EAMQ: Environment-based Adaptive Model Quantization on Federated Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Federated Reinforcement Learning (FRL) enables agents to collaboratively train models across distributed environments without sharing raw data. However, existing quantization methods like QuARL, ReLeQ, and VQQL struggle in environments with varying state transitions and rewards, affecting model robustness. In this paper, we introduce Environment-based Adaptive Model Quantization (EAMQ), a method that dynamically adjusts compression ratios based on environmental variability. EAMQ uses a reward-weighted sensitivity analysis to assign lower compression ratios to sensitive parameters in sparse reward environments while applying higher compression in dense reward settings. We also propose a learnable quantization technique that adapts based on a Temporal Difference (TD) loss function. Experiments show that EAMQ outperforms traditional methods across diverse environments, reducing communication and storage costs while maintaining performance, even under heterogeneous conditions.

1 Introduction

Federated Reinforcement Learning (FRL) [7] is a decentralized approach where multiple agents collaboratively train a reinforcement learning model across distributed environments without sharing raw data. FRL has been applied in real-world scenarios such as smart grid management, multi-agent large language models, and the Internet of Things (IoT) [11]. Model quantization, such as QuARL [4], ReLeQ [1], and VQQL [2], have been developed to compress models during reinforcement learning (RL) training in order to reduce the communication and storage costs. However, traditional algorithms perform badly when FRL is applied in environmental heterogeneity situations [3], because they don't consider the influence of the changing environments. Models trained in different environments have different robustness to quantization, models in some environments may be insensitive to higher compression rates, while others rely heavily on accurate parameter representations. In this work, we focus on quantizing model parameters during the training of several FRL functions [3] considering the changing environments. To the best of our knowledge, this is the first effort to apply model quantization specifically in environmental heterogeneity situations.

In this paper, we simulate the packet loss conditions in a Federated Reinforcement Learning (FRL) system and propose a novel model compression algorithm called "Environment-based Adaptive Model Quantization (EAMQ)", inspired by learnable quantization techniques from [6]. Our approach first identifies the sensitivity of model parameters during reinforcement learning training in different environments. Parameters that exhibit significant variation are classified as sensitive and assigned a lower compression ratio, preserving higher precision by quantizing from float32 to int8, while less sensitive parameters are compressed more aggressively, for instance, from float32 to int4. Additionally, we introduce a learnable quantization mechanism that adaptively adjusts the quantization range by

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

minimizing the environment-based Temporal Difference loss function which will be adaptively adjusted according to the changing environment. [9].

Our main contributions are as follows: First, we address the impact of environmental heterogeneity on model quantization by developing a compression ratio allocation strategy tailored to different environments. Second, we propose a novel learnable quantization algorithm that dynamically adjusts the quantization range, along with an environment-aware Temporal Difference loss function that accounts for both RL performance and environmental variability. Third, we adapt traditional model quantization algorithms for heterogeneous environments and compare them with our proposed EAMQ algorithm, establishing a new baseline for future research in model quantization under environmental heterogeneity. Fourth, we deployed our FRL algorithm on a real-world wireless distributed system to evaluate its performance, bridging the gap between theoretical analysis and practical application [5].

2 Method

2.1 Reward-Weighted Sensitivity Based on Environment Reward Distribution

To assign adaptive quantization compression rates based on environment reward distribution differences, we calculate a reward-weighted variance for each parameter. The goal is to assign lower compression rates (e.g., float32 to int4) for parameters that are sensitive in sparse reward environments, and higher compression rates (e.g., float32 to int8) for parameters that are less sensitive in dense reward environments. For each environment e, we first compute the average reward across all time steps:

$$R^e = \frac{1}{T}\sum_{t=1}^{T} r_t^e$$

where r_t^e is the reward at time step t in environment e, and T is the total number of time steps. Next, we define the reward sparsity factor α^e for each environment based on the inverse of the average reward:

$$\alpha^e = \frac{1}{R^e + \epsilon}$$

where ϵ is a small constant added to avoid division by zero. A higher α^e indicates a more sparse reward environment, while a lower α^e indicates a dense reward environment. Using the reward sparsity factor α^e, we compute the reward-weighted variance $\text{Var}_R(\theta_k)$ for each parameter θ_k across all environments:

$$\text{Var}_R(\theta_k) = \frac{1}{n}\sum_{e=1}^{E} \alpha^e \cdot (\theta_k^e - \mu_k)^2$$

where θ_k^e is the value of parameter θ_k in environment e, μ_k is the mean value of θ_k across all environments, and E is the total number of environments.

Finally, we assign compression rates based on the reward-weighted sensitivity. Parameters with higher reward-weighted variance are assigned lower compression rates (e.g., float32 to int4), while parameters with lower reward-weighted variance are assigned higher compression rates (e.g., float32 to int8).

2.2 Learnable model quantization

Symmetric linear quantization is a widely used data quantization technique [13], where the quantization range is centered around zero, treating positive and negative values symmetrically. In this method, the mapping between original and quantized values follows a linear relationship. However, a key limitation of traditional symmetric linear quantization is that the quantization range is predetermined before quantization. This fixed range may not be optimal for preserving model performance across all data distributions, as it may fail to adapt to the specific characteristics of the data.

In this algorithm, we propose a novel learnable linear quantization that optimizes the quantization range for each data using a loss function called the "environment-based Temporal Difference (TD) loss function," which can be adjusted based on different environments. The formula for this loss function is:

$$L_{\text{total}} = L_{\text{task}} + \lambda_{\text{env}} L_{\text{quant}} + \lambda_{\text{reg}} L_{\text{reg}} \tag{1}$$

Where L_{task} is the standard loss in reinforcement learning (e.g., TD error), L_{quant} is the difference between the original and quantized parameters using discrete cosine distance, and L_{reg} is the regularization term introduced to ensure stable quantization decisions across training iterations.

The regularization term L_{reg} is defined as:

$$L_{\text{reg}} = \sum_k \left(\theta_k^{\text{current}} - \theta_k^{\text{previous}}\right)^2$$

Where $\theta_k^{\text{current}}$ represents the parameter values in the current iteration, and $\theta_k^{\text{previous}}$ represents the parameter values from the previous iteration.

In addition, λ_{env} is a weight dynamically adjusted based on the environment's sensitivity, and λ_{reg} is the weight assigned to the regularization term.

The environment-adaptive weight $\lambda_{\text{env},t}$ is calculated as:

$$\lambda_{\text{env},t} = \alpha \cdot \left(\frac{\delta_t}{\max(\delta)}\right) + \beta \cdot \left(\frac{\Delta G}{\max(\Delta G)}\right) \tag{2}$$

Where δ_t is the TD error at step t, representing the difference between predicted and actual rewards; ΔG is the cumulative reward drop rate across environments; and α and β are hyperparameters to balance between TD error and cumulative reward drop.

3 Experiments

3.1 Experiment Setting

In this experiment, we will first use tabular environments to verify the result of our EAMQ algorithm on quantifying the model in PAvg and QAvg. Next, we evaluate the algorithm's effectiveness in deep reinforcement learning tasks, specifically in DQNAvg. The functions and environment configurations are consistent with those used in [3]. We compare our results against the following baselines: QuARL [4], ReLeQ [1], QFL [12], Fixar [10], FedDQ [8], and VQQL [2]. All model quantization algorithms are evaluated under the same compression ratios for a fair comparison. The original models are in float32 format, and we apply different quantization levels: int16 (50% compression), int8 (75% compression), int6 (81.25% compression), and int4 (87.5% compression). For the 81.25% compression ratio, our EAMQ method quantizes half of the data to int8 and the other half to int4, while the other algorithms quantize all data directly to int6.

In our experiments, we apply quantization to either the Q-table (QAvg) or the policy function (PAvg), using two environments: RandomMDPs and WindyCliffs. To simulate varying degrees of heterogeneity across environments, we introduce the parameter κ. As κ increases, the environments become more diverse, reflecting greater dissimilarity in state transition probabilities and reward distributions. Tables 7, 7, 4, and 6 demonstrate that, across different environments and compression ratios, our quantization algorithm consistently outperforms traditional model quantization methods.

For the deep reinforcement learning environment, we quantify the Deep Q-Network (DQN) in two scenarios: Acrobot and CartPole. The performance of DQNAvg is evaluated over 20 episodes. The curve illustrates the generation objective value, which represents the averaged performance across 10 environments with newly generated state transitions. A higher objective value indicates better performance. Our results demonstrate that EAMQ outperforms other model quantization algorithms in terms of overall performance across these environments.

3.2 Ablation Study and Analysis

In this section, we conducted ablation experiments to evaluate the effectiveness of our proposed algorithms. Figures 2a and 2b demonstrate that at a compression ratio of 81.25%, applying a

Table 1: Q-Avg over RandomMDPs for different compression ratios under $\kappa = 0.4$ and $\kappa = 0.6$ larger κ indicates environments with larger environment heterogeneity. The number stands for the average cumulative reward of the algorithm, higher is better

	$\kappa = 0.4$				$\kappa = 0.6$			
Compression Rate	50%	75%	81.25%	87.5%	50%	75%	81.25%	87.5%
QPI	27.88	25.99	23.23	20.89	27.48	25.29	23.03	20.79
QuARL	29.01	26.17	25.33	21.34	28.01	26.07	25.33	21.34
ReLeA	26.58	25.52	24.12	22.99	26.58	24.52	23.02	21.19
VAQL	28.89	27.05	26.33	23.22	27.88	26.15	25.47	23.12
VOQL	29.87	28.57	27.91	23.21	28.86	27.47	26.92	25.25
EAMQ	34.05	33.80	32.15	30.82	34.14	32.80	31.80	30.80

Table 2: P-Avg over WindyCliffs at a compression ratio of 81.25% under $\kappa = 0.6$ and $\kappa = 0.8$ in an FRL system with a high packet loss wireless network.

	$\kappa = 0.6$				$\kappa = 0.8$			
Loss package Rate	50%	70%	80%	90%	50%	70%	80%	90%
No quantization	106.52	95.51	93.11	90.08	10.58	10.72	9.21	9.19
ReLeA	116.52	113.51	104.13	102.05	15.58	14.72	11.91	10.21
VAQL	122.77	116.05	105.31	103.01	28.01	25.95	24.47	23.02
VOQL	121.17	114.51	109.98	105.02	20.06	17.07	14.82	14.15
EAMQ	135.57	133.05	127.14	121.87	31.05	27.28	25.19	23.48

Table 3: P-Avg over RandomMDPs for different compression ratios under $\kappa = 0.4$ and $\kappa = 0.6$ larger κ indicates environments with larger environment heterogeneity. The number stands for the average cumulative reward of the algorithm, higher is better

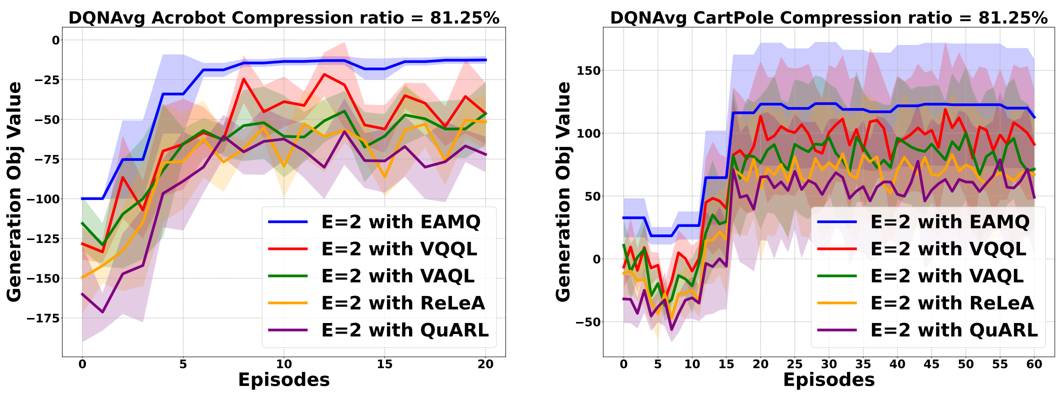
	$\kappa = 0.4$				$\kappa = 0.6$			
Compression Rate	50%	75%	81.25%	87.5%	50%	75%	81.25%	87.5%
QPI	25.18	24.89	22.21	20.81	26.48	22.29	21.03	19.79
QuARL	26.21	25.16	24.32	21.24	28.01	26.07	25.03	21.04
ReLeA	26.52	25.51	24.11	23.98	25.58	24.82	23.00	20.19
VAQL	27.82	26.05	25.39	24.21	28.01	25.95	24.47	23.02
VOQL	28.17	27.51	26.98	25.22	28.06	27.07	26.82	25.15
EAMQ	33.58	32.07	30.15	29.83	32.14	31.81	30.79	29.80

Table 4: Q-Avg over WindyCliffs for different compression ratios under $\kappa = 0.6$ and $\kappa = 0.8$ larger κ indicates environments with larger environment heterogeneity. The number stands for the average cumulative reward of the algorithm, higher is better

	$\kappa = 0.6$				$\kappa = 0.8$			
Compression Rate	50%	75%	81.25%	87.5%	50%	75%	81.25%	87.5%
QPI	125.18	124.89	122.21	120.81	126.48	122.29	121.03	119.79
QuARL	126.21	125.16	124.32	121.24	128.01	126.07	125.03	118.01
ReLeA	126.52	125.41	124.01	123.68	125.58	124.82	123.02	119.09
VAQL	127.81	126.15	125.09	123.19	128.01	125.95	124.47	123.02
VOQL	128.87	127.53	126.08	125.02	128.06	127.07	126.82	125.05
EAMQ	133.96	132.07	130.15	129.83	133.65	131.81	130.79	129.81

Table 5: P-Avg over WindyCliffs for different compression ratios under $\kappa = 0.6$ and $\kappa = 0.8$ larger κ indicates environments with larger environment heterogeneity. The number stands for the average cumulative reward of the algorithm, higher is better

	$\kappa = 0.6$				$\kappa = 0.8$			
Compression Rate	50%	75%	81.25%	87.5%	50%	75%	81.25%	87.5%
QPI	125.18	124.89	122.21	120.01	126.08	122.19	121.93	119.39
QuARL	126.21	125.16	124.32	21.44	128.01	126.07	125.03	121.04
ReLeA	126.52	125.51	124.11	123.08	125.58	124.72	123.91	121.29
VAQL	127.82	126.05	125.39	124.01	28.01	25.95	24.47	23.02
VOQL	128.17	127.51	126.98	125.12	28.06	27.07	26.82	25.15
EAMQ	139.58	132.07	131.15	131.89	32.14	31.81	30.79	29.80



(a) DQNAvg in Acrobot environment

(b) DQNAvg in CartPole environment

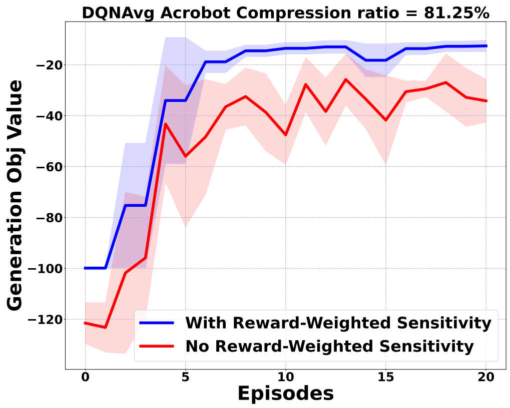
Figure 1: Performance of DQNAvg in different environments. The y-axis shows the cumulative reward of the agents. E=2 means the agents' models are averaged every 2 episodes. Different colors show the performance of the algorithm after quantification by different quantization algorithms, the standard error (a measure of variability or uncertainty) is depicted as a shadow around the line, with the shadow width being 1.65 times the standard error.

uniform compression ratio across all data yields inferior results compared to utilizing Reward-Weighted Sensitivity for adaptive compression allocation, as shown in Figure 7, our learnable model quantization method significantly outperforms traditional symmetric linear quantization.

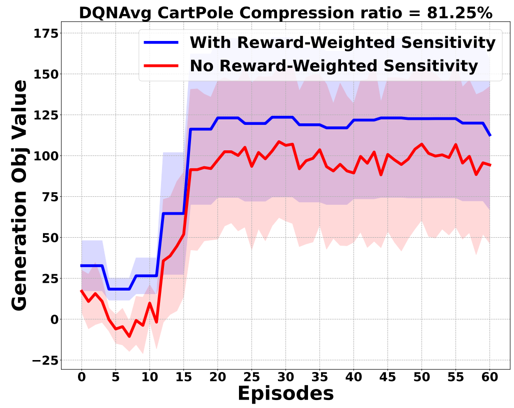
Table 2 demonstrates that FRL performance declines in high packet loss networks due to information loss during communication, a significant challenge in real-world IoT systems [10]. We implemented our algorithm in a real wireless distributed system, controlling the packet loss ratio to simulate communication loss between agents in different environments. Results show that our model quantization algorithm enhances FRL robustness in poor network conditions, highlighting both the effectiveness and efficiency of our approach.

4 Conclusion and Future Work

In this paper, we introduced Environment-based Adaptive Model Quantization (EAMQ) to tackle the challenges of model quantization in heterogeneous environments within Federated Reinforcement Learning (FRL). EAMQ uses reward-weighted sensitivity and a learnable quantization method to adapt compression rates based on the environment, ensuring strong performance across different scenarios. Our experiments show that EAMQ outperforms traditional methods, reducing communication costs while preserving or improving model effectiveness. We hope our algorithm will encourage further exploration of Federated Reinforcement Learning model quantization in heterogeneous environments, as a promising and innovative direction for advancing model compression techniques.



(a) DQNAvg in Acrobot environment



(b) DQNAvg in CartPole environment

Figure 2: Performance of DQNAvg in different environments. We compare the performance in the same compression ratio between we not using Reward-Weighted Sensitivity analysis and using Reward-Weighted Sensitivity analysis

Table 6: P-Avg over WindyCliffs for different compression ratios under $\kappa = 0.6$ and $\kappa = 0.8$ larger κ indicates environments with larger environment heterogeneity. The number stands for the average cumulative reward of the algorithm, higher is better

	$\kappa = 0.6$				$\kappa = 0.8$			
Compression Rate	50%	75%	81.25%	87.5%	50%	75%	81.25%	87.5%
QPI	125.18	124.89	122.21	120.01	26.08	22.19	21.93	19.39
QuARL	126.21	125.16	124.32	21.44	26.01	26.07	25.03	21.04
ReLeA	126.52	125.51	124.11	123.08	25.58	24.72	23.91	21.29
VAQL	127.82	126.05	125.39	124.01	28.01	25.95	24.47	23.02
VOQL	128.17	127.51	126.98	125.12	28.06	27.07	26.82	25.15
EAMQ	139.58	132.07	131.15	131.89	32.14	31.81	30.79	29.80

Table 7: Q-Avg and P-Avg over RandomMDPs for different compression ratios under $\kappa = 0.4$ and $\kappa = 0.6$, the result we use Learnable model quantization or directly using Symmetric linear quantization

	$\kappa = 0.4$				$\kappa = 0.6$			
Compression Rate(Q-Avg)	50%	75%	81.25%	87.5%	50%	75%	81.25%	87.5%
Linear quantization	26.17	26.07	25.01	21.21	25.86	24.47	22.02	21.15
EAMQ(ours)	33.58	32.07	30.15	29.83	32.14	31.81	30.79	29.80

	$\kappa = 0.4$				$\kappa = 0.6$			
Compression Rate(P-Avg)	50%	75%	81.25%	87.5%	50%	75%	81.25%	87.5%
Linear quantization	29.17	28.07	28.01	22.21	27.86	26.47	26.02	24.15
EAMQ(ours)	34.05	33.80	32.15	30.82	34.14	32.80	31.80	30.80

References

[1] Ahmed Elthakeb, Prannoy Pilligundla, FatemehSadat Mireshghallah, Amir Yazdanbakhsh, Sicuan Gao, and Hadi Esmaeilzadeh. Releq: an automatic reinforcement learning approach for deep quantization of neural networks. In *NeurIPS ML for Systems workshop, 2018*, 2019.

[2] Fernando Fernández and Daniel Borrajo. Vqql. applying vector quantization to reinforcement learning. In *RoboCup-99: Robot Soccer World Cup III 3*, pages 292–303. Springer, 2000.

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

[4] Srivatsan Krishnan, Sharad Chitlangia, Maximilian Lam, Zishen Wan, Aleksandra Faust, and Vijay Janapa Reddi. Quantized reinforcement learning (quarl). *arXiv preprint arXiv:1910.01055*, 2019.

[5] Euclides Carlos Pinto Neto, Somayeh Sadeghi, Xichen Zhang, and Sajjad Dadkhah. Federated reinforcement learning in iot: applications, opportunities and open challenges. *Applied Sciences*, 13(11):6497, 2023.

[6] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*, 2018.

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

[8] Linping Qu, Shenghui Song, and Chi-Ying Tsui. Feddq: Communication-efficient federated learning with descending quantization. In *GLOBECOM 2022-2022 IEEE Global Communications Conference*, pages 281–286. IEEE, 2022.

[9] Limin Wang, Zhan Tong, Bin Ji, and Gangshan Wu. Tdn: Temporal difference networks for efficient action recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1895–1904, 2021.

[10] Je Yang, Seongmin Hong, and Joo-Young Kim. Fixar: A fixed-point deep reinforcement learning platform with quantization-aware training and adaptive parallelism. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 259–264. IEEE, 2021.

[11] Rui Ye, Wenhao Wang, Jingyi Chai, Dihan Li, Zexi Li, Yinda Xu, Yaxin Du, Yanfeng Wang, and Siheng Chen. Openfedllm: Training large language models on decentralized private data via federated learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6137–6147, 2024.

[12] Cui Zhang, Wenjun Zhang, Qiong Wu, Pingyi Fan, Qiang Fan, Jiangzhou Wang, and Khaled B Letaief. Distributed deep reinforcement learning based gradient quantization for federated learning enabled vehicle edge computing. *IEEE Internet of Things Journal*, 2024.

[13] Xiandong Zhao, Ying Wang, Xuyi Cai, Cheng Liu, and Lei Zhang. Linear symmetric quantization of neural networks for low-precision integer hardware. In *International Conference on Learning Representations*, 2020.

A Algorithm

B Detail of Q-Avg function

B.1 Q-Avg Algorithm in Federated Reinforcement Learning

In this paper, we apply the Q-Avg algorithm, a variant of the Q-learning algorithm adapted for Federated Reinforcement Learning (FRL). Q-Avg is designed to address the challenges of training multiple agents across distributed and heterogeneous environments by periodically averaging the Q-value updates from each agent. This approach aims to reduce communication costs and improve the overall performance of the system in scenarios with varying environment dynamics.

Algorithm 1 Gradient Descent for Quantization Range Optimization with Regularization and Environment Sensitivity

Require: Initialized quantization ranges $S = \{S_1, S_2, \ldots, S_m\}$, learning rate η, number of iterations T, λ_{reg}, λ_{env}.

Initialize parameters S_k for each parameter k
for each iteration $t = 1 \to T$ **do**
    Compute task loss L_{task}
    Quantize parameters using current quantization ranges S_k
    Compute quantization loss L_{quant}
    Compute regularization term:

$$L_{\text{reg}} = \sum_k \left(\theta_k^{\text{current}} - \theta_k^{\text{previous}}\right)^2$$

    Compute total loss:

$$L_{\text{total}} = L_{\text{task}} + \lambda_{\text{env}} L_{\text{quant}} + \lambda_{\text{reg}} L_{\text{reg}}$$

    for each parameter S_k **do**
        Compute gradient $\frac{\partial L_{\text{total}}}{\partial S_k}$
        Update quantization range:

$$S_k \leftarrow S_k - \eta \frac{\partial L_{\text{total}}}{\partial S_k}$$

    end for
end for
Return optimized quantization ranges S

Q-Value Averaging: In each environment, agents independently learn Q-values by interacting with the environment. After a set number of episodes, the Q-value updates from each agent are transmitted to a central server where the **Q-Avg** algorithm computes the averaged Q-values across all participating agents. This ensures that all agents benefit from each other's learning experiences, even in environments with heterogeneous state transitions and reward functions. The Q-Avg formula is given by:

$$Q_{\text{avg}}(s, a) = \frac{1}{N} \sum_{i=1}^{N} Q_i(s, a) \tag{3}$$

where N is the number of agents, $Q_i(s, a)$ is the Q-value of agent i for state s and action a, and $Q_{\text{avg}}(s, a)$ represents the averaged Q-value after aggregation.

Handling Heterogeneous Environments: One of the key advantages of Q-Avg is its ability to handle heterogeneous environments. In standard reinforcement learning, models are trained in homogeneous environments, but in FRL, agents operate in environments with different state transition dynamics and reward structures. To address this, Q-Avg adapts by incorporating the agents' experiences across diverse environments. This allows agents to generalize better to new environments and ensures robustness in learning.

Communication Efficiency: A major challenge in FRL is the communication overhead due to frequent parameter updates. Q-Avg mitigates this by reducing the frequency of communication between agents and the server, only averaging the Q-values after a predefined number of episodes. By doing so, Q-Avg minimizes the communication costs while still benefiting from collaborative learning across agents.

Algorithm Overview: The overall steps of the Q-Avg algorithm can be summarized as follows:

1. **Initialization:** Each agent initializes its Q-table $Q_i(s, a)$ and begins interacting with its local environment.

2. **Learning:** Each agent updates its Q-values using the standard Q-learning update rule:

$$Q_i(s,a) \leftarrow Q_i(s,a) + \alpha \left(r + \gamma \max_{a'} Q_i(s', a') - Q_i(s,a) \right) \tag{4}$$

where α is the learning rate, r is the reward, and γ is the discount factor.

3. **Averaging:** After a fixed number of episodes, each agent sends its updated Q-values to the central server, which computes the average Q-values:

$$Q_{\text{avg}}(s,a) = \frac{1}{N} \sum_{i=1}^{N} Q_i(s,a) \tag{5}$$

4. **Update:** The central server sends the averaged Q-values $Q_{\text{avg}}(s,a)$ back to the agents, which update their Q-tables accordingly.
5. **Reiteration:** The process continues, with agents periodically sending their updated Q-values for averaging and receiving the averaged Q-values from the server.

Advantages

- **Collaborative Learning:** Q-Avg enables agents to leverage the experiences of other agents, improving overall learning performance in federated environments.
- **Scalability:** The algorithm scales efficiently with the number of agents, as the Q-value averaging process is simple and communication is minimized.
- **Adaptability:** Q-Avg is well-suited to handle heterogeneous environments, making it robust in real-world scenarios where environment dynamics vary between agents.

Overall, Q-Avg offers a simple yet effective solution for federated Q-learning, particularly in scenarios where communication costs and environment diversity are key challenges.

C Detail of P-Avg function

C.1 P-Avg Algorithm in Federated Reinforcement Learning

In this paper, we utilize the **P-Avg** algorithm, a federated averaging approach specifically designed for policy-based reinforcement learning in distributed environments. P-Avg focuses on averaging policy parameters across multiple agents, allowing them to collaboratively improve their policies while interacting with heterogeneous environments. This method is particularly useful for handling policy gradients in Federated Reinforcement Learning (FRL), where agents work in diverse environments and need to share their policy updates efficiently.

Policy Averaging: The core idea of P-Avg is to periodically average the policy parameters from each agent to form a global policy. Each agent learns its local policy by interacting with its environment, and after a set number of episodes, the policies are shared with a central server for averaging. The **P-Avg** update rule is as follows:

$$\pi_{\text{avg}} = \frac{1}{N} \sum_{i=1}^{N} \pi_i \tag{6}$$

where N is the number of agents, π_i represents the policy parameters of agent i, and π_{avg} is the averaged policy after aggregation. This global policy is then distributed back to the agents for further updates.

Handling Environmental Heterogeneity: P-Avg is particularly effective in **heterogeneous environments**, where each agent operates in a different environment with its own dynamics and reward structures. Since each agent learns a policy suited to its local environment, averaging these policies helps agents generalize across different environments. This approach ensures that all agents benefit from each other's experiences, improving the robustness of the global policy.

Policy Gradient Updates: In P-Avg, each agent updates its local policy parameters using the **policy gradient** method. For each agent i, the policy is updated using the following rule:

$$\theta_i \leftarrow \theta_i + \alpha \nabla_{\theta_i} J(\theta_i) \tag{7}$$

where θ_i are the policy parameters for agent i, α is the learning rate, and $\nabla_{\theta_i} J(\theta_i)$ is the policy gradient computed based on the agent's experience. After a fixed number of updates, the local policies are shared and averaged, as described earlier.

Algorithm Overview: The overall process of P-Avg can be summarized as follows:

1. **Initialization:** Each agent initializes its policy parameters π_i and begins interacting with its local environment.
2. **Policy Update:** Each agent updates its policy using the policy gradient method:
$$\theta_i \leftarrow \theta_i + \alpha \nabla_{\theta_i} J(\theta_i)$$
where θ_i are the policy parameters, and $J(\theta_i)$ is the objective function.
3. **Averaging:** After a predefined number of episodes, each agent sends its updated policy parameters π_i to the central server, which computes the averaged policy:
$$\pi_{\text{avg}} = \frac{1}{N} \sum_{i=1}^{N} \pi_i$$
4. **Update:** The central server sends the averaged policy π_{avg} back to the agents, which update their local policies accordingly.
5. **Reiteration:** The process repeats, with agents periodically sending their policy updates for aggregation and receiving the averaged policy from the server.

Communication Efficiency: Like Q-Avg, P-Avg reduces communication costs by minimizing the frequency of parameter exchanges between agents and the server. Instead of continuously transmitting policy updates, agents only communicate their policies after a set number of episodes, reducing the overall communication overhead in large-scale distributed systems.

Advantages

- **Collaborative Learning:** P-Avg enables agents to share and combine their policies, leveraging the collective knowledge from diverse environments.
- **Adaptability to Heterogeneous Environments:** By averaging policies across agents working in different environments, P-Avg improves the generalization of policies to unseen or diverse conditions.
- **Scalability:** The algorithm scales efficiently with the number of agents, as policy averaging is computationally lightweight and reduces the need for frequent communication.

Overall, P-Avg provides a scalable and efficient solution for policy-based reinforcement learning in federated settings, particularly when agents operate in environments with varying dynamics and reward structures.

D Detail of DQNAvg function

D.1 DQNAvg Algorithm in Federated Reinforcement Learning

The **DQNAvg** algorithm is an adaptation of the standard Deep Q-Network (DQN) for Federated Reinforcement Learning (FRL) environments, where multiple agents learn independently in distributed environments and periodically share their model parameters with a central server for aggregation. DQNAvg is particularly useful in continuous action spaces, where learning a robust policy across heterogeneous environments is crucial for improving performance while reducing communication costs.

Deep Q-Network (DQN): DQN is a model-free reinforcement learning algorithm where a neural network is used to approximate the Q-values $Q(s, a)$ for each state-action pair. The network is trained to minimize the difference between the predicted Q-values and the target Q-values, which are based on the Bellman equation:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s', a') - Q(s,a) \right) \tag{8}$$

where α is the learning rate, γ is the discount factor, r is the reward, and s', a' represent the next state and action, respectively.

Averaging Q-Networks: In DQNAvg, each agent trains its own local Q-network based on its interactions with the environment. After a certain number of episodes, the local Q-networks are sent to the central server, where they are aggregated to form a global Q-network. This global Q-network is then distributed back to the agents for further training, ensuring that agents benefit from the experiences of others.

The aggregation of Q-networks follows a simple averaging scheme:

$$Q_{\text{avg}}(s,a) = \frac{1}{N} \sum_{i=1}^{N} Q_i(s,a) \tag{9}$$

where N is the number of agents, $Q_i(s, a)$ represents the Q-values of agent i, and $Q_{\text{avg}}(s, a)$ is the averaged Q-value for each state-action pair after aggregation.

Handling Environmental Heterogeneity: DQNAvg is designed to handle **heterogeneous environments**, where each agent operates in an environment with different state-transition dynamics and reward functions. By averaging the Q-values across agents, DQNAvg ensures that the global Q-network reflects experiences from diverse environments, improving the generalization of policies to new and unseen conditions.

The key advantage of this approach is that agents can learn robust policies even in environments with varying dynamics, as the averaging process incorporates knowledge from multiple sources.

Experience Replay: Each agent in DQNAvg uses an **experience replay buffer** to store past transitions, which are sampled randomly to break the correlation between consecutive experiences. The Q-network is updated using mini-batches of experiences from the replay buffer, ensuring more stable and efficient learning.

Algorithm Overview: The main steps of the DQNAvg algorithm can be summarized as follows:

1. **Initialization:** Each agent initializes its Q-network Q_i and its experience replay buffer.
2. **Learning:** Each agent interacts with its local environment, updates its Q-network using the DQN update rule, and stores experiences in the replay buffer:

$$Q_i(s,a) \leftarrow Q_i(s,a) + \alpha \left(r + \gamma \max_{a'} Q_i(s', a') - Q_i(s,a) \right)$$

3. **Averaging:** After a fixed number of episodes, each agent sends its Q-network Q_i to the central server. The server computes the average Q-network as:

$$Q_{\text{avg}}(s,a) = \frac{1}{N} \sum_{i=1}^{N} Q_i(s,a)$$

4. **Update:** The averaged Q-network Q_{avg} is sent back to all agents, which update their local Q-networks accordingly.
5. **Reiteration:** The process continues with agents periodically sending their Q-networks for aggregation and receiving the averaged Q-network for further training.

Communication Efficiency: To reduce communication overhead, DQNAvg only averages Q-networks after a predefined number of episodes. This minimizes the frequency of model transmissions, significantly lowering communication costs in federated settings, especially when applied to large-scale environments.

Advantages:

- **Collaborative Learning:** DQNAvg allows agents to leverage the experiences of others, enabling faster and more robust learning in distributed environments.
- **Adaptability to Heterogeneous Environments:** By averaging Q-values across diverse environments, DQNAvg ensures that the global model can generalize to a wide range of conditions.
- **Reduced Communication Costs:** The periodic averaging of Q-networks ensures that communication is minimized, making DQNAvg well-suited for large-scale federated learning applications.

Overall, DQNAvg extends the traditional DQN approach to a federated learning framework, allowing multiple agents to collaborate and learn efficiently across diverse environments while reducing communication costs and improving policy generalization.

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: **[TODO]**

   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: **[TODO]**

   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 the 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 addressing 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: [Yes]

Justification: **[TODO]**

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 the 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: **[TODO]**

   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: **[TODO]**Code will be available after acceptance

Guidelines:

- The answer NA means that the 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, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: **[TODO]**

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: **[TODO]**

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: **[TODO]**

   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: **[TODO]**

   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: **[TODO]**

    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: [Yes]

    Justification: **[TODO]**

    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: **[TODO]**

    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: **[TODO]**

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: **[TODO]**

    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: **[TODO]**

    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.