it via a model selection strategy, thereby achieving marked improvement in FDRL performance.

A.8 Comparison of four implementations of our method

This section investigates the performance of different implementations of our method. The baseline algorithms include FRLLS, AFedPG, FFRL, and AFRL. Evaluations are carried out on the RobotArm-v1 and RobotArm-v2 benchmarks, each configured with 20 tasks. Table 7 presents the variant



(a) Nav-v1 (b) Nav-v1 (c) Nav-v1

(d) Nav-v2 (e) Nav-v2 (f) Nav-v2

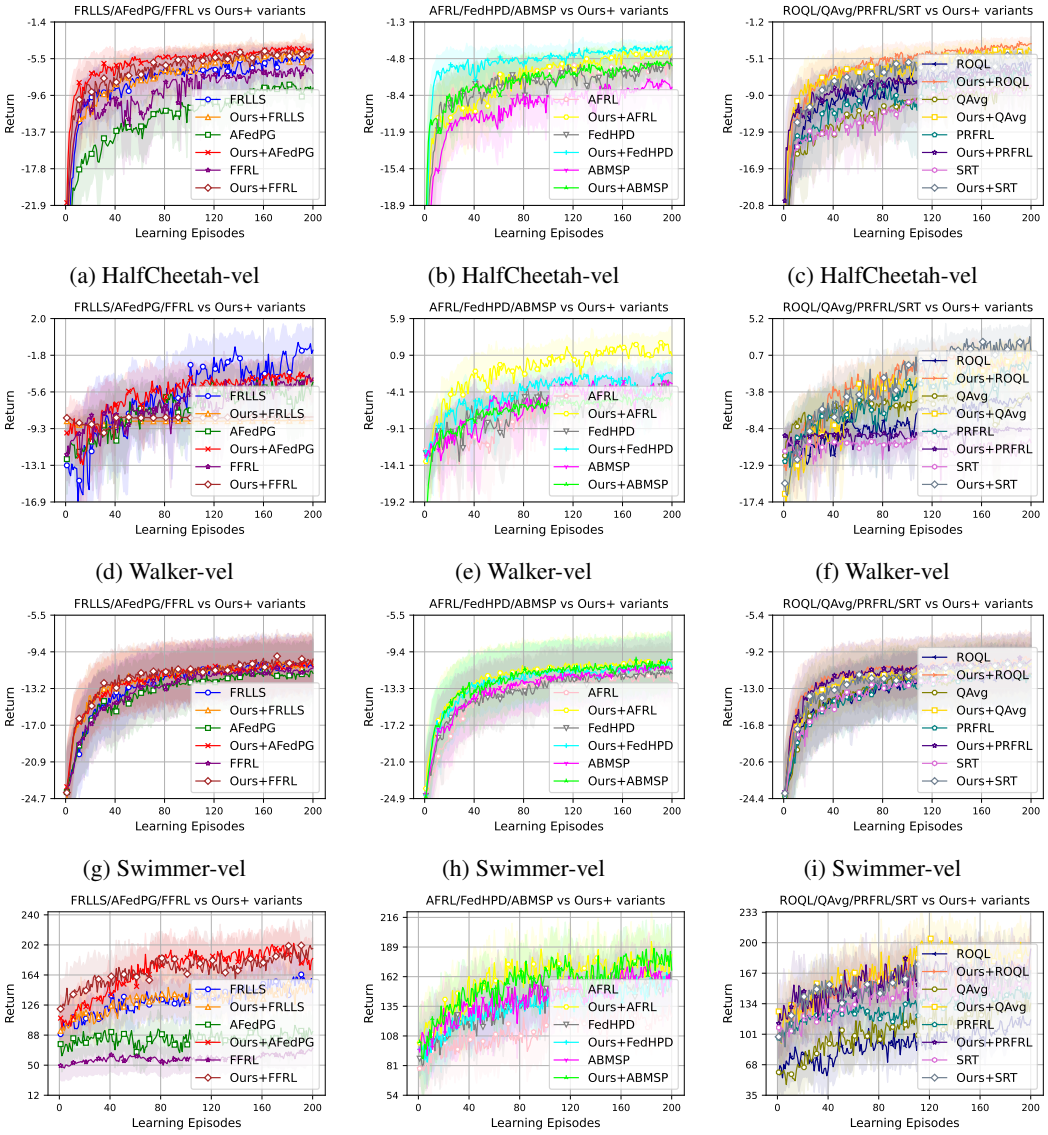
(g) Nav-Reacher-v1 (h) Nav-Reacher-v1 (i) Nav-Reacher-v1

(j) Nav-Reacher-v2 (k) Nav-Reacher-v2 (l) Nav-Reacher-v2

Figure 3: Episode returns on static navigation tasks, comparing our approach with baseline FDRL methods. Results are averaged over 20 tasks and five random seeds. The bold curve denotes the mean return per episode, and the shaded region corresponds to the standard deviation. All subsequent figures adopt this presentation style.

Table 6: Comparison of SOTA expansion-based CFL approaches (Mean $\pm$ Standard deviation).

Methods	Nav-v2	Nav-Reacher-v1	Nav-Reacher-v2	RobotArm-v1	RobotArm-v2
FedHPD+FedCliP	-0.720 ± 0.013	-14.267 ± 0.031	-13.779 ± 0.035	-1.062 ± 0.022	-1.383 ± 0.028
FedHPD+FedWeIT	-1.959 ± 0.054	-13.230 ± 0.028	-14.638 ± 0.050	-1.797 ± 0.039	-2.730 ± 0.084
FedHPD+FCL-BL	-4.520 ± 0.111	-24.639 ± 0.249	-27.621 ± 0.156	-4.512 ± 0.094	-11.006 ± 0.271
Ours+FedHPD	$\mathbf{-0.513} \pm 0.004$	$\mathbf{-9.670} \pm 0.015$	$\mathbf{-10.065} \pm 0.017$	$\mathbf{-0.831} \pm 0.009$	$\mathbf{-1.058} \pm 0.010$
AFRL+FedCliP	-117.249 ± 0.791	-74.044 ± 0.706	-76.753 ± 0.614	-69.444 ± 1.197	-386.944 ± 4.095
AFRL+FedWeIT	-2.162 ± 0.073	-13.681 ± 0.035	-13.839 ± 0.050	-1.805 ± 0.039	-2.354 ± 0.049
AFRL+FCL-BL	-4.927 ± 0.076	-31.009 ± 0.154	-27.018 ± 0.187	-5.084 ± 0.089	-10.742 ± 0.203
Ours+AFRL	$\mathbf{-0.512} \pm 0.003$	$\mathbf{-10.575} \pm 0.022$	$\mathbf{-11.295} \pm 0.023$	$\mathbf{-0.747} \pm 0.003$	$\mathbf{-1.016} \pm 0.007$
AFedPG+FedCliP	-8.228 ± 0.012	-43.161 ± 0.022	-41.169 ± 0.029	-12.715 ± 0.020	-18.408 ± 0.032
AFedPG+FedWeIT	-1.775 ± 0.034	-12.851 ± 0.035	-14.091 ± 0.073	-2.274 ± 0.102	-3.138 ± 0.101
AFedPG+FCL-BL	-4.157 ± 0.119	-27.114 ± 0.118	-30.921 ± 0.240	-4.478 ± 0.071	-9.252 ± 0.152
Ours+AFedPG	$\mathbf{-0.517} \pm 0.004$	$\mathbf{-9.497} \pm 0.015$	$\mathbf{-9.925} \pm 0.016$	$\mathbf{-0.770} \pm 0.005$	$\mathbf{-1.041} \pm 0.008$



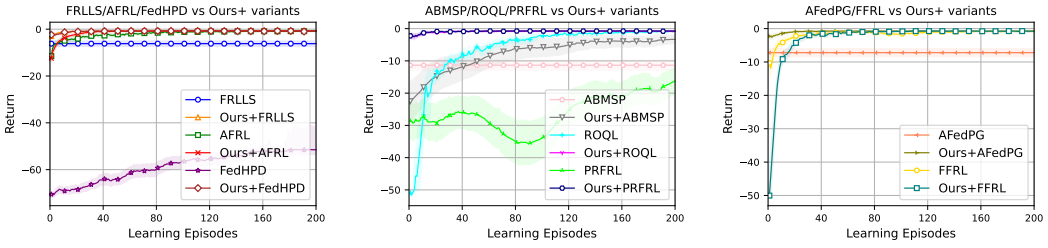
(a) HalfCheetah-vel (b) HalfCheetah-vel (c) HalfCheetah-vel

(d) Walker-vel (e) Walker-vel (f) Walker-vel

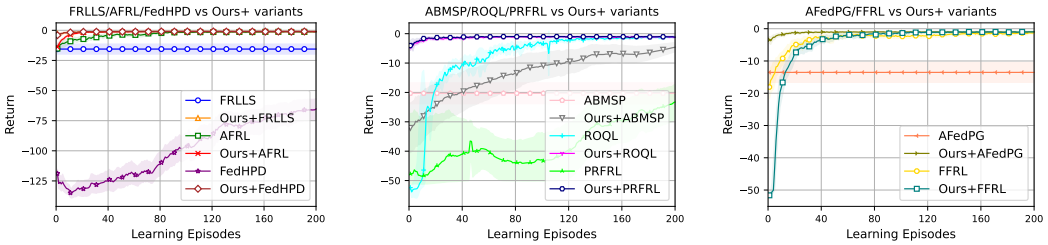
(g) Swimmer-vel (h) Swimmer-vel (i) Swimmer-vel

(j) Humanoid-vel (k) Humanoid-vel (l) Humanoid-vel

Figure 4: Episode returns on MuJoCo locomotion tasks, comparing our approach with baseline FDRL methods.

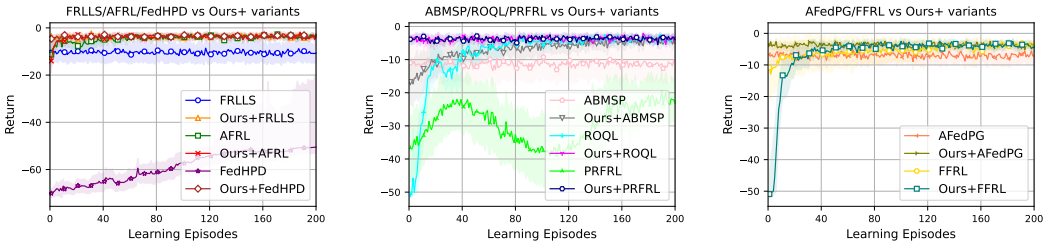


(a) RobotArm-v1 (b) RobotArm-v1 (c) RobotArm-v1

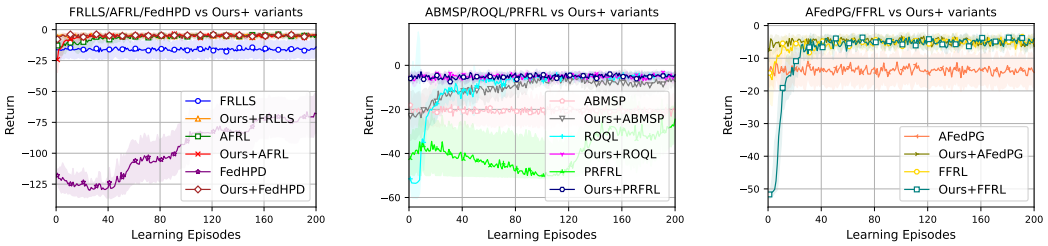


(d) RobotArm-v2 (e) RobotArm-v2 (f) RobotArm-v2

Figure 5: Episode returns on static robotic arm tasks, comparing our approach with baseline FDRL methods.

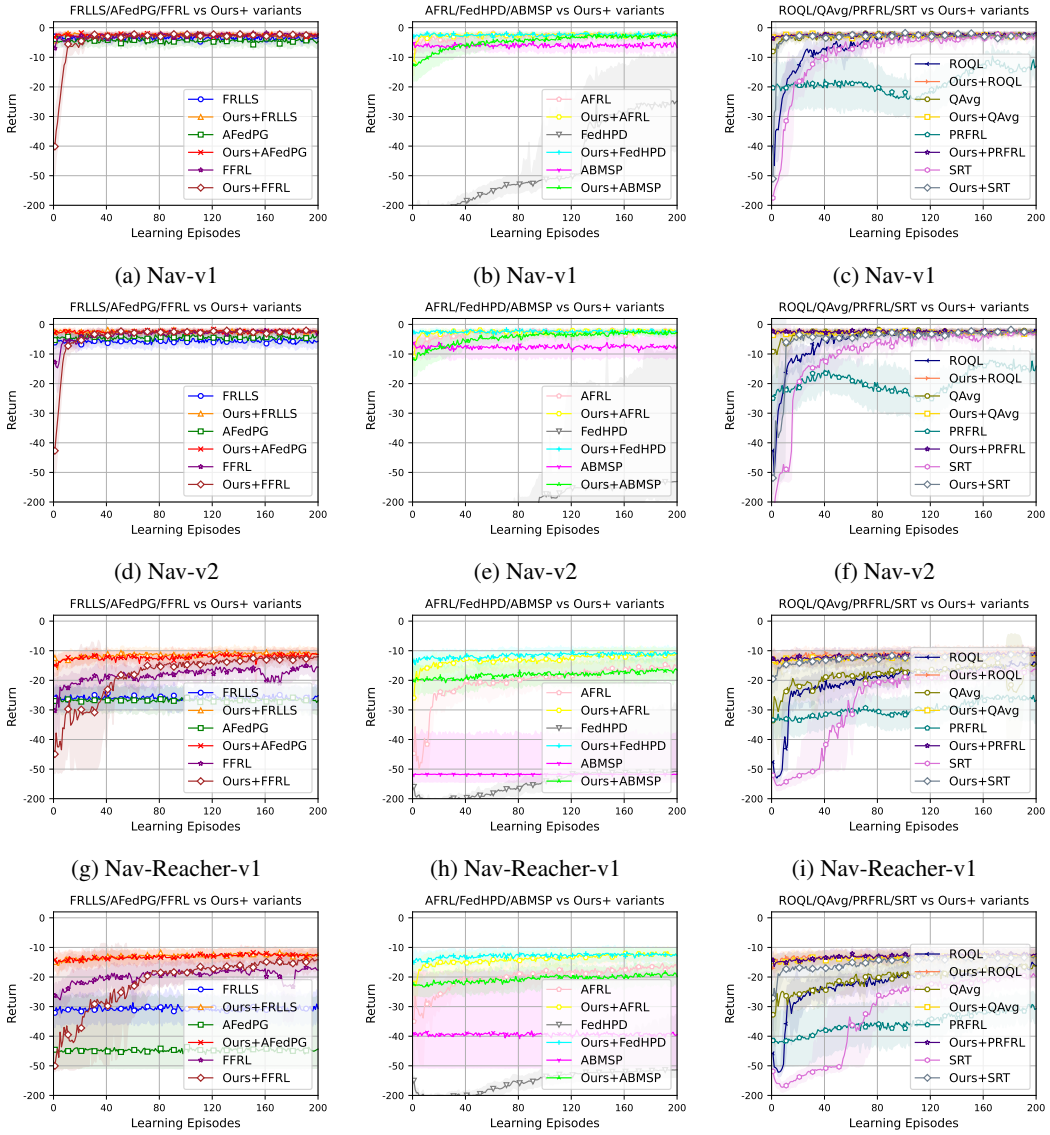


(a) RobotArm-v1 (b) RobotArm-v1 (c) RobotArm-v1



(d) RobotArm-v2 (e) RobotArm-v2 (f) RobotArm-v2

Figure 6: Episode returns on dynamic robotic arm tasks, comparing our approach with baseline FDRL methods.



(a) Nav-v1 (b) Nav-v1 (c) Nav-v1

(d) Nav-v2 (e) Nav-v2 (f) Nav-v2

(g) Nav-Reacher-v1 (h) Nav-Reacher-v1 (i) Nav-Reacher-v1

(j) Nav-Reacher-v2 (k) Nav-Reacher-v2 (l) Nav-Reacher-v2

Figure 7: Episode returns on dynamic navigation tasks, comparing our approach with baseline FDRL methods.

Table 7: Variants analysis of our method. Ours+FRLLS (I) denotes the integration of the first implementation of our method into FRLLS, and the other notations follow the same convention.

Methods	RobotArm-v1	RobotArm-v2
FRLLS	-6.182 ± 0.000	-15.573 ± 0.000
Ours+FRLLS (I)	-0.856 ± 0.013	-1.167 ± 0.030
Ours+FRLLS (II)	-0.783 ± 0.005	-1.011 ± 0.008
Ours+FRLLS (III)	-0.855 ± 0.012	-1.235 ± 0.030
Ours+FRLLS (IV)	-0.846 ± 0.013	-1.214 ± 0.025
AFedPG	-7.210 ± 0.000	-13.539 ± 0.000
Ours+AFedPG (I)	-0.862 ± 0.016	-1.230 ± 0.034
Ours+AFedPG (II)	-0.770 ± 0.005	-1.041 ± 0.008
Ours+AFedPG (III)	-0.972 ± 0.019	-1.099 ± 0.017
Ours+AFedPG (IV)	-0.832 ± 0.012	-1.235 ± 0.038
FFRL	-11.347 ± 0.000	-20.292 ± 0.000
Ours+FFRL (I)	-0.817 ± 0.008	-1.544 ± 0.089
Ours+FFRL (II)	-0.778 ± 0.005	-1.028 ± 0.007
Ours+FFRL (III)	-0.804 ± 0.005	-1.338 ± 0.034
Ours+FFRL (IV)	-0.826 ± 0.009	-1.129 ± 0.016
AFRL	-1.025 ± 0.019	-1.309 ± 0.019
Ours+AFRL (I)	-0.818 ± 0.010	-1.138 ± 0.036
Ours+AFRL (II)	-0.747 ± 0.003	-1.016 ± 0.007
Ours+AFRL (III)	-0.804 ± 0.006	-1.186 ± 0.019
Ours+AFRL (IV)	-0.874 ± 0.017	-1.257 ± 0.025

Table 8: Performance comparison under varying client quantities. Here, Ours+FFRL (20) denotes the Ours+FFRL method with 20 clients.

Methods	RobotArm-v1	RobotArm-v2
FRLLS (20)	-0.776 ± 0.005	-1.299 ± 0.016
Ours+FRLLS (20)	$\mathbf{-0.758} \pm 0.007$	$\mathbf{-1.012} \pm 0.006$
AFedPG (20)	-58.236 ± 1.059	-345.138 ± 7.516
Ours+AFedPG (20)	$\mathbf{-0.796} \pm 0.011$	$\mathbf{-1.062} \pm 0.012$
FFRL (20)	-3.535 ± 0.081	-7.991 ± 0.292
Ours+FFRL (20)	$\mathbf{-0.815} \pm 0.009$	$\mathbf{-1.025} \pm 0.008$
AFRL (20)	-19.796 ± 0.000	-33.283 ± 0.000
Ours+AFRL (20)	-30.646 ± 0.215	-135.969 ± 0.064
FRLLS (40)	-0.777 ± 0.003	-1.277 ± 0.015
Ours+FRLLS (40)	$\mathbf{-0.769} \pm 0.013$	$\mathbf{-1.049} \pm 0.010$
AFedPG (40)	-77.232 ± 1.721	-350.388 ± 6.536
Ours+AFedPG (40)	$\mathbf{-0.774} \pm 0.007$	$\mathbf{-1.015} \pm 0.006$
FFRL (40)	-5.104 ± 0.125	-8.564 ± 0.269
Ours+FFRL (40)	$\mathbf{-0.974} \pm 0.014$	$\mathbf{-1.018} \pm 0.006$
AFRL (40)	-21.451 ± 0.000	-38.786 ± 0.000
Ours+AFRL (40)	$\mathbf{-15.681} \pm 0.272$	$\mathbf{-17.065} \pm 0.295$

Table 9: Scalability analysis. Here, Ours+FFRL (50) denotes the Ours+FFRL method trained over 50 tasks.

Methods	RobotArm-v1	RobotArm-v2
FRLLS (50)	-0.820 ± 0.004	-1.617 ± 0.060
Ours+FRLLS (50)	$\mathbf{-0.791} \pm 0.008$	$\mathbf{-1.038} \pm 0.007$
FRLLS (70)	-1.143 ± 0.008	-1.966 ± 0.022
Ours+FRLLS (70)	$\mathbf{-0.815} \pm 0.004$	$\mathbf{-1.079} \pm 0.004$
AFedPG (50)	-84.957 ± 1.498	-414.923 ± 2.739
Ours+AFedPG (50)	$\mathbf{-0.782} \pm 0.004$	$\mathbf{-1.039} \pm 0.007$
AFedPG (70)	-75.832 ± 1.183	-494.368 ± 5.465
Ours+AFedPG (70)	$\mathbf{-0.824} \pm 0.005$	$\mathbf{-1.080} \pm 0.004$
FFRL (50)	-1.319 ± 0.023	-1.433 ± 0.025
Ours+FFRL (50)	$\mathbf{-0.831} \pm 0.008$	$\mathbf{-1.107} \pm 0.006$
FFRL (70)	-1.169 ± 0.021	-1.408 ± 0.021
Ours+FFRL (70)	$\mathbf{-0.835} \pm 0.004$	$\mathbf{-1.093} \pm 0.005$
AFRL (50)	-10.645 ± 0.000	-18.331 ± 0.000
Ours+AFRL (50)	$\mathbf{-0.793} \pm 0.005$	$\mathbf{-1.034} \pm 0.005$
AFRL (70)	-12.294 ± 0.000	-20.006 ± 0.000
Ours+AFRL (70)	$\mathbf{-0.810} \pm 0.004$	$\mathbf{-1.063} \pm 0.003$

analysis results of our method in terms of average returns, with the number of clients fixed at 5. As illustrated in Table 7, the four implementations achieve distinct performance due to their different ways of computing model importance, and all surpass the baseline approaches, thereby verifying the effectiveness of each implementation.

A.9 Comparison under different numbers of participants

This section evaluates the improvement of our method over baseline approaches as the number of participants in FDRL increases. The baselines include FRLLS, AFedPG, FFRL, and AFRL, and the benchmarks are RobotArm-v1 and RobotArm-v2, each comprising 20 non-stationary tasks. Table 8 presents the experimental results in terms of average returns, with client numbers set to 20 and 40. Our results indicate that our method improves the performance of baseline FDRL methods in the vast majority of cases. This is because, even with an increasing number of clients, FDRL methods still suffer from catastrophic forgetting when sequentially learning multiple tasks. In contrast, our approach employs a model buffer and a model selection strategy to mitigate catastrophic forgetting and enhance knowledge utilization efficiency, respectively. Consequently, it enables FDRL methods to achieve stronger CL capability and thus higher returns.

Table 10: Variants analysis of our method. Ours+FRLLS (I) represents integrating the first implementation of our method into FRLLS.

Methods	RobotArm-v1	RobotArm-v2
FRLLS	-6.182 ± 0.000	-15.573 ± 0.000
Ours+FRLLS (I)	-0.856 ± 0.013	-1.167 ± 0.030
Ours+FRLLS (II)	-0.783 ± 0.005	-1.011 ± 0.008
AFedPG	-7.210 ± 0.000	-13.539 ± 0.000
Ours+AFedPG (I)	-0.862 ± 0.016	-1.230 ± 0.034
Ours+AFedPG (II)	-0.770 ± 0.005	-1.041 ± 0.008
FFRL	-11.347 ± 0.000	-20.292 ± 0.000
Ours+FFRL (I)	-0.817 ± 0.008	-1.544 ± 0.089
Ours+FFRL (II)	-0.778 ± 0.005	-1.028 ± 0.007
AFRL	-1.025 ± 0.019	-1.309 ± 0.019
Ours+AFRL (I)	-0.818 ± 0.010	-1.138 ± 0.036
Ours+AFRL (II)	-0.747 ± 0.003	-1.016 ± 0.007

Table 11: Ablation analysis. Ours+FRLLS (R) denotes integrating our method with random model sampling into FRLLS.

Methods	RobotArm-v1	RobotArm-v2
FRLLS	-6.182 ± 0.000	-15.573 ± 0.000
Ours+FRLLS (R)	-1.085 ± 0.047	-1.531 ± 0.069
Ours+FRLLS	$\mathbf{-0.783} \pm 0.005$	$\mathbf{-1.011} \pm 0.008$
FFRL	-11.347 ± 0.000	-20.292 ± 0.000
Ours+FFRL (R)	-1.005 ± 0.035	-1.402 ± 0.058
Ours+FFRL	$\mathbf{-0.778} \pm 0.005$	$\mathbf{-1.028} \pm 0.007$
AFedPG	-7.210 ± 0.000	-13.539 ± 0.000
Ours+AFedPG (R)	-5.278 ± 0.495	-8.647 ± 0.687
Ours+AFedPG	$\mathbf{-0.770} \pm 0.005$	$\mathbf{-1.041} \pm 0.008$
AFRL	-1.025 ± 0.019	-1.309 ± 0.019
Ours+AFRL (R)	-2.238 ± 0.298	-4.626 ± 0.394
Ours+AFRL	$\mathbf{-0.747} \pm 0.003$	$\mathbf{-1.016} \pm 0.007$

A.10 Comparison under different numbers of tasks

This section evaluates the improvement of our method over baseline approaches as the number of tasks in the CFDRL task sequence increases. The baselines include FRLLS, AFedPG, FFRL, and AFRL, and the benchmarks are RobotArm-v1 and RobotArm-v2, with the number of tasks per benchmark increased to 50 and 70, respectively. Table 9 presents the results in terms of average returns, with the number of clients fixed at 5. Our results indicate that our method improves the performance of baseline FDRL methods regardless of the number of tasks to be learned sequentially. This improvement stems from the fact that, during sequential task learning, FDRL methods suffer from catastrophic forgetting, which leads to suboptimal policy performance. In contrast, our approach employs a model buffer and a model selection strategy to mitigate catastrophic forgetting and enhance knowledge utilization efficiency, respectively. These mechanisms jointly endow FDRL methods with stronger CL capability in sequential task settings, thereby yielding higher returns.

A.11 Variants and ablation analysis of our method

This section investigates the performance of different implementations of our method and presents ablation studies. The baseline algorithms cover FRLLS, AFedPG, FFRL, and AFRL. Evaluations are conducted on the RobotArm-v1 and RobotArm-v2 benchmarks, each configured with 20 tasks. Table 10 and Table 11 report the variant analysis and ablation results of our method in terms of average returns, with the number of clients fixed at 5. As shown in Table 10, the two implementations deliver different performance. The second implementation outperforms the first in most cases, and both outperform all baseline approaches to validate the effectiveness of our method. For Table 11, we do not create an ablation variant without the model buffer, as our method degenerates to vanilla baselines such as FRLLS once the model buffer is removed. Results from Table 11 indicate that disabling the model buffer or randomly sampling one model from the buffer for each new task cannot achieve competitive performance against the full version of our method. These observations verify the efficacy of the model buffer and the model selection strategy. The model buffer stores learned knowledge, and the selection strategy picks the best prior knowledge for solving individual tasks.