Federated Ensemble-Directed Offline Reinforcement Learning

Desik Rengarajan [1] **Nitin Ragothaman** [1] **Dileep Kalathil** [1] **Srinivas Shakkottai** [1]

Abstract

We consider the problem of federated offline reinforcement learning (RL), where clients must collaboratively learn a control policy only using data collected using unknown behavior policies. Naïvely combining a standard offline RL approach with a standard federated learning approach to solve this problem can lead to poorly performing policies. We develop Federated Ensemble-Directed Offline Reinforcement Learning Algorithm (FEDORA), which distills the collective wisdom of the clients using an ensemble learning approach. We show that FEDORA significantly outperforms other approaches, including offline RL over the combined data pool, in various complex continuous control and real-world environments.

1. Introduction

Federated learning is a collaborative approach where clients share their locally trained models (not data) with a central server. The server combines these models periodically and returns a federated model to the clients for further improvement (Kairouz et al., 2021; Wang et al., 2021). Federated learning has been successful in supervised learning, leading to well-trained models while maintaining privacy and reducing communication overheads. There has also been interest in applying federated learning to *online* reinforcement learning, wherein clients learn via sequential interactions with their environments and federating learned policies across clients (Khodadadian et al., 2022; Nadiger et al., 2019; Qi et al., 2021). However, such online interactions with real-world systems are often infeasible, and each client might only posses pre-collected local operational data. The fundamental problem of federated *offline* RL is to learn the optimal policy using pre-collected offline data from heterogeneous policies at clients, without sharing the data.

[1]Department of Electrical and Computer Engineering, Texas A&M University, College Station, Texas, United States. Correspondence to: Desik Rengarajan <desik@tamu.edu>.

Workshop of Federated Learning and Analytics in Practice, colocated with 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA.

Offline RL algorithms (Levine et al., 2020), offer an approach to learn using existing datasets at each client. However, we will see that naïvely federating such offline RL algorithms using standard federation approach, such as FedAvg (McMahan et al., 2017) can lead to a policy that is even worse than the constituent policies. We hence identify the following basic challenges of federated offline RL: (i) *Ensemble heterogeneity:* Heterogeneous client datasets will generate policies of different performance levels. It is vital to capture the collective wisdom of these policies. (ii) *Pessimistic value computation:* Offline RL algorithms employs a pessimistic approach toward computing the value of state-actions poorly represented in their dataset. However, federation must be ambitious in extracting the highest values as represented in the ensemble of clients. (iii) *Data heterogeneity:* Multiple local gradient steps on heterogeneous data at each client may lead to biased models.

We propose Federated Ensemble-Directed Offline RL Algorithm (FEDORA), which collaboratively produces high-quality policies. FEDORA recognizes that individual client policies and critics are of varying qualities, and, attempts to maximize the overall objective, while regularizing by the entropy of the weights for combining the constituents. FEDORA ensures optimism across using the federated and local critic at each client and so sets ambitious targets to train against. It addresses data heterogeneity and prunes the influence of irrelevant data. To the best of our knowledge, no other work systematically identifies these fundamental challenges of offline federated RL, or designs methods to explicitly tackle each of them.

We demonstrate the superior performance of FEDORA on a variety of high dimensional and challenging environments. We also demonstrate FEDORA's excellent performance via real-world experiments on a TurtleBot (Amsters & Slaets, 2020). **We provide our codebase, and video of the robot experiments via an anonymous github link[1].**

2. Preliminaries

Federated Learning: The goal of federated learning is to minimize the following objective, $F(\theta) = \mathbb{E}_{i\sim\mathcal{P}}\left[F_i(\theta)\right]$

[1]`https://github.com/DesikRengarajan/FEDORA`

where θ represents the parameter of the federated (server) model, F_i denotes the local objective function of client i, and $\mathcal{P}$ is the distribution over the set of clients $\mathcal{N}$. FedAvg algorithm (McMahan et al., 2017) is a popular method to solve the objective in a federated way. FedAvg divides the training process into rounds, where at the beginning of each round t, the server sends its current model θ^t to all the clients, and each client initializes its current local model to the current server model and performs multiple local updates on its own dataset $\mathcal{D}_i$ to obtain an updated local model θ_i^t. The server then averages these local models proportional to the size of their local dataset to obtain the server model θ^{t+1} for the next round of federation, as

$$\theta^{t+1} = \sum_{i=1}^{|\mathcal{N}|} w_i \theta_i^t, \quad w_i = \frac{|\mathcal{D}_i|}{|\mathcal{D}|}, \quad |\mathcal{D}| = \sum_{i=1}^{|\mathcal{N}|} |\mathcal{D}_i|. \quad (1)$$

Reinforcement Learning: We model RL using the Markov Decision Process (MDP) framework denoted as $(\mathcal{S}, \mathcal{A}, R, P, \gamma, \mu)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ denotes the transition probability function, γ is the discount factor, and μ is initial state distribution. A policy π is a function that maps states to actions (deterministic policy) or states to a distribution over actions (stochastic policy). The goal of RL is to maximize the infinite horizon discounted reward of policy π, defined as $J(\pi) = \mathbb{E}_{\pi,P,\mu}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right]$. The state-action value function (or Q function) of a policy π at state s and executing action a is defined as: $Q^\pi(s, a) = \mathbb{E}_{\pi,P}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s, a_0 = a\right]$.

Offline Reinforcement Learning: The goal of offline RL is to learn a policy π only using a static dataset $\mathcal{D}$ of transitions (s, a, r, s') collected using a behavior policy π_b, without any environment interaction. Offline RL algorithms typically utilize some kind of regularization with respect to the behavior policy to prevent distribution shift (Levine et al., 2020).

TD3-BC (Fujimoto & Gu, 2021) is a behavior cloning (BC) regularized version of the TD3 algorithm (Fujimoto et al., 2018). The TD3-BC objective can be written as

$$\pi = \arg\max_{\pi} U_{\mathcal{D}}(\pi), \quad (2)$$
$$\text{where } U_{\mathcal{D}}(\pi) = \mathbb{E}_{s,a\sim\mathcal{D}}\left[\lambda Q^\pi(s, \pi(s)) - (\pi(s) - a)^2\right].$$

3. Federated Offline Reinforcement Learning

A federated offline RL algorithm aims to learn the optimal policy using data distributed across clients. We denote the set of clients as $\mathcal{N}$. Each client $i \in \mathcal{N}$ has an offline dataset $\mathcal{D}_i = \{(s_j, a_j, r_j, s'_j)_{j=1}^{m_i}\}$ generated according to a behavior policy π_i^b. We assume that the underlying MDP model P and reward function $R(\cdot, \cdot)$ are identical for all the clients, and the statistical differences between the offline datasets $\mathcal{D}_i$ are only due to the difference in behavior policies π_i^b.

In federated offline RL, each client has to learn using its local data $\mathcal{D}_i$, thus we consider a client objective function that is consistent with a standard offline RL algorithm objective. Our approach is compatible with most offline RL algorithms. We choose TD3-BC (Eq. (2)) (Fujimoto & Gu, 2021) as the client objective due to its simplicity and good empirical performance. We define the federated offline RL objective as $U(\pi_{\text{fed}}) = \sum_{i=1}^{|\mathcal{N}|} w_i U_{\mathcal{D}_i}(\pi_{\text{fed}})$ where w_i are weights of federation.

One approach to leveraging experiences across users without sharing data would be to combine existing federated learning techniques with offline RL algorithms. *Is such a naïve federation strategy sufficient to learn an excellent federated policy collaboratively? Furthermore, is federation even necessary?* In this section, we aim to understand the challenges of federated offline RL with the goal of designing an algorithmic framework to address these challenges.

We start with an example illustrating the issues in designing a federated offline RL algorithm. We consider the Hopper environment from MuJoCo (Todorov et al., 2012), with $|\mathcal{N}| = 10$, $|\mathcal{D}_i| = 5000$, and we use the data from the D4RL dataset (Fu et al., 2020). We consider a setting where five clients use the data from the hopper-expert-v2 dataset (generated using a completely trained (expert) SAC policy) and five clients use the data from the hopper-medium-v2 dataset (generated using a partially trained (medium) policy achieving only a third of the expert performance). The clients and the server are unaware of the quality (expert or medium) of the data. Fig. 2 shows the performance comparison of multiple algorithms, where the mean and the standard deviation are calculated over 4 seeds.

Combining All Data (Centralized): Combining data and learning centrally is the ideal scenario in supervised learning. However, as seen in Fig. 2, performing centralized training over combined data generated using different behavior policies in offline RL can be detrimental. This is consistent with (Yu et al., 2021) that proves that pooling data from behavior policies with different expertise levels can exacerbate the distributional shift between the learned policy and the individual datasets, leading to poor performance.

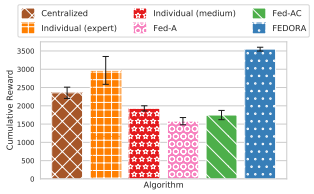


Figure 2. Performance comparison of federated and centralized offline RL algorithms.

Individual Offline RL: Agents learn using their local data without collaboration. In Fig. 2, we observe that clients with

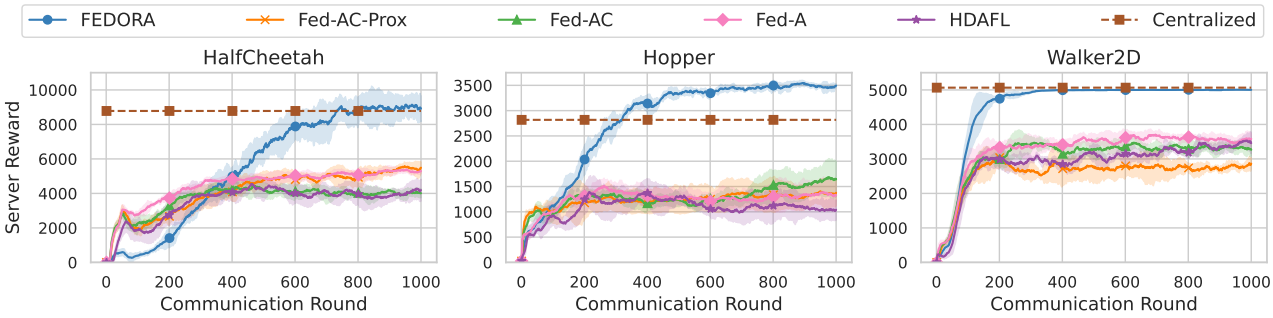


Figure 1. Evaluation on different MuJoCo environments.

either expert or medium data do not learn well and exhibit a large standard deviation due to insufficient data.

Naïve Federated Offline RL: We perform FedAvg (Eq. (1)) with TD3-BC as the local objective. We conduct experiments where we federate only the actor (Fed-A) or both the actor and the critic (Fed-AC). Surprisingly, these naïve strategies result in federated policies that perform worse than individual offline RL, as seen in Fig. 2.

We now outline the issues of with Federated Offline RL. **1. Ensemble Heterogeneity:** Performing offline RL over heterogeneous data yields a set of policies of different qualities. It is crucial to leverage the information contained in these varied policies rather than simply averaging them. However, federation after a single-step local gradient at each client using weights in the manner of FedAvg, $w_i = |\mathcal{D}_i|/|\sum_{i=1}^{|\mathcal{N}|} |\mathcal{D}_i|$, is equivalent to solving the offline RL problem using the combined dataset of all clients (Wang et al., 2021). This approach leads to poor performance due to the resulting distribution shift. *How should we optimally federate the ensemble of policies learned by the clients?* **2. Pessimistic Value Computation:** Most offline RL algorithms involve a pessimistic term with respect to the offline data for minimizing the distribution shift. Training a client's critic using only the local data would make it pessimistic towards actions poorly represented in its dataset but well represented in other clients' data. *How do we effectively utilize the federated critic along with the locally computed critic to set ambitious targets for offline RL at each client?* **3. Data Heterogeneity:** Performing multiple local gradient steps would bias a client's local model to its dataset. *How should we regularize local policies to prevent this?*

4. FEDORA Design Approach

We develop Federated Ensemble-Directed Offline RL Algorithm (FEDORA) that addresses the above issues.

Our solution is to follow the principle of maximum entropy to choose weights that best represent the current knowledge about the relative merits of the clients' policies. We prevent the weights from collapsing over a few clients by adding an entropy regularization over the weights with temperature parameter β resulting in the following objective,

$$U(\pi_{\text{fed}}) = \sum_{i=1}^{|\mathcal{N}|} w_i U_{\mathcal{D}_i}(\pi_{\text{fed}}) - \frac{1}{\beta} \sum_{i=1}^{|\mathcal{N}|} w_i \log w_i. \quad (3)$$

We can then show using a Lagrange dual approach that this objective is maximized when $w_i = \frac{e^{\beta U_{\mathcal{D}_i}(\pi_{\text{fed}})}}{\sum_{i=1}^{|\mathcal{N}|} e^{\beta U_{\mathcal{D}_i}(\pi_{\text{fed}})}}$. Based on these soft-max type of weights suggested by the entropy-regularized objective, we now design FEDORA. In what follows, $\pi_i^{(t,k)}$ denotes the policy of client i in round t of federation after k local policy update steps. Since all clients initialize their local policies to the federated policy, $\pi_i^{(t,0)} = \pi_{\text{fed}}^t$ for each client i. We also denote $\pi_i^t = \pi_i^{(t,K)}$, where K is the maximum number of local updates. We can similarly define $Q_i^{(t,k)}$, $Q_i^{(t,0)} = Q_{\text{fed}}^t$, and $Q_i^t = Q_i^{(t,K)}$ for the local critic.

Ensemble-Directed Learning over Client Policies: We utilize the performance of the final local policy $J_i^t = \mathbb{E}_{s\sim\mathcal{D}_i}\left[Q_i^t(s, \pi_i^t(s))\right]$, as a proxy for $U_{\mathcal{D}_i}(\pi_{\text{fed}})$. Here, Q_i^t is the local critic after updates. Our approach toward computing Q_i^t and π_i^t are described later. The accuracy of the local estimates J_i^t are highly dependent on the number of data samples available at i, thus, we account for the size of the dataset $|\mathcal{D}_i|$ while computing weights as follows,

$$w_i^t = \frac{e^{\beta J_i^t}|\mathcal{D}_i|}{\sum_{i=1}^{|\mathcal{N}|} e^{\beta J_i^t}|\mathcal{D}_i|}, \quad \pi_{\text{fed}}^{t+1} = \sum_{i=1}^{|\mathcal{N}|} w_i^t \pi_i^t. \quad (4)$$

Federated Optimism for Critic Training: A critic based on offline data suffers from extrapolation errors as state-action pairs not seen in the local dataset will be erroneously estimated. Since the federated policy is derived from the set of local policies, it may take actions not seen in any client's local dataset. This problem is exacerbated when the local policy at the beginning of each communication round is

initialized to the federated policy. We introduce the notion of *federated optimism* to train local critics, wherein critics leverage the wisdom of the crowd and are encouraged to be optimistic. We achieve this federated optimism via two steps. First, we compute the federated critic as follows,

$$Q_{\text{fed}}^{t+1} = \sum_{i=1}^{|\mathcal{N}|} w_i^t Q_i^t. \tag{5}$$

Such entropy-regularized averaging ensures that the critics from clients with good policies significantly influence the federated critic. Second, for the local critic update, we choose the target value as the maximum value between the local critic and the federated critic, given by $\tilde{Q}_i^{(t,k)}(s,a) = \max\left(Q_i^{(t,k)}(s,a), Q_{\text{fed}}^t(s,a)\right)$, where $\tilde{Q}_i^{(t,k)}(s,a)$ is the target value of state s and action a at the t^{th} round of federation after k local critic updates. This ensures that the local critic has an optimistic (but likely feasible) target seen by the system. Using this optimistic target in the Bellman error, we update the local critic as

$$Q_i^{(t,k+1)} = \arg\min_Q \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}_i}[(r + \gamma\tilde{Q}_i^{(t,k)}(s',a') - Q(s,a))^2], \text{where } a' = \pi_i^{(t,k)} \tag{6}$$

Proximal Policy Update for Heterogeneous Data: An optimistic critic might erroneously estimate the value of $\tilde{Q}_i^{(t,k)}$. Therefore, regularizing the local policy update w.r.t. both the local data and the federated policy is crucial. For regularization w.r.t. to the local offline data, we use the TD3-BC loss as the local loss function $\mathcal{L}_{\text{local}}(\pi) = \mathbb{E}_{(s,a)\sim\mathcal{D}_i}[-Q_i^{(t,k)}(s,\pi(s)) + (\pi(s)-a)^2]$. We then define the actor loss $\mathcal{L}_{\text{actor}}$ in Eq. (7), where the second term is a regularization w.r.t. to the federated policy (Li et al., 2020). The local policy is updated using $\mathcal{L}_{\text{actor}}$,

$$\mathcal{L}_{\text{actor}}(\pi) = \mathcal{L}_{\text{local}}(\pi) + \mathbb{E}_{(s,a)\sim\mathcal{D}_i}[(\pi(s) - \pi_{\text{fed}}^t(s))^2],$$
$$\pi_i^{t,k+1} = \arg\min_\pi \mathcal{L}_{\text{actor}}(\pi). \tag{7}$$

Decaying the Influence of Local Data: Training on local data may hamper the updated policy's performance since the local dataset may be generated according to a non-expert behavior policy. Hence, we decay the influence of $\mathcal{L}_{\text{local}}$ (influence of its local data) by a factor δ if $J_i^{\text{fed},t} \geq J_i^t$. Where $J_i^{\text{fed},t} = \mathbb{E}_{s\sim\mathcal{D}_i}[Q_{\text{fed}}^t(s, \pi_{\text{fed}}^t(s))]$ is a proxy for the performance of the federated policy. We summarize FEDORA in Algorithm 1 and 2 (Appendix A).

5. Experimental Evaluation

We conduct experiments to analyze the performance of FEDORA. We develop a framework to implement FEDORA over distributed resources and on a single system. (See Appendix C). We consider the following baselines. The local objective of all baselines is TD3-BC (Eq. 2). **(i) Fed-A:** Server performs FedAvg over only the actor's parameters. **(ii) Fed-AC:** Server performs FedAvg over the parameters of both the actor and the critic. **(iii) Fed-AC-Prox:** We add a proximal term to Fed-AC, which has been shown to help in federated supervised learning when clients have heterogeneous data (Li et al., 2020). **(iv) Heterogeneous Data-Aware Federated Learning (HDAFL)** We extend HDAFL (Yang et al., 2020) to the offline RL setting by federating only a part of the actor during each round. **(v) Centralized:** We perform offline RL (TD3-BC) over the pooled data from all clients.

Experiments on Simulated Environments: We run experiments on MuJoCo environments (Todorov et al., 2012) with $|\mathcal{N}| = 50$, and $|\mathcal{D}_i| = 5000$. Of these 50 clients, 25 are provided with data from the D4RL (Fu et al., 2020) expert dataset, while the other 25 are provided with data from the D4RL medium dataset. The clients (and the server) are unaware of the nature of their datasets. We choose $|\mathcal{N}_t| = 20$ clients at random to participate in each round t of federation. For each plot, we evaluate the performance with four different seeds. We provide details in Appendix C. In Fig. 1, we plot the cumulative episodic reward of the federated policy during each round of federation. We observe that FEDORA outperforms all federated baselines and achieves performance equivalent to or better than centralized training. Furthermore, the federated baselines fail to learn a good server policy even after training for many communication rounds and plateau at lower levels compared to FEDORA, emphasizing that the presence of heterogeneous data hurts their performance.

We present ablation studies and additional experiments in Appendix D due to space constraints.

Real-World Experiments on TurtleBot: We evaluate the performance of FEDORA on a TurtleBot (Amsters & Slaets, 2020) to collaboratively learn a control policy to navigate waypoints while avoiding obstacles using offline data distributed across multiple robots. This scenario is relevant to real-world applications such as cleaning robots where collaborative learning is essential because a single robot might not have enough data/seen diverse scenarios to learn from, and sharing data comes with privacy concerns.

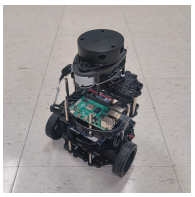
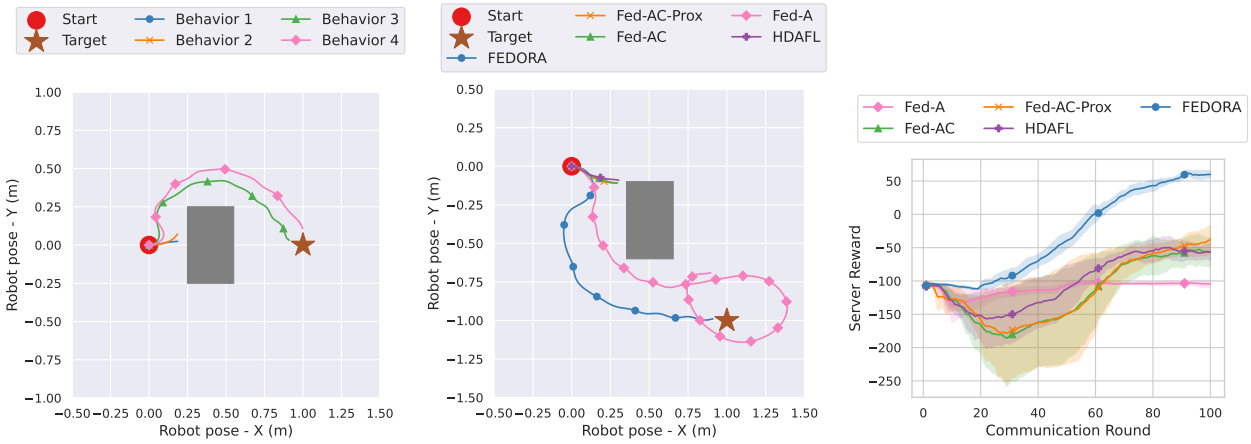


Figure 3. TurtleBot3 Burger.

We collect data in the real-world using four behavior policies with varying levels of expertise (Fig. 4(a).). We train over 20 clients for 100 communication rounds, each consisting of 20 local epochs (Fig. 4(c)). Fig. 4(b) shows the trajectories obtained by the learned policies of different algorithms in the real-world. FEDORA is able to successfully reach the target by avoiding the ob-



(a) Trajectories of behavior policies (b) Trajectories of learned policies (c) Comparison with baselines

Figure 4. Real-world experiments on a TurtleBot.

stacle. We provide more details in Appendix E. We discuss limitations and societal impact of our work in Appendix F.

6. Conclusion

We presented an approach for federated offline RL with heterogeneous client data. We solved multiple challenging issues by systematically developing a well-performing ensemble-directed approach entitled FEDORA, which extracts the collective wisdom of the policies and critics and discourages excessive reliance on irrelevant local data. We demonstrated its performance on several simulation and real-world tasks.

7. Acknowledgement

This work was supported in part by NSF Grants CNS 1955696 and ECCS 2038963, ARO Grant W911NF-19-1-0367, ARL Grant W911NF-21-2-0064. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

Portions of this research were conducted with the advanced computing resources provided by Texas A&M High Performance Research Computing.

References

Amsters, R. and Slaets, P. Turtlebot 3 as a robotics education platform. In *Robotics in Education: Current Research and Innovations*, pp. 170–181. Springer, 2020.

Beutel, D. J., Topal, T., Mathur, A., Qiu, X., Parcollet, T., and Lane, N. D. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*, 2020.

Chen, X., Zhou, Z., Wang, Z., Wang, C., Wu, Y., and Ross, K. Bail: Best-action imitation learning for batch deep reinforcement learning. *Advances in Neural Information Processing Systems*, 33:18353–18363, 2020.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Fujimoto, S. and Gu, S. S. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596, 2018.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pp. 2052–2062, 2019.

Hebert, L., Golab, L., Poupart, P., and Cohen, R. Fedformer: Contextual federation with attention in reinforcement learning. *arXiv preprint arXiv:2205.13697*, 2022.

Hu, Y., Hua, Y., Liu, W., and Zhu, J. Reward shaping based federated reinforcement learning. *IEEE Access*, 9: 67259–67267, 2021.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S., Stich, S., and Suresh, A. T. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pp. 5132–5143, 2020.

Khodadadian, S., Sharma, P., Joshi, G., and Maguluri, S. T. Federated reinforcement learning: Linear speedup under markovian sampling. In *International Conference on Machine Learning*, pp. 10997–11057, 2022.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Kumar, A., Fu, J., Soh, M., Tucker, G., and Levine, S. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 1179–1191, 2020a.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 1179–1191, 2020b.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020.

Lim, H.-K., Kim, J.-B., Ullah, I., Heo, J.-S., and Han, Y.-H. Federated reinforcement learning acceleration method for precise control of multiple devices. *IEEE Access*, 9: 76296–76306, 2021.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282, 2017.

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

Peng, X. B., Kumar, A., Zhang, G., and Levine, S. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Qi, J., Zhou, Q., Lei, L., and Zheng, K. Federated reinforcement learning: techniques, applications, and open challenges. *arXiv preprint arXiv:2108.11887*, 2021.

Qin, R., Gao, S., Zhang, X., Xu, Z., Huang, S., Li, Z., Zhang, W., and Yu, Y. Neorl: A near real-world benchmark for offline reinforcement learning. *arXiv preprint arXiv:2102.00714*, 2021.

Reddi, S. J., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečný, J., Kumar, S., and McMahan, H. B. Adaptive federated optimization. In *International Conference on Learning Representations*, 2021.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pp. 1889–1897, 2015.

Tarasov, D., Nikulin, A., Akimov, D., Kurenkov, V., and Kolesnikov, S. CORL: Research-oriented deep offline reinforcement learning library. In *3rd Offline RL Workshop: Offline RL as a "Launchpad"*, 2022. URL `https://openreview.net/forum?id=SyAS49bBcv`.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033, 2012.

Vázquez-Canteli, J. R., Dey, S., Henze, G., and Nagy, Z. Citylearn: Standardizing research in multi-agent reinforcement learning for demand response and urban energy management. *arXiv preprint arXiv:2012.10504*, 2020.

Wang, J., Charles, Z., Xu, Z., Joshi, G., McMahan, H. B., Al-Shedivat, M., Andrew, G., Avestimehr, S., Daly, K., Data, D., et al. A field guide to federated optimization. *arXiv preprint arXiv:2107.06917*, 2021.

Wang, Q., Xiong, J., Han, L., Liu, H., Zhang, T., et al. Exponentially weighted imitation learning for batched historical data. *Advances in Neural Information Processing Systems*, 2018.

Wang, X., Li, R., Wang, C., Li, X., Taleb, T., and Leung, V. C. Attention-weighted federated deep reinforcement learning for device-to-device assisted heterogeneous collaborative edge caching. *IEEE Journal on Selected Areas in Communications*, 39(1):154–169, 2020.

Wu, Y., Tucker, G., and Nachum, O. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

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

Yang, L., Beliard, C., and Rossi, D. Heterogeneous data-aware federated learning. *arXiv preprint arXiv:2011.06393*, 2020.

Yu, T., Kumar, A., Chebotar, Y., Hausman, K., Levine, S., and Finn, C. Conservative data sharing for multi-task offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34:11501–11516, 2021.

Zhou, D., Zhang, Y., Sonabend-W, A., Wang, Z., Lu, J., and Cai, T. Federated offline reinforcement learning. *arXiv preprint arXiv:2206.05581*, 2022.

Appendix

We present several results and details in the appendix that illustrates the performance and further describe FEDORA. These include, the FEDORA algorithm (Appendix A), related works (Appendix B), details of our experimental setup (Appendix C), additional experiments studying different components of FEDORA and illustrating its performance in different settings (Appendix D), details of our real-world experiments using a TurtleBot (Appendix E), and discussion on limitations, societal impact and future work (Appendix F).

A. FEDORA: Algorithm

Algorithm 1 Outline of Client i's Algorithm

function train_client(π^t_{fed}, Q^t_{fed})
  $\pi_i^{(t,0)} = \pi^t_{\text{fed}}, \quad Q_i^{(t,0)} = Q^t_{\text{fed}}$
  for $1 \leq k < K$ **do**
    Update Critic by one gradient step w.r.t. Eq. (6)
    Update Actor by one gradient step w.r.t. Eq. (7)
  end for
  Decay $\mathcal{L}_{\text{local}}$ by δ if $J_i^{\text{fed},t} \geq J_i^t$
end function

Algorithm 2 Outline of Server Algorithm

Initialize $\pi^1_{\text{fed}}, Q^1_{\text{fed}}$
for $t \in 1 \ldots$ **do**
  Send π^t_{fed} and Q^t_{fed} to $i \in \mathcal{N}$
  Sample $\mathcal{N}_t \subset \mathcal{N}$
  for $i \in \mathcal{N}_t$ **do**
    i.train_client $(\pi^t_{\text{fed}}, Q^t_{\text{fed}})$
  end for
  Compute π^{t+1}_{fed} and Q^{t+1}_{fed} for clients in $\mathcal{N}_t$ using Eq. (4) and (5) respectively.
end for

B. Related Work

Offline RL: The goal of offline RL is to learn a policy from a fixed dataset generated by a behavior policy (Levine et al., 2020). One of the key challenges of the offline RL approach is the distribution shift problem where the state-action visitation distribution of learned policy may be different from that of the behavior policy which generated the offline data. It is known that this distribution shift may lead to poor performance of the learned policy (Levine et al., 2020). A common method used by offline RL algorithms to tackle this problem is to learn a policy that is close to the behavior policy that generated the data via regularization either on the actor or critic (Fujimoto & Gu, 2021; Fujimoto et al., 2019; Kumar et al., 2020a; 2019; Wu et al., 2019). Some offline RL algorithms perform weighted versions of behavior cloning or imitation learning on either the whole or subset of the dataset (Wang et al., 2018; Peng et al., 2019; Chen et al., 2020).

Federated Learning: McMahan et al. (McMahan et al., 2017) introduced FedAvg, a federation strategy where clients collaboratively learn a joint model without sharing data. A generalized version of FedAvg was presented in (Reddi et al., 2021). A key problem in federated learning is data heterogeneity wherein clients have non-identically distributed data, which causes unstable and slow convergence (Wang et al., 2021; Karimireddy et al., 2020; Li et al., 2020). To tackle the issue of data heterogeneity, (Li et al., 2020) proposed FedProx, a variant of FedAvg, where a proximal term is introduced reduce deviation by the local model from the server model. (Karimireddy et al., 2020) tackled client drift in local updates caused by data heterogeneity using control variates.

Federated Reinforcement Learning: Federated learning has recently been extended to the online RL setting. (Khodadadian

et al., 2022) analyzed the performance of federated tabular Q-learning. (Qi et al., 2021) combined traditional online RL algorithms with FedAvg for multiple applications. Some works propose methods to vary the weighting scheme of FedAvg according to performance metrics such as the length of a rally in the game of Pong (Nadiger et al., 2019) or average return in the past 10 training episodes (Lim et al., 2021) to achieve better performance or personalization. (Wang et al., 2020) proposed a method to compute weights using attention over performance metrics of clients such as average reward, average loss, and hit rate for an edge caching application. (Hebert et al., 2022) used a transformer encoder to learn contextual relationships between agents in the online RL setting. (Hu et al., 2021) proposed an alternative approach to federation where reward shaping is used to share information among clients. (Xie & Song, 2023) proposed a KL divergence-based regularization between the local and global policy to address the issue of data heterogeneity in an online RL setting.

In the offline RL setting, (Zhou et al., 2022) propose federated dynamic treatment regime algorithm by formulating offline federated learning using a multi-site MDP model constructed using linear MDPs. However, this approach relies on running the local training to completion followed by just one step of federated averaging. Unlike this work, our method does not assume linear MDPs, which is a limiting assumption in many real-world problems. Moreover, we use the standard federated learning philosophy of periodic federation followed by multiple local updates. *To the best of our knowledge, ours is the first work to propose a general federated offline RL algorithm for clients with heterogeneous data.*

C. Experimental Setup

Algorithm Implementation: We use the PyTorch framework to program the algorithms in this work, based on a publicly-available TD3-BC implementation. The actor and the critic networks have two hidden layers of size 256 with ReLu non-linearities. We use a discount factor of 0.99, and the clients update their networks using the Adam optimizer with a learning rate of 3×10^{-4}. For training FEDORA, we fixed the decay rate $\delta = 0.995$ and the temperature $\beta = 0.1$. TD3-BC trains for 5×10^5 time steps in the centralized setup. The batch size is 256 in both federated and centralized training.

The training data for clients are composed of trajectories sampled from the D4RL dataset. In situations where only a fraction of the clients partake in a round of federation, we uniformly sample the desired number of clients from the entire set.

Federation Structure: We implement FEDORA over the Flower federated learning platform (Beutel et al., 2020), which supports learning across devices with heterogeneous software stacks, compute capabilities, and network bandwidths. Flower manages all communication across clients and the server and permits us to implement the custom server-side and client-side algorithms of FEDORA easily. However, since Flower is aimed at supervised learning, it only transmits and receives a single model at each federation round, whereas we desire to federate both policies and critic models. We solve this limitation by simply appending both models together, packing and unpacking them at the server and client sides appropriately.

While 'FEDORA-over-Flower' is an effective solution for working across distributed compute resources, we also desire a simulation setup that can be executed on a single machine. This approach sequentially executes FEDORA at each selected client, followed by a federation step, thereby allowing us to evaluate the different elements of FEDORA in an idealized federation setup.

Compute Resources: Each run on the MuJoCo environments (as in Fig. 1) takes around 7 hours to complete when run on a single machine (AMD Ryzen Threadripper 3960X 24-Core Processor, 2x NVIDIA 2080Ti GPU). This time can be drastically reduced when run over distributed compute using the Flower framework.

D. Additional Experiments

D.1. Effects of data from multiple behavior policies

To understand the effect of data coming from multiple behavior policies on centralized training, we consider a scenario where 50 clients with datasets of size $|D_i| = 5000$ participate in federation, with 25 clients having expert data and the other 25 having random data, i.e., data generated from a random policy. From Fig. 5, we notice that combining data of all clients deteriorates performance as compared to FEDORA. This observation highlights the fact that performing centralized training with data collected using multiple behavior policies can be detrimental.

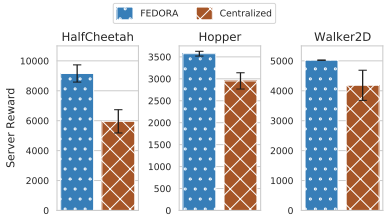


Figure 5. Comparison of FEDORA and centralized training with heterogeneous data.

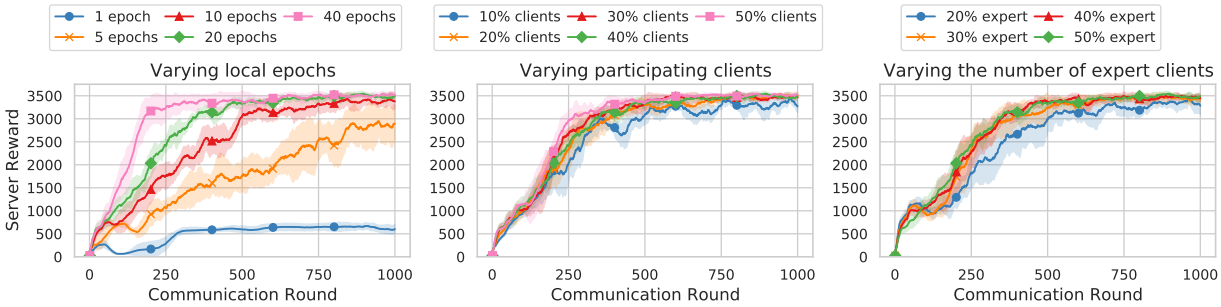
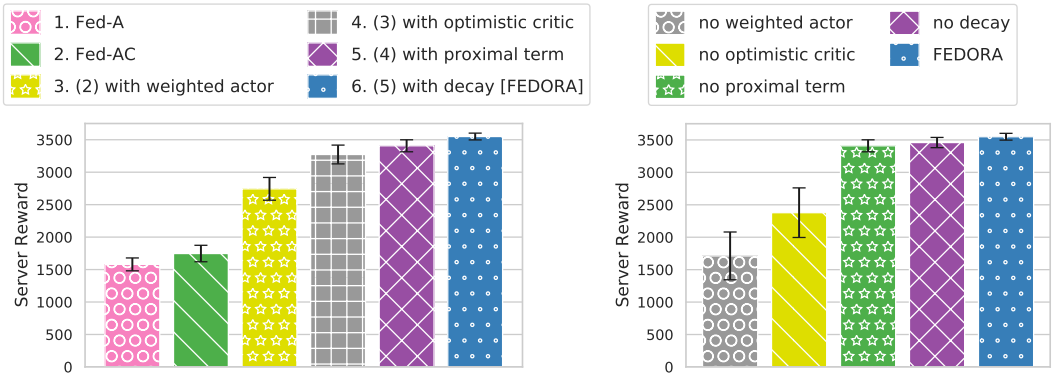


Figure 6. Effect of varying the number of (a) local gradient steps, (b) participating clients in each round, and (c) expert clients in FEDORA.

D.2. Sensitivity to Client Updates and Data Quality

We study the sensitivity of FEDORA to client update frequency and data quality in the Hopper environment in the same setting as in Fig. 1. Communication efficiency is a crucial to federated learning. Increasing the number of local training steps can improve communication efficiency, but is detrimental in under heterogeneous data due to client drift (Karimireddy et al., 2020). In Fig. 6(a), we study the effect of varying the number of local training epochs. We observe that increasing the number of epochs leads to faster learning, emphasizing that FEDORA can effectively learn with heterogeneous data. Not all clients may participate in every round of federation due to communication/compute constraints. In Fig.6(b), we evaluate how FEDORA learns under various fractions of clients' participation in each round . We observe that FEDORA is robust towards variations in the fraction of clients during federation. Finally, in Fig. 6(c) we study the effect of data heterogeneity by varying the percentage of clients with expert datasets. We observe that FEDORA performs well even when only 20% of the total clients have expert-quality data.

D.3. Importance of Individual Algorithm Component



(a) Effect of sequentially adding one algorithm component at a time

(b) Effect of removing one individual algorithm components from FEDORA

Figure 7. Ablation Studies.

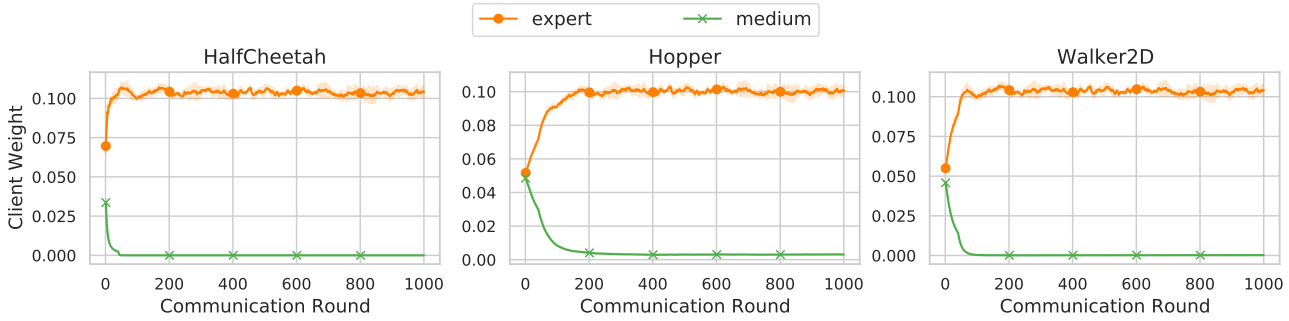
We perform an ablation study to examine the different components of our algorithm and understand their relative impacts on the performance of the federated policy. We use the experimental framework with 10 clients and the Hopper environment described in Section 3, and plot the performance of the federated policy with mean and standard deviation over 4 seeds. The ablation is performed in two ways: (a) We build up FEDORA starting with Fed-A, the naïve method which federates only the actor, and add one new algorithm component at a time and evaluate its performance. (b) We exclude one component of FEDORA at a time and evaluate the resulting algorithm.

We observe in Fig. 7a that using priority-weighted averaging of the client's policy to compute the federated policy (Eq. (4)), and an optimistic critic (Eq. (5) - (6)) significantly improves the performance of the federated policy. This is consistent

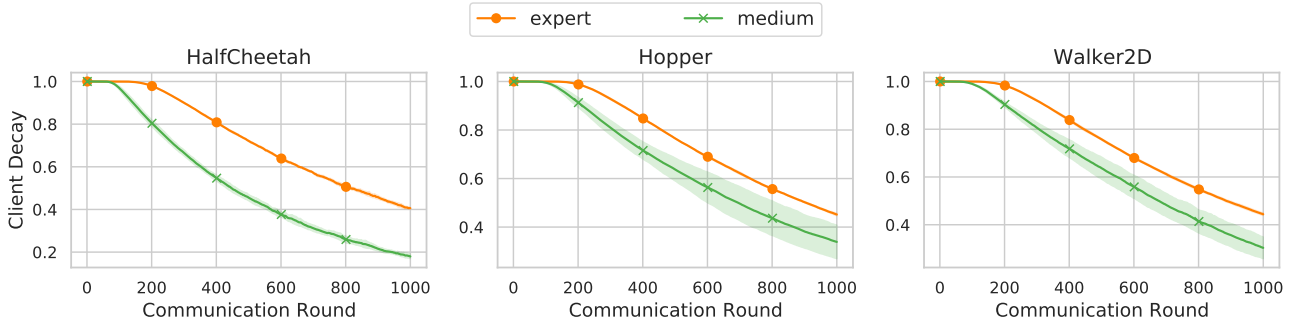
with our intuition that the most important aspect is extracting the collective wisdom of the policies and critics available for federation, and ensuring that the critic sets optimistic targets. The proximal term helps regularize local policy updates (Eq. (7)) by choosing actions close to those seen in the local dataset or by the federated policy. Additionally, decaying the influence of local updates enables the local policy to leverage the federated policy's vantage by choosing actions not seen in the local dataset.

From Fig. 7b, we observe that removing priority-weighted actor from FEDORA causes the steepest drop in performance, followed by the optimistic critic. Again, this is consistent with our intuition on these being the most important effects. Excluding the proximal term and local decay also results in a reduction in server performance along with a greater standard deviation.

D.4. Analysis of Client Performance



(a) Client ratio



(b) Client decay

Figure 8. Analysis of client performance during federation. The average of the performance metric is computed across expert and medium clients participating in a given round of federation.

We train FEDORA on MuJoCo environments using a setup similar to Section 5 where 20 out of the 50 clients are randomly chosen to participate in each round of federation. Our goal is to analyze the contribution of clients with expert data and those with medium data to the learning process. As before, the clients and the algorithm are unaware of the data quality.

We plot the mean weights w_i^t across the expert and medium dataset clients participating in a given round of federation in Fig. 8a. We observe that the weights of medium clients drop to 0, while the weights of expert clients rise to 0.1. This finding emphasizes the fact that clients are combined based on their relative merits.

In Fig. 8b, we plot the mean of the decay value associated with $\mathcal{L}_{\text{local}}$ across participating expert and medium dataset clients. The decay of both sets of clients drops as training progresses. A reduction in decay occurs each time the local estimate of the federated policy's performance $J_i^{\text{fed},t}$ is greater than the estimated performance of the updated local policy J_i^t. A decreasing decay implies that the federated policy offers a performance improvement over local policies more often as the rounds t advance. Thus, training only on local data is detrimental, and participation in federation can help learn a superior policy.

D.5. Federated Offline RL experiments with CityLearn

Real-world environments often have a large state space and are stochastic in nature. We run federated experiments on CityLearn (Vázquez-Canteli et al., 2020) to assess the effectiveness of FEDORA on such large-scale systems. CityLearn is an OpenAI Gym environment with the goal of urban-scale energy management and demand response, modeled on data from residential buildings. The goal is to reshape the aggregate energy demand curve by regulating chilled water tanks and domestic hot water, two modes of thermal energy storage in each building. The energy demand of residential buildings changes as communities evolve and the weather varies. Hence, the controller must update its policy periodically to perform efficient energy management. Federated learning would allow utilities that serve communities in close proximity to train a policy collaboratively while preserving user data privacy, motivating the use of FEDORA for this environment.

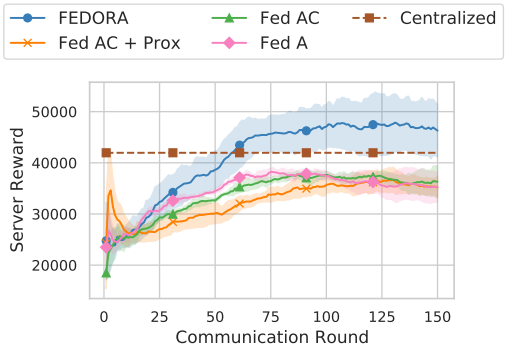


Figure 9. Evaluation of algorithms on CityLearn.

In our experiments, we have 10 clients with 5000 training examples such that they all participate in 150 rounds of federation. The training data for the clients is obtained from NeoRL, an offline RL benchmark (Qin et al., 2021). 5 clients each have data from the CityLearn High and CityLearn Low datasets, which are collected by a SAC policy trained to 75% and 25% of the best performance level, respectively. During each round of federation, each client performs 20 local epochs of training. The server reward at the end of each federation round is evaluated online and shown in Fig. 9. We observe that FEDORA outperforms other federated offline RL algorithms as well as centralized training, which learns using TD3-BC on the data aggregated from every client. These findings indicate that FEDORA can perform well in large-scale stochastic environments.

D.6. Effect of multiple behavior policies and proportion of clients participating in federation

In this section, we study the effects of clients having data from multiple behavior policies for varying proportions of clients participating in federation. We consider a scenario with 50 clients having $D_i = 5000$ in the Hopper-v2 environment where,

- 12 clients have expert data (samples from a policy trained to completion with SAC.).
- 12 clients have medium data (samples from a policy trained to approximately 1/3 the performance of the expert).
- 14 clients have random data (samples from a randomly initialized policy).
- 12 clients have data from the replay buffer of a policy trained up to the performance of the medium agent.

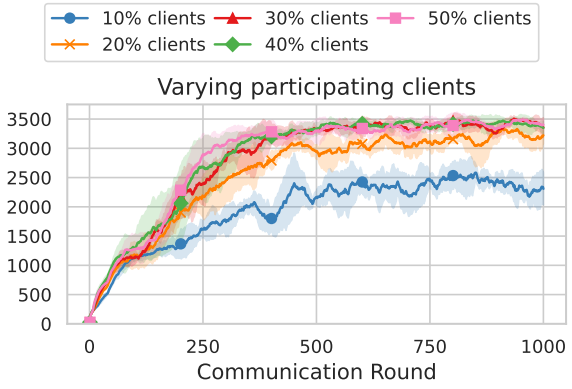


Figure 10. Effect of varying the number of participating clients in each round on FEDORA

We run FEDORA by varying the the percentage of clients participating in each round of federation. We observe that the FEDORA is fairly robust to the fraction of clients participating in federation even when the fraction is as low as 20%.

D.7. Centralized training with other Offline RL algorithms

We consider a scenario similar to the one in Fig. 5 for the Hopper-v2 environment with 50 clients, having $|D_i| = 5000$ participating in federation, where 25 clients have expert data, and 25 clients have random data. We compare the performance of different Offline RL algorithms over the pooled data with FEDORA. The algorithms we choose are Conservative Q-Learning for Offline Reinforcement Learning (CQL) (Kumar et al., 2020b) and Offline Reinforcement Learning with Implicit Q-Learning (IQL) (Kostrikov et al., 2021) whose implementations are obtained from the CORL library (Tarasov et al., 2022). We can observe from Fig. 11 that pooling data from different behavior policies affects both offline RL algorithms.

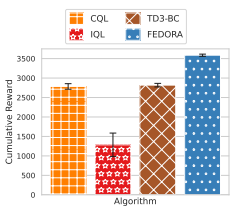


Figure 11. Comparison with Offline RL algorithms

E. Details of Real-World Robot Experiments

E.1. Demonstration Data Collection

We train four behavior policies of varying levels of expertise using TRPO (Schulman et al., 2015) on a custom simulator for mobile robots described in section E.2. The first policy is capable of waypoint navigation but collides with obstacles. The second policy can reach waypoints while avoiding obstacles present at one fixed position. The third policy has not fully generalized to avoiding obstacles at various positions. Finally, the fourth policy can navigate to the goal without any collision. We execute the behavior policies in the real-world by varying the waypoint (target location) and location of the obstacle to gather demonstration data, which we then use to train FEDORA and other baselines. Each client has a dataset consisting of 300 data points collected using a single behavior policy. After training, we test the learned policies in the real-world on a TurtleBot to ascertain its feasibility.

E.2. Simulator Design

We develop a first-order simulator for mobile robots using the OpenAI Gym framework, which enables the training of RL algorithms. The robot's pose is represented by its X- and Y-coordinates in a 2D space and its orientation with respect to the X-axis, θ. The pose is updated using differential drive kinematics

$$\begin{aligned} x_{t+1} &= x_t + \Delta t\ v \cos\theta_t \\ y_{t+1} &= y_t + \Delta t\ v \sin\theta_t \\ \theta_{t+1} &= \theta_t + \Delta t\ \omega, \end{aligned} \tag{8}$$

where (x_t, y_t, θ_t) is the pose at time t, v and w are the linear and angular velocity of the robot respectively, and Δt is time discretization of the system.

The simulator uses a functional LIDAR to detect the presence of obstacles. We simulate the LIDAR using a discrete representation of the robot and obstacles in its immediate environment. For each scanning direction around the LIDAR, we use Bresenham's line algorithm to generate a path comprising of discrete points. The simulator determines LIDAR measurements by counting the number of points along each path, starting from the robot and continuing until it encounters and obstacle or reaches the maximum range.

The reward function is designed to encourage effective waypoint navigation while preventing collisions. We define a boundary grid that extends for 1m beyond the start and the goal positions in all directions. The reward function at time t for navigating to the goal position (x_g, y_g) is chosen to be

$$R_t = \begin{cases} +100, & \text{if } |x_t - x_g| \leq thresh \texttt{ and } |y_t - y_g| \leq thresh \\ -10, & \text{if robot outside boundary} \\ -100, & \text{if robot collides} \\ -(\texttt{c.t.e}_t^2 + \texttt{a.t.e}_t + \texttt{h.e}_t) + \sum \texttt{lidar}_t, & \text{otherwise} \end{cases} \tag{9}$$

where $\texttt{c.t.e}_t$ is the cross-track error, $\texttt{a.t.e}_t$ is the along-track error, $\texttt{h.e}_t$ is the heading error, $\texttt{lidar}_t$ is the array of LIDAR measurements at time t, and $thresh$ is the threshold error in distance, chosen as 0.1m. Let the L-2 distance to the goal and the heading to the goal at time t be d_t^g and θ_t^g respectively. Then, we have

$$\begin{aligned} d_t^g &= \sqrt{(x_g - x_t)^2 + (y_g - y_t)^2}, \\ \theta_t^g &= \tan^{-1}\left(\tfrac{y_g - y_t}{x_g - x_t}\right), \\ \texttt{c.t.e}_t &= d_t^g \sin(\theta_g - \theta_t), \\ \texttt{a.t.e}_t &= |x_g - x_t| + |y_g - y_t|, \\ \texttt{h.e}_t &= \theta_t^g - \theta_t. \end{aligned} \tag{10}$$

E.3. Mobile Robot Platform

We evaluate the trained algorithms on a Robotis TurtleBot3 Burger mobile robot (Amsters & Slaets, 2020), an open-source differential drive robot. The robot has a wheel encoder-based pose estimation system and is equipped with an RPLIDAR-A1 LIDAR for obstacle detection. We use ROS as the middleware to set up communication. The robot transmits its state (pose and LIDAR information) over a wireless network to a computer, which then transmits back the corresponding action suggested by the policy being executed.

F. Limitations, Societal Impacts, and Future work

F.1. Limitations and Future Work

In this work, we examine the issue of Federated Offline RL. We make the assumption that all clients share the same MDP model (transition kernel and reward model), and any statistical variances between the offline datasets are due to differences in the behavior policies used to collect the data. Moving forward, we aim to broaden this to cover scenarios where clients have different transition and reward models. To achieve this, we plan to extend ideas from offline meta RL to the federated learning scenario. Furthermore, we plan to explore personalization in federated offline RL as an extension to our research. We also believe that our approach may also be useful in the context of federated supervised learning, especially when the data is sourced from varying qualities, and we intend to formally investigate this in the future as a seperate line of work.

F.2. Ethics Statement and Societal Impacts

In this work, we introduce a novel algorithm for federated offline reinforcement learning. The domain of federated offline RL offers the potential for widespread implementation of RL algorithms while upholding privacy by not sharing data, as well as reducing the need for communication. Throughout our study, no human subjects or human-generated data were involved. As a result, we do not perceive any ethical concerns associated with our research methodology.

While reinforcement learning holds great promise for the application in socially beneficial systems, caution must be exercised when applying it to environments involving human interaction. This caution arises from the fact that guarantees in such scenarios are probabilistic, and it is essential to ensure that the associated risks remain within acceptable limits to ensure safe deployments.