Federated Ensemble-Directed Offline Reinforcement Learning

Desik Rengarajan* **Nitin Ragothaman** **Dileep Kalathil** **Srinivas Shakkottai**
Department of Electrical and Computer Engineering, Texas A&M University

Abstract

We consider the problem of federated offline reinforcement learning (RL), a scenario under which distributed learning agents must collaboratively learn a high-quality control policy only using small pre-collected datasets generated according to different 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. In response, we develop the Federated Ensemble-Directed Offline Reinforcement Learning Algorithm (FEDORA), which distills the collective wisdom of the clients using an ensemble learning approach. We develop the FEDORA codebase to utilize distributed compute resources on a federated learning platform. We show that FEDORA significantly outperforms other approaches, including offline RL over the combined data pool, in various complex continuous control environments and real-world datasets. Finally, we demonstrate the performance of FEDORA in the real-world on a mobile robot. We provide our code and a video of our experiments at `https://github.com/DesikRengarajan/FEDORA`.

1 Introduction

Federated learning is an approach wherein clients learn collaboratively by sharing their locally trained models (not their data) with a federating agent, which periodically combines their models and returns the federated model to the clients for further refinement [10, 30]. Federated learning has seen recent success in supervised learning applications due to its ability to generate well-trained models using small amounts of data at each client, while preserving privacy and reducing the usage of communication resources. There has also been interest in federated learning for *online* reinforcement learning (RL), wherein clients learn via sequential interactions with their environments and federating learned policies across clients [12, 21, 23]. However, such online interactions with real-world systems are often infeasible, and each client might only posses pre-collected operational data generated according to a client-specific behavior policy. The fundamental problem of federated *offline* RL is on how to learn the optimal policy only using such offline data collected by heterogeneous policies at clients, without actually sharing any of the data.

Offline RL algorithms [17], such as CQL [15] and TD3-BC [5] offer an actor-critic learning approach that only utilizes existing datasets at each client. However, in our case, this approach taken across many small datasets at clients will produce an ensemble of policies of heterogeneous (unknown) qualities across the clients, along with their corresponding critics of variable accuracy. We will see that naïvely federating such offline RL trained policies and critics using a standard federation approach, such as FedAvg [20] 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 this ensemble of policies, not average them. (ii) *Pessimistic value*

*Corresponding author. `Email:desik.29@gmail.com`

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

computation: Offline RL employs a pessimistic approach toward computing the value of actions poorly represented in the data to minimize distribution shift (and so reduce the probability of taking these actions). However, federation must be ambitious in extracting the highest values as represented in the ensemble of critics (and so promote high-value actions). (iii) *Data heterogeneity:* As with other federated learning, multiple local gradient steps based on heterogeneous data at each client between federation rounds may lead to biased models. We must regularize local policies to reduce such drift.

In this work, we propose Federated Ensemble-Directed Offline RL Algorithm (FEDORA), which collaboratively produces a high-quality control policy and critic function. FEDORA estimates the performance of client policies using only local data (of unknown quality) and, at each round of federation, produces a weighted combination of the constituent policies that maximizes the overall objective, while regularizing by the entropy of the weights. The same approach is followed to federate client critics. Following the principle of maximum entropy in this manner produces both federated policies and critics that extract the collective wisdom of the ensemble. In doing so, it constructs a federated policy and a critic based on the relative merits of each client policy in an ensemble learning manner. FEDORA ensures optimism across evaluation by the federated and local critic at each client and so sets ambitious targets to train against. It addresses data heterogeneity by regularizing client policies with respect to both the federated policy and the local dataset. Finally, FEDORA prunes the influence of irrelevant data by decaying the reliance on a dataset based on the quality of the policy it can generate. 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 develop a framework for implementing FEDORA either on a single system or over distributed compute resources. We evaluate FEDORA on a variety of MuJoCo environments and real-world datasets and show that it outperforms several other approaches, including performing offline RL on a pooled dataset. We also demonstrate FEDORA's excellent performance via real-world experiments on a TurtleBot robot [1].

2 Related Work

Offline RL: The goal of offline RL is to learn a policy from a fixed dataset generated by a behavior policy [17]. 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 [17]. 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 [5, 7, 16, 14, 33]. Some offline RL algorithms perform weighted versions of behavior cloning or imitation learning on either the whole or subset of the dataset [31, 22, 3]. [38, 37] propose data rebalancing methods designed to prioritize highly-rewarding transitions that can be augmented to offline RL algorithms to alleviate the distribution shift issue for heterogeneous data settings.

Federated Learning: [20] introduced FedAvg, a federation strategy where clients collaboratively learn a joint model without sharing data. A generalized version of FedAvg was presented in [25]. A key problem in federated learning is data heterogeneity wherein clients have non-identically distributed data, which causes unstable and slow convergence [30, 11, 18]. To tackle the issue of data heterogeneity, [18] proposed FedProx, a variant of FedAvg, where a proximal term is introduced reduce deviation by the local model from the server model.

Federated Reinforcement Learning: Federated learning has recently been extended to the online RL setting. [12] analyzed the performance of federated tabular Q-learning. [23] 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 [21] or average return in the past 10 training episodes [19] to achieve better performance or personalization. [32] 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. [8] used a transformer encoder to learn contextual relationships between agents in the online RL setting. [9] proposed an alternative approach to federation where reward shaping is used to share information among clients. [34] 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, [39] 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.*

3 Preliminaries

Federated Learning: The goal of federated learning is to minimize the following objective,

$$F(\theta) = \mathbb{E}_{i\sim\mathcal{P}}\left[F_i(\theta)\right], \tag{1}$$

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 $\mathcal{N}$. The FedAvg algorithm [20] is a popular method to solve Eq. (1) in a federated way. FedAvg divides the training process into rounds, where at the beginning of each round t, the server broadcasts its current model θ^t to all the clients, and each client initializes its current local model to the current server model. Clients perform multiple local updates on their 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

$$\theta^{t+1} = \sum_{i=1}^{|\mathcal{N}|} w_i\theta_i^t, \quad w_i = \frac{|\mathcal{D}_i|}{|\mathcal{D}|}, \quad |\mathcal{D}| = \sum_{i=1}^{|\mathcal{N}|}|\mathcal{D}_i|. \tag{2}$$

Reinforcement Learning: We model RL using the Markov Decision Process (MDP) framework denoted as a tuple $(\mathcal{S}, \mathcal{A}, R, P, \gamma, \mu)$, 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 that gives the probability of transitioning to a state s' by taking action a in state s, γ is the discount factor, and μ is the distribution of the initial state s_0. 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 $J(\pi) = \mathbb{E}_{\pi,P,\mu}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t, a_t)\right]$, which is the expected cumulative discounted reward obtained by executing policy π. The state-action value function (or Q function) of a policy π at state s and executing action a is the expected cumulative discounted reward obtained by taking action a in state s and following policy π thereafter: $Q^\pi(s, a) = \mathbb{E}_{\pi,P}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t, a_t)|s_0 = s, a_0 = a\right]$.

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 additional interactions with the environment. Offline RL algorithms typically utilize some kind of regularization with respect to the behavior policy to ensure that the learned policy does not deviate from the behavior policy. This regularization is done to prevent distribution shift, a significant problem in offline RL, where the difference between the learned policy and behavior policy can lead to erroneous Q-value estimation of state-action pairs not seen in the dataset [16, 17].

Our approach is compatible with most offline RL algorithms, such as CQL [15] or TD3-BC [5]. We choose TD3-BC for illustration, motivated by its simplicity and its superior empirical performance in benchmark problems. The TD3-BC algorithm is a behavior cloning (BC) regularized version of the TD3 algorithm [6]. The policy in TD3-BC is updated using a linear combination of TD3 objective and behavior cloning loss, where the TD3 objective ensures policy improvement and the BC loss prevents distribution shift. More precisely, the TD3-BC objective can be written as

$$\pi = \arg\max_{\pi}\ U_{\mathcal{D}}(\pi), \tag{3}$$

$$\text{where} \quad U_{\mathcal{D}}(\pi) = \mathbb{E}_{s,a\sim\mathcal{D}}\left[\lambda Q^\pi(s, \pi(s)) - (\pi(s) - a)^2\right], \tag{4}$$

and λ is a hyperparameter that determines the relative weight of the BC term.

4 Federated Offline Reinforcement Learning

In real-world offline RL applications, data is typically obtained from the operational policies of multiple agents (clients) with different (unknown) levels of expertise. Clients often prefer not to share data. We aim to learn the optimal policy for the underlying RL problem using only such offline data, without the clients knowing the quality of their data, or sharing it with one another or the server. Furthermore, neither the clients nor server have access to the underlying model or the environment. We denote the set of clients as $\mathcal{N}$. Each client $i \in \mathcal{N}$ has the 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 used for collecting the data.

In a standard federated learning algorithm such as FedAvg, each client performs multiple parameter updates before sending its parameters to the server. It is known that performing multiple local updates in federated learning can reduce the communication cost significantly without compromising on the optimality of the converged solution [10, 30]. In federated offline RL, since each client has to perform multiple steps of policy evaluation and policy update using its local offline data $\mathcal{D}_i$, it is reasonable to consider a client objective function that is consistent with a standard offline RL algorithm objective. We choose the objective function used in the TD3-BC algorithm [5], i.e., $U_{\mathcal{D}_i}$ given in Eq. (3), as the client objective function. Our choice is motivated by the simplicity of the TD3-BC objective function and its empirical success in a variety of environments. Similar to the standard federated learning objective given in Eq. (1), we can now define the federated offline RL objective as

$$U(\pi_{\text{fed}}) = \sum_{i=1}^{|\mathcal{N}|} w_i U_{\mathcal{D}_i}(\pi_{\text{fed}}), \tag{5}$$

where w_i are weights to be determined.

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 by illustrating the issues in designing a federated offline RL algorithm. We consider the Hopper environment from MuJoCo [28], with $|\mathcal{N}| = 10$, $|\mathcal{D}_i| = 5000$, and we use the data from the D4RL dataset [4]. However, instead of using the data generated by the same policy for all clients, we consider the setting where five clients use the data from the hopper-expert-v2 dataset (which was generated using a completely trained (expert) SAC policy) and five clients use the data from the hopper-medium-v2 dataset (which was 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. 1 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. 1, performing centralized training over combined data generated using different behavior policies in offline RL can be detrimental. This is consistent with [36] 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. Similar deterioration due to combining data has also been observed in other offline RL literature [5, 16]. We also explore centralized algorithms with data re-balancing, and observer that FEDORA is still superior (See Appendix B.9). We would like to further emphasise that combining the data from all clients is a hypothetical base in the federated setting, as data is distributed amongst clients and cannot be combined.

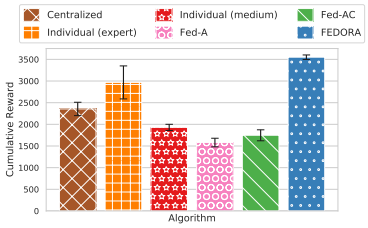


Figure 1: Performance comparison of federated and centralized offline RL algorithms.

Individual Offline RL: Here, agents apply offline RL to their own datasets without collaborating with others. In Fig. 1, we observe that clients with either expert or medium data do not learn well and

exhibit a large standard deviation. This observation may be attributed to no client having sufficient data to learn a good policy.

Naïve Federated Offline RL: A simple federation approach is to use the offline RL objective as the local objective and apply FedAvg (Eq. (2)). However, offline RL algorithms typically comprise two components – an actor and a critic. It is unclear a priori which components should be federated, so 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. 1.

4.1 Issues with Federated Offline RL

Our example illustrates several fundamental issues that must be addressed while designing viable federated offline RL algorithms, including:

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 [30]. This approach leads to poor performance due to the resulting distribution shift, as shown in Fig. 1. *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 critic using only the local data with this pessimistic term could 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: Federated learning calls for performing multiple local gradient steps at each client before federation to enhance communication efficiency. However, numerous epochs would bias a client's local model to its dataset. This client drift effect is well known in federated (supervised) learning and could lead to policies that are not globally optimal. In turn, this could cause the federated policy's performance to be worse than training locally using only the client's data, as seen in Fig. 1. *How should we regularize local policies to prevent this?*

5 FEDORA Design Approach

We desire to develop a Federated Ensemble-Directed Offline RL Algorithm (FEDORA) that addresses the issues outlined in Section 4 in a systematic manner. Three fundamental requirements drive our approach. First, the clients jointly possess an ensemble of local policies of different (unknown) qualities, and the server must leverage the collective knowledge embedded in this ensemble during federation. Second, the quality of these policies must be assessed using an ensemble of critics that depend on local data for policy evaluation. Finally, after each round of federation, clients must update their local policies via offline RL utilizing both their local data and the received federated policy.

Maximizing the federated offline reinforcement learning (RL) objective in Eq. (5) using FedAvg would set weights as in Eq. (2), i.e., each client's contribution is weighted by the size of its dataset. This is is equivalent to solving the offline RL problem using the combined dataset of all clients. However, such an approach exacerbates the distribution shift problem that affects offline RL algorithms, leading to poor performance. This issue has been verified analytically and empirically in [36]. We illustrated this phenomenon in Fig. 1, where offline RL over pooled data resulted in a sub-optimal policy. The recommendation in [36] is to share data conservatively by identifying which samples are likely to result in policy improvement. However, we cannot share any of the data across clients in the federated offline RL setting.

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. Here, the weights are prevented from collapsing over a few clients that have the best current performance by adding an entropy

regularization over the weights with temperature parameter β resulting in the following objective:

$$U(\pi_{\text{fed}}) = \sum_{i=1}^{|\mathcal{N}|} w_i U_{\mathcal{D}_i}(\pi_{\text{fed}}) - \frac{1}{\beta} \sum_{i=1}^{|\mathcal{N}|} w_i \log w_i. \tag{6}$$

We can then show using a Lagrange dual approach that this objective is maximized when

$$w_i = \frac{e^{\beta U_{\mathcal{D}_i}(\pi_{\text{fed}})}}{\sum_{i=1}^{|\mathcal{N}|} e^{\beta U_{\mathcal{D}_i}(\pi_{\text{fed}})}}. \tag{7}$$

Based on these soft-max type of weights suggested by the entropy-regularized objective, we now design FEDORA accounting for each of the three requirements indicated above.

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 at the beginning of each round of federation, $\pi_i^{(t,0)} = \pi_{\text{fed}}^t$ for each client i. We also denote $\pi_i^t = \pi_i^{(t,K)}$, where K is the maximum number of local updates. Since all clients initialize their local critics to the federated critic, 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.

5.1 Ensemble-Directed Learning over Client Policies

We first require a means of approximating $U_{\mathcal{D}_i}(\pi_{\text{fed}})$ in order to determine the weight w_i of client i as shown in Eq. (7). We utilize the performance of the final local policy $J_i^t = \mathbb{E}_{s\sim\mathcal{D}_i}\left[Q_i^t(s, \pi_i^t(s))\right]$, which also characterizes the relative performance at client i, as a proxy for $U_{\mathcal{D}_i}(\pi_{\text{fed}})$. Here, Q_i^t is the local critic function at round t after K local updates. It is hard to directly obtain such an unbiased local critic Q_i^t in offline RL, since we do not have access to the environment for executing the policy and evaluating its performance. 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, and so in the usual manner of federated averaging, we need to account for the size of the dataset $|\mathcal{D}_i|$ while computing weights. We thus have client weights and federated policy update as

$$w_i^t = \frac{e^{\beta J_i^t}|\mathcal{D}_i|}{\sum_{i=1}^{|\mathcal{N}|} e^{\beta J_i^t}|\mathcal{D}_i|}, \quad \pi_{\text{fed}}^{t+1} = \sum_{i=1}^{|\mathcal{N}|} w_i^t \pi_i^t. \tag{8}$$

5.2 Federated Optimism for Critic Training

The critic in our algorithm plays two major roles. First, offline RL for policy updates at each client requires policy evaluation using local data. Second, policy evaluation by the critic determines weight w_i^t of the local policy at client i for ensemble learning during each round t of federation. We desire a local critic at each client that can utilize the knowledge from the ensemble of critics across all clients while also being tuned to the local data used for policy evaluation.

A critic based on offline data suffers from extrapolation errors as state-action pairs not seen in the local dataset will be erroneously estimated, greatly impacting actor-critic style policy updates in federated offline RL. 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 use an ensemble-directed federation of the critics, where the local critic of client i at round t is weighed according to its merit to compute the federated critic as

$$Q_{\text{fed}}^{t+1} = \sum_{i=1}^{|\mathcal{N}|} w_i^t Q_i^t. \tag{9}$$

Such entropy-regularized averaging ensures that the critics from clients with good policies significantly influence the federated critic.

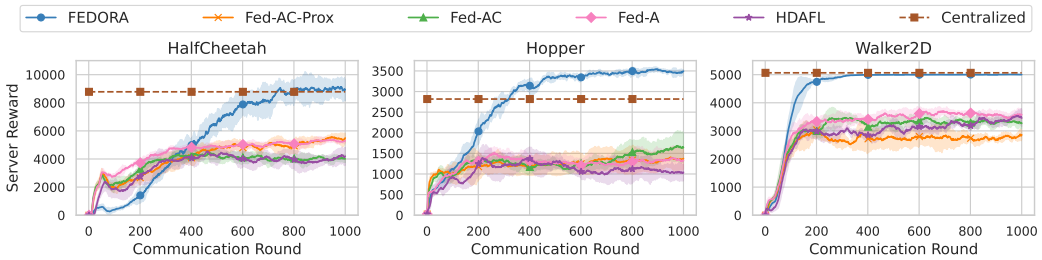


Figure 2: Evaluation of algorithms on different MuJoCo environments.

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 tth 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

$$Q_i^{(t,k+1)} = \arg\min_Q \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}_i}[(r + \gamma\tilde{Q}_i^{(t,k)}(s',a') - Q(s,a))^2], \tag{10}$$

where $a' = \pi_i^{(t,k)}$. In practice, we obtain $Q_i^{(t,k+1)}$ after a single gradient update.

5.3 Proximal Policy Update for Heterogeneous Data

While essential in order to set ambitious estimates, 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 same method as in the TD3-BC algorithm and define 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. (11), where the second term is a regularization w.r.t. to the federated policy. The local policy is updated using $\mathcal{L}_{\text{actor}}$,

$$\mathcal{L}_{\text{actor}}(\pi) = \mathcal{L}_{\text{local}}(\pi) + \mathbb{E}_{(s,a)\sim\mathcal{D}_i}[(\pi(s) - \pi_{\text{fed}}^t(s))^2], \quad \pi_i^{t,k+1} = \arg\min_\pi \mathcal{L}_{\text{actor}}(\pi). \tag{11}$$

5.4 Decaying the Influence of Local Data

FEDORA uses a combination of local data loss and a proximal term for its policy update Eq. (11). However, the local data loss might hamper the updated policy's performance since the local dataset may be generated according to a non-expert behavior policy. Hence, a client must decay the influence of its local data if it is reducing the performance of the updated policy by lowering the influence of $\mathcal{L}_{\text{local}}$ in $\mathcal{L}_{\text{actor}}$. To do so, we first evaluate the performance of the federated policy using the federated critic and local data at round t. For this evaluation, we use the proxy estimate $J_i^{\text{fed},t} = \mathbb{E}_{s\sim\mathcal{D}_i}[Q_{\text{fed}}^t(s,\pi_{\text{fed}}^t(s))]$. We compare this value with the performance of the updated policy, J_i^t, which is obtained using the updated critic. This difference provides us with an estimate of the improvement the local data provides. We decay the influence of $\mathcal{L}_{\text{local}}$ by a factor δ if $J_i^{\text{fed},t} \geq J_i^t$.

We summarize FEDORA in Algorithm 1 and 2. We would like to emphasize that in our algorithm, as in any offline RL setting, the clients or server do not have access to the environment or the MDP. Further, we would like to point out that the clients are not aware of the quality of data they possess.

Algorithm 1 Outline of Client i's Algorithm

function train_client($\pi^t_{\text{fed}}, Q^t_{\text{fed}}$)
  $\pi_i^{(t,0)} = \pi^t_{\text{fed}}, \quad Q_i^{(t,0)} = Q^t_{\text{fed}}$
  for $1 \leq k < K$ **do**
    Update Critic by one gradient step w.r.t. Eq. (10)
    Update Actor by one gradient step w.r.t. Eq. (11)
  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 $(\pi^t_{\text{fed}}, Q^t_{\text{fed}})$ (Client side)
  end for
  Compute π^{t+1}_{fed} and Q^{t+1}_{fed} for clients in $\mathcal{N}_t$ using Eq. (8) and (9) respectively.
end for

6 Experimental Evaluation

We conduct experiments to answer three broad questions: **(i) Comparative Performance:** How does FEDORA perform compared to other approaches with client data generated by heterogeneous behavior policies?, **(ii) Sensitivity to client updates and data quality:** How does the performance depend on the number of local gradient steps at clients, the randomness in the available number of agents for federation, and the quality of the data at the clients?, and **(iii) Ablation:** How does the performance depend on the different components of FEDORA? We implement FEDORA over the Flower federated learning platform [2] which supports learning across devices. We also provide a simulation setup that can be executed on a single machine (See Appendix A).

Baselines: We consider the following baselines. **(i) Fed-A:** The local objective of all clients follows TD3-BC (Eq. 3). The server performs FedAvg over the actor's parameters, whereas each client learns the critic locally. **(ii) Fed-AC:** The local objective of all clients follows TD3-BC and the 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 [18]. **(iv) Heterogeneous Data-Aware Federated Learning (HDAFL)** We extend HDAFL [35] to the offline RL setting by dividing the actor network into generic and client-specific parts and then federating only the generic part during each round. **(v) Centralized:** We perform offline RL (TD3-BC) over the pooled data by combining the data present in all clients.

6.1 Experiments on Simulated Environments

Experimental Setup: We focus on a scenario where clients are collaboratively learning to solve the same task, but the behavior policies used to collect data for each client could differ. We run experiments with the number of clients $|\mathcal{N}| = 50$, with each client having a local dataset of size $|\mathcal{D}_i| = 5000$. Of these 50 clients, 25 are provided with data from the D4RL [4] expert dataset, while the other 25 are provided with data from the D4RL medium dataset. The clients (and the server) are unaware of the quality of their datasets. Further, both the client and server do not have access to the environment. We choose $|\mathcal{N}_t| = 20$ clients at random to participate in each round t of federation. The server obtains weights from clients in $|\mathcal{N}_t|$ and computes the federated weight π^{t+1}_{fed} and Q^{t+1}_{fed}. For each plot, we evaluate the performance with four different seeds. We evaluate the performance of FEDORA and baselines over three MuJoCo tasks: Hopper, HalfCheetah, and Walker2D. During a round of federation, each client performs 20 epochs of local training in all algorithms, which is roughly 380 local gradient steps in our experimental setup.

Comparative Performance of FEDORA: In Fig. 2, we plot the cumulative episodic reward of the server/federated policy during each round of communication/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.

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

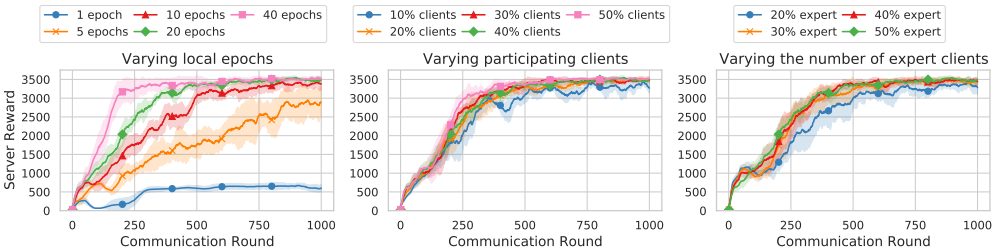
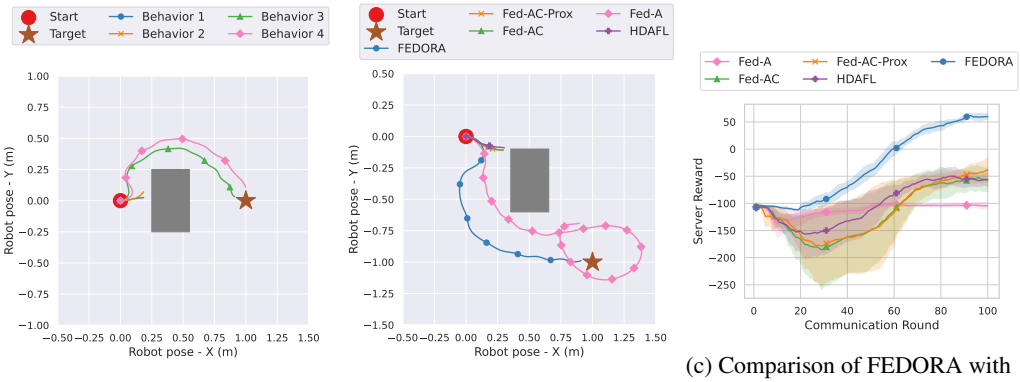


Figure 4: Effect of varying the number of (a) local gradient steps, (b) participating clients in each round, and (c) expert clients in FEDORA.

random policy. From Fig. 3, 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.



(a) Trajectories of behavior policies (b) Trajectories of learned policies (c) Comparison of FEDORA with federated baseline algorithms

Figure 5: Evaluation of FEDORA and other federated baselines for a mobile robot navigation task in the presence of an obstacle.

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. 2. Increasing the number of local training steps can improve communication efficiency, but is detrimental under heterogeneous data due to client drift [11]. In Fig. 4(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.4(b), we study the effect of the fraction of clients participating in federation. We observe that FEDORA is robust towards variations in the fraction of clients during federation. Finally, in Fig. 4(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. We present several ablation studies and additional experiments in appendix B.

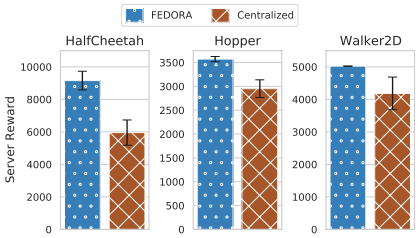


Figure 3: Comparison of FEDORA and centralized training with heterogeneous data.

6.2 Real-World Experiments on TurtleBot

We evaluated the performance of FEDORA on TurtleBot [1], a two-wheeled differential drive robot (Fig. 6) to collaboratively learn a control policy to navigate waypoints while avoiding obstacles using offline data distributed across multiple robots (clients). This scenario is relevant to several real-world applications, such as cleaning robots in various houses, which aim to collaboratively

learn a control policy to navigate and avoid obstacles using data distributed across different robots. Collaborative learning is essential, because a single robot might not have enough data to learn from or have encountered adequately different scenarios. Additionally, federated learning overcomes the privacy concerns associated with sharing data among the robots.

We collect data in the real-world using four behavior policies with varying levels of expertise (Fig. 5(a)). We train over 20 clients for 100 communication rounds, each consisting of 20 local epochs (see Fig. 5(c)). Fig. 5(b) shows the trajectories obtained by the learned policies of different algorithms in the real-world, and only FEDORA is able to successfully reach the target by avoiding the obstacle. We provide more details in Appendix C. We provide a video of our experiments at `https://github.com/DesikRengarajan/FEDORA`.

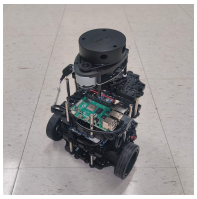


Figure 6: TurtleBot3 Burger.

7 Conclusion

We presented an approach for federated offline RL, accounting for the heterogeneity in the quality of the ensemble of policies that generated the data at the clients. 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.

8 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.

9 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 separate line of work.

10 Acknowledgement

This work was supported in part by NSF Grants CNS 2312978, ECCS 2038963, ARO Grant W911NF-19-1-0367, and NSF-CAREER-EPCN-2045783. 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

[1] Robin Amsters and Peter Slaets. Turtlebot 3 as a robotics education platform. In *Robotics in Education: Current Research and Innovations*, pages 170–181. Springer, 2020.

[2] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Titouan Parcollet, and Nicholas D Lane. Flower: A friendly federated learning research framework. *arXiv preprint arXiv:2007.14390*, 2020.

[3] Xinyue Chen, Zijian Zhou, Zheng Wang, Che Wang, Yanqiu Wu, and Keith Ross. Bail: Best-action imitation learning for batch deep reinforcement learning. *Advances in Neural Information Processing Systems*, 33:18353–18363, 2020.

[4] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[5] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

[6] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596, 2018.

[7] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pages 2052–2062, 2019.

[8] Liam Hebert, Lukasz Golab, Pascal Poupart, and Robin Cohen. Fedformer: Contextual federation with attention in reinforcement learning. *arXiv preprint arXiv:2205.13697*, 2022.

[9] Yiqiu Hu, Yun Hua, Wenyan Liu, and Jun Zhu. Reward shaping based federated reinforcement learning. *IEEE Access*, 9:67259–67267, 2021.

[10] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[11] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143, 2020.

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

[13] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

[14] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

[15] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

[16] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

[17] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[18] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020.

[19] Hyun-Kyo Lim, Ju-Bong Kim, Ihsan Ullah, Joo-Seong Heo, and Youn-Hee Han. Federated reinforcement learning acceleration method for precise control of multiple devices. *IEEE Access*, 9:76296–76306, 2021.

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

[21] Chetan Nadiger, Anil Kumar, and Sherine Abdelhak. Federated reinforcement learning for fast personalization. In *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pages 123–127. IEEE, 2019.

[22] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

[23] Jiaju Qi, Qihao Zhou, Lei Lei, and Kan Zheng. Federated reinforcement learning: techniques, applications, and open challenges. *arXiv preprint arXiv:2108.11887*, 2021.

[24] Rongjun Qin, Songyi Gao, Xingyuan Zhang, Zhen Xu, Shengkai Huang, Zewen Li, Weinan Zhang, and Yang Yu. Neorl: A near real-world benchmark for offline reinforcement learning. *arXiv preprint arXiv:2102.00714*, 2021.

[25] Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and Hugh Brendan McMahan. Adaptive federated optimization. In *International Conference on Learning Representations*, 2021.

[26] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 1889–1897, 2015.

[27] Denis Tarasov, Alexander Nikulin, Dmitry Akimov, Vladislav Kurenkov, and Sergey Kolesnikov. CORL: Research-oriented deep offline reinforcement learning library. In *3rd Offline RL Workshop: Offline RL as a "Launchpad"*, 2022.

[28] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033, 2012.

[29] José R Vázquez-Canteli, Sourav Dey, Gregor Henze, and Zoltán Nagy. Citylearn: Standardizing research in multi-agent reinforcement learning for demand response and urban energy management. *arXiv preprint arXiv:2012.10504*, 2020.

[30] Jianyu Wang, Zachary Charles, Zheng Xu, Gauri Joshi, H Brendan McMahan, Maruan Al-Shedivat, Galen Andrew, Salman Avestimehr, Katharine Daly, Deepesh Data, et al. A field guide to federated optimization. *arXiv preprint arXiv:2107.06917*, 2021.

[31] Qing Wang, Jiechao Xiong, Lei Han, Han Liu, Tong Zhang, et al. Exponentially weighted imitation learning for batched historical data. *Advances in Neural Information Processing Systems*, 2018.

[32] Xiaofei Wang, Ruibin Li, Chenyang Wang, Xiuhua Li, Tarik Taleb, and Victor CM Leung. 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.

[33] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

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

[35] Lixuan Yang, Cedric Beliard, and Dario Rossi. Heterogeneous data-aware federated learning. *arXiv preprint arXiv:2011.06393*, 2020.

[36] Tianhe Yu, Aviral Kumar, Yevgen Chebotar, Karol Hausman, Sergey Levine, and Chelsea Finn. Conservative data sharing for multi-task offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34:11501–11516, 2021.

[37] Yang Yue, Bingyi Kang, Xiao Ma, Gao Huang, Shiji Song, and Shuicheng Yan. Offline prioritized experience replay. *arXiv preprint arXiv:2306.05412*, 2023.

[38] Yang Yue, Bingyi Kang, Xiao Ma, Zhongwen Xu, Gao Huang, and Shuicheng Yan. Boosting offline reinforcement learning via data rebalancing. *arXiv preprint arXiv:2210.09241*, 2022.

[39] Doudou Zhou, Yufeng Zhang, Aaron Sonabend-W, Zhaoran Wang, Junwei Lu, and Tianxi Cai. Federated offline reinforcement learning. *arXiv preprint arXiv:2206.05581*, 2022.

Appendix

We present several results and details in the appendix that illustrates the performance of FEDORA. These include details of our experimental setup (Appendix A), additional experiments studying different components of FEDORA and illustrating its performance in different settings (Appendix B), and details of our real-world experiments using a TurtleBot (Appendix C).

A 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.

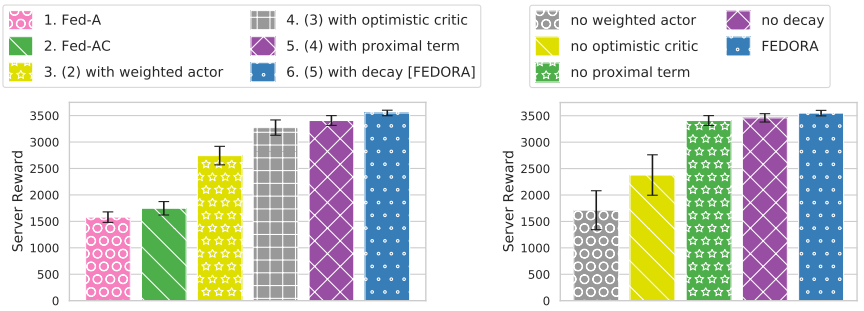
Federation Structure: We implement FEDORA over the Flower federated learning platform [2], which supports learning across devices with heterogeneous software stacks, compute capabilities, and network bandwidths. Flower manages all communication across clients and the server and permits us to implement 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. 2) 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.

B Additional Experiments

B.1 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.

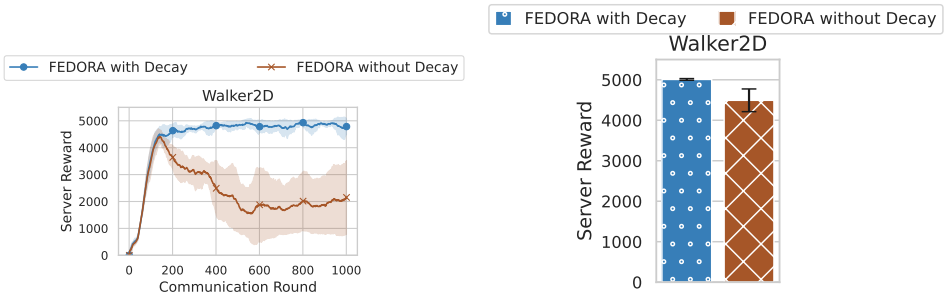
We perform an ablation study to examine the different components of our algorithm and understand their relative impacts on the performance of the federated policy. We use the experimental framework with 10 clients and the Hopper environment described in Section 4, and plot the performance of the federated 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. (8)), and an optimistic critic (Eq. (9) - (10)) 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. (11)) 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.

B.2 Ablation of Decaying mechanism on Walker Environment

We study the effect of decaying the influence of local data (5.4) in the Walker2D environment in Figure 8. Although the decaying mechanism seems to give only a small improvement in Figure 7, which pertains to experiments on the Hopper environment, we observe that it provides a significant improvement in the Walker2D environment.



(a) Training curve

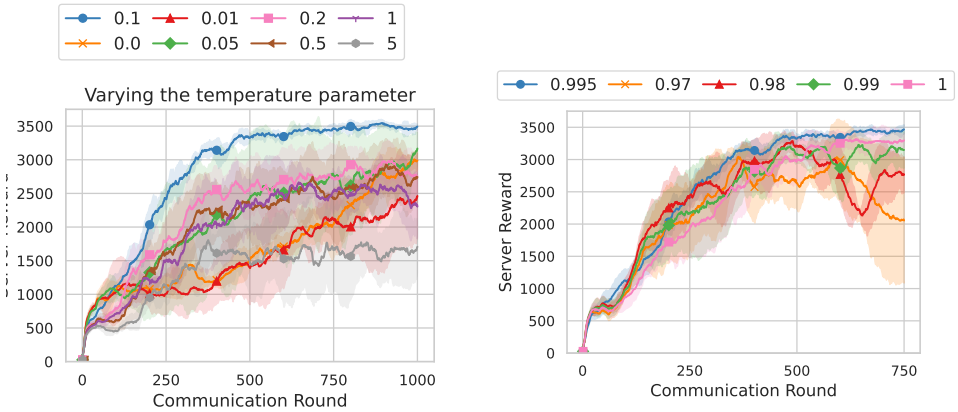
(b) Peak performance

Figure 8: Ablation study of decaying mechanism on Walker2d environment (setting similar to Fig 7).

B.3 Hypterparameter sweep

In Fig. 9a run FEDORA for different values of β, which is the temperature parameter of federation. We consider a scenario similar to Fig. 1, where we consider the Hopper-v2 environment with 10 clients having $|D_i| = 5000$ participating in federation, where 5 clients have expert data, and 5 clients have medium data. When $\beta = 0$, it boils down to a uniform weighting scheme, where the quality of data present in each client is not considered during federation. As $\beta \to \infty$ it tends to a max weighting scheme, where the federated policy is the same as an individual client's policy with the highest quality data.

In Fig. 9b we run FEDORA for different values of the decay parameter δ. The decay parameter controls the influence of the local data in $\mathcal{L}_{\text{actor}}$ by decaying the influence of $\mathcal{L}_{\text{local}}$. In Fig. 9b, we consider a scenario similar to Fig. 1, where we consider the Hopper-v2 environment with 10 clients having $|D_i| = 5000$ participating in federation, where 5 clients have expert data, and 5 clients have medium data. We run all the algorithms for 750 rounds of federation. We observe that FEDORA is robust to the variations in the values of δ.

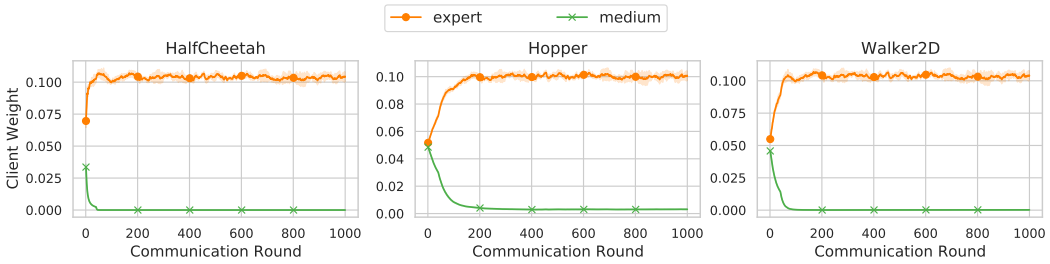


(a) Varying β

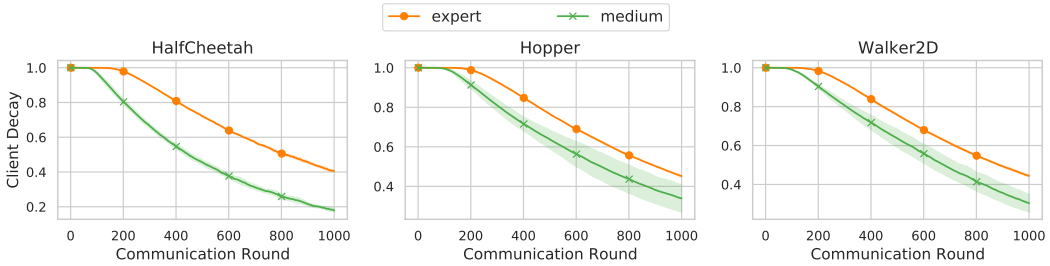
(b) Varying δ

Figure 9: Hyperparameter sweep

B.4 Analysis of Client Performance



(a) Client ratio



(b) Client decay

Figure 10: 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 6 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. 10a. 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. 10b, we plot the mean of the decay value associated with $\mathcal{L}_{\text{local}}$ across participating expert and medium dataset clients (Section 5.4). 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.

B.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 [29] 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.

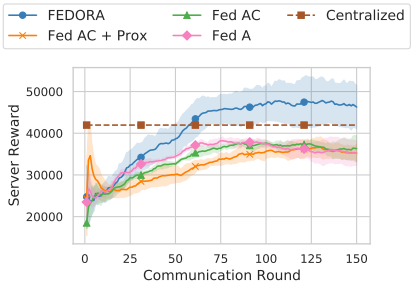


Figure 11: Evaluation of algorithms on CityLearn.

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 [24]. 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. 11. 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.

B.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.

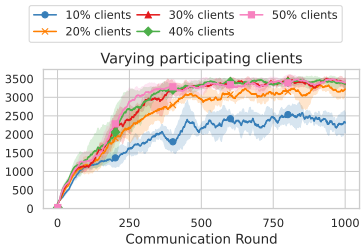


Figure 12: Effect of varying the number of participating clients in each round on FEDORA

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%.

B.7 Variable Size Datasets

In Fig. 13, we run FEDORA with clients having variable dataset sizes and compare it with FEDORA with a fixed dataset size.

FEDORA_Var_Dataset: We consider a scenario where we have 10 clients in the Hopper-v2 environment. 5 clients have the expert dataset with dataset sizes $4000, 5000, 6000, 7000, 8000$ and 5 clients have the medium dataset with dataset sizes $4000, 5000, 6000, 7000, 8000$.

FEDORA: We consider the scenario described in Fig.1, with a constant dataset size of 5000.

From Fig. 13 we can observe that FEDORA is robust can handle variable dataset sizes.

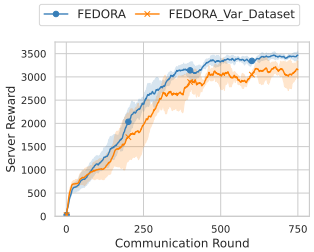


Figure 13: Variable Dataset Size

B.8 Centralized training with other Offline RL algorithms

We consider a scenario similar to the one in Fig. 3 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) [15] and Offline Reinforcement Learning with Implicit Q-Learning (IQL) [13] whose implementations are obtained from the CORL library [27]. We can observe from Fig. 14 that pooling data from different behavior policies affects both offline RL algorithms.

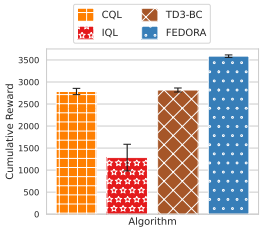


Figure 14: Comparison with different Offline RL algorithms

B.9 Data Rebalancing

We consider a scenario similar to Fig. 1, where we consider the Hopper-v2 environment with 10 clients having $|D_i| = 5000$ participating in federation, where 5 clients have expert data, and 5 clients have medium data. We compare the performance of TD3-BC, TD3-BC with data rebalancing (TD3-BC_RB) [38, 37], and FEDORA. From Fig. 15 we can notice that the addition of data rebalancing does help the performance of offline RL algorithms when data is collected using multiple behavior policies. We also notice that the performance of TD3-BC with data rebalancing does not match the performance of FEDORA. We hypothesise that this could be due to the superior weighting mechanism employed by FEDORA, and that data rebalancing cannot completely solve the distribution shift issue caused by data coming from multiple behavior policies.

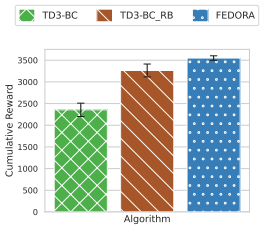


Figure 15: Comparison with Offline RL with data rebalancing

B.10 Different Weighing Mechanisms

We consider a scenario similar to Fig. 1, where we consider the Hopper-v2 environment with 10 clients having $|D_i| = 5000$ participating in federation, where 5 clients have expert data, and 5 clients have medium data. We run all the algorithms for 750 rounds of federation. We compare the performance of FEDORA with two additional baselines in which we change only the weighing mechanism of FEDORA to be based on average reward of the dataset at each client. **(1.) FEDORA_RAvg:** We combine clients based on the average reward in their dataset. We choose the weights of federation of client i, $w_i = \frac{R_i}{\sum_{k \in \mathcal{N}_t} R_k}$, where R_i corresponds to the average reward of client i's dataset and $\mathcal{N}_t$ is the set of clients participating in federation at round t. **(2.) FEDORA_RC:** We extend the weighing scheme proposed

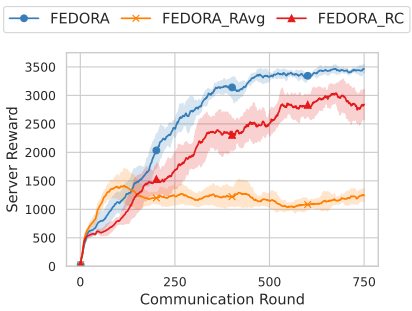


Figure 16: Comparison with different weighing mechanism based on average reward in the dataset

in [38] to the federated setting. In this scenario, we choose the weights of federation for client i, $w_i = \frac{p_i}{\sum_{k \in \mathcal{N}_t} p_k}$. Where $p_i = \frac{R_i - R_{\min}}{R_{\max} - R_{\min}}$, here R_i corresponds to the average reward of client i's dataset, $R_{\min} = \min_{i \in \mathcal{N}_t} R_i$, and $R_{\max} = \max_{i \in \mathcal{N}_t} R_i$.

From Fig. 16 we notice that FEDORA outperforms both baselines. We observe that weighing based on average reward of the datasets does not yield good results, this can be attributed to the fact that the average rewards in the dataset do not vary much. For instance, the average reward of clients with the expert dataset is 3.6, while that of the medium dataset is 3.11. Thus combining based solely on the average reward boils down to using a uniform weighing scheme. We observe that extending the approach proposed in [38] does help, but this weighing scheme is still inferior to that of FEDORA.

FEDROA's weighing scheme is superior as it combines the policies based on the performance of the learned policy from the dataset, rather than the average reward in the dataset. In other words, FEDORA weighs dataset from which better performing policy can be learnt higher.

C Details of Real-World Robot Experiments

C.1 Demonstration Data Collection

We train four behavior policies of varying levels of expertise using TRPO [26] on a custom simulator for mobile robots described in section C.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.

C.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

$$\begin{aligned} x_{t+1} &= x_t + \Delta t \; v \cos\theta_t \\ y_{t+1} &= y_t + \Delta t \; v \sin\theta_t \\ \theta_{t+1} &= \theta_t + \Delta t \; \omega, \end{aligned} \tag{12}$$

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

$$R_t = \begin{cases} +100, & \text{if } |x_t - x_g| \leq thresh \texttt{ and } |y_t - y_g| \leq thresh \\ -10, & \text{if robot outside boundary} \\ -100, & \text{if robot collides} \\ -(\texttt{c.t.e}_t^2 + \texttt{a.t.e}_t + \texttt{h.e}_t) + \sum \texttt{lidar}_t, & \text{otherwise} \end{cases} \tag{13}$$

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

$$\begin{aligned} d_t^g &= \sqrt{(x_g - x_t)^2 + (y_g - y_t)^2}, \\ \theta_t^g &= \tan^{-1}\left(\frac{y_g - y_t}{x_g - x_t}\right), \\ \texttt{c.t.e}_t &= d_t^g \sin(\theta_g - \theta_t), \\ \texttt{a.t.e}_t &= |x_g - x_t| + |y_g - y_t|, \\ \texttt{h.e}_t &= \theta_t^g - \theta_t. \end{aligned} \tag{14}$$

C.3 Mobile Robot Platform

We evaluate the trained algorithms on a Robotis TurtleBot3 Burger mobile robot [1], 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.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: [NA]

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Section 9

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be

used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: [NA]

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: See Appendix A and code `https://github.com/DesikRengarajan/FEDORA`.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example

(a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: See code `https://github.com/DesikRengarajan/FEDORA`.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: See section 6, Appendix A, and code in `https://github.com/DesikRengarajan/FEDORA`.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: See section 6.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: See Appendix A.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: [NA]

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See section 8

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Code at `https://github.com/DesikRengarajan/FEDORA` and Appendix A.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: See code at `https://github.com/DesikRengarajan/FEDORA`

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

Justification: [NA]

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.