Federated Ensemble Model-based Reinforcement Learning in Edge Computing

Jin Wang, Jia Hu, Jed Mills, Geyong Min, Ming Xia, and Nektarios Georgalas

Abstract—Federated learning (FL) is a privacy-preserving distributed machine learning paradigm that enables collaborative training among geographically distributed and heterogeneous devices without gathering their data. Extending FL beyond the supervised learning models, federated reinforcement learning (FRL) was proposed to handle sequential decision-making problems in edge computing systems. However, the existing FRL algorithms directly combine model-free RL with FL, thus often leading to high sample complexity and lacking theoretical guarantees. To address the challenges, we propose a novel FRL algorithm that effectively incorporates model-based RL and ensemble knowledge distillation into FL for the first time. Specifically, we utilise FL and knowledge distillation to create an ensemble of dynamics models for clients, and then train the policy by solely using the ensemble model without interacting with the environment. Furthermore, we theoretically prove that the monotonic improvement of the proposed algorithm is guaranteed. The extensive experimental results demonstrate that our algorithm obtains much higher sample efficiency compared to classic model-free FRL algorithms in the challenging continuous control benchmark environments under edge computing settings. The results also highlight the significant impact of heterogeneous client data and local model update steps on the performance of FRL, validating the insights obtained from our theoretical analysis.

Index Terms—Edge computing, distributed machine learning, federated learning, deep reinforcement learning

1 Introduction

The advancements in deep learning (DL) [1] algorithms and high-performance computing technologies are fundamental to the tremendous successes of artificial intelligence (AI) in many aspects of our societies, including transportation, healthcare, education, etc. The emerging AI-empowered applications such as smart manufacturing, autonomous driving, and smart healthcare generate large volumes of data on the user side. To enable real-time data processing for these emerging applications, edge computing was proposed to shift computation and storage resources from the remote Cloud to the network edge in the proximity of end-users. Traditional centralized AI approaches need to collect data from end-users and save it centrally at edge servers to effectively train DL models for various applications. However, users are often unwilling to share their sensitive data with others due to the growing concern on data privacy, thus rendering these centralized approaches impractical in many cases.

To address the aforementioned issue, federated learning (FL) was proposed to collaboratively train DL models in a distributed fashion without sensitive data leaving the user devices. In FL, models are trained locally at clients (i.e., user devices) and only the model parameters are uploaded by clients to the server. The existing FL works [2]–[6] predominantly consider training supervised learning models (e.g., Convolutional Neural Networks and Long Short-Term Memory) for solving perception problems such as image classification and linguistic prediction.

More recently, federated reinforcement learning (FRL) was proposed to extend FL to train reinforcement learning (RL) models for solving sequential decision-making problems in edge computing, such as resource allocation [7], [8], content caching [9], and user access control [10]. Those studies directly combine model-free RL (learning without using a system dynamics model) with FL. Specifically, they train policies locally for all collaborating devices, using the model-free RL objective, and average the policy parameters on the server to generate a global policy for the next round of local training. However, traditional model-free RL algorithms generally have high sample complexity whilst obtaining samples is costly in many real-world edge computing scenarios such as smart factories and intelligent transport. For example, when applying RL methods to solve the task offloading problem in edge computing [8], the immediate reward for an agent can only be obtained once the offloaded task is executed. Obtaining an effective offloading policy via model-free RL may require numerous trial-and-error steps where the agent interacts with the targeted edge computing system, resulting in huge costs. Besides, the theoretical properties (such as monotonic improvement) of these model-free FRL algorithms were not well understood. These issues hinder the practical use of model-free FRL in real-world edge computing scenarios.

Compared to model-free methods, model-based RL [11]–[13] is much more sample efficient. Model-based RL learns an estimated dynamics model and then derives an optimal policy based on the learned model. Since the dynamics model is trained by using supervised learning, it can be naturally adapted to the current federated supervised learn-

- *Jin Wang, Jia Hu, Jed Mills, and Geyong Min are with the Department of Computer Science, University of Exeter, United Kingdom. E-mail: {jw855, j.hu, jm729, g.min}@exeter.ac.uk*
- *Ming Xia is with Google, California, U.S.A. E-mail: xiaming2006@gmail.com*
- *Nektarios Georgalas is with Applied Research Department, British Telecom, United Kingdom. E-mail: nektarios.georgalas@bt.com*
- *Corresponding authors: Jia Hu and Geyong Min.*

ing setting where many state-of-the-art FL algorithms are available. In addition, when applying model-based FRL in edge computing, training of the RL policy can be offline (as the training process is based on interactions with the learned dynamics model), saving the huge costs of interacting directly with the edge computing system.

Despite its promising benefits, there are several major challenges for effectively integrating model-based RL into FL. First, model bias (caused by overfitting in regions where insufficient data is available to train the model) is a key factor that affects model-based RL methods [12]. Handling RL model bias in the federated setting is even more challenging due to the highly heterogeneous client data. Second, a rigorous theoretical analysis of federated RL is lacking. Especially, monotonic improvement of RL algorithms has not been proven to hold in the federated setting. Third, it is unclear how non-independent and identically distributed (non-IID) client data will affect the performance of federated model-based RL.

In this paper, we extend model-based RL to the revolutionary FL paradigm, proposing a novel federated ensemble model-based reinforcement learning (FEMRL) algorithm. In FEMRL, the dynamics model is trained by FL, and then the RL policy is trained by solely using the dynamics model without interacting with the environment. To address the problem of model bias, we create an ensemble of dynamics models uploaded by clients. In addition, an ensemble distillation method is used to enhance the performance of model aggregation during FL. We summarise the key contributions of our work as follows:

- To the best of our knowledge, this is the first of its kind that effectively extends model-based RL to the popular FL setting. In particular, we integrate FL and knowledge distillation techniques to create an ensemble of dynamics models for clients and then train the policy by solely using the ensemble without relying on the costly process of sampling data from the environment.
- We provide a rigorous theoretical analysis to prove that the monotonic improvement of FEMRL is guaranteed. The discrepancy bound of the return from the environment and the learned dynamics identifies and highlights the impacts of non-IID client data on the policy improvement for federated RL.
- We perform extensive experiments using four challenging continuous control environments [14] under edge computing settings. The results demonstrate the superior sampling efficiency (hence lower computation and communication cost) of FEMRL compared to classic model-free FRL algorithms. The results also highlight the significant impacts of non-IID client data and local model update steps on the rate of reward improvement for federated RL, validating the insights obtained from our theoretical analysis.

The rest of the paper is organised as follows. Section 2 introduces the related work including federated learning in edge computing systems and model-based reinforcement learning. We next overview some necessary background knowledge related to FL and RL in Section 3. Section 4 presents details of the proposed FEMRL including the algorithm design and theoretical analysis. We then evaluate FEMRL with four standard RL environments and give the discussion about the experimental results in section 5. Finally, we summarise the paper in section 6.

2 Related Work

The related work focuses on extending RL algorithms to FL settings in edge computing systems, namely federated reinforcement learning. However, directly combining model-free RL with FL has low sample complexity. This work aims to improve the sample efficiency by adapting model-based RL to FL settings and further improve the training stability by utilizing federated ensemble distillation. In Table 1, we summarize the related research topics of Federated Reinforcement Learning, Model-based Reinforcement Learning, and Federated Ensemble Distillation and present the detailed review in the following paragraphs.

Federated Reinforcement Learning: Several previous studies have investigated training RL policies in the FL setting. Nadiger *et al.* [15] proposed a system for training virtual Pong players (controlled via a Deep Q-network) in the FL setting to match the skill levels of (simulated) players. The authors in [16] designed the FedRL system for training a policy, where individual FL clients do not have access to the full state-space of the RL task. Some researchers focus on domain-specific federated reinforcement learning in edge computing system. In [7], the authors combined federated reinforcement learning and blockchain to solve resource allocation problem in edge computing system, providing reliable and secure training process. Wang *et al.* [9] proposed an attention-weighted federated deep reinforcement learning model to solve the heterogeneous collaborative edge caching problem by jointly optimising the node selection and cache replacement in device-to-device assisted mobile networks. In [10], the authors proposed an intelligent user access control scheme based on FRL in radio access networks to optimise the overall throughput and avoid frequent handovers. Whilef these works contribute to the development of model-free RL in the FL setting, they suffer from high sample complexity and lack theoretical guarantees.

Model-based Reinforcement Learning: RL algorithms are generally built on Markov Decision Processes (MDP) and can be divided into two categories: model-free RL algorithms, which directly train a value function or policy by trial-and-error in the environment; and model-based RL algorithms that explicitly learn a dynamics model based on the sampled data and derive a policy from the model. Model-based RL has been demonstrated to have significantly higher sample efficiency than model-free RL, and has been successfully applied to robotics [18], video games [13], etc., using a variety of dynamics models including Gaussian processes [24], linear models [25], [26], mixtures of Gaussians [27], and Deep Neural Networks (DNNs) [28]–[30]. One key challenge for model-based RL is how to handle uncertainty of the dynamics model [12], [17]. To address this challenge, ensembles of DNNs [12], [17], [18] have been used to handle model uncertainty given data collected from the environment. In our FEMRL

TABLE 1
A summary of differences between the related work and our work.

Related research topics	References	Support federated training	Support decision-making	Sample efficiency
Federated reinforcement learning	[7], [9], [10], [15], [16]	✓	✓	low
Model-based reinforcement learning	[11], [12], [17]–[21]	×	✓	high
Federated ensemble distillation	[22], [23]	✓	×	high
Federated ensemble model-based reinforcement learning	our work	✓	✓	high

algorithm, we approximate the model dynamics using DNNs and create an ensemble using the models uploaded by FL clients. From the theoretical perspective, previous works [11], [19]–[21] have provided general frameworks for analysing model-based RL, which include monotonic improvement guarantees. We extend the analyses of these works to our FEMRL algorithm, proving the monotonic improvement of FEMRL, which also demonstrates the influence of non-IID client data on the policy improvement.

Federated Ensemble Distillation: FL aims to train a global model by sharing users' locally-trained models, rather than their private data. A crucial step in FL is how to aggregate local models into a global model. The seminal FedAvg algorithm [31] averages local models after each communication round to produce a new global model. However, directly averaging model parameters may not be the most effective method of creating the global model, due to non-IID client data, which is a significant challenge in FL and can come in many forms [32]. Some recent works focus on using ensemble distillation techniques to create more robust global models. [22] proposed a novel aggregation approach using Bayesian model ensembles and knowledge distillation. [23] proposed a similar algorithm for distillation on the server, using the average logits of the client models on an unlabelled dataset as the distillation target. Inspired by the above methods, we aggregate the client models into a single global model using knowledge distillation. Moreover, in our method, we sample fictional experience (as opposed to real experience) from the ensemble of models for knowledge distillation, further helps reduce the privacy risks of FEMRL.

3 Preliminaries

In this section, we provide some necessary background about the formulations of FL and RL problems.

3.1 Federated Learning

In FL, clients collaboratively train a model without exchanging their training data in any way. The FL objective is to find the minimiser $\boldsymbol{w}$ of the average client loss function f:

$$\min_{w\in\mathbb{R}^d} f(\boldsymbol{w}) = \frac{1}{K}\sum_{k=1}^{K} p_k f_k(\boldsymbol{w}), \tag{1}$$

where K is the total number of clients, p_k and f_k are the fraction of total samples ($\sum_k p_k = 1$) and average loss over samples on client k, respectively. Therefore, FL aims to compute the minimiser of the average loss over all samples on all participating clients (i.e., the same objective as would be achieved by centralised training on pooled data). However, in real-world FL data is non-IID across clients, as the behaviour of each client influences how its local samples are generated. Non-IID client data has been extensively shown to hinder the convergence of the FL model, and is one of the key challenges to FL. In our FEMRL algorithm, we use FL to train the dynamics model of the MDP.

3.2 Reinforcement Learning

A sequential decision-making problem solved by RL is generally modelled as an MDP, which is given by the six-tuple $\mathcal{M} := (\mathcal{S}, \mathcal{A}, T, R, \rho_0, \gamma)$. Here, $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, respectively. $T(s'|s,a)$ represents the dynamics that specifies the conditional distribution of the next state s' given the current state s and action a. $R(s,a)$ is the reward function, ρ_0 represents the initial state distribution, and $\gamma \in (0,1)$ denotes the discount-factor. Denote $\pi(\cdot|s)$ as the policy that specifies the conditional distribution over action space given a state s. The goal of RL algorithms is to find the optimal policy that maximises the expected discounted return defined by $\mathbb{E}_{\pi,T,\rho_0}\left[\sum_{t=0}^{\infty}\gamma^t R(S_t,A_t)\right]$. Define the value function following policy π with MDP $\mathcal{M} := (\mathcal{S}, \mathcal{A}, T, R, \rho_0, \gamma)$ as:

$$V_\pi^{\mathcal{M}}(s) = \mathop{\mathbb{E}}_{\substack{S_{t+1}\sim T(\cdot|S_t,A_t)\\ A_t\sim\pi(\cdot|S_t)}} \left[\sum_{t=0}^{\infty}\gamma^t R(S_t,A_t)\middle| S_0 = s\right]. \tag{2}$$

Thus $V_\pi^{\mathcal{M}} := V_\pi^{\mathcal{M}}(s_0)$ is the total return given policy π, where $s_0 \sim \rho_0$ is the initial state.

4 Federated Ensemble Model-based Reinforcement Learning (FEMRL)

In this section, we describe the proposed FEMRL algorithm in detail, and then provide a theoretical analysis guaranteeing monotonic improvement of the policy produced by FEMRL.

4.1 Algorithm Design

Our algorithm intends to train a model-based RL policy in an edge computing environment involving multiple client devices, and a corresponding edge server. In our setting, all participating clients share the same environment with different state transitions. There are many real-world applications corresponding to this setting, including unmanned aerial vehicles [33], edge caching [34], user access control [10], and resource management [7], [8] in edge computing systems. Fig. 1 illustrates the operation of FEMRL, which consists of two major sub-components: FL loop for the training of dynamics model, and RL loop for policy training.

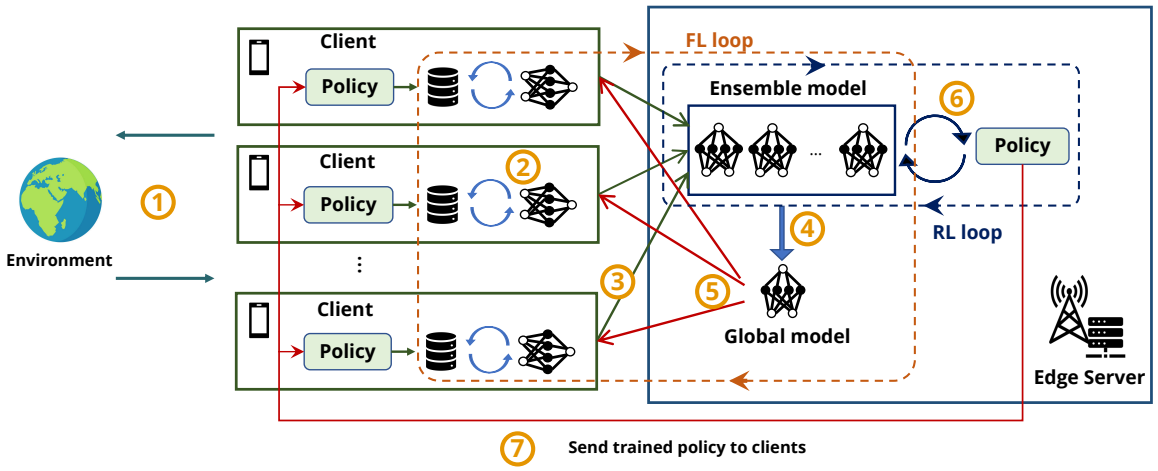


Fig. 1. Overview of the FEMRL algorithm. **Step 1:** each client samples data from the environment based on the local sample policy and stores the data locally. **Step 2:** local dynamics models are trained based on the sampled data. **Step 3:** the parameters of the local dynamics models are sent to the server. **Step 4:** an ensemble of dynamics models are created on the server using the uploaded client models, and a single global model is then created via knowledge distillation. **Step 5:** the parameters of the global model are sent to clients. Then, starting step 2 again for T_c rounds of FL loops. **Step 6:** after rounds of FL training, the server then trains the policy using a policy-gradient algorithm (e.g., TRPO) and the ensemble of dynamics models. **Step 7:** the parameters of the new policy are sent to clients for the next round of sampling (i.e., Step 1).

Formally, define the MDP with learned dynamics $\widehat{T}(s'|s,a;\boldsymbol{w})$ as $\widehat{\mathcal{M}} := (\mathcal{S},\mathcal{A},\widehat{T},R,\rho_0,\gamma)$, where $\boldsymbol{w}$ are the parameters of the learned model. Define $\widehat{T}(s,a;\boldsymbol{w})$ as the function that produces the unique value of s'. The goal of the FL loop is to learn the optimal $\boldsymbol{w}$ such that the discrepancy between the learned dynamics and real dynamics is minimal. This minimisation is a typical supervised learning process, which can be solved through maximum likelihood estimation or other techniques from generative and dynamics modelling. In this paper, we apply a multi-step prediction loss that is similar to [19] for model learning, and use a predefined reward function, as in the works [12], [17], [19]. Concretely,s for a state s_t and action sequence $a_{t:t+h}$, the h-step prediction $\hat{s}_{t+h}$ as $\hat{s}_t = s_t$, and for $h \geq 0$, $\hat{s}_{t+h+1} = \widehat{T}(\hat{s}_{t+h}, a_{t+h}; \boldsymbol{w})$, the H-step loss is defined as:

$$f(\boldsymbol{w}) = \frac{1}{H}\sum_{i=1}^{H} \|(\hat{s}_{t+i} - \hat{s}_{t+i-1}) - (s_{t+i} - s_{t+i-1})\|_2 \,. \quad (3)$$

The FL loop involves T_c rounds of communication between client devices and the edge server. Within each round of federated training, each client parallelly conducts the local update procedure as shown in Algorithm 2. The client first samples trajectories from the environment using the current policy $\pi_D \leftarrow \pi_\theta$, where π_θ is the updated policy received from the server. The client then collects all the sampled trajectories into the local replay buffer, D_k. Note that the distribution of sampling trajectories is determined by the values of the policy parameters θ and the dynamics of the environment $T(s'|s,a)$ as:

$$P(s_0,a_0,s_1,...,s_n,a_n,s_{n+1}) = \rho_0 \prod_{t=0}^{n} \pi_\theta(a_t|s_t) T(s_{t+1}|s_t,a_t). \quad (4)$$

Next, the client conducts E local update steps to train the local dynamics model with mini-batch gradient descent. The returned local dynamics model is then uploaded to the server for further processing. All uploaded models are then aggregated into a single global model on the server-side. Instead of simply averaging the local models as in FedAvg [31], we create an ensemble model $\{\boldsymbol{w}_k\}_{k=1}^m$ based on the uploaded local models, where $\boldsymbol{w}_k$ is the local model updated by the k^{th} client. This ensemble serves two purposes: 1) creating a single global dynamics model that benefits from knowledge distillation; 2) generating fictitious data for policy training. Using the model ensemble, therefore, benefits both the FL and policy training processes by producing a robust aggregate model and alleviating the model bias problem in policy training. In our proposed FEMRL, the policy is trained through interacting with the learned dynamics model rather than the actual environment. Therefore, the model error has a significant impact on the learned policy. To reduce the impact of the model error, the ensemble method provides an effective regularization for policy training: by using the ensemble dynamics model, the policy is able to perform well over many possible alternative futures, making the learned policy more robust.

The ensemble knowledge distillation method involves a typical student-teacher learning scheme. Denote the sampled fictitious data as $\mathcal{D} = \{s_0, a_0, ..., s_n, a_n\}$, $s_0 \sim \rho_0$, $a_t \sim \pi(a_t|s_t)$, $s_{t+1} = \widehat{T}(s_t, a_t; \{\boldsymbol{w}_k\}_{k=1}^m)$. The student model (i.e., the single global dynamics model) is trained with Adam [35] following the loss function:

$$L(\overline{\boldsymbol{w}}) = \left\| \frac{1}{m}\sum_{k=0}^{m} T(s_t,a_t;\boldsymbol{w}_k) - T(s_t,a_t;\overline{\boldsymbol{w}}) \right\|_2 , \quad (5)$$

where $T(s_t,a_t;\boldsymbol{w}_k)$ is the learned local dynamics of client k and $T(s_t,a_t;\overline{\boldsymbol{w}})$ is the global dynamics represented by the student model.

After T_c rounds of federated training, we then use a policy-gradient algorithm (Trust Region Policy Optimization (TRPO) [36]) to train the policy by interacting with the ensemble of models. Next, the parameters of the updated policy are sent to all participating clients, which will then start the next round of sampling procedure using the up-

dated policy. We adopt asynchronous model aggregation where the server does not wait for all clients to finish sending their updated local models. At each training round, only a fraction, α (i.e., policy synchronisation rate), of clients update their policy using the newest global policy. This design is practical since clients can be unreliable edge devices that may not always be able to reach the server (e.g., a smartphone loses its network connection) in the FL scenario. For $\alpha < 1$, clients' data distributions become non-IID, as some clients will be performing local updates on the environment model using a 'stale' (unsynchronised) policy. We present the detailed server-side algorithm of FEMRL in Algorithm 1. Specifically, we conduct training with n_{outer} epochs. Each epoch involves n_{inner} rounds of inner loops. Within each inner loop, we alternatively conduct T_{c} rounds of FL loops and G rounds of RL loops.

It is noteworthy that the model-free RL methods can also be integrated into the framework as follows. First, the server receives locally trained policy networks from clients and creates an ensemble of policy networks. Next, a single global policy network is created via knowledge distillation. Finally, the parameters of the global policy network are sent to clients, starting next-round local training. In the following sections, we provide a theoretical guarantee of monotonic policy improvement for FEMRL, before performing a thorough empirical evaluation of the algorithm.

4.2 Theoretical analysis

Proving monotonic improvement guarantee is an important aspect of RL algorithms. In this section, we provide the conditions under which FEMRL is guaranteed to provide monotonic improvement for π. To prove monotonic improvement of a model-based RL algorithm, we wish to find a lower bound of $V_{\pi}^{\mathcal{M}}$:

$$V_{\pi}^{\mathcal{M}} \geq V_{\pi}^{\widehat{\mathcal{M}}} - B \tag{6}$$

where B is the bounded value.

Since the model is trained with supervised learning, the distance between the true model and the learned model can be quantified by standard Probably Approximately Correct (PAC) generalization error [37]. PAC bounds the difference in generalisation and empirical error by a constant with high probability. In FEMRL, this generalisation error can be defined as the distance between the learned dynamics and the environment dynamics. The recent literature provides two main ways to measure this distance, each with different assumptions. One assumes that the dynamics model is a complex probability distribution, and measures the distance using Total Variation Distance (TVD) [11]. The other assumes deterministic dynamics and directly uses 1-Wasserstein distance [19]. In addition, [38] uses a general measurement, Integral Probability Metric, where TVD and 1-Wasserstein distance are two special cases. Since TVD requires weaker assumptions and is typically more practical than 1-Wasserstein distance, we use TVD in our analysis. Overall, we make the following assumptions:

Assumption 1. The generalisation error is measured by the TVD, defined as $\epsilon_m := D_{\text{TV}}(\widehat{T}(\cdot|s,a)|T(\cdot|s,a)) = \frac{1}{2}\sum_{s'}\left|\widehat{T}(s'|s,a) - T(s'|s,a)\right|$

Algorithm 1 FEMRL running on K clients (indexed by k) for E epochs, each consisting of T_{c} rounds of federated communication and G steps of policy update.

Procedure FEMRL
- **for** n_{outer} epochs **do**
  - **for** n_{inner} iterations **do**
    - $\{\boldsymbol{w}^{(k)}\}_{k=1}^{K} \leftarrow$ FedEnLearning(T_{c})
    - **for** G iterations **do**
      - Generate fictitious samples $\mathcal{D} \leftarrow$ *GenerateFictitiousData*$(\{\boldsymbol{w}_k\}_{k=1}^{K}, \pi_\theta)$.
      - Update policy π_θ using TRPO and $\mathcal{D}$
    - **end**
  - **end**
  - Send the updated policy π_θ to clients with synchronisation rate α.
- **end**

Procedure FedEnLearning (T_{c})
- Initialise parameters of the student model $\overline{\boldsymbol{w}}$
- **for** T_{c} iterations **do**
  - **for** each client $k \in K$ **in parallel do**
    - ▷ *LocalUpdate is detailed in Algorithm 2*
    - ▷ *At each local update round, the student model $\overline{\boldsymbol{w}}$ works as initial model of all participated clients.*
    - $\boldsymbol{w}_k \leftarrow$ *LocalUpdate*$(k, \overline{\boldsymbol{w}}, E)$
  - **end**
  - Create ensemble of models $\{\boldsymbol{w}_k\}_{k=1}^{K}$
  - **for** N iterations **do**
    - Generate fictitious samples $\mathcal{D} \leftarrow$ *GenerateFictitiousData*$(\{\boldsymbol{w}_k\}_{k=1}^{K}, \pi_\theta)$.
    - ▷ *The updated student model $\overline{\boldsymbol{w}}$ is then used by LocalUpdate procedure for next-round of local training.*
    - Update the student model $\overline{\boldsymbol{w}}$ using loss function from Eq. (5) on $\mathcal{D}$.
  - **end**
- **end**
- **return** $\{\boldsymbol{w}_k\}_{k=1}^{K}$

Procedure GenerateFictitiousData $(\{\boldsymbol{w}_k\}_{k=1}^{K}, \pi_\theta)$
- Sample initial state s_0 from the initial state distribution $s_0 \sim \rho_0$
- **for** $t \leftarrow 0$ **to** N **do**
  - Sample $a_t \sim \pi_\theta(a_t|s_t)$ from policy π_θ
  - Randomly sample a dynmics model $\boldsymbol{w}^{(k)}$ from the ensemlbe of models $\{\boldsymbol{w}_k\}_{k=1}^{K}$
  - Using the dynamics model $\boldsymbol{w}_k$ to predict the next state $s_{t+1} \sim \widehat{T}(s_{t+1}|s_t, a_t; \boldsymbol{w}_k)$
  - Get reward r_t by the reward function $r_t = R(s_t, a_t)$
  - Add the transition to fictitious dataset $\mathcal{D} \bigcup \{s_t, a_t, r_t, s_{t+1}\}$
- **end**
- **return** $\mathcal{D}$

Assumption 2. The dependency of two policies π and π_D is measured by the TVD $\epsilon_\pi = D_{\text{TV}}(\pi(a|s)|\pi_{\text{D}}(a|s))$, and is bounded by a constant δ_π, where $D_{\text{TV}}(\pi(a|s)|\pi_D(a|s)) \leq \delta_\pi$.

Assumption 3. The reward function of the MDP is bounded:

Algorithm 2 Procedures of client side

Procedure LocalUpdate(k, ω_k^0, E)
  Sample initial state s_0 from the initial state distribution $s_0 \sim \rho_0$.
  for $t \leftarrow 0$ ***to*** N **do**
    Sample $a_t \sim \pi_D(a_t|s_t)$ with current policy $\pi_D \leftarrow \pi_\theta$.
    Apply a_t to the environment and get the next state s_{t+1} and reward r_t.
    Store the transition to the local replay buffer $D_k \leftarrow D_k \cup (s_t, a_t, r_t, s_{t+1})$.
  end
  for $i \leftarrow 1$ ***to*** E **do**
    Random sample a batch of training data ξ_i from D_k
    Conduct mini-batch gradient descent: $\omega_k^i \leftarrow \omega_k^{i-1} - \eta_i \nabla f_k(\omega_k^{i-1}; \xi_i)$
  end
  return ω_k^E

$\forall s \in \mathcal{S}, \forall a \in \mathcal{A}, R(s,a) \leq r_{\max}$.

Assumption 4. The loss function of the FL dynamics model is convex and bounded by L, $|f(\boldsymbol{w})| \leq L$, $\forall \boldsymbol{w}$.

Based on previous works [11], [19], [38], we have the following Lemma to build the lower bound of the discrepancy of the total returns from the true model and the learned model in conventional model-based RL:

Lemma 4.1. *Denote ϵ_m as the generalization error of the dynamics model and $\epsilon_m^{\max}$ as the maximal value of ϵ_m. Denote ϵ_π as the discrepancy between target policy π and sample policy π_D. For any policy π, the return of the environment $V_\pi^{\mathcal{M}}$ and the return of the learned dynamics $V_\pi^{\widehat{\mathcal{M}}}$ are bounded as:*

$$V_\pi^{\mathcal{M}} \geq V_\pi^{\widehat{\mathcal{M}}} - \underbrace{\left[\frac{2\gamma r_{\max}}{1-\gamma}\epsilon_m + \frac{4\gamma^2 r_{\max}}{(1-\gamma)^3}\epsilon_\pi \epsilon_m^{\max}\right]}_{B}. \quad (7)$$

Proof. See Appendix A.2. □

Lemma 4.1 gives a theoretical guarantee for the monotonic improvement of the model-based RL algorithm. As long as we improve the returns under the learned model by more than B, we can guarantee improvement under the environment [11]. The bound B is proportional to the generalization error of the dynamics model, ϵ_m, and the discrepancy between the sample policy and target policy, ϵ_π. However, Lemma 4.1 holds only if the generalization error ϵ_m is bounded. Conventional model-based RL methods use normal centralised supervised learning to train the dynamics model, however, in FEMRL we use FL to train the dynamics model through an ensemble of models created from the clients' local models to approximate the learned model, $\widehat{T}(s'|s,a;\{\boldsymbol{w}^{(k)}\}_{k=1}^K)$. Therefore, it is necessary to investigate if ϵ_m is bounded in the FL setting and what factors influence ϵ_m in FEMRL.

We now derive a bound on the generalisation error of the ensemble of client models.

Theorem 4.2. *Denote the global data distribution as D. Let D_k be the local data distribution of client k. Let π_D^k be the sample policy for client k. Let $\overline{\pi}_D$ be the virtual global sample policy. Therefore, we have $D = \mathbb{P}_{s,a,s'} = \sum_{s,a} T(s'|s,a)\overline{\pi}_D(a|s)$ and $D_k = \mathbb{P}_{s,a,s'} = \sum_{s,a} T(s'|s,a)\pi_D^k(a|s)$. Denote $S_k \sim D_k^m$ as local empirical distribution for client k. Let $\hat{S}$ be the global empirical distribution, each local empirical distribution has equal contribution to the global distribution, thus $\hat{S} = \frac{1}{K}\sum_{k=1}^K S_k$. Let $\mathcal{H}$ be a hypothesis class with limited Vapnik–Chervonenkis (VC) dimension, $VCdim(\mathcal{H}) \leq d < \infty$. The hypothesis $h \in \mathcal{H}$ learned on S_k and $\hat{S}_k$ is denoted by h_{S_k} and $\hat{h}_{S_k}$, respectively. Then, the generalisation error of the ensemble model is bounded with probability at least $1-\delta$:*

$$\begin{aligned}\epsilon_m &:= \epsilon_D(\frac{1}{K}\sum_k h_{S_k}) \\ &\leq \epsilon_{\hat{S}_k}(h_{\hat{S}_k}) + C\sqrt{\frac{d + log(1/\delta)}{m}} + \frac{L}{K}\Gamma,\end{aligned} \quad (8)$$

where C and L are constants, m is the number of training samples per local data distribution, and $\Gamma = \sum_{k=1}^K D_{\mathrm{TV}}(\overline{\pi}_D||\pi_D^k)$ which is affected by the sample policies.

Proof. See Appendix B. □

Theorem 4.2 shows the generalisation error is bounded, thus the monotonic improvement (i.e., Lemma 4.1) still holds for FEMRL. There are three key factors affecting the maximal value of generalisation error ϵ_m: the virtual global empirical error $\epsilon_{\hat{S}_k}(h_{\hat{S}_k})$, the number of training samples m, and the sum of TVDs between the clients' sample policies and the virtual global sample policy, Γ.

Note that, The virtual global empirical error can in principle be estimated and optimised approximately by the training loss. $\Gamma = \sum_k D_{\mathrm{TV}}(\overline{\pi}_D||\pi_D^k) = \sum_k ||D - D_k||_1$ can be a measurement of the degree of non-IID of clients' datasets. When the data distribution is IID on all clients, $||D - D_k||_1 = 0$, $D_{\mathrm{TV}}(\overline{\pi}_D||\pi_D^k) = 0$, $\forall k$, which means all clients share the same sample policy. When the data distribution of clients becomes heterogeneous, $\Gamma > 0$. Specifically, the higher degree of non-IID of data distribution, the higher Γ is.

We now analyse the effect of policy synchronisation rate α on the measure of non-IID client data distributions, Γ. Denote the sample policy before and after the global update as π_D and π_D', respectively. After policy synchronisation (with rate α), αK clients have the latest sample policy π_D' and $(1-\alpha)K$ clients use the old sample policy π_D. Therefore, the virtual global sample policy is given as:

$$\overline{\pi}_D = \frac{1}{K}\left[\sum_{k=1}^{\alpha K}\pi_D' + \sum_{k=1}^{(1-\alpha)K}\pi_D\right] = \alpha\pi_D' + (1-\alpha)\pi_D. \quad (9)$$

Using the the definition of Γ:

$$\begin{aligned}\Gamma &:= \sum_{k=1}^K D_{\mathrm{TV}}(\overline{\pi}_D||\pi_D^k) \\ &= \sum_{k=1}^{\alpha K} D_{\mathrm{TV}}(\overline{\pi}_D||\pi_D') + \sum_{k=1}^{(1-\alpha)K} D_{\mathrm{TV}}(\overline{\pi}_D||\pi_D).\end{aligned} \quad (10)$$

Replacing $\overline{\pi}$ using Eq. (9), we have for the synchronised component:

$$\begin{aligned} D_{\mathrm{TV}}(\overline{\pi}_D||\pi'_D) &= \frac{1}{2}\sum_{s,a}|\alpha\pi'_D + (1-\alpha)\pi_D - \pi'_D| \\ &= \frac{1}{2}(1-\alpha)D_{\mathrm{TV}}(\pi_D||\pi'_D). \end{aligned} \tag{11}$$

Similarly, for the unsynchronised component:

$$\begin{aligned} D_{\mathrm{TV}}(\overline{\pi}_D||\pi_D) &= \frac{1}{2}\sum_{s,a}|\alpha\pi'_D + (1-\alpha)\pi_D - \pi_D| \\ &= \frac{1}{2}\alpha D_{\mathrm{TV}}(\pi_D||\pi'_D). \end{aligned} \tag{12}$$

Combining Eqs. (10), (11), and (12), we have

$$\Gamma = \alpha(1-\alpha)KD_{\mathrm{TV}}(\pi_D||\pi'_D). \tag{13}$$

Eq. (13), shows that Γ is influenced both by the policy discrepancy $D_{\mathrm{TV}}(\pi_D||\pi'_D)$ and the policy synchronous rate α. Γ takes the maximal value with respect to α at $\alpha = 0.5$. For this value, we would expect the convergence of FEMRL to be most hindered due to highly heterogeneous clients. In the next section, we will show how the degree of non-IID of clients' data distributions affects the rate of the reward improvement for FEMRL.

5 Experimental Evaluation

In this section, we evaluate the proposed FEMRL with model-free FRL algorithms in standard RL environments. We first give the implementation details about all the algorithms and environments. Next, we give the comparative assessment about FEMRL. Finally, we investigate the impact of non-IID client data, local update steps, and ensemble knowledge distillation.

5.1 Implementation details

We evaluate the performance of FEMRL on four realistic continuous control tasks (i.e., HalfCheetah, Ant, Hopper, and Swimmer) from the rllab framework [14] which are widely used to evaluate the RL algorithms [17], [18], [39]. For all these tasks, we set the maximal episode length to 500. One important application scenario of edge computing is in smart manufacturing where robots are widely used to improve production automation and productivity [40]. In the context of smart manufacturing, the proliferation of terminal devices (e.g., mobile robots and mechanical arms) has given rise to new challenges for the real-time operation and maintenance, scalability, and reliability. Edge computing aims to address these challenges by providing edge servers with networking, computing, and storage capabilities close to the manufacturing unit to meet key performance requirements. Therefore, in our experiments, we consider robotics environments for the local learning clients, which together with the simulated edge server can reflect a typical edge computing scenario in smart manufacturing. We assume all environments running on an edge computing platform which includes multiple user devices and an edge server. All user devices share the same environment dynamics as we discussed in section 4.

We implement FEMRL and all baseline algorithms by using Pytorch ($\geq$1.7.0). Specially, the dyanmics of the MDP is approximated by the feed-forward neural network with two hidden layers and each layer includes 500 units. The activation function at each layer is ReLU. Instead of directly predicting the next state, the network predicts the normalised differences between the next state s_{t+1} and s_t as in previous works [12], [19]. Each client maintains its own normalised statistics (i.e., the mean μ, and standard deviation σ) based on the sampled local dataset. The normalised difference can be calculated as $((s_{t+1} - s_t) - \mu)/\sigma$. The policy neural network is also implemented by a feed-forward neural network with two hidden layers, each of which has 128 hidden units. We use ReLU as the activation function and the output of the policy neural network is a Gaussian distribution $\mathcal{N}(\mu(s), \sigma^2)$ where σ is a state-independent trainable vector.

For other default settings of FEMRL, we set the number of inner loops as $n_{\mathrm{inner}} = 20$ at each training epoch. Each client conducts 500 environment steps using the sample policy and stores the sampled data locally. We set the batch size of local updates for the dynamics model as 128 for all clients. Each client conducts $E = 80$ steps of local training with Adam (with learning rate 10^{-3}) and then uploads parameters of the local dynamics model to the server. The server then aggregates the uploaded models into a single global model through knowledge distillation. Specifically, we use the sample policy to sample trajectories based on the ensemble of client models and then apply the student-teacher scheme to train a single global model on the fictional trajectories. The learning rate and batch size of the knowledge distillation are set as 10^{-3} and 128, respectively. At each epoch, we optimise the dynamics model and policy alternatively for $n_{\mathrm{inner}} = 20$ times. At each inner loop, we conduct $T_{\mathrm{c}} = 5$ communication rounds between clients and server for training the dynamic models. After the training for the dynamics model, we then use a policy gradient algorithm (TRPO) to train the policy. We set the number of iterations for policy training as $G = 20$.

In our framework, edge devices run two computation tasks: sampling data from the environment and training the local dynamic models. The sampling process is conducted by forward propagation of the policy network with the linear time complexity $O(n)$. Here n is the size of the sampled data and is usually small for an edge device. In addition, the process of training the local dynamic models is the same as most of federated learning algorithms that have the linear computation complexity $O(m)$, where m is the number of local training samples. Therefore, the computation overhead of our method is acceptable for edge devices.

For the settings of the model-free baseline algorithms, we use two advanced policy gradient methods: PPO and TRPO. Both PPO and TRPO use General Advantage Estimator (GAE) [41] to measure advantages. The policy networks of all baseline algorithms share the same settings as FEMRL. The hyperparameters settings for centralised TRPO and PPO are listed in Tables 2 and 3, respectively. Fed-TRPO and Fed-PPO share most of the hyperparameters settings as their centralised counterparts except batch size (TRPO) and environment steps per epoch (PPO). Since Fed-TRPO and Fed-PPO do not collect data from clients, thus the batch

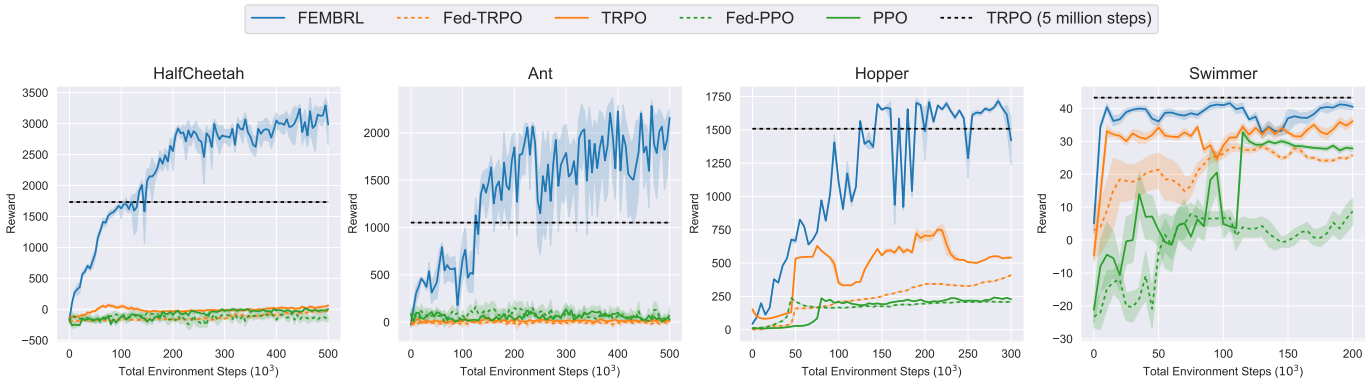


Fig. 2. Global total reward during training for FEMRL (blue) and the four baselines on continuous control benchmarks. Solid curves show the average over 10 trials, and shaded regions show the standard deviation of the mean. The dotted horizontal lines give the final total reward of TRPO after 5 million environment steps.

size of Fed-TRPO on each client is set as 500 while the environment steps per epoch of Fed-PPO on each client is set as 500.

TABLE 2
Hyperparameters for TRPO.

Hyperparameter	Value	Hyperparameter	Value
Batch Size	5000	Max KL Divergence	0.01
Discount γ	0.99	GAE λ	0.95
Conj. Gradient Damping	0.1	Conj. Gradient Steps	10

TABLE 3
Hyperparameters for PPO.

Hyperparameter	Value	Hyperparameter	Value
Batch Size	100	Env. Steps per Epoch	5000
Learning Rate	0.001	Optimizer	Adam
GAE λ	0.95	Discount γ	0.99
Ent. Coefficient	0.01	Clipping Value ϵ	0.2

5.2 Comparative assessment

We first compare the sampling efficiency of FEMRL to 4 other algorithms: 1) TRPO [36], a model-free policy-gradient based algorithm running centrally, where all client samples are collected on the server (thus breaking the FL assumption). The policy is updated using the gathered samples. 2) Proximal Policy Optimisation (PPO) [42], a model-free RL algorithm also running centrally. 3) Federated TRPO (Fed-TRPO), where each client collects samples from the environment and updates the local policy based on the collected samples. After the local update of the policies on the clients, the server averages all uploaded client policies, creating a global policy for the next round of training. 4) Federated PPO (Fed-PPO), again applying PPO to the FL setting. Both Fed-TRPO and Fed-PPO are model-free FRL methods. The existing federated RL methods, e.g., [9], [16], [43], share the same FL architecture as Fed-TRPO and Fed-PPO, but differ in the model-free RL algorithm used.

In FEMRL, after the policy update on the server, the parameters of the policy network are sent to clients to update their local policies. However, the update of local policies at clients can be asynchronous: some clients receive the updated policy, others do not receive it and thus will use the old policy for sampling. As a consequence, clients will have heterogeneous sampling policies. We denote the policy synchronous rate as α, where only αK clients will receive the updated sample policy at each training epoch. As the default setting of FEMRL, we set $\alpha = 0.3$, the number of local update steps of FL $E = 80$, and the number of FL communication rounds $T_c = 5$. We use $K = 10$ clients for all algorithms. Each client performs 500 environment steps at each epoch, which therefore has 5000 total environment steps. FEMRL first trains the dynamics model based on the sampled data, and then uses this model to generate fictitious data for policy updating. In contrast, the model-free algorithms (i.e., TRPO, PPO, Fed-TRPO, Fed-PPO) directly use the sampled data for policy update. Due to the sparse reward signal of RL, they generally require huge numbers of interactions with the environment to obtain effective policies, leading to sample inefficiency.

Fig. 2 shows the policy improvement rate of FEMRL and the four baseline algorithms. The learning parameters of all the algorithms in Fig. 2 use the default settings as given in the previous paragraph and Section 5.1. The dotted lines demonstrate the final performance of (centralised) TRPO after 5 million environment steps. The performance of (centralised) TRPO or PPO acts as a soft upper bound of the federated counterpart (i.e., Fed-TRPO, or Fed-PPO). FEMRL learns substantially faster and achieves the best performance with 0.5 million or fewer environment steps. For example, FEMRL achieves the same performance at 120k environment steps as TRPO does after 5 million environment steps in the HalfCheetah and Ant environments. FEMRL is an FL variant of the model-based RL [19], where we train the environment dynamics model via FL and optimize the policy by interacting with the learned dynamics model directly. Therefore, FEMRL can achieve better sample efficiency than model-free RL methods and their FL variants.

5.3 The impact of non-IID client data

Clients with non-IID datasets possess unique, non-identical minimisers to their local objectives. During the local-update

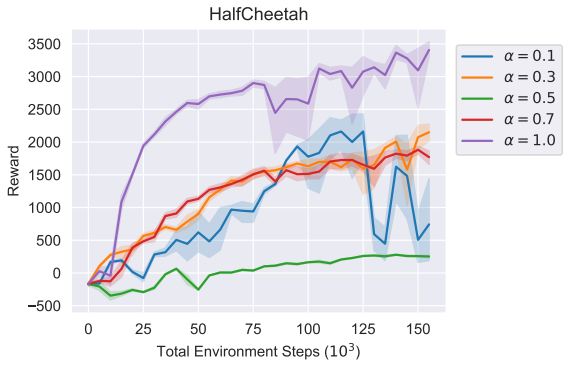


Fig. 3. Performance of FEMRL with different policy synchronous rates (i.e., α) on HalfCheetah.

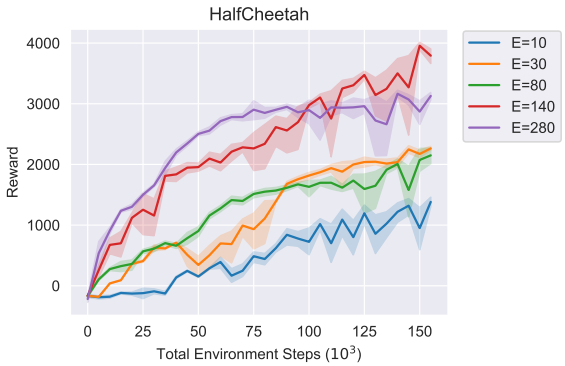


Fig. 5. Performance of FEMRL with different numbers of local update steps (i.e., E) on HalfCheetah.

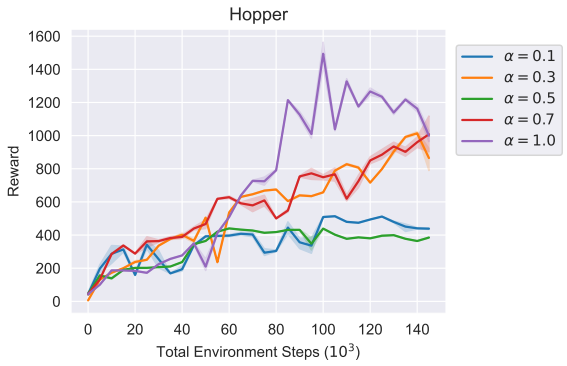


Fig. 4. Performance of FEMRL with different policy synchronous rates (i.e., α) on Hopper.

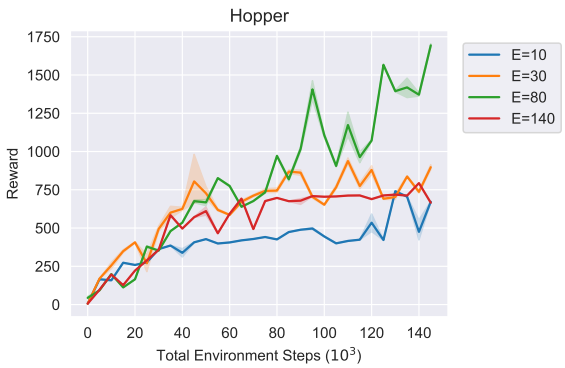


Fig. 6. Performance of FEMRL with different numbers of local update steps (i.e., E) on Hopper.

phase of FL, each participating client's model will diverge from the global model and move towards their local minimiser. This divergence is termed 'client-drift' [44] and has been extensively shown to harm the performance of the global model. The greater the degree of non-IID client data, and the more local steps clients perform, the greater the level of client-drift. In this section, we investigate how non-IID client data impacts the performance of FEMRL.

Eq. (13) shows that the degree of non-IID is determined by α for FEMRL. Therefore, we evaluate FEMRL with varying α on HalfCheetah and Fig. 3 shows the training curves. When $\alpha = 0.5$, the rate of policy improvement is slowest due to the highly non-IID client data: higher model error results in a worse policy. The rate of policy improvement naturally is the fastest when $\alpha = 1.0$, as Γ is 0 (according to Eq. (13)) that represents an IID scenario. When $\alpha = 0.1$, although Γ is small, performance is still low because the discrepancy (i.e., ϵ_π) between the sample policy and target policy is large. Lemma 4.1 reveals the relationship between ϵ_π and the returns of the dynamics model and the environment. The $\alpha \in \{0.3, 0.7\}$ curves show that the policy improvement rate of FEMRL falls gracefully as $\alpha \to 0.5$.

Fig. 4 shows the performance of FEMRL on Hopper with varying policy synchronisation rates. As expected, when $\alpha = 1.0$, the client data is purely IID, therefore FEMRL can achieve the best performance. In contrast, when $\alpha = 0.5$, the degree of non-IID is maximal, therefore, FEMRL obtains the worst performance.

5.4 The impact of local update steps

Previous works have shown that the number of local steps of SGD that clients perform, E, is a key factor affecting the convergence of FL algorithms [31], [32], [44]. Larger E allows clients to do more work locally and make more progress, but the final performance of the global model is harmed when the data on clients is non-IID.

Fig. 5 shows the convergence of FEMRL with a varying number of local steps E, for a fixed number of communication rounds $T_c = 5$. As expected, as E increases, the initial rate of policy improvement increases as clients make more progress in training the dynamics model. However, as E becomes very large ($E = 280$), the final reward plateaus at 3000, as the environment model reaches a local optimum and the mininum error it can achieve is harmed. In this scenario, the value of $E = 140$ strikes a good trade-off between policy improvement rate and maximum reward.

Fig. 6 shows the performance of FERML on varying numbers of local update steps, for a fixed number of communication rounds $T_c = 5$. As expected, both small ($E = 10, 30$) and large ($E = 140$) number of local update steps can harm the convergence rate. The value of $E = 80$ achieves the best performance in this scenario.

5.5 The impact of ensemble knowledge distillation

We then investigate how the ensemble knowledge distillation method affects the performance. We train FEMRL on

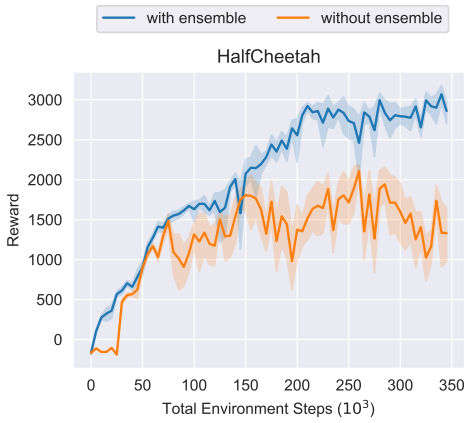


Fig. 7. Performance of FEMRL with or without ensemble knowledge distillation on HalfCheetah.

HalfCheetah without using ensemble knowledge distillation. Specifically, we directly average the uploaded parameters of clients' models and create a single global dynamics model for the FL training process. After $T = 5$ rounds of FL training, we use the global dynamics model for the policy-updating process using TRPO. The rest of the hyperparameter settings are the same as the default settings. Fig. 7 shows the results of FEMRL on HalfCheetah with or without ensemble knowledge distillation. We find that the ensemble model distillation method can significantly improve the performance of FEMRL, indicating the importance and necessity of combining ensemble knowledge distillation into our method.

5.6 Discussion

The proposed FEMRL is a general federated RL method that is not limited to a specific problem. We can adapt FEMRL to other edge computing scenarios by modifying the structure of the policy network and dynamics model to fit the dimension of the state and action space of the specific edge computing application. For example, task offloading is a typical edge computing application, which enables to offload computation-intensive tasks of mobile applications from user devices to edge servers. However, unlike the continual action space of robotics control tasks defined in our experiments, the action space of the task offloading problem is generally discrete [45]. To adapt FEMRL to the task offloading problem, we need to redesign the input/output layers of the policy network and dynamics model to fit the discrete state and action space defined in the task offloading problem, especially replacing the output layer of the policy network from a Gaussian distribution to a Categorical distribution. While the training process of FEMRL remains the same.

Although FEMRL has many benefits to MEC applications, there are several challenges requiring further exploration. In particular, the performance of the trained RL policy might deteriorate when handling fast-changing environments. In fact, how to enhance the generalisation ability of DRL methods for fast-changing environments is still an open problem in RL [46]. We feel that a useful solution for generalisation objectives would constitute a whole new paper, so we leave this to future work: we intend to combine meta-learning [47], [48] into our framework to solve the out-of-distribution problem for enhancing its generalisation ability.

6 Conclusion

In this paper, we proposed a novel federated RL algorithm, FEMRL, for edge computing systems, which incorporates model-based RL, and ensemble distillation technologies into FL. In FEMRL, clients train their local dynamics model based on their locally sampled data. An ensemble of the dynamics models is then created at the edge server based on the updated local models. We use the ensemble model for both policy training and FL model aggregation (by an ensemble distillation method). The updated policy is then sent to clients for the next-round of sampling process. We provide a rigorous theoretical analysis to prove the monotonic improvement of FEMRL in federated setting with non-IID client data. Finally, we evaluate FEMRL based on four challenging continuous control tasks. Experiment results demonstrate that FEMRL can achieve much higher sample efficiency than federated model-free counterparts.

7 Acknowledgement

This work was supported in part by EU Horizon 2020 INITIATE Project under Grant 101008297, in part by Royal Society International Exchanges Project under Grant IEC/NSFC/ 211460, in part by EPSRC New Horizons fund EP/X019160/1, and in part by UKRI Project EP/X038866/1.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] S. J. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," in *Proc. International Conference on Learning Representations (ICLR)*, 2021.

[3] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," in *Proc. International Conference on Learning Representations (ICLR)*, 2020.

[4] P. Kairouz, H. B. McMahan *et al.*, "Advances and open problems in federated learning," *Foundations and Trends in Machine Learning*, vol. 14, no. 1-2, pp. 1–210, 2021.

[5] W. Liu, L. Chen, Y. Chen, and W. Zhang, "Accelerating federated learning via momentum gradient descent," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 8, pp. 1754–1766, 2020.

[6] M. Duan, D. Liu, X. Chen, R. Liu, Y. Tan, and L. Liang, "Self-balancing federated learning with global imbalanced data in mobile systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, pp. 59–71, 2020.

[7] L. Cui, Z. Chen, S. Yang, R. Chen, and Z. Ming, "A secure and decentralized dlaas platform for edge resource scheduling against adversarial attacks," *IEEE Transactions on Computers*, 2021.

[8] S. Yu, X. Chen, Z. Zhou, X. Gong, and D. Wu, "When deep reinforcement learning meets federated learning: Intelligent multitimescale resource management for multiaccess edge computing in 5g ultradense network," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2238–2251, 2020.

[9] X. Wang, R. Li, C. Wang, X. Li, T. Taleb, and V. C. M. Leung, "Attention-weighted federated deep reinforcement learning for device-to-device assisted heterogeneous collaborative edge caching," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 154–169, Jan 2021.

[10] Y. Cao, S.-Y. Lien, Y.-C. Liang, K.-C. Chen, and X. Shen, "User access control in open radio access networks: A federated deep reinforcement learning approach," *IEEE Transactions on Wireless Communications*, 2021.

[11] M. Janner, J. Fu, M. Zhang, and S. Levine, "When to trust your model: Model-based policy optimization," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019.

[12] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel, "Model-ensemble trust-region policy optimization," in *Proc. International Conference on Learning Representations (ICLR)*, 2018.

[13] L. Kaiser, M. Babaeizadeh, P. Milos *et al.*, "Model based reinforcement learning for atari," in *Proc. International Conference on Learning Representations (ICLR)*, 2020.

[14] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *Proc. International Conference on Machine Learning (ICML)*, vol. 48, 2016, pp. 1329–1338.

[15] C. Nadiger, A. Kumar, and S. Abdelhak, "Federated reinforcement learning for fast personalization," in *IEEE International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, 2019, pp. 123–127.

[16] H. H. Zhuo, W. Feng, Q. Xu, Q. Yang, and Y. Lin, "Federated reinforcement learning," *arXiv preprint arXiv:1901.08277*, 2019.

[17] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018.

[18] Y. Zhang, I. Clavera, B. Tsai, and P. Abbeel, "Asynchronous methods for model-based reinforcement learning," in *Proc. Conference on Robot Learning (CoRL)*, vol. 100, 2020, pp. 1338–1347.

[19] Y. Luo, H. Xu, Y. Li, Y. Tian, T. Darrell, and T. Ma, "Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees," in *Proc. International Conference on Learning Representations (ICLR)*, 2019.

[20] W. Sun, G. J. Gordon, B. Boots, and J. A. Bagnell, "Dual policy iteration," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018.

[21] R. Kidambi, A. Rajeswaran, P. Netrapalli, and T. Joachims, "Morel: Model-based offline reinforcement learning," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020.

[22] H.-Y. Chen and W.-L. Chao, "Fedbe: Making bayesian model ensemble applicable to federated learning," in *Proc. International Conference on Learning Representations (ICLR)*, 2021.

[23] T. Lin, L. Kong, S. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020.

[24] M. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *Proc. International Conference on Machine Learning (ICML)*, 2011, pp. 465–472.

[25] S. Levine and P. Abbeel, "Learning neural network policies with guided policy search under unknown dynamics." in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, vol. 27, 2014.

[26] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *Proc. IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 4906–4913.

[27] S. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with gaussian mixture models," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 943–957, 2011.

[28] S. Depeweg, J. M. Hernández-Lobato, F. Doshi-Velez, and S. Udluft, "Learning and policy search in stochastic dynamical systems with bayesian neural networks," in *Proc. International Conference on Learning Representations (ICLR)*, 2017.

[29] A. Draeger, S. Engell, and H. Ranke, "Model predictive control using neural networks," *IEEE Control Systems Magazine*, vol. 15, no. 5, pp. 61–66, 1995.

[30] A. Nagabandi, G. Kahn, R. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 7559–7566.

[31] B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," *Proc. International Conference on Artifical Intelligence and Statistics (AISTATS)*, 2017.

[32] K. Hsieh, A. Phanishayee, O. Mutlu, and P. Gibbons, "The non-IID data quagmire of decentralized machine learning," in *Proc. International Conference on Machine Learning (ICML)*, vol. 119, Jul 2020, pp. 4387–4398.

[33] N. I. Mowla, N. H. Tran, I. Doh, and K. Chae, "Afrl: Adaptive federated reinforcement learning for intelligent jamming defense in fanet," *Journal of Communications and Networks*, vol. 22, no. 3, pp. 244–258, 2020.

[34] X. Wang, C. Wang, X. Li, V. Leung, and T. Taleb, "Federated deep reinforcement learning for internet of things with decentralized cooperative edge caching," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9441–9455, 2020.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. International Conference on Learning Representations (ICLR)*, 2015.

[36] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. International Conference on Machine Learning (ICML)*, vol. 37, 2015, pp. 1889–1897.

[37] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge University Press, 2014.

[38] T. Yu, G. Thomas, L. Yu, S. Ermon, J. Y. Zou, S. Levine, C. Finn, and T. Ma, "Mopo: Model-based offline policy optimization," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020.

[39] Y. Meng, S. Kuppannagari, R. Kannan, and V. Prasanna, "Ppoaccel: A high-throughput acceleration framework for proximal policy optimization," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 9, pp. 2066–2078, 2021.

[40] B. Chen, J. Wan, A. Celesti, D. Li, H. Abbas, and Q. Zhang, "Edge computing in iot-based manufacturing," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 103–109, 2018.

[41] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[42] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[43] B. Liu, L. Wang, and M. Liu, "Lifelong federated reinforcement learning: a learning architecture for navigation in cloud robotic systems," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4555–4562, 2019.

[44] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *Proc. International Conference on Machine Learning (ICML)*, vol. 119, 2020, pp. 5132–5143.

[45] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE communications surveys & tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[46] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, and T. Hester, "Challenges of real-world reinforcement learning: definitions, benchmarks and analysis," *Machine Learning*, vol. 110, no. 9, pp. 2419–2468, 2021.

[47] T. Jeong and H. Kim, "Ood-maml: Meta-learning for few-shot out-of-distribution detection and classification," *Advances in Neural Information Processing Systems (NIPS)*, vol. 33, pp. 3907–3916, 2020.

[48] A. Nagabandi, I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn, "Learning to adapt in dynamic, real-world environments through meta-reinforcement learning," in *Proc. International Conference on Learning Representations (ICLR)*, 2019.

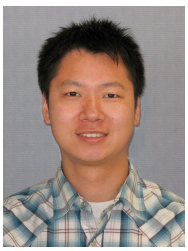
[49] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of machine learning*. MIT press, 2018.

Jin Wang received the BEng and MEng degrees in computer science from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2014 and 2017, respectively. He is currently working toward the PhD degree in computer science at the University of Exeter. His research interests include deep reinforcement learning, applied machine learning, cloud and edge computing, and computer system optimization.

Jia Hu received the BEng and MEng degrees in electronic engineering from the Huazhong University of Science and Technology, China, in 2006 and 2004, respectively, and the PhD degree in computer science from the University of Bradford, UK, in 2010. He is a senior lecturer of computer science at the University of Exeter. His research interests include edge-cloud computing, resource optimization, applied machine learning, and network security.

Jed Mills is a Computer Science Ph.D. student in the Department of Computer Science at the University of Exeter, UK. He received a B.Sc. in Natural Science from the University of Exeter in 2018. His research interests include machine learning, federated learning and mobile edge computing.

Geyong Min received the BSc degree in computer science from the Huazhong University of Science and Technology, China, in 1995, and the PhD degree in computing science from the University of Glasgow, United Kingdom, in 2003. He is a professor of high performance computing and networking with the Department of Computer Science within the College of Engineering, Mathematics and Physical Sciences at the University of Exeter, United Kingdom. His research interests include computer networks, wireless communications, parallel and distributed computing, ubiquitous computing, multimedia systems, modeling and performance engineering.



Ming Xia received the Ph.D. degree in computer science from the University of California, Davis, in 2009. He is currently a Staff Software Engineer at Google, California, U.S.A. Before joining Google, he held several positions at Alibaba, Microsoft, Ericsson Research, and the National Institute of Information and Communications Technology (NICT), Tokyo, Japan. He serves as associate editor for the Springer Journal of Telecommunication Systems, Photonic Network Communications, and guest editor for Journal of Computers & Electrical Engineering S.I. on Ubiquitous Computing and Communications. He co-organized several IEEE conferences and symposiums including ANTS, Globecom workshop on SDN and optics, ICNC optical and grid computing, etc. His research interests include cloud infrastructure, data centers health, and networking.

Nektarios Georgalas is a Principal Researcher in the Applied Research department of British Telecom. In his current role, he leads two collaborative research programmes with key BT partners delivering innovations in the areas of Cloud, Data Centres, Network Virtualisation, Smart Cities, IoT and Mobility. During his career with BT, since 1998, he has managed numerous collaborative and internal research projects in areas such as network management, market-driven data management systems, policy-based management, distributed information systems, SOA/Web Services, Model Driven Design and Development of telecoms OSS, Cloud and NFV. Nektarios is inventor and co-inventor of 11 patents. He has also authored more than 60 papers in international journals and conferences. He has served as General Co-chair, Programme Co-Chair, Programme Committee and Keynote Speaker or Invited Panelist in top international IEEE academic and TMForum conferences.

APPENDIX A
MONOTONIC IMPROVEMENT GUARANTEE

In this section, we first provide some useful Lemmas for the theoretical analysis of monotonic improvement guarantee for FEMRL and then give the proof of Lemma 4.1.

A.1 Lemmas

Lemma A.1. *(Importance sampling inequality) For any distribution $\rho(s)$ and $\rho'(s)$ and a function $f(s)$, we have $\mathbb{E}_{s\sim\rho(s)}f(s) \leq \mathbb{E}_{s\sim\rho'(s)}f(s) + |\rho(s)-\rho'(s)|f_{\max}$, where $f_{\max}$ is the maximal value of $f(s)$.*

Proof.

$$\begin{aligned}
\mathbb{E}_{s\sim\rho(s)}f(s) &= \mathbb{E}_{s\sim\rho'(s)}\frac{\rho(s)}{\rho'(s)}f(s)\\
&= \mathbb{E}_{s\sim\rho'(s)}\frac{\rho(s)-\rho'(s)+\rho'(s)}{\rho'(s)}f(s)\\
&= \mathbb{E}_{s\sim\rho'(s)}f(s) + \mathbb{E}_{s\sim\rho'(s)}(\rho(s)-\rho'(s))f(s)\\
&\leq \mathbb{E}_{s\sim\rho'(s)}f(s) + \sum_s |\rho(s)-\rho'(s)|f_{\max}\\
&\leq \mathbb{E}_{s\sim\rho'(s)}f(s) + ||\rho(s)-\rho'(s)||_1 f_{\max}.
\end{aligned}$$

□

Lemma A.2. *(Bounded difference of discounted state distributions). Let π and π_D be two different policies and $\epsilon_\pi = D_{\mathrm{TV}}(\pi||\pi_D)$, we have:*

$$\|\rho_\pi^{\mathcal{M}} - \rho_{\pi_D}^{\mathcal{M}}\|_1 \leq \frac{2\gamma}{(1-\gamma)^2}\epsilon_\pi.$$

Proof. Define $\mathbb{P}_\pi^{\mathcal{M}}$ and $\mathbb{P}_{\pi_D}^{\mathcal{M}}$ as the transition kernels of the MDP $\mathcal{M}$ following policies π and π_D, respectively. Let $\mathbf{G} = (\mathbf{I} + \gamma\mathbb{P}_\pi^{\mathcal{M}} + (\gamma\mathbb{P}_\pi^{\mathcal{M}})^2 + ...) = (\mathbf{I} - \gamma\mathbb{P}_\pi^{\mathcal{M}})^{-1}$ and $\mathbf{G_D} = (\mathbf{I} + \gamma\mathbb{P}_{\pi_D}^{\mathcal{M}} + (\gamma\mathbb{P}_{\pi_D}^{\mathcal{M}})^2 + ...) = (\mathbf{I} - \gamma\mathbb{P}_{\pi_D}^{\mathcal{M}})^{-1}$. Let $\mathbf{\Delta} = \mathbb{P}_{\pi_D}^{\mathcal{M}} - \mathbb{P}_\pi^{\mathcal{M}}$. We start with some algebraic manipulations as:

$$\mathbf{G}^{-1} - \mathbf{G_D}^{-1} = (\mathbf{I} - \gamma\mathbb{P}_\pi^{\mathcal{M}}) - (\mathbf{I} - \gamma\mathbb{P}_{\pi_D}^{\mathcal{M}}) = \gamma\mathbf{\Delta}.$$

Left-multiplying by $\mathbf{G}$ and right-multiplying by $\mathbf{G_D}$, then multiplying by ρ_0:

$$\mathbf{G_D}\rho_0 - \mathbf{G}\rho_0 = \gamma\mathbf{G\Delta G_D}\rho_0.$$

Note that $\rho_\pi^{\mathcal{M}} = \mathbf{G}\rho_0$. By definition we have $\|\mathbf{G}\|_1 = (1-\gamma)^{-1}$, $\|\mathbf{\Delta}\|_1 = 2D_{\mathrm{TV}}(\pi||\pi_D)$, and $\|\rho_0\| = 1$. Hence:

$$\begin{aligned}
\|\rho_\pi^{\mathcal{M}} - \rho_{\pi_D}^{\mathcal{M}}\|_1 &= \|\gamma\mathbf{G\Delta G_D}\rho_0\|_1 \leq \gamma\|\mathbf{G}\|_1\|\mathbf{\Delta}\|_1\|\mathbf{G_D}\|_1\|\rho_0\|_1\\
&\leq \frac{2\gamma}{(1-\gamma)^2}D_{\mathrm{TV}}(\pi||\pi_D) = \frac{2\gamma}{(1-\gamma)^2}\epsilon_\pi.
\end{aligned}$$

□

A.2 Proof of Lemma 4.1

Proof. Let $\rho_\pi^{\mathcal{M}}$ be the discounted visitation frequencies [36] over the state space as $\rho_\pi^{\mathcal{M}}(s) = \sum_{t=0}^{\infty}\gamma^t P(S_t = s|\pi, \mathcal{M})$, where $P(S_t = s|\pi, \mathcal{M})$ denotes the probability of being in state s at time step t in the MDP $\mathcal{M} := (\mathcal{S}, \mathcal{A}, T, R, \rho_0, \gamma)$ following the policy π. We can define the expected discounted return as:

$$\begin{aligned}
V_\pi^{\mathcal{M}} &= \mathop{\mathbb{E}}_{\substack{S_{t+1}\sim T(\cdot|S_t,A_t)\\ A_t\sim\pi(\cdot|S_t)}}\left[\sum_{t=0}^{\infty}\gamma^t R(S_t,A_t)\Big|S_0 = s_0\right] \\
&= \mathbb{E}_{s\sim\rho_\pi^{\mathcal{M}}(s),a\sim\pi(a|s)}\left[R(s,a)\right],
\end{aligned} \tag{14}$$

where s_0 is the initial state.

Let W_j be the discounted total reward when executing π on the dynamics model $\mathcal{M}$ for the first j steps and the rest of the steps on $\widehat{\mathcal{M}}$. That is:

$$W_j = \mathop{\mathbb{E}}_{\substack{\forall t\geq 0, A_t\sim\pi(\cdot|S_t)\\ \forall j>t\geq 0, S_{t+1}\sim T(\cdot|S_t,A_t)\\ \forall t\geq j, S_{t+1}\sim\widehat{T}(\cdot|S_t,A_t)}}\left[\sum_{t=0}^{\infty}\gamma^t R(S_t,A_t)|S_0 = s_0\right].$$

Note that we define $V_\pi^{\mathcal{M}} = \mathbb{E}_{s_0\sim\rho_0}\left[V_\pi^{\mathcal{M}}(s_0)\right]$. By definition we have $W_\infty = V_\pi^{\mathcal{M}}$ and $W_0 = V_\pi^{\widehat{\mathcal{M}}}$, thus:

$$V_\pi^{\widehat{\mathcal{M}}} - V_\pi^{\mathcal{M}} = \sum_{j=0}^{\infty}\left(W_{j+1} - W_j\right).$$

We rewrite W_j and W_{j+1} as:

$$W_j = R_j + \mathop{\mathbb{E}}_{A_j,S_j\sim\pi,T}\left[\mathop{\mathbb{E}}_{\hat{S}_{j+1}\sim\widehat{T}(\cdot|S_j,A_j)}\left[\gamma^{j+1}V_\pi^{\widehat{\mathcal{M}}}(\hat{S}_{j+1})\right]\right],$$

$$W_{j+1} = R_j + \mathop{\mathbb{E}}_{A_j,S_j\sim\pi,T}\left[\mathop{\mathbb{E}}_{S_{j+1}\sim T(\cdot|S_j,A_j)}\left[\gamma^{j+1}V_\pi^{\mathcal{M}}(S_{j+1})\right]\right].$$

we define $G_{\widehat{\mathcal{M}}}^\pi(S,A) = \mathop{\mathbb{E}}_{S'\sim T(\cdot|S,A)}\left[V_\pi^{\mathcal{M}}(S')\right] - \mathop{\mathbb{E}}_{\hat{S}'\sim\widehat{T}(\cdot|S,A)}\left[V_\pi^{\widehat{\mathcal{M}}}(\hat{S}')\right]$. Therefore:

$$W_{j+1} - W_j = \gamma^{j+1}\mathop{\mathbb{E}}_{A_j,S_j\sim\pi,T}\left[G_{\widehat{\mathcal{M}}}^\pi(S,A)\right],$$

where R_j is the expected accumulative reward of the first j steps, which are taken w.r.t. dynamics model $\mathcal{M}$. Thus we have:

$$\begin{aligned}
V_\pi^{\widehat{\mathcal{M}}} - V_\pi^{\mathcal{M}} &= \sum_{j=0}^{\infty}\left(W_{j+1} - W_j\right)\\
&= \sum_{j=0}^{\infty}\gamma^{j+1}\mathop{\mathbb{E}}_{A_j,S_j\sim\pi,T}\left[G_{\widehat{\mathcal{M}}}^\pi(S,A)\right]\\
&= \gamma\mathop{\mathbb{E}}_{\substack{S\sim\rho_\pi^{\mathcal{M}},\\ A\sim\pi(\cdot|S)}}\left[G_{\widehat{\mathcal{M}}}^\pi(S,A)\right],
\end{aligned}$$

where the last equality is from applying Eq. (14). For simplicity, we define $T(S,A) = T(s'|s,a)$ as the dynamics of the environment and $\widehat{T}(S,A) = \widehat{T}(s'|s,a)$ as the dynamics of the learned model. The reward function is bounded by $r_{\max}$ according to Assumption 3, we then have for any value function: $||V_\pi|| \leq \frac{1}{1-\gamma}r_{\max}$. Next, we bound $G_{\widehat{\mathcal{M}}}^\pi(S,A)$ as:

$$\begin{aligned}
G_{\widehat{\mathcal{M}}}^\pi(S,A) &= \sum_{S'}T(S,A)V_\pi^{\mathcal{M}}(S') - \sum_{S'}\widehat{T}(S,A)V_\pi^{\widehat{\mathcal{M}}}(S')\\
&\leq \frac{r_{\max}}{1-\gamma}\sum_{S'}\left[T(S,A) - \widehat{T}(S,A)\right]\\
&\leq \frac{2r_{\max}}{1-\gamma}D_{\mathrm{TV}}(T(S,A)\|\widehat{T}(S,A)).
\end{aligned}$$

Therefore:

$$V_\pi^{\widehat{\mathcal{M}}} - V_\pi^{\mathcal{M}} \leq \frac{2\gamma r_{\max}}{1-\gamma} \mathop{\mathbb{E}}_{\substack{S\sim\rho_\pi^{\mathcal{M}},\\ A\sim\pi(\cdot|S)}} \left[D_{\mathrm{TV}}(T(S,A)\|\widehat{T}(S,A))\right]. \tag{15}$$

We define $\epsilon_m = \mathbb{E}_{S\sim\rho_{\pi_D}^{\mathcal{M}},A\sim\pi(\cdot|S)}\left[D_{\mathrm{TV}}\left(T(S,A)\|\widehat{T}(S,A)\right)\right]$ and $\epsilon_m^{\max} = \max_{S\sim\rho_{\pi_D}^{\mathcal{M}}}\left[D_{\mathrm{TV}}\left(T(S,A)\|\widehat{T}(S,A)\right)\right]$. In our algorithm we use the sample policy π_D to sample trajectories from the environment, so we bound the following using **Lemma A.1** and **Lemma A.2** as:

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\substack{S\sim\rho_\pi^{\mathcal{M}},\\ A\sim\pi(\cdot|S)}} \left[D_{\mathrm{TV}}\left(T(S,A)\|\widehat{T}(S,A)\right)\right] \\
&\quad\leq \mathop{\mathbb{E}}_{\substack{S\sim\rho_{\pi_D}^{\mathcal{M}},\\ A\sim\pi(\cdot|S)}} \left[D_{\mathrm{TV}}\left(T(S,A)\|\widehat{T}(S,A)\right)\right] \\
&\quad+ \left\|\rho_\pi^{\mathcal{M}} - \rho_{\pi_D}^{\mathcal{M}}\right\|_1 \max_{S\sim\rho_{\pi_D}^{\mathcal{M}}} \left[D_{\mathrm{TV}}\left(T(S,A)\|\widehat{T}(S,A)\right)\right] \\
&\quad\leq \epsilon_m + \frac{2\gamma}{(1-\gamma)^2}\epsilon_\pi\epsilon_m^{\max}.
\end{aligned}$$

Combining the above inequality with Eq. (15), we have:

$$V_\pi^{\widehat{\mathcal{M}}} - V_\pi^{\mathcal{M}} \leq \frac{2\gamma r_{\max}}{1-\gamma}\epsilon_m + \frac{4\gamma^2 r_{\max}}{(1-\gamma)^3}\epsilon_\pi\epsilon_m^{\max}.$$

□

APPENDIX B
GENERALISATION ANALYSIS OF THE ENSEMBLE DYNAMICS

In this section, we derive a bound on the generalisation error of the environment model trained during our FEMRL learning process. Since the training of the model is a supervised learning process, we can utilise the Probably Approximately Correct (PAC) learning framework for our analysis. First, we give the general bounds for Vapnik–Chervonenkis (VC)-dimension and the discrepancy of the generalisation error between two different data domains. We then give the proof of Theorem 4.2.

B.1 Preliminaries

Theorem B.1. *(Uniform VC-dimension error bound [49]) Let $\mathcal{H}$ be a hypothesis class with $VCdim(\mathcal{H}) \leq d < \infty$. Let D be the probability measures over the sample space. Let S be the empirical dataset sampled from D, $S \sim D^m$ where m is the size of the dataset. Then for any $\delta > 0$, with probability at least $1-\delta$, the following holds for all $h \in \mathcal{H}$:*

$$|\epsilon_D(h) - \epsilon_S(h)| \leq C\sqrt{\frac{d + \log(1/\delta)}{m}}, \tag{16}$$

where C is a constant factor.

We now give a bound of learning between different domains.

Lemma B.2. *Let $\mathcal{H}$ be a hypothesis class. D and D' denote two probability measures over the sample space. Let $\epsilon_D h$ denote the general error of h over D. If the loss function $l(\cdot)$ is bounded by L, then for every $h \in \mathcal{H}$ we have:*

$$\epsilon_D(h) \leq \epsilon_{D'}(h) + L||D - D'||_1. \tag{17}$$

Proof.

$$\begin{aligned}
\epsilon_D(h) &\leq \epsilon_{D'}(h) + |\epsilon_D(h) - \epsilon_{D'}(h)| \\
&\leq \epsilon_{D'}(h) + \int |l(y,h(x))|\,|\mathbb{P}_{(x,y)\sim D} - \mathbb{P}_{(x,y)\sim D'}| \\
&= \epsilon_{D'}(h) + L||D - D'||_1.
\end{aligned} \tag{18}$$

□

B.2 Proof of Theorem 4.2

Proof. According to the definition of Empirical Risk Minimisation (ERM), we have $\epsilon_{S_k}(h_{S_k}) \leq \epsilon_{S_k}(h_{\hat{S}})$, where $h_{\hat{S}}$ is the model learned based on the virtual global empirical dataset $\hat{S}$, where $\hat{S} = \frac{1}{K}\sum_{k=1}^K S_k$. Therefore, we have:

$$\frac{1}{K}\sum_{k=1}^K \epsilon_{S_k}(h_{S_k}) \leq \frac{1}{K}\sum_{k=1}^K \epsilon_{S_k}(h_S) = \epsilon_{\hat{S}}(h_{\hat{S}}). \tag{19}$$

Next we give the bound of the generalisation error of the model ensemble, by considering the distance between the generalisation error of the ensemble of client models, $\epsilon_D(\frac{1}{K}\sum_k h_{S_k})$, and the generalisation error of the model learned from the virtual global dataset, $\epsilon_D(h_{S_k})$. By convexity of the loss function f and Jensen's inequality, we have the probability of at least $1-\delta$ over $\{S_k \sim D_k^m\}_{k=1}^K$ that:

$$\begin{aligned}
\epsilon_D\left(\frac{1}{K}\sum_k h_{S_k}\right) &\leq \frac{1}{K}\sum_k \epsilon_D(h_{S_k}) \\
&\leq \frac{1}{K}\sum_k \left(\epsilon_{S_k}(h_{S_k}) + C\sqrt{\frac{d+log(1/\delta)}{m}} + L||D - D_k||_1\right) \\
&\leq \frac{1}{K}\sum_k \epsilon_{S_k}(h_{S_k}) + C\sqrt{\frac{d+log(1/\delta)}{m}} + \frac{1}{K}\sum_k L||D - D_k||_1 \\
&\leq \epsilon_{\hat{S}_k}(h_{\hat{S}_k}) + C\sqrt{\frac{d+log(1/\delta)}{m}} + \frac{L}{K}\sum_k ||D - D_k||_1.
\end{aligned}$$

The distribution of the virtual global dataset can be calculated using $D = \mathbb{P}_{s,a,s'} = \sum_{s,a} T(s'|s,a)\overline{\pi}_D(a|s)$:

$$\begin{aligned}
||D - D_k||_1 &= \sum_{s',s,a} |\mathbb{P}_{s,a,s'\sim D} - \mathbb{P}_{s,a,s'\sim D_k}| \\
&= \sum_{s'}\sum_{s,a}(T(s'|s,a)\overline{\pi}_D(a|s) - T(s'|s,a)\pi_D^k(a|s)) \\
&= \sum_{s'} T(s'|s,a)\sum_{s,a}(\overline{\pi}_D(a|s) - \pi_D^k(a|s)) \\
&= \sum_{s,a}(\overline{\pi}_D(a|s) - \pi_D^k(a|s)) \\
&= D_{\mathrm{TV}}(\overline{\pi}_D||\pi_D^k).
\end{aligned} \tag{20}$$

Denote the discrepancy between the sample policy of client k, π_D^k, and the virtual global sample policy $\overline{\pi}_D$ as

$D_{\mathrm{TV}}(\overline{\pi}_D||\pi_D^k)$. Let $\Gamma = \sum_k D_{\mathrm{TV}}(\overline{\pi}_D||\pi_D^k)$. Therefore, we have:

$$\epsilon_D\left(\frac{1}{K}\sum_k h_{S_k}\right) \leq \epsilon_{\hat{S}_k}(h_{\hat{S}_k}) + C\sqrt{\frac{d+log(1/\delta)}{m}} + \frac{L}{K}\Gamma, \tag{21}$$

where Γ can be used to measure the degree of the non-IID client data.

□