Learning From Your Peers: Incorporating Critic Feedback Signals in Multi-Agent Reinforcement Learning

Yubin Hu, Chirag Sharma

December 2022

Abstract

In this paper, we develop two methods of incorporating peer feedback in a multi-agent Soft Actor-Critic system, which is a type of deep reinforcement learning algorithm. The goal is to explore the notion of peer-influenced learning and communication in the context of actor-critic methods. Our methods are tested using the HalfCheetah-v4 environment in OpenAI's gym, and the results show that incorporating peer feedback can lead to faster training and better performance.

The first method involves training a critic network for each agent, which takes in feedback from the other critic networks in the system as inputs to predict Q-values. This allows each agent to learn from the predictions of its peers.

The second method seeks to allow the transmission of a compressed latent layer between peers to enrich the communication process. It allocates the task of message generation to the sender's network and decoding to the receiver's network by adding extra outputs and inputs to these networks. Both networks are trained when the receiver's network is updated.

We compare the performance of our proposed methods to that of a baseline single-agent Soft Actor-Critic model, and find that incorporating peer feedback leads to faster training and better performance for the second method though improvements were unclear for the first method. Our work contributes to the broader study of ensemble-based and multi-agent reinforcement learning, as it is the first to explore peer-influenced reinforcement learning in which agents performing the same task take in feedback from other "peers" as contextual inputs.

Our code could be found at https://github.com/chirag-sharma-00/CS285-Project

1 Introduction

When learning to perform a complex task for which existing knowledge is limited, humans benefit significantly from having a peer group. They can often better navigate unfamiliar topics or environments by taking in their peers' thoughts and actions. More generally, communication between agents learning to perform the same task can lead to benefits due to knowledge or opinion sharing, which expands the learner's horizon beyond their own experiences and allows for shortcuts in learning. For example, for tasks in which there is a large scope for exploration, one would expect that a multi-agent system in which each agent explores a different section of the state space and is able to communicate their learnings with the other agents would perform better than a setting in which each agent's performance is independent of the others.

We seek to explore this notion of peer-influenced learning and communication in the context of off-policy actor-critic methods in deep reinforcement learning (RL). In RL, having agents that are trained to optimize for the same task communicate with each other could potentially lead to faster and better results due to knowledge transfer. In this project, we develop two methods of incorporating this peer feedback in a multi-agent Soft Actor-Critic system and run experiments in which each agent is trained to maximize rewards on the HalfCheetah-v4 environment in OpenAI's gym.

2 Background and Preliminaries

This work falls within the broader study of ensemble-based and multi-agent RL. Existing studies show that simple ensemble methods often yield better results for the ensemble than single agents [WH08] but these typically involve simple aggregation of agent outputs without any stronger notion of communication. Some work in multi-agent RL has explored the effect of collaborative agents trained on the same task, but this has been restricted to communication between actors in a distributed Actor-Critic scheme in which the models are collectively trained to optimize the average of the values predicted by the critics [PP10] and a distributed tabular Q-learning schemse, where each network's Q-value updates are influenced linearly by their deviation from the other networks' predicted Q-values [KMP13].

As far as we know, our project is the first to explore a distributed off-policy RL setting in which the training of different agents includes critic networks taking in feedback from other critics in the system as inputs to predict Q-values.

We focus on policy learning in continuous action spaces (our experiments are run on the HalfCheetah-v4 environment in OpenAI's gym). We consider a Markov decision process (MDP) with continuous state space $\mathcal{S}$ and action space $\mathcal{A}$. Each transition from taking action $\mathbf{a}_t \in \mathcal{A}$ in state $\mathbf{s}_t \in \mathcal{S}$ results in a bounded reward $r(\mathbf{s}_t, \mathbf{a}_t)$. Our policy is then a conditional distribution $\pi(\mathbf{a}_t|\mathbf{s}_t)$. We consider only neural-network policies in this project, so the policy for agent i in an N-agent system is π_{ϕ_i}, where ϕ_i is the parameter vector for the neural network representing the policy. The policy is trained on a replay buffer $\mathcal{D}$ by sampling trajectories $\{(\mathbf{s}_t, \mathbf{a}_t, r_t)\}$ from it. The same policy is also used to collect and add new trajectories to the buffer in-between training phases.

In this work, we incorporate peer communication in the state-of-the-art RL algorithm Soft Actor-Critic (SAC), which trains a critic network $Q_{\theta_i}(\mathbf{s}_t, \mathbf{a}_t)$ in addition to the policy. SAC augments the standard RL maximization objective $\sum_t \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\mathcal{D}}[r(\mathbf{s}_t, \mathbf{a}_t)]$ with an expected entropy term $\mathbb{E}_{\mathbf{s}_t\sim\mathcal{D}}[\mathcal{H}(\pi(\cdot|\mathbf{s}_t)]$ to encourage better exploration and multi-modality. Further details about the SAC setup used in this project are provided in the next section.

3 Multi-Agent Critic Communication Approach

Our general approach towards incorporating critic feedback sharing between agents in a system involves setting up an architecture consisting of multiple vanilla SAC agents, which are all trained to optimize on the same task. The actor/policy networks π_{ϕ_i} with parameters ϕ_i are trained as usual by minimizing the expected KL divergence [Haa+18]

$$J_{\pi_{\phi_i}}(\phi_i) = \mathbb{E}_{\mathbf{s}_t\sim\mathcal{D}}\left[D_{KL}\left(\pi_{\phi_i}(\cdot|\mathbf{s}_t)\middle\|\frac{\exp(Q_{\theta_i}(\mathbf{s}_t,\cdot))}{Z_{\theta_i}(\mathbf{s}_t)}\right)\right] \tag{1}$$

which after applying the reparametrization trick with some input Gaussian noise vector ϵ_t, becomes

$$J_{\pi_{\phi_i}}(\phi_i) = \mathbb{E}_{\mathbf{s}_t\sim\mathcal{D},\epsilon_t\sim\mathcal{N}}[\log \pi_{\phi_i}(f_{\phi_i}(\epsilon_t;\mathbf{s}_t)|\mathbf{s}_t) - Q_{\theta_i}(\mathbf{s}_t, f_{\phi_i}(\epsilon_t;\mathbf{s}_t))] \tag{2}$$

where the optimization is over the neural network function f_{ϕ_i}.

However, each agent's critic network Q_{θ_i} undergoes a slight modification from the vanilla SAC training algorithm to enable communication between all of the critics in the system. We explore two different communication methods as detailed below.

3.1 Method 1: Incorporating peer critic feedback via value signals

In the first method, we set up a multi-agent system in which each critic network Q_{θ_i} is trained via the standard Bellman residual minimization objective but we add an additional input α_t to the critic network so that the objective is now

$$J_{Q_{\theta_i}}(\theta_i) = \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \mathcal{D}} \left[\frac{1}{2} \left(Q_{\theta_i}(\mathbf{s}_t, \mathbf{a}_t, \alpha_t) - \hat{Q}(\mathbf{s}_t, \mathbf{a}_t) \right)^2 \right] \tag{3}$$

where the target is still the same as the vanilla entropy-regularized SAC target [Haa+18]

$$\hat{Q}(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) \sim \mathcal{D}} \left[Q_{\theta_i}(\mathbf{s}_t, \mathbf{a}_t) - \log \pi_{\phi_i}(\mathbf{a}_{t+1} | \mathbf{s}_{t+1}) \right] \tag{4}$$

Note that in practice, we use the double-Q trick with 2 Q-networks and 2 target networks to improve the stability of the optimization procedure.

The key difference here is that the additional input α_t is "advice" that the current critic receives from the other critics in the system to help influence its training step update. In particular, the advice α_t is obtained via another neural network encoder f_{ψ_i}, called the "advice network", that takes as input the Q-values predicted by the other critics in the system (hence the terminology "feedback via value signals" for this method). So in a system of N agents, during the training step update on a specific sample $(\mathbf{s}_t, \mathbf{a}_t)$ while training critic Q_{θ_i}, we have

$$\alpha_t = f_{\psi_i}(\{Q_{\theta_j}(\mathbf{s}_t, \mathbf{a}_t) : j \neq i\}) \tag{5}$$

The choice of the dimension of α_t as well as architecture choices for the encoder f_{ψ_i} are hyperparameters that we explore variations on in our experiments.

Furthermore, we introduce another hyperparameter ϵ, which controls how often the critics in the system have to work by themselves during training. Specifically, while training critic Q_{θ_i} on any sample, α_t is set according to Equation 5 with probability $1-\epsilon$ and with probability ϵ, α_t is set to 0. This way, ϵ allows us to control the degree of communication in the system. Moreover, during evaluation, we always set α_t to be 0, to enforce the constraint that critics only receive feedback while training but perform the task independently during evaluation.

3.2 Method 2: Incorporating peer critic feedback via compressed critic state

We propose method 2 for enriching communication between agents in a peer reinforcement learning setting. Specifically, we augment the value signals exchanged between agents with custom messages generated by the sender's critic network. This is achieved by adding additional heads to the network and expanding its output dimensionality from 1 to $1 + d_m$, where d_m represents the dimension of the message vector α_t.

To ensure that the messages are informative, we do not detach them from the sender's network during training. This allows gradients to propagate through the messages to the sender's networks, allowing for learning to give better advice. Additionally, we randomly select one of the other critic networks to generate the advice message, encouraging diversity in the messages and promoting the formation of a "convention" among all agents.

The modified advice-generating mechanism is formulated as follows:

$$\begin{bmatrix} q \\ \alpha_t \end{bmatrix} = Q_{\theta_j}(\mathbf{s}_t, \mathbf{a}_t), j \neq i \tag{6}$$

where Q_{θ_j} represents the critic network of agent j and j is randomly selected among all agents. Note that the q value produced by the critic is discarded and only the message vector α_t is passed on. The target and loss calculations for learning remain unchanged and we continue to use the hyperparameter ϵ to regularize the frequency of self-training.

4 Experiments

4.1 Method 1 Performance Evaluation

To evaluate the performance of Method 1, we compare the mean performance of 2 agent, 3 agent, and 5 agent version 1 peer learning systems with hyperparameter $\epsilon = 0.6$ against their corresponding no-communication baselines ($\epsilon = 1$) and ensembles. The no-communication baseline corresponds to the average performance of 2, 3, or 5 agents that have been trained independently and roll out evaluation trajectories independently. The ensemble baseline is obtained using the classic continuous action space approach of training independent policies and then rolling out a single evaluation trajectory by taking the mean action of all of the agents at each step. The curves for this experiment are plotted in Figure 1.

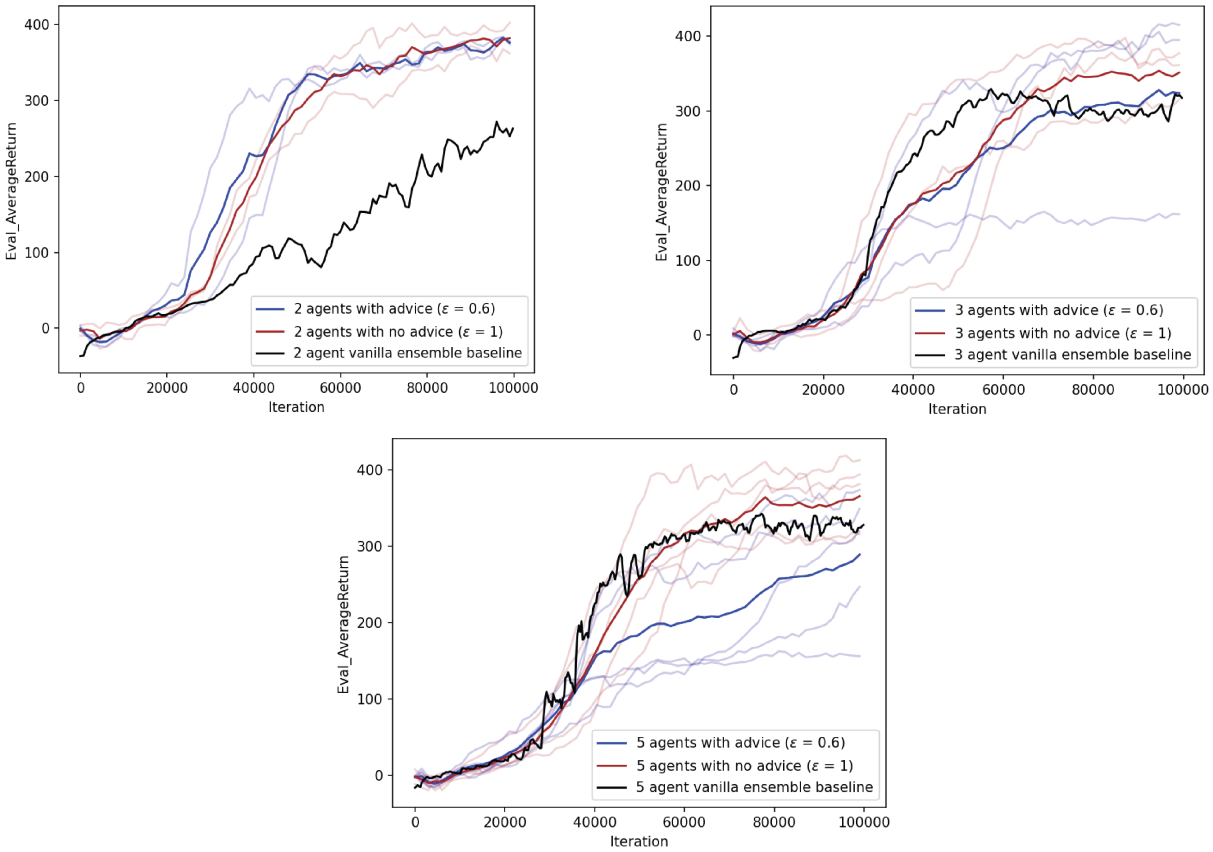


Figure 1: Evaluation performance of peer learning using the peer critic feedback via value signals implementation (Method 1) on the HalfCheetah-v4 environment for $\epsilon = 1$ (no advice/communication) and $\epsilon = 0.6$ (agents uses advice with probability 0.6 at each training step) for 2, 3, and 5 agent systems. An ensemble baseline is also plotted in each case.

4.2 Method 1 Hyperparameter Search

We explore the performance of Method 1 as a function of 3 hyperparameters: ϵ, the advice network size s (the number of neurons in the hidden layer – we fix f_{ψ_i} to always have only 1 hidden layer to minimize complexity), and the dimension of the advice α_t, i.e., the dimension of the output layer of f_{ψ_i}. We plot evaluation return curves for $\epsilon \in \{0, 0.1, 0.3, 0.6, 1\}$, $s \in \{16, 32, 64\}$, and $\dim(\alpha_t) \in \{4, 8, 16\}$ in Figure 2. Note that $\epsilon = 1$ corresponds to the no-communication baseline, as discussed above.

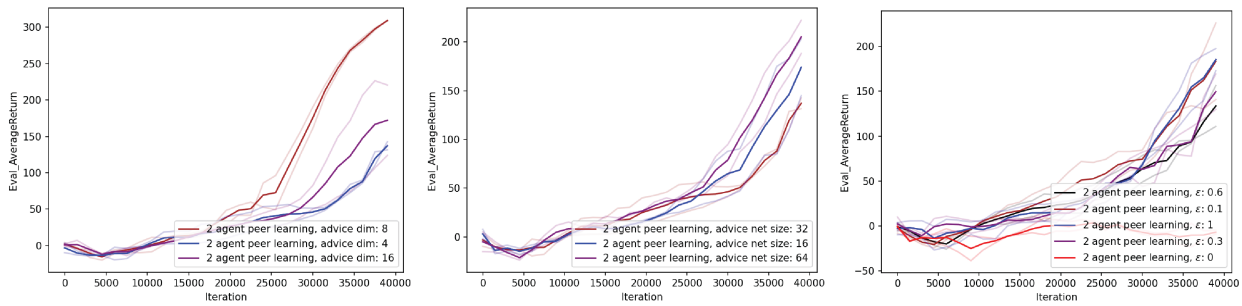
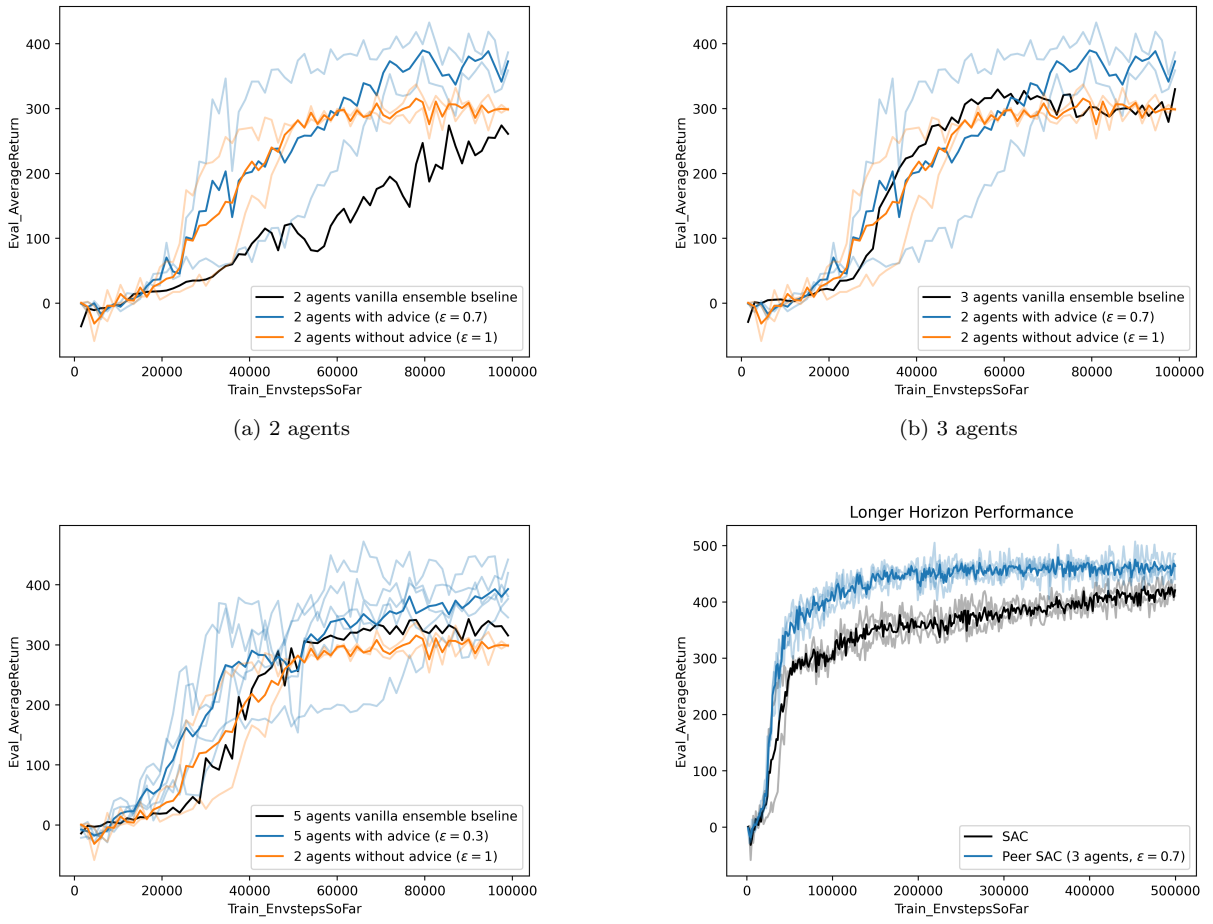


Figure 2: Evaluation performance of 2 agent peer learning using Method 1 on the HalfCheetah-v4 environment for different choice of hyperparameters ϵ, advice network size s, and advice dimension $\dim(\alpha_t)$.

4.3 Method 2 Performance Evaluation



(a) 2 agents

(b) 3 agents

(c) 5 agents

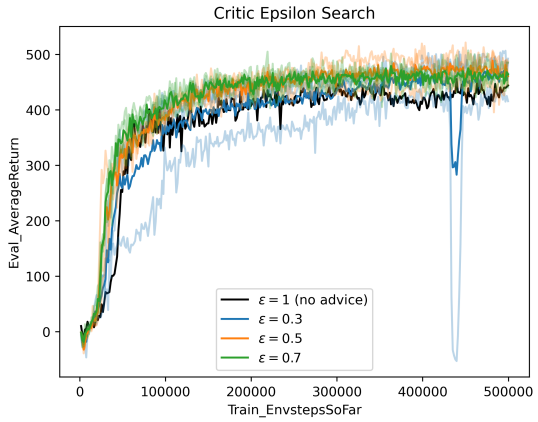
(d) Longer horizon study

Figure 3: Evaluation performance of peer learning using the peer critic feedback via compressed critic state implementation (Method 2) on the HalfCheetah-v4 environment for $\epsilon = 1$ (no advice/communication) and $\epsilon = 0.3$ (agents uses advice with probability 0.3 at each training step) for 2, 3, and 5 agent systems. An ensemble baseline is also plotted in each case.

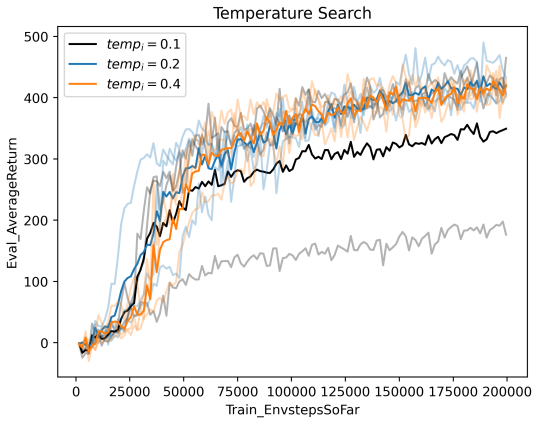
To evaluate the performance of Method 2, we compare the mean performance of 2, 3, and 5 agent versions of the peer learning system with hyperparameter $\epsilon = 0.3$ against no-communication baselines ($\epsilon = 1$) and ensembles. The no-communication baseline represents the average performance of 2 agents that have been trained independently and evaluate their performance independently. The ensemble baselines are the same as the ones described in the first method. The results of this experiment are plotted in Figure 3.

4.4 Method 2 Hyperparameter Search

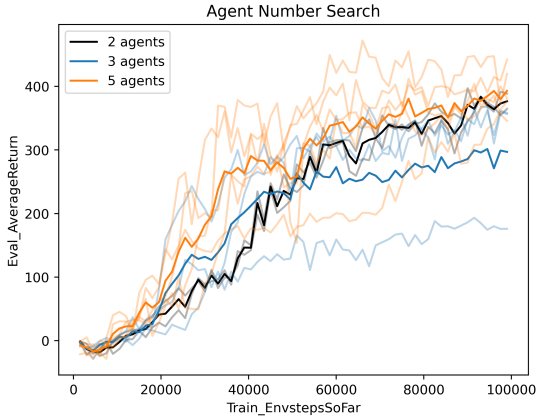
We explore the performance of Method 2 as a function of 3 hyperparameters: level of independence ϵ, the initial SAC temperature $temp_i$, and the number of agents n_a, i.e., the dimension of the output layer of f_{ψ_i}. We plot evaluation return curves for $\epsilon \in \{0.3, 0.5, 0.7, 1\}$, $temp_i \in \{0.1, 0.2, 0.4\}$, and $n_a \in \{2, 3, 5\}$ in Figure 4. Note that $\epsilon = 1$ corresponds to the no-communication baseline, as discussed above.



(a) Level of independence



(b) Initial temperature



(c) Number of agents

Figure 4: Evaluation performance of peer learning using Method 2 on the HalfCheetah-v4 environment for different choices of hyperparameters ϵ, $temp_i$, and n_a.

5 Discussion

5.1 Method 1 Results

The Method 1 experiments plotted in Figure 1 unfortunately seem to indicate no clear difference between the results obtained in a multi-agent system when critics have access to value signal feedback from other critics during training and when they don't ($\epsilon = 0.6$ versus $\epsilon = 1$). However, it is encouraging that this method is able to recover multi-agent ensemble performance for each agent independently during evaluation time. This could lead to faster inference while maintaining ensemble-level performance during evaluation by using only a single agent that was originally trained in a multi-agent peer learning scenario.

It is also certainly possible that the chosen environment (HalfCheetah-v4) does not have an exploration landscape that is conducive to obtaining large benefits due to feedback. In particular, one would expect that in an environment that has multiple modes of optimal behavior, having agents separately learn to cover all of the modes and then communicate information about them with each other could lead to better overall performance

and faster training for each agent. However, it is not clear that the HalfCheetah environment has this characteristic. Future work into testing this method on different environments would be an interesting direction to explore.

The hyperparameter search illustrated in the Figure 2 plots suggests that the model performance is not very sensitive to the choice of these values. Only the advice dimension $\dim(\alpha_t)$ hyperparameter displays a large gap, with an intermediate size of 8 appearing to be the best. Furthermore, a larger advice network size s seems to be preferable.

5.2 Method 2 Results

The Method 2 experiments plotted in Figure 3 indicate clear improvement of the peer SAC structure over vanilla ensembling. It also demonstrates that on a longer time scale it is able to outperform a pure SAC model, meaning that the agents successfully used peer advice to improve upon their training. While this trend is not clear for all agent numbers from the graphs, the trend of improvements of peer-sac over single sac is clearer over longer horizons.

While our hyperparameter search was limited by our computing power, early results show that epsilon and the initial temperature have a clear effect on peer learning.

These results serve as a proof of concept for peer learning and invite further study into finding the optimal structure and regime of peer learning. For example, the advice dimension should be explored and these experiments should be done in different environments with different task complexities.

Since peer learning enables easy leverage of parallelism, studying peer learning in the context of similar but not exactly the same environments would potentially allow us to leverage real-world scenarios where environments may not behave exactly the same but we still want to accelerate learning by creating duplicates.

6 Contributions

Both authors contributed equally. Hu implemented peer-SAC methods and wrote sections corresponding to the second method in the report. Sharma implemented the vanilla ensemble and wrote sections corresponding to the first method.

References

[Haa+18] Tuomas Haarnoja et al. *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor*. 2018. DOI: 10.48550/ARXIV.1801.01290. URL: https://arxiv.org/abs/1801.01290.

[KMP13] Soummya Kar, José M. F. Moura, and H. Vincent Poor. "$\mathcal{QD}$-Learning: A Collaborative Distributed Strategy for Multi-Agent Reinforcement Learning Through Consensus + Innovations". In: *IEEE Transactions on Signal Processing* 61.7 (2013), pp. 1848–1862. DOI: 10.1109/tsp.2013.2241057. URL: https://doi.org/10.1109%2Ftsp.2013.2241057.

[PP10] Paris Pennesi and Ioannis Ch. Paschalidis. "A Distributed Actor-Critic Algorithm and Applications to Mobile Sensor Network Coordination Problems". In: *IEEE Transactions on Automatic Control* 55.2 (2010), pp. 492–497. DOI: 10.1109/TAC.2009.2037462.

[WH08] Marco A Wiering and H. V. Hasselt. "Ensemble Algorithms in Reinforcement Learning". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 38 (2008), pp. 930–936.