UNLEASHING THE POTENTIAL OF DATA SHARING IN ENSEMBLE DEEP REINFORCEMENT LEARNING

Zhixuan Lin,* **Pierluca D'Oro, Evgenii Nikishin & Aaron Courville**
Mila, Université de Montréal

ABSTRACT

This work studies a crucial but often overlooked element of ensemble methods in deep reinforcement learning: data sharing between ensemble members. We show that data sharing enables *peer learning*, a powerful learning process in which individual agents learn from each other's experience to significantly improve their performance. When given access to the experience of other ensemble members, even the worst agent can match or outperform the previously best agent, triggering a virtuous circle. However, we show that peer learning can be unstable when the agents' ability to learn is impaired due to overtraining on early data. We thus employ the recently proposed solution of periodic resets and show that it ensures effective peer learning. We perform extensive experiments on continuous control tasks from both dense states and pixels to demonstrate the strong effect of peer learning and its interaction with resets.

1 INTRODUCTION

Amid their popularity in supervised learning, ensemble methods have also proven to be effective in deep reinforcement learning (RL) for a wide range of purposes (Osband et al., 2016; Chua et al., 2018; Chen et al., 2021; Schmitt et al., 2020; Agarwal et al., 2020; Liu et al., 2020). Underlying some of these methods is a simple principle: training a set of independent agents while *sharing the data* among them. This simple strategy allows different agents to concurrently explore different policies while sharing their knowledge. It has been shown to be particularly useful for promoting deep exploration (Osband et al., 2016) and thus improving the learning efficiency.

Intuitively, data sharing allows different agents to learn from each other's experience. This learning process, which we refer to as *peer learning*, happens covertly during training and thus is often overlooked, despite being essential for exploiting the diverse data brought by deep exploration. Moreover, previous work often applies additional techniques (Osband et al., 2018; Lee et al., 2021; Januszewski et al., 2021) on top of the basic ensemble strategy, which further obscures the role of peer learning in the success of these methods. As a result, almost no effort has been put into understanding how peer learning affects individual agents, under what conditions is it effective, and how we can better benefit from it.

This work presents a dedicated study of the effect of data sharing in ensemble deep reinforcement learning. Through carefully designed experiments, we present a clear view of the peer learning process in which agents of different levels learn from each other to improve their own performance. Specifically, we show that a transition from separate to shared replay buffers during training can often cause all ensemble members to match or outperform the previously best agent in the ensemble, and thus resulting in a significant improvement in the overall performance. Based on these observations, we show that oftentimes data sharing *alone* can already produce a significant gain over just training a single agent, without the use of any additional technique such as policy aggregation (Januszewski et al., 2021), bootstrapping (Osband et al., 2016), or UCB exploration (Chen et al., 2018; Lee et al., 2021).

A natural question is whether peer learning is always stable in practice. Recently, Nikishin et al. (2022) has shown that deep RL agents can suffer from the primacy bias, i.e., a tendency to overfit early data that impairs the agent's ability to learn from its own future experience. We show that,

*Correspondence to Zhixuan Lin <zxlin.cs@gmail.com>

in the context of ensemble methods, the primacy bias can also prevent an agent from learning from other agents, even when the other agents have good performance and are producing high-quality samples. Fortunately, we find that the simple resetting mechanism proposed in Nikishin et al. (2022) provides a robust solution and ensures exploitation of the shared data. Our experiments on continuous control tasks from both dense states and pixels show that ensemble and resets are a particularly robust combination and always significantly improve the performance over the baseline methods while only incurring a minor increase in computation time.

The contribution of this paper can be summarized as follows:

1. We expose peer learning as a prominent effect when using data sharing in ensemble methods and demonstrate its impact on individual agents' performance;
2. We show that data sharing alone has the potential to provide a significant improvement in performance, without the use of any additional technique;
3. We find that peer learning can be unreliable in the presence of the primacy bias, and show that the recently proposed resetting mechanism provides a robust solution;
4. We empirically verify the effect of peer learning and its interaction with resets with extensive experiments on continuous control tasks from both dense states and pixels.

2 Related Work

Ensemble methods in RL Ensemble methods have been employed in deep reinforcement learning for a wide range of purposes including promoting deep exploration (Osband et al., 2016; Peng et al., 2020), improving value estimations (Chen et al., 2021; Lan et al., 2020; Peer et al., 2021; Liang et al., 2022; Anschel et al., 2016) or model predictions (Chua et al., 2018; Kurutach et al., 2018), and accelerating hyperparameter sweeps (Schmitt et al., 2020; Liu et al., 2020). While there are various distinct ways to apply an ensemble, the simple ensemble strategy (i.e., training a set of independent agents while sharing the data) considered in this work underlies many previous works. For example, Osband et al. (2016) trains a multi-head DQN (Mnih et al., 2015) with all heads sharing a single replay buffer, to improve exploration. Januszewski et al. (2021) trains multiple DDPG (Lillicrap et al., 2016) policies with a shared replay buffer, and combine them into an average policy at test-time. Schmitt et al. (2020) and Liu et al. (2020) accelerate hyperparameter sweeps by training agents with different hyperparameters sharing a central replay buffer. However, most previous work does not investigate the role of data sharing, or only marginally touches on its usefulness, without uncovering its effect on individual agents in detail or analyzing its failure cases.

Data sharing in offline and multi-task RL Besides its wide adoption in ensemble RL methods, data sharing has also been explored in offline RL (Yu et al., 2022; 2021; Dorfman & Tamar, 2020; Mitchell et al., 2021) and multi-task RL (Andrychowicz et al., 2017; Eysenbach et al., 2020; Li et al., 2020; Liu et al., 2019; Kalashnikov et al., 2021; Chebotar et al., 2021). Both paradigms provide natural motivations for data sharing: offline RL involves reusing previously collected data, while multi-task RL involves sharing experience across different tasks. These scenarios pose various challenges that are very different from the online, single-task setting that we study in this work.

The primacy bias in deep RL The recently studied primacy bias (Nikishin et al., 2022) is a tendency of deep RL agents to overfit early data that impairs the agents' ability to learn from their own future experience. Such overfitting can have multiple different causes and various negative effects. For example, Nikishin et al. (2022) shows that a high replay ratio can often cause significant overfitting due to the large number of gradient updates relative to the number of environment interactions collected. Interestingly, Nikishin et al. (2022) proposes a surprisingly simple but effective solution: periodically reinitializing part of the agents, *while* preserving the replay buffer. This simple mechanism, which they refer to as "resetting", can reliably eliminate the negative effect of the primacy bias and result in significant improvements. In this work, we show that the primacy bias can damage the peer learning process as well, and that resetting also robustly fixes this issue.

3 PRELIMINARIES

The RL formulation We consider the standard RL framework (Sutton & Barto, 2018) that models the environment as a Markov decision process (MDP). At each timestep, the agent observes the current environment state S_t and executes an action A_t sampled from its current policy π. It then receives a reward signal R_t and the next state S_{t+1} from the environment. The goal of a reinforcement learning algorithm is then to find a policy π that maximizes the expected discounted sum of rewards $\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t R_t]$ based on its past interactions with the environment. This work focuses on deep RL algorithms that come with a *replay buffer* and support *off-policy* training, as they provide natural infrastructures and mechanisms for data sharing. Specifically, we use the popular Soft Actor-Critic (SAC) algorithm (Haarnoja et al., 2018) (for dense states) and the DrQ algorithm (Kostrikov et al., 2021) (for image observations) as the base algorithms in all our experiments.

Ensemble strategy In this work, we consider a simple form of ensemble that underlies many previous works (Osband et al., 2016; Schmitt et al., 2020; Januszewski et al., 2021; Peng et al., 2020; Liu et al., 2020). Given a base RL algorithm and the ensemble size N, we train N independent agents interacting with N environment instances concurrently. Each agent has their own copy of the parameters (e.g., policy, value, and target networks) and optimizers, and any learning updates of the base algorithm are performed independently for each agent, but in parallel. However, we allow different agents to share the same image encoder when using images as input. Unless otherwise stated, the agents share a central replay buffer as well as the training batch. Note that for fair comparison, whenever we use an ensemble with data sharing, we keep the total interaction steps the same as when we train a single agent. For example, if the total interaction budget is 1×10^6 steps and the ensemble size is 10, each ensemble member can only interact with the environment for 1×10^5 steps. For evaluation, we do not perform any policy aggregation such as voting (Osband et al., 2016) or averaging (Januszewski et al., 2021). Instead, we always use the policies of individual ensemble members and report either the best or average performance within ensemble in most cases. A detailed description of this ensemble strategy can be found in Algorithm 1 in our Appendix.

Resetting The resetting mechanism is originally proposed in Nikishin et al. (2022) to counter the negative effect of the primacy bias. Similar to Nikishin et al. (2022), whenever we refer to "resetting an agent", we mean to reinitialize all or part of the parameters and optimizer states of all components of the agent, *while* preserving the replay buffer. Specifically, for SAC, this means reinitializing everything while preserving the buffer. This is also the case for DrQ except that we retain the parameters and optimizer states of the image encoder.

Replay ratio (RR) An important hyperparameter that deserves special attention is the *replay ratio*[1]. It refers to the ratio of the number of gradient updates to the number of environment steps at any point in training. Crucially, Nikishin et al. (2022) shows that a high replay ratio can often worsen the negative effect of primacy bias due to overtraining on early data. Though in the single-agent case replay ratio is well-defined, in the context of ensemble methods there exists some ambiguity because it involves multiple agents. We therefore give an exact definition of replay ratio that we use throughout this paper in Appendix D.

4 THE PEER LEARNING PROCESS

When training an ensemble of independent data sharing agents, one can expect the following two processes to happen:

1. A *diversification* process in which random initialization and stochastic training introduce diversity to the ensemble;
2. A *peer learning* process in which different ensemble members learn from each other's experience to improve themselves.

Intuitively, the first process creates a diverse set of agents and thus diverse data, which peer learning might exploit to improve the performance of individual ensemble members. These two processes

[1] Also referred to as update-to-data (UTD) ratio.

cooperate closely and *jointly* contribute to the high performance of ensemble methods. However, most focus and credit have been given to the diversification aspect and its exploration benefits (Osband et al., 2016; Chen et al., 2018; Lee et al., 2021), while the more subtle peer learning process has often been overlooked despite being essential for the exploitation of the diverse data brought by deep exploration. A better understanding of peer learning can reveal the inner workings behind the success of ensemble methods and guide the design of better ensemble algorithms.

The rest of this section is divided into two parts. In the first part, we demonstrate the strong effect of peer learning on the performance of individual ensemble members. In the second part, we demonstrate a failure case of peer learning in which the agents' ability to learn is impaired due to overtraining on early data, and that the recently proposed solution of periodic resets robustly fixes this issue.

4.1 The Effect of Peer Learning

We now demonstrate the effect of peer learning on the individual ensemble members. Normally, peer learning happens covertly during training and its effect is only indirectly reflected in the overall performance. In order to make the impact of peer learning visible, we design a simple setup that disentangles the diversification process and the peer learning process. Specifically, we start training with the N ensemble members with separate replay buffers. After a certain timestep, we either continue training with separate replay buffers, or *merge* the separate replay buffers into one shared replay buffer. We train an ensemble of $N = 10$ SAC agents with a replay ratio of 1, and select four environments from the DeepMind Control Suite (Tassa et al., 2018) that are representative of the effects of ensemble and resets. We carefully control the interaction and gradient update schedule as well as the buffer sizes before and after merging to ensure the results are fair and meaningful. More details and results can be found in Appendix C and Appendix F respectively.

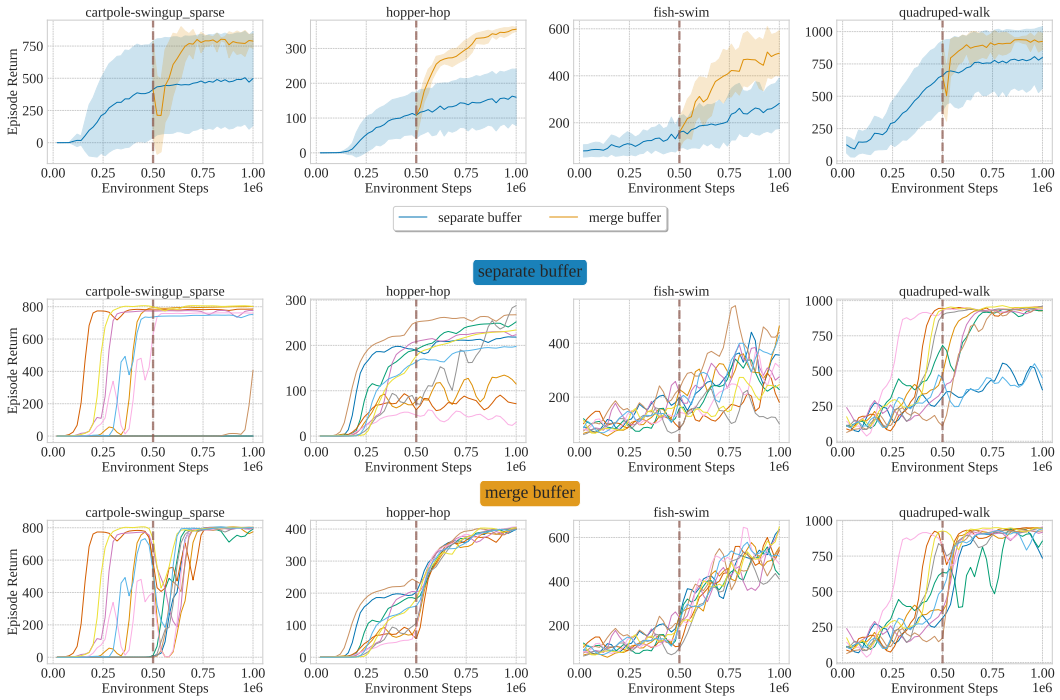

Figure 1: **Top**: the effect of buffers merging. Vertical lines indicate the step for merging. For each experiment we use 5 seeds and $N = 10$ agents in the ensemble. Performance is averaged across all ensemble member and all seeds. To quantify the effect of buffer merging on the variance within ensembles, we first calculate the standard deviation of the performance within each ensemble and then average over all seeds. Shaded area thus represents this average standard deviation. **Bottom**: individual learning curves from one run with or without buffer merging. The individual learning curves are Gaussian smoothed to reduce visual clutter.

Figures 1 shows the effect of buffer merging on the overall performance of the ensemble as well as the individual learning curves of all ensemble members from a single run with or without buffer merging. Before merging, different agents develop distinct policies and have different performance; upon merging, the overall performance improves significantly, usually with a drastic reduction in the variance within the ensemble. In all cases, the improvement happens robustly and rapidly, demonstrating the strong effect of peer learning. Inspecting the individual learning curves reveals the diverse effects of peer learning: on `cartpole-swingup_sparse` and `quadruped-walk`, the under-performing agents learn from the data of the optimal agents in the ensemble and quickly converge to the optimal policy. On `fish-swim` and `hopper-hop`, the entire ensemble even outperforms the previously best agents before merging.

Note the this experiment is for demonstration purposes only. In practice, one would always directly start with a shared replay buffer for best sample efficiency. Given the above results, we can imagine how peer learning improves performance in that case: starting with random initialization, the stochastic training process continues to diversify the ensemble; concurrently, peer learning selects and integrates the solutions discovered and improves the performance of all agents in the ensemble. The result of this simultaneous diversification and peer learning is significantly improved sample efficiency relative to just training a single agent throughout the entire training process, as observed in previous work (Osband et al., 2016) and confirmed in Section 5 of this work.

The above results have an important implication. As long as peer learning is stable, we can expect almost *all* ensemble members to be more high-performing than if we only train a single agent. Therefore, one can simply train an ensemble of agents without any additional technique and still benefit from it. This however does not mean that other techniques are not useful or not worth investigating, as previous work has shown that techniques such as policy aggregation and diversity regularization can further improve performance (Januszewski et al., 2021; Peng et al., 2020).

4.2 A Failure Case and the Solution

Intuitively, successful peer learning requires an agent to maintain the ability to quickly adapt their policy. One can imagine that this is less likely to happen when the agent is stuck in some suboptimal policy, loses their representation capacity (Lyle et al., 2022), or experiences significant value overestimation (Kumar et al., 2019). This work investigates one particular effect that might impair an agent's ability to learn: the primacy bias (Nikishin et al., 2022). Specifically, Nikishin et al. (2022) shows that overtraining on early data, e.g. when using a high replay ratio, can significantly impair the agent's ability to learn from its own future experience. This creates a vicious circle, as the data generated by a poor agent will also be poor, which further eliminates the possibility of improvement. We hypothesize that the primacy bias can also damage the agent's ability to learn from other agents' data, *even if it has access to the high quality samples generated by other high-performing agents*. Notably, Nikishin et al. (2022) proposes a simple solution to counter the primacy bias: periodically resetting part of the agent, while preserving the replay buffer. This strategy is shown to be highly effective. We therefore expect it to also provide a solution that ensures stable peer learning.

To verify these hypotheses, we use the same setup as in the previous experiment, except for a replay ratio of 16 instead of 1 to make the effect of the primacy bias clearly detectable. To quantify the effect of resets, we include two additional variants: in the first one, we continue training with separate buffers, and reset the agents; in the second one, we merge the buffers and reset the agents. Results are shown in Figure 2. Naively merging the buffer is much less effective at a replay ratio of 16 than it is when using a replay ratio of 1, except for `cartpole-swingup_sparse`, which has the simplest environment dynamics among the four. On `hopper-hop` and `quadruped-walk`, merging the buffer provides almost no benefit. The individual learning curves show that there exists a significant variance in the ensemble members' performance *even if the data is shared* within the ensemble, indicating that peer learning is less effective in this case. Specifically, for `hopper-hop` and `quadruped-walk`, our inspection shows that the under-performing agents are incapable to learn from the better agents due to significant value overestimation either before or after merging, while for `fish-swim` the cause is less clear. Nevertheless, resetting restores the agents' ability to effectively learn from each other and achieves high performance with a low variance. Note that even though resetting without merging the buffer also provides a reasonable gain (e.g., on `quadruped-walk`), it is not sufficient for reaching optimal performance in all cases.

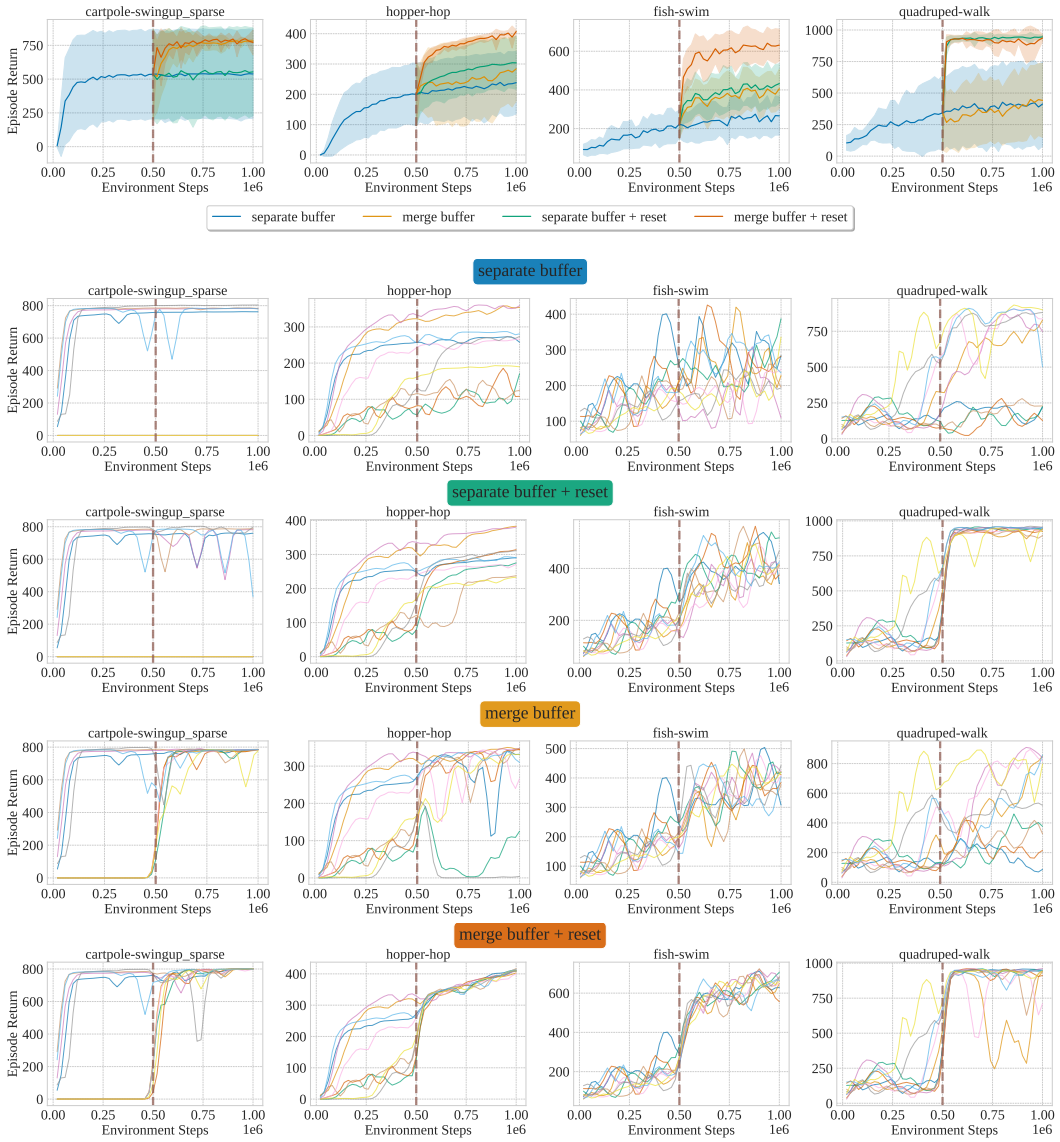

Figure 2: **Top**: the effect of buffer merging and resets with replay ratio 16. Vertical lines indicate the step for merging. For each experiment we use 5 seeds and $N = 10$ agents in the ensemble. Performance is averaged across all ensemble member and all seeds. To quantify the effect of buffer merging on the variance within ensembles, we first calculate the standard deviation of the performance within each ensemble and then average over all seeds. Shaded area thus represents this average standard deviation. **Bottom**: individual learning curves from one run. The individual learning curves are Gaussian smoothed to reduce visual clutter.

Besides ensuring effective peer learning, another reason why resetting can be particularly important when using an ensemble is the increased off-policyness and hence more severe primacy bias. Note that when using an ensemble, we keep the total interaction steps the same as when training a single agent (e.g., 1×10^5 interactions for each of the 10 members when using a 1×10^6 step total interaction budget). Due to the natural design choice of keeping the number of gradient updates for each agent unchanged, each agent receives much fewer on-policy samples compared to the number of gradient updates it experiences. This increased off-policyness may exacerbate issues such as significant overestimation (Ostrovski et al., 2021; Kumar et al., 2019), which has been shown to be one of the causes of the primacy bias (Nikishin et al., 2022). We support this statement with the counter-intuitive fact that sometimes using an ensemble can be much worse than just training a single agent. For example, in Figure 12 in our Appendix, we show that combining SAC with ensemble is much worse than SAC on the three `humanoid` tasks when using a high replay ratio.

Through these experiments and analysis, we learned more about the mechanics of peer learning and how to overcome its limitations. In the next section, we perform experiments with a standard ensemble strategy (i.e., start training with a shared buffer) to demonstrate the effect of peer learning in practice and its interaction with resets.

5 Experiments

Our experimental results can be summarized as follows. First, we verify the strong effect of peer learning by showing that data sharing alone has the potential to provide a significant gain over just training a single agent, without the use of any additional technique. Second, we show that the primacy bias can damage peer learning and weaken the advantages of ensembles, while resetting ensures the exploitation of their benefits. Then, we present an analysis of the performance distribution within ensemble and show the dominance of ensemble methods in terms of sample efficiency.

5.1 Setup

General setup We focus on continuous control tasks from the DeepMind Control Suite (Tassa et al., 2018). We select 16 tasks for state-based control and 13 tasks for pixel-based control. Please refer to Appendix A for the full environment lists. For state-based tasks, each algorithm is allowed a budget of 1×10^6 environment steps; for pixel-based tasks, we use 2×10^6 environments steps. We use SAC (Haarnoja et al., 2018) and DrQ (Kostrikov et al., 2021) as our base algorithms for the state-based and pixel-based tasks respectively. For each base algorithm X, where X can either be SAC or DrQ, we also consider 3 variants: X + ensemble, X + resets, and X + ensemble + resets. For all experiments we use 10 random seeds.

We use $N = 10$ for all our main results with ensemble. We maintain completely independent ensemble members for SAC, while for DrQ we share the image encoder among ensemble members. As stated earlier, when using an ensemble we keep the total number of environment interactions the same as when we train a single agent to ensure fair comparison. For SAC, in order to probe the effect of primacy bias, we experiment with five different replay ratios $\{1, 2, 4, 8, 16\}$. Recently, Anonymous (2022) shows that SAC + resets exhibits scalability with respect to the replay ratio, and thus it would also be interesting to see whether this still applies with the addition of ensemble. Following a similar strategy in Anonymous (2022), when using resets, we set the reset period of SAC according to the replay ratio such that we reset every 1.92×10^6 gradient updates; for DrQ, we reset the whole agent except for the image encoder per 2×10^5 environment steps as done in Nikishin et al. (2022). We always skip the last reset to avoid reporting on undertrained agents.

Evaluation protocol For the main results, we show the performance when using the best agent in each ensemble for evaluation (best), as well when using the average performance of each ensemble as the corresponding run's performance (avg). To avoid bias in the reported results, when reporting with the best ensemble members, we perform two independent rounds of evaluation for all ensemble members. We select the best agent based on the first round of evaluation, and report its performance in the second round. For our main results, we report the point estimates and 95% confidence intervals of the interquartile mean (IQM) (Agarwal et al., 2021) of the evaluation results of the last timestep across all environments. All evaluation uses 10 episodes. Other experimental details can be found in Appendix A

5.2 Main Results

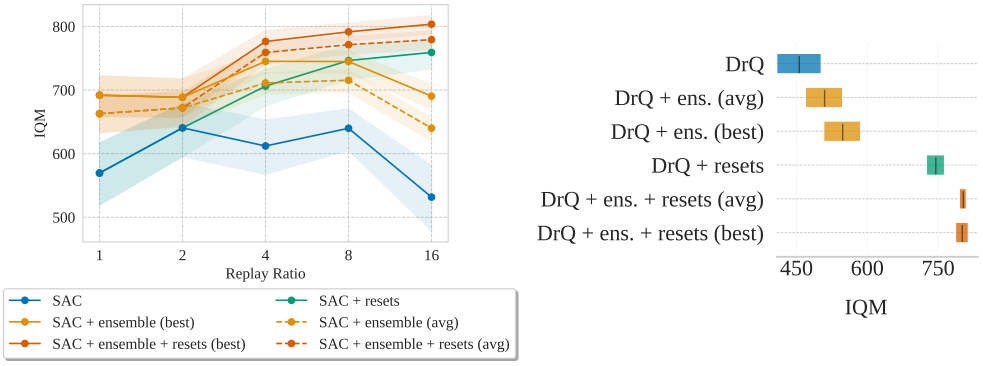

Figure 3: Point estimates and 95% confidence intervals for IQM of the performance of SAC, DrQ, and their variants. The performance of SAC and DrQ is aggregated across 16 and 13 environments respectively. We show the performance when using the best agent in each ensemble for evaluation (best), as well as when using the average performance of each ensemble as the corresponding run's performance (avg). Note that for the reset period defined previously, for SAC we do not perform resets for replay ratios 1 and 2, and therefore some data points overlap under these two replay ratios. All results are aggregated over 10 random seeds.

Our main results for SAC and DrQ are presented in Figure 3. More results, including an ablation of the ensemble size and comparison with state-of-the-art results, can be found in Appendix E.

Effect of ensemble We first focus on the overall effect of ensemble. As shown in Figure 3, using an ensemble while sharing data always provides a significant gain over just training a single agent, *regardless of the base algorithm, replay ratio, or whether we do resets* (e.g., SAC + ensemble + resets is better than SAC + resets for all replay ratios). This provides strong evidence that data sharing alone has the potential to provide a significant improvement in performance without the need of any additional trick, which also validates the strong effect of peer learning and its important role in ensemble methods.

Effect of the primacy bias and resets From Figure 3 (left) we can see the impact of the replay ratio when using an ensemble. For low replay ratios such as 1, 2 and 4, SAC + ensemble is able to robustly outperform SAC and SAC + resets. However, as we step into the high replay ratio regime in which the primacy bias becomes severe, the performance of SAC + ensemble starts to degrade and is outperformed by SAC + resets. Note SAC + ensemble still dominates SAC even with replay ratio 16, indicating that the benefits of using an ensemble still exist but are weakened by the primacy bias, likely due to a combination of its damage to the peer learning process as shown in Section 4 and its other the direct negative effects. As expected, SAC + ensemble + resets does not suffer from the damaged peer learning process even under a high replay ratio (e.g. 16) and remains high performing.

Scalability w.r.t. computation Similar to SAC + resets, we see that SAC + ensemble + resets exhibits the same scalability with respect to the replay ratio (Anonymous, 2022). However, it is much more high-performing: SAC + ensemble + resets at replay ratio 4 already outperforms SAC + resets at replay ratio 16, despite the fact that the latter is much more expensive. Besides, it is also more robust, as can be seen from the much narrower confidence intervals.

5.3 Analysis

Performance distribution within ensemble In Figure 4, we show the results when using the n-th best agent in the ensemble for evaluation, where n ranges from 1 to N. For SAC, most ensemble members are high-performing and stay close to the best agent in the ensemble. This is especially true when resets are applied, in which case the performance distribution of agents of different ranks are highly concentrated. Without resets, the worst and the second worst ensemble members gradually

deviate from other members as replay ratio increases. For DrQ, with resets there is almost no variance within the ensembles, while without resets the performance of different ensemble members is clearly spread out. The low variance in performance across ensemble members when using resets validates our claim that resets can ensure effective peer learning.

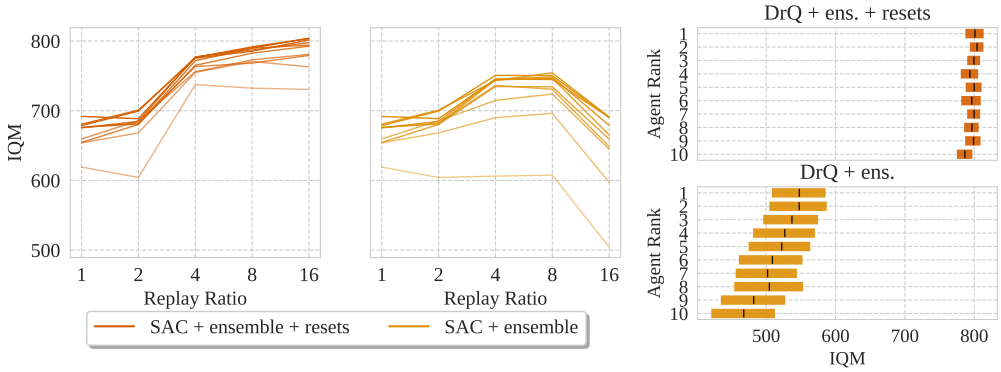

Figure 4: **Left**: Performance of SAC + ensemble with or without resets when using the n-th best agent in each ensemble for evaluation, where n ranges from 1 to N. Darker color indicates smaller n. **Right**: Performance of DrQ + ensemble with or without resets when using the n-th best agent for evaluation. Agent rank n indicates the n-th best agent. All results are aggregated over 10 random seeds. Similar to the process for selecting the best agent, we use two rounds of evaluation for selecting the n-th best agent.

Sample efficiency In Figure 5 we show the learning curves of the different methods. For SAC we show the results of an intermediate replay ratio 4, and more results can be found in Appendix E. From Figure 5 we can see that the dominance of ensemble methods over the single-agent methods often starts from the beginning of the training and continues until the end, consistent with previous work's observation (Osband et al., 2016). This confirms that the interaction between diversification and peer learning is a continual process that happens throughout the entire training process.

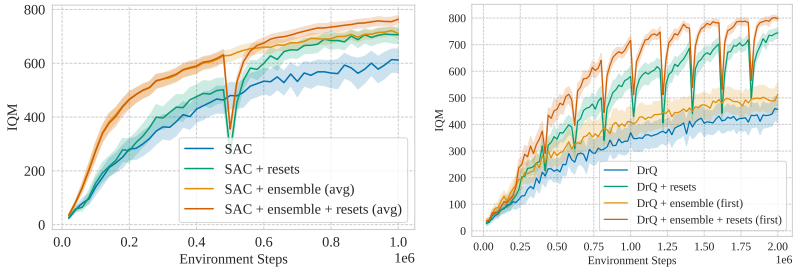

Figure 5: Learning curves for SAC (left) and DrQ (right), and their variants. Shaded area indicates 95% bootstrapped confidence interval. We report average performance within the ensemble for SAC and the performance of first indexed agent for DrQ. All results are aggregated over 10 random seeds.

6 Conclusion

The work studies a crucial element behind the success of many ensemble methods: data sharing between ensemble members. We highlight the powerful peer learning process enabled by data sharing and demonstrate how it improves individual agents' performance. We also identify a failure case of peer learning caused by overtraining on early data, and show that the recently proposed solution of periodic resets robustly alleviates the problem and boosts performance. Our experiments on continuous control tasks from both dense states and pixels demonstrate the strong effect of peer learning and its interaction with resets. We hope our work will encourage the RL community to better recognize the effectiveness of data sharing and fully exploit its potential.

REFERENCES

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *ICML*, 2020.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C. Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *NeurIPS*, 2021.

Marcin Andrychowicz, Dwight Crow, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Joshua Tobin, P. Abbeel, and Wojciech Zaremba. Hindsight experience replay. *ArXiv*, abs/1707.01495, 2017.

Anonymous. Sample-efficient reinforcement learning by breaking the replay ratio barrier. In *Submitted to The Eleventh International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=OpC-9aBBVJe`. under review.

Oron Anschel, Nir Baram, and Nahum Shimkin. Deep reinforcement learning with averaged target dqn. *ArXiv*, abs/1611.01929, 2016.

Yevgen Chebotar, Karol Hausman, Yao Lu, Ted Xiao, Dmitry Kalashnikov, Jacob Varley, Alex Irpan, Benjamin Eysenbach, Ryan C. Julian, Chelsea Finn, and Sergey Levine. Actionable models: Unsupervised offline reinforcement learning of robotic skills. *ArXiv*, abs/2104.07749, 2021.

Richard Y. Chen, Szymon Sidor, P. Abbeel, and John Schulman. Ucb exploration via q-ensembles. *arXiv: Learning*, 2018.

Xinyue Chen, Che Wang, Zijian Zhou, and Keith W. Ross. Randomized ensembled double q-learning: Learning fast without a model. *ArXiv*, abs/2101.05982, 2021.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *NeurIPS*, 2018.

Ron Dorfman and Aviv Tamar. Offline meta reinforcement learning. *ArXiv*, abs/2008.02598, 2020.

Benjamin Eysenbach, Xinyang Geng, Sergey Levine, and Ruslan Salakhutdinov. Rewriting history with inverse rl: Hindsight inference for policy improvement. *ArXiv*, abs/2002.11089, 2020.

Tuomas Haarnoja, Aurick Zhou, P. Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 2018.

Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2020. URL `http://github.com/google/flax`.

Piotr Januszewski, Mateusz Olko, Michal Królikowski, Jakub Bartlomiej Swiatkowski, Marcin Andrychowicz, Lukasz Kuci'nski, and Piotr Milo's. Continuous control with ensemble deep deterministic policy gradients. *ArXiv*, abs/2111.15382, 2021.

Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *ArXiv*, abs/2104.08212, 2021.

Ilya Kostrikov. JAXRL: Implementations of Reinforcement Learning algorithms in JAX, 10 2021. URL `https://github.com/ikostrikov/jaxrl`.

Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *ArXiv*, abs/2004.13649, 2021.

Aviral Kumar, Justin Fu, G. Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *NeurIPS*, 2019.

Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and P. Abbeel. Model-ensemble trust-region policy optimization. *ArXiv*, abs/1802.10592, 2018.

Qingfeng Lan, Yangchen Pan, Alona Fyshe, and Martha White. Maxmin q-learning: Controlling the estimation bias of q-learning. *ArXiv*, abs/2002.06487, 2020.

Kimin Lee, Michael Laskin, A. Srinivas, and P. Abbeel. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. In *ICML*, 2021.

Alexander Li, Lerrel Pinto, and P. Abbeel. Generalized hindsight for reinforcement learning. *ArXiv*, abs/2002.11708, 2020.

Litian Liang, Yaosheng Xu, Stephen McAleer, Dailin Hu, Alexander T. Ihler, P. Abbeel, and Roy Fox. Reducing variance in temporal-difference value estimation via ensemble of deep networks. *ArXiv*, abs/2209.07670, 2022.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Manfred Otto Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2016.

Ge Liu, Rui Wu, Heng-Tze Cheng, Jing Wang, Jayden Ooi, Lihong Li, Ang Li, Wai Lok Sibon Li, Craig Boutilier, and Ed H. Chi. Data efficient training for reinforcement learning with adaptive behavior policy sharing. *ArXiv*, abs/2002.05229, 2020.

Hao Liu, Alexander Trott, Richard Socher, and Caiming Xiong. Competitive experience replay. *ArXiv*, abs/1902.00528, 2019.

Clare Lyle, Mark Rowland, and Will Dabney. Understanding and preventing capacity loss in reinforcement learning. *ArXiv*, abs/2204.09560, 2022.

Eric Mitchell, Rafael Rafailov, Xue Bin Peng, Sergey Levine, and Chelsea Finn. Offline meta-reinforcement learning with advantage weighting. In *ICML*, 2021.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charlie Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron C. Courville. The primacy bias in deep reinforcement learning. In *ICML*, 2022.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In *NIPS*, 2016.

Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. *ArXiv*, abs/1806.03335, 2018.

Georg Ostrovski, Pablo Samuel Castro, and Will Dabney. The difficulty of passive learning in deep reinforcement learning. In *NeurIPS*, 2021.

Oren Peer, Chen Tessler, Nadav Merlis, and Ron Meir. Ensemble bootstrapping for q-learning. In *ICML*, 2021.

Zhenghao Peng, Hao Sun, and Bolei Zhou. Non-local policy optimization via diversity-regularized collaborative exploration. *ArXiv*, abs/2006.07781, 2020.

Simon Schmitt, Matteo Hessel, and Karen Simonyan. Off-policy actor-critic with shared experience replay. In *ICML*, 2020.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy P. Lillicrap, and Martin A. Riedmiller. Deepmind control suite. *ArXiv*, abs/1801.00690, 2018.

Tianhe Yu, Aviral Kumar, Yevgen Chebotar, Karol Hausman, Sergey Levine, and Chelsea Finn. Conservative data sharing for multi-task offline reinforcement learning. In *NeurIPS*, 2021.

Tianhe Yu, Aviral Kumar, Yevgen Chebotar, Karol Hausman, Chelsea Finn, and Sergey Levine. How to leverage unlabeled data in offline reinforcement learning. In *ICML*, 2022.

A General Experimental Details

Table 1: Environment list for SAC and DrQ

SAC	DrQ
acrobot-swingup	acrobot-swingup
cartpole-swingup_sparse	cartpole-swingup_sparse
cheetah-run	cheetah-run
finger-turn_hard	finger-turn_easy
fish-swim	finger-turn_hard
hopper-hop	hopper-hop
hopper-stand	hopper-stand
humanoid-run	pendulum-swingup
humanoid-stand	quadruped-run
humanoid-walk	quadruped-walk
pendulum-swingup	reacher-easy
quadruped-run	reacher-hard
quadruped-walk	walker-run
reacher-hard	
swimmer-swimmer6	
walker-run	

Our selection of the environments for state-based and pixel-based control are listed in Table 1. We try to include as many commonly used environments as possible while excluding the overly simple environments (e.g., `cartpole-balance`).

The implementation of all our SAC and DrQ variants are based on an open source JAX implementation (Kostrikov, 2021). We adopt all their default hyperparameters except for the replay buffer size of DrQ, for which we use 1×10^6 instead of 1×10^5 as we find it to work much better. For the implementation of resets, we follow the the official implementation from the authors[2]. We implement the parallel computation of the ensembles using the `nn.vmap` function of the Flax (Heek et al., 2020) library.

B Details of the Basic Ensemble Strategy

Please see Algorithm 1 for a detailed description of the basic ensemble strategy considered in this work.

C Details for the Buffer Merging Experiment

For all experiments we train for a total of 1×10^6 steps. For the first 5×10^5 steps where we perform training with separate buffers, *each* agent will interact with the environment for 5×10^5 steps. For the last 5×10^5 steps, if we continue training with separate buffers, then again each agent will interact with the environment for 5×10^5 steps. However, if we merge buffer, the 5×10^5 interaction steps will be shared across the N agents, i.e., each agent only gets to interact with the environment for $5 \times 10^5/N$ steps, but in a round-robin way. We control the gradient update schedule according to the definition of replay ratio we give in Appendix D, so that all agents receive the equal amount of gradient updates at the same x-axis position in Figure 1 and Figure 2.

Also, upon merging we subsample the shared replay buffer by N to avoid the unfair advantage of a much larger replay buffer. For example, with $N = 10$ the shared buffer will contain $10 \times 5 \times 10^5 = 5 \times 10^6$ samples (merged from the 10 separate replay buffers), which we will downsample to 5×10^5 samples. We uniformly sample the transitions at constant intervals from each replay buffer, and therefore each separate replay buffer will have an equal portion in the merged replay buffer.

[2] `https://github.com/evgenii-nikishin/rl_with_resets`

Algorithm 1: The Basic Ensemble Strategy

Input: Total interaction steps M, the ensemble size N, replay ratio R, N environment instances $\{\text{env}_i\}_{i=1}^N$, N agents with parameters $\{\theta_i\}_{i=1}^N$, a base RL algorithm f, a shared replay buffer

Output: The updated agents with parameters $\{\theta_i\}_{i=1}^N$

$s_i \leftarrow \text{init}(\text{env}_i)$ for $i = 1, \ldots, N$ `// Reset environments`
for $m \leftarrow 1$ ***to*** M **do**
  $i \leftarrow m \bmod N$ `// Select an ensemble member for acting`
  $a_i \sim \pi_{\theta_i}(\cdot|s_i)$
  `/* For simplicity, we assume the environment has no terminal states */`
  $s'_i, r_i = \text{step}(\text{env}_i, a_i)$
  Add (s_i, a_i, r_i, s'_i) to the shared replay buffer
  $s_i \leftarrow s'_i$
  for $k \leftarrow 1$ ***to*** R **do**
    Sample a batch of transition B from the shared replay buffer
    for $i \leftarrow 1$ ***to*** N **do**
      $\theta_i \leftarrow f(\theta_i, B)$ `// Update the agents`
return $\{\theta_i\}_{i=1}^N$

For each environment and setting we use 5 seeds and $N = 10$ agents in the ensemble. Reported results is thus average over 50 agents.

D Definition of Replay Ratio

Here we give an exact definition of replay ratio that we use throughout this paper: 1) when training a single agent or an ensemble of agents with *separate* replay buffers, replay ratio refers to the ratio of the number of gradient updates that each agent will experience to the number of *their own* interaction steps; 2) when training an ensemble with a shared replay buffer, replay ratio refers to the number of gradient updates that each agent will experience to the *total number* of interaction steps of the entire ensemble. We find this definition to make most sense because it considers the actual number of environment transitions that the agent has access to. It also reflects how previous work typically sets this hyperparameter in practice (Osband et al., 2016; Peng et al., 2020).

E Additional Results for the Main Experiment

E.1 Numerical Results

Please see Table 2 for results for SAC and Table 3 for results for DrQ. We highlight the row with the highest IQM.

E.2 Ensemble size ablation

In Figure 6 we presents an ablation of the ensemble size N. For SAC, we use replay ratio 1 and thus we do not perform resets. For DrQ, we perform resets as it reaches better performance. As shown in the figure, most benefits of using an ensemble can be gained from a small ensemble size of 5, while going to 10 also provides additional, albeit marginal, improvements.

E.3 Comparison with Other Algorithms

In Table 4 we show a comparison with SR-SAC (Anonymous, 2022), REDQ (Chen et al., 2021), and DDPG (Lillicrap et al., 2016). Note that this comparison is on the DMC15-1M benchmark used in Anonymous (2022), which contains one less environment (`cartpole-swingup_sparse`) than the set of 16 environments used in this work. Therefore all our results in this table are re-calculated

Table 2: Numerical results for SAC. The results are aggregated over the 16 environments for SAC in Table 1 at 1M environment steps (DMC16-1M).

DMC16-1M			
Method	IQM	Mean	Median
SAC (RR=1)	570 (520, 616)	586 (494, 678)	528 (498, 556)
SAC + ens. (best, RR=1)	692 (657, 724)	667 (654, 752)	609 (586, 632)
SAC + ens. (avg, RR=1)	663 (633, 690)	648 (619, 729)	594 (572, 615)
SAC (RR=2)	641 (595, 681)	618 (557, 719)	576 (549, 602)
SAC + ens. (best, RR=2)	689 (658, 718)	718 (651, 799)	612 (591, 632)
SAC + ens. (avg, RR=2)	672 (643, 698)	709 (644, 789)	605 (585, 625)
SAC (RR=4)	612 (567, 652)	583 (517, 677)	562 (534, 590)
SAC + resets (RR=4)	706 (676, 732)	715 (660, 769)	628 (608, 647)
SAC + ens. (best, RR=4)	745 (728, 761)	781 (755, 811)	659 (645, 670)
SAC + ens. (avg, RR=4)	711 (693, 728)	741 (709, 780)	646 (633, 658)
SAC + ens. + resets (best, RR=4)	776 (756, 794)	803 (794, 820)	682 (667, 696)
SAC + ens. + resets (avg, RR=4)	759 (742, 775)	790 (777, 820)	674 (659, 686)
SAC (RR=8)	640 (606, 670)	636 (542, 755)	584 (562, 606)
SAC + resets (RR=8)	746 (717, 771)	715 (697, 790)	652 (630, 671)
SAC + ens. (best, RR=8)	745 (726, 761)	797 (760, 812)	663 (648, 676)
SAC + ens. (avg, RR=8)	716 (696, 732)	769 (704, 795)	648 (634, 660)
SAC + ens. + resets (best, RR=8)	792 (776, 805)	802 (792, 824)	695 (679, 710)
SAC + ens. + resets (avg, RR=8)	771 (757, 784)	780 (766, 824)	686 (673, 699)
SAC (RR=16)	532 (479, 582)	497 (384, 609)	515 (484, 546)
SAC + resets (RR=16)	759 (734, 779)	734 (710, 797)	664 (643, 682)
SAC + ens. (best, RR=16)	690 (670, 710)	766 (690, 808)	631 (617, 645)
SAC + ens. (avg, RR=16)	640 (621, 657)	728 (635, 775)	588 (574, 600)
SAC + ens. + resets (best, RR=16)	**804 (788, 817)**	**812 (804, 830)**	**711 (695, 724)**
SAC + ens. + resets (avg, RR=16)	779 (764, 793)	779 (766, 833)	697 (682, 710)

Table 3: Numerical results for DrQ. The results are aggregated over the 13 environments for DrQ in Table 1 at 2M environment steps (DMC13-Pixel-2M).

DMC13-Pixel-2M			
Method	IQM	Mean	Median
DrQ	456 (409, 503)	402 (283, 569)	466 (439, 494)
DrQ + resets	745 (726, 763)	711 (658, 803)	680 (663, 694)
DrQ + ens. (best)	548 (508, 585)	408 (343, 576)	536 (510, 563)
DrQ + ens. (avg)	509 (469, 548)	399 (308, 564)	508 (482, 534)
DrQ + ens. + resets (best)	801 (787, 814)	837 (807, 847)	720 (711, 729)
DrQ + ens. + resets (avg)	**803 (795, 809)**	**830 (818, 839)**	**716 (710, 722)**

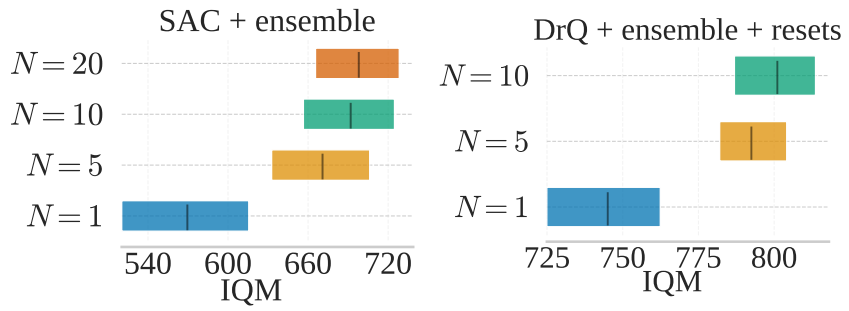

Figure 6: Ensemble size ablation for SAC and DrQ. We report point estimate of the IQM as well as the 95% confidence interval. We report the performance of the best agent based on two rounds of evaluation as previously mentioned. All results are aggregated over 10 random seeds.

without `cartpole-swingup_sparse`. Note that SAC + ens. + resets with replay ratio 16 almost approaches the performance of SR-SAC with replay ratio 128, which is the state of art algorithm on the DeepMind Control Suite. SR-SAC is essentially just SAC + resets but with a different reset frequency and a much higher high replay ratio and therefore much more computationally expensive.

Table 4: Comparison with other algorithms on the DMC15-1M (Anonymous, 2022) benchmark. The results of SR-SAC, REDQ, and DDPG are taken from Anonymous (2022), which uses the same codebase (Kostrikov et al., 2021) as ours. All other results come from this work. For each algorithm used in this work, we report the results under the replay ratio with which it reaches the highest IQM. For each algorithm we also give the replay ratio used. For completeness we also report the performance of SAC under replay ratio 1.

	DMC15-1M		
Method	IQM	Median	Mean
SR-SAC (RR=128)	**805 (726, 867)**	**729 (628, 790)**	**710 (643, 775)**
REDQ (RR=20)	586 (514, 649)	546 (490, 596)	539 (498, 576)
DDPG (RR=1)	514 (450, 572)	492 (440, 540)	489 (450, 526)
SAC + ens. + resets (best, RR=16)	**799 (781, 814)**	**826 (815, 839)**	**705 (688, 720)**
SAC + ens. + resets (avg, RR=16)	780 (763, 794)	822 (811, 842)	694 (678, 707)
SAC + resets (RR=16)	762 (738, 782)	793 (783, 816)	667 (648, 684)
SAC + ens. (best, RR=4)	732 (711, 748)	771 (722, 816)	650 (634, 663)
SAC + ens. (avg, RR=4)	700 (682, 717)	738 (684, 794)	639 (626, 650)
SAC (RR=2)	644 (604, 679)	648 (570, 755)	582 (558, 605)
SAC (RR=1)	569 (524, 613)	696 (515, 739)	531 (505, 557)

E.4 Learning curves

For all sample efficiency curves, 1) for SAC, we use the average of all ensemble members within the ensemble as the ensemble's reported performance 2) for DrQ, we use the first indexed agent for evaluation

SAC In Figure 7 we show the aggregate sample efficiency curves for each replay ratio. In Figure 8, 9, 10, 11, 12 we show the per-environment results for different replay ratios.

DrQ In Figure 13 we show the per-environment results for DrQ.

F Additional Results for the Buffer Merging Experiment

We show results for all 16 environments for the buffer merging environment in Figure 14 and Figure 15 for replay ratio 1 and replay ratio 16, respectively. Note that the sudden merging can have negative impacts on some environments, with the most obvious being the `humanoid` tasks. We hypothesize that this is due to negative effect of the sudden exposure to off-policy data. Note however these experiments are for demonstration purposes only. In practice people will use a shared replay buffer throughout training so this issue will not exist.

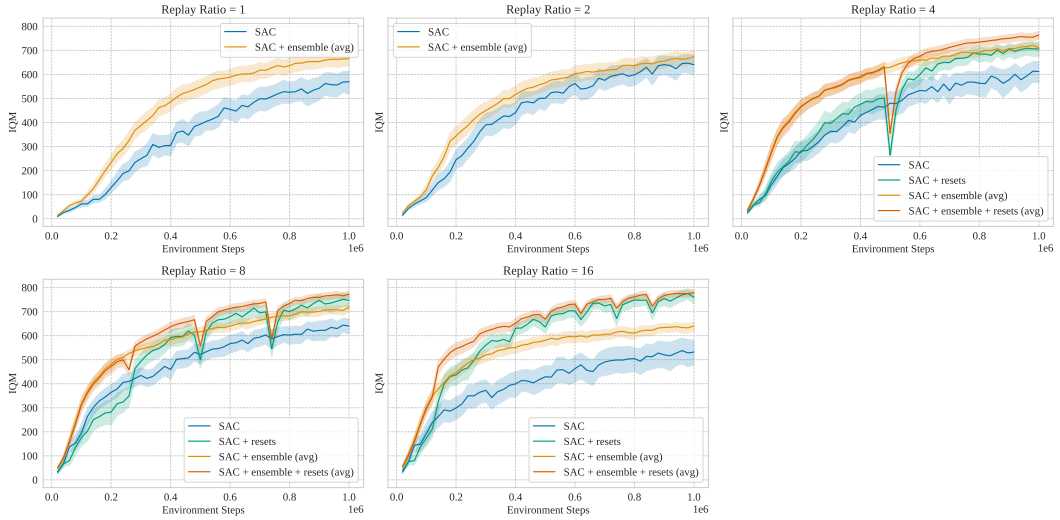

Figure 7: Per replay ratio sample efficiency curves for SAC

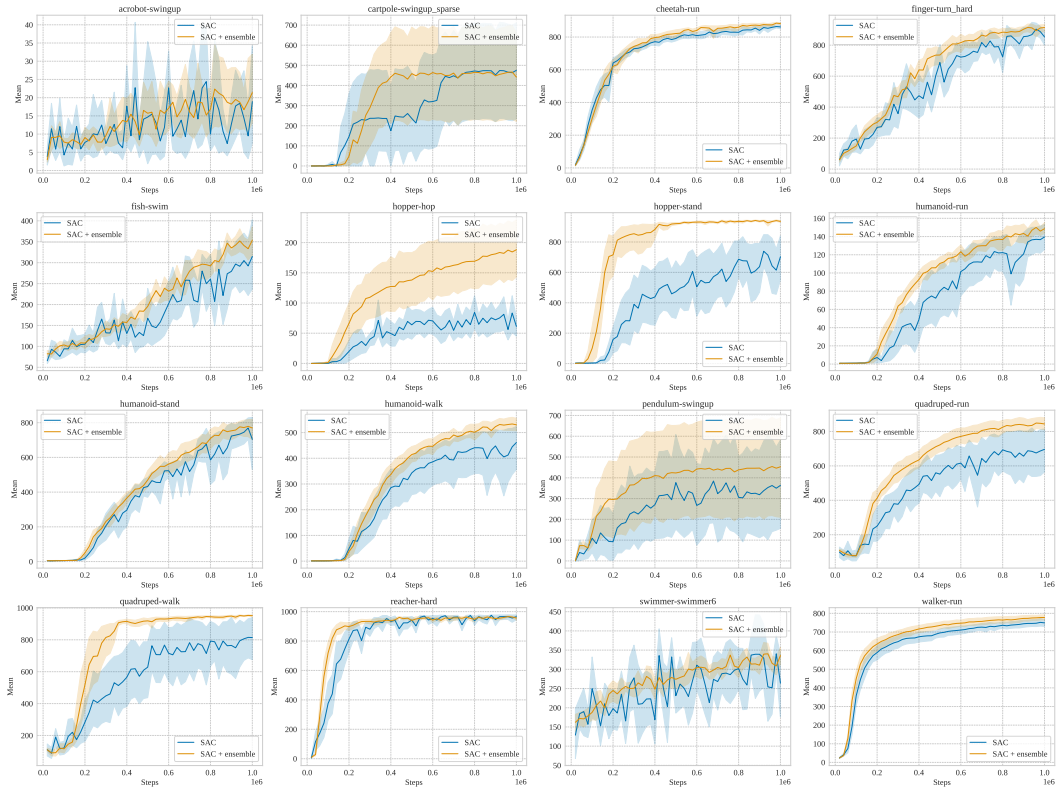

Figure 8: Per-environment results for SAC with replay ratio 1

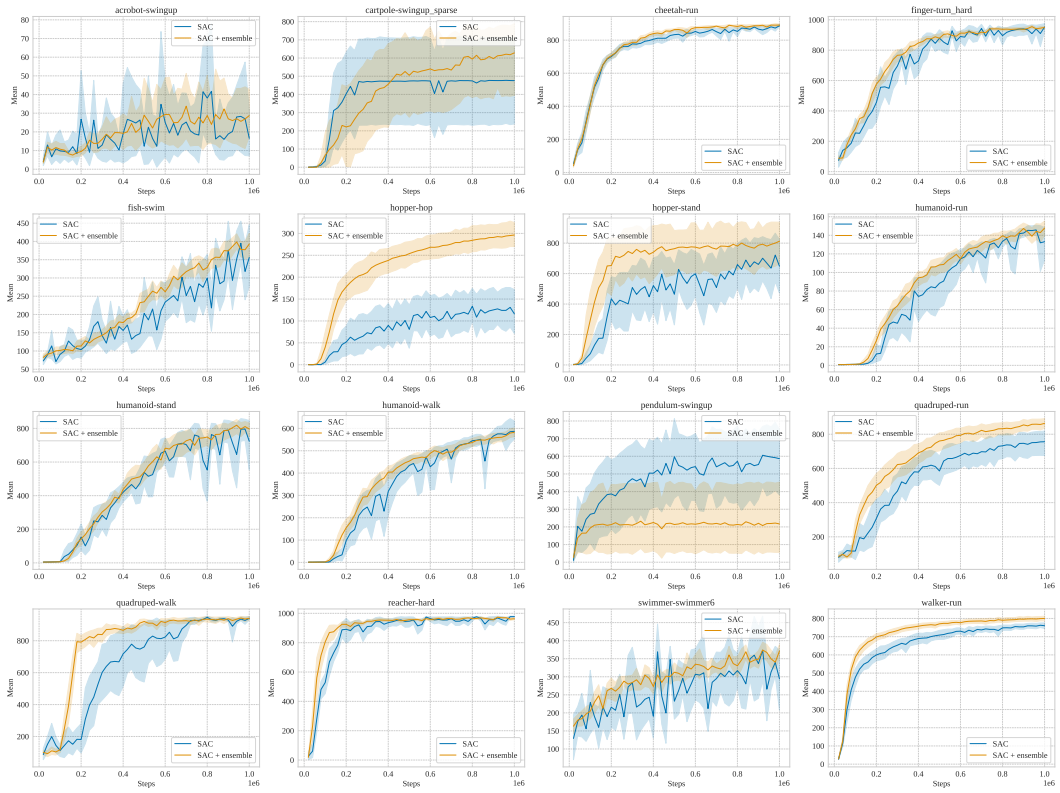

Figure 9: Per-environment results for SAC with replay ratio 2

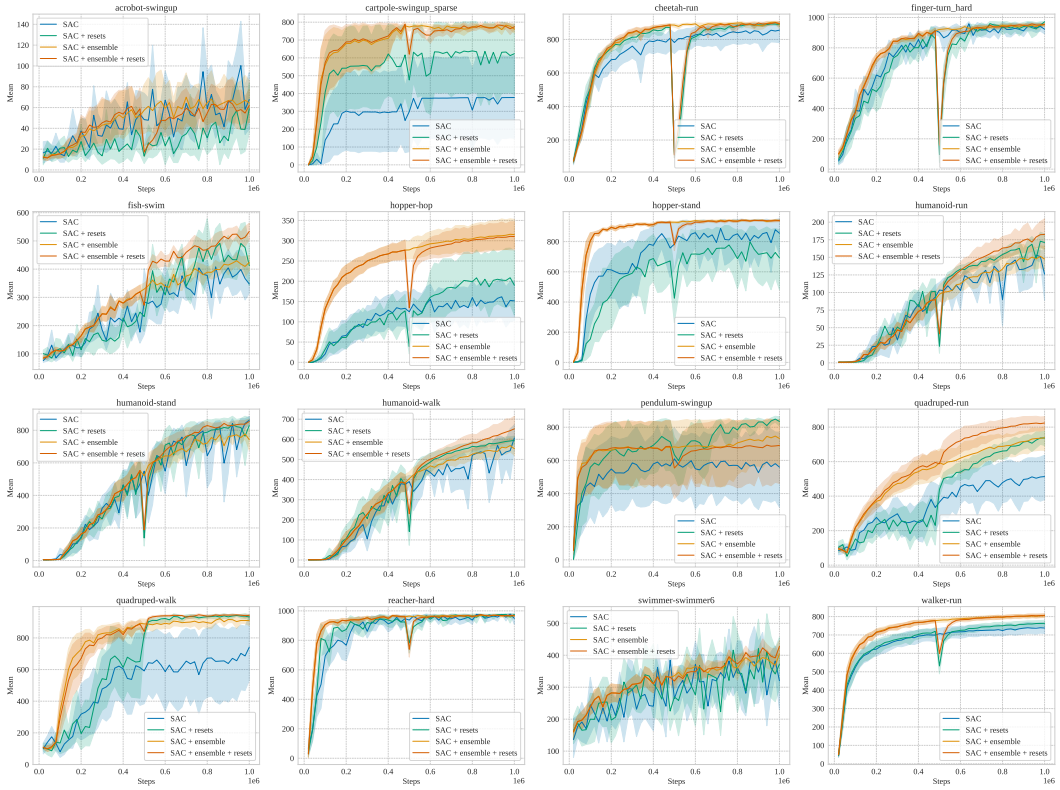

Figure 10: Per-environment results for SAC with replay ratio 4

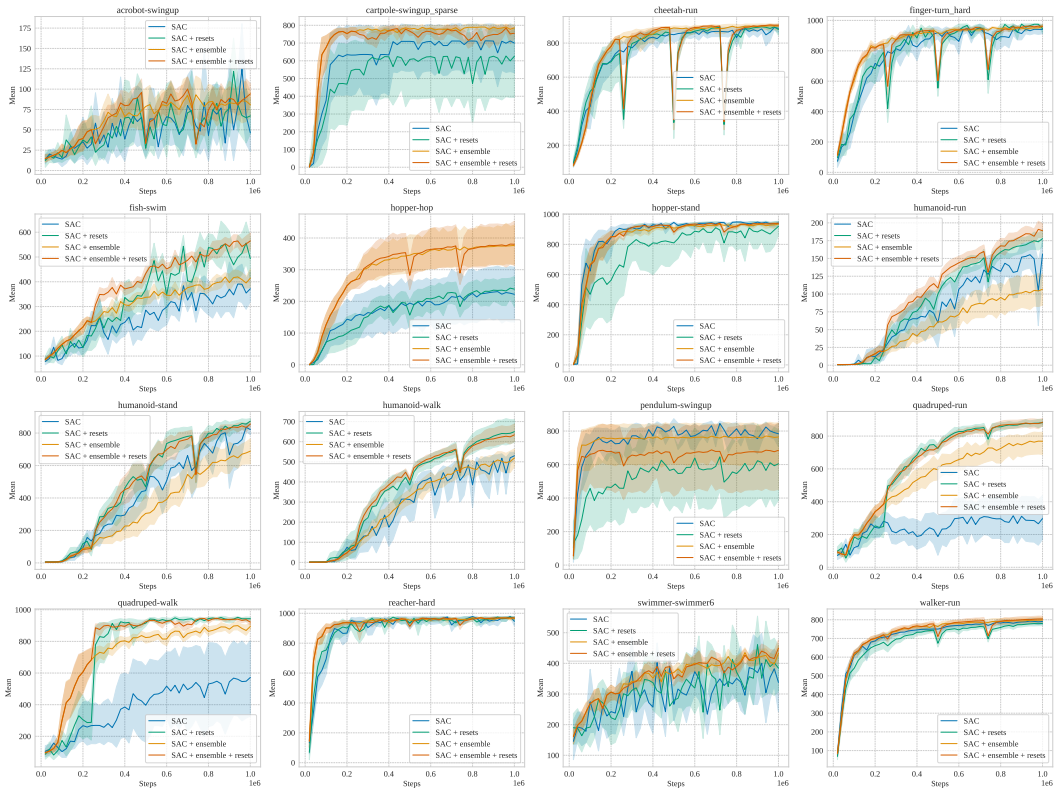

Figure 11: Per-environment results for SAC with replay ratio 8

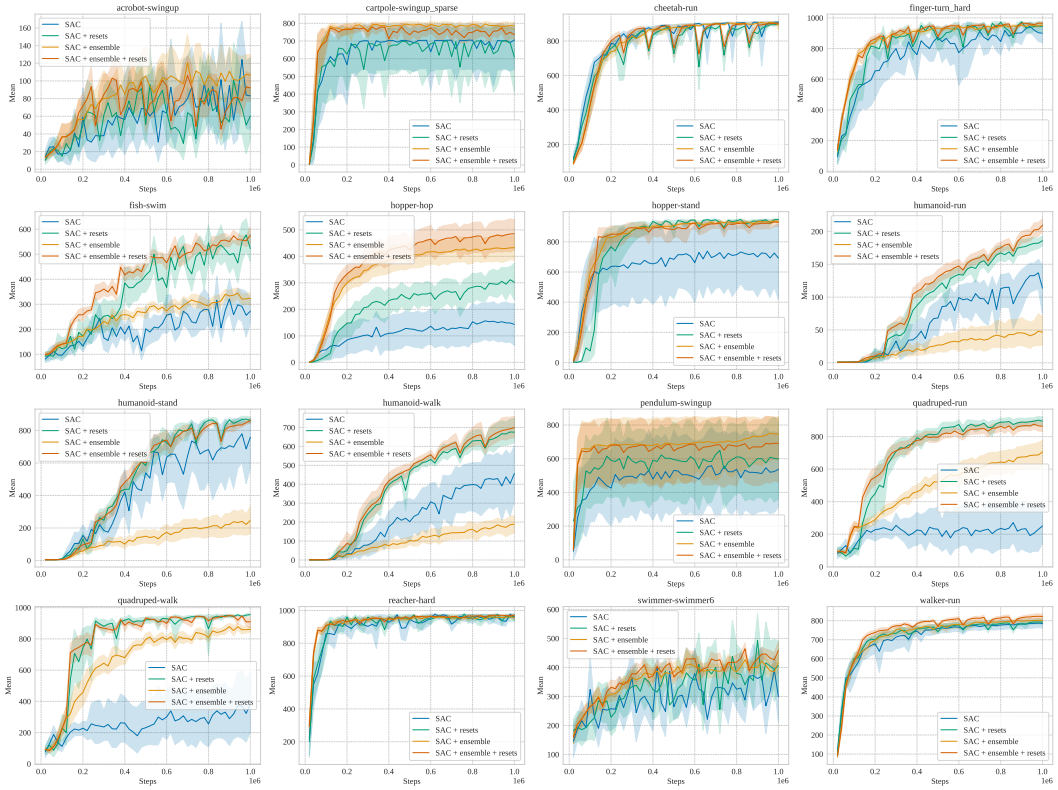

Figure 12: Per-environment results for SAC with replay ratio 16

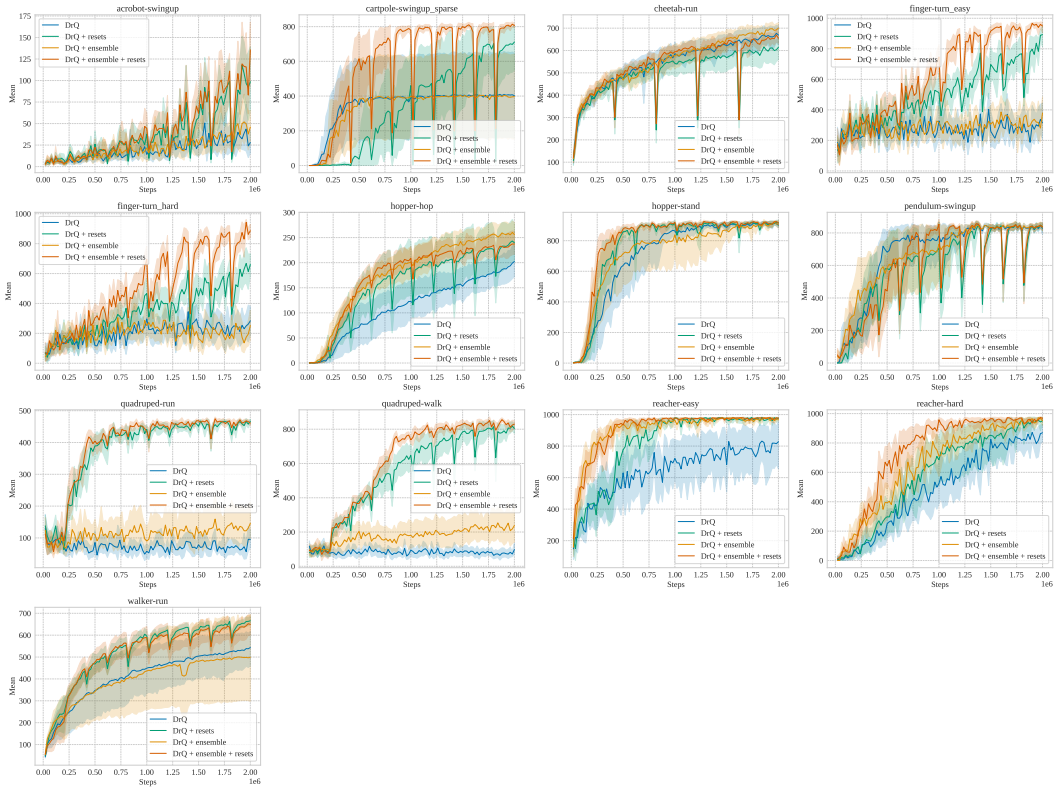

Figure 13: Per-environment results for DrQ

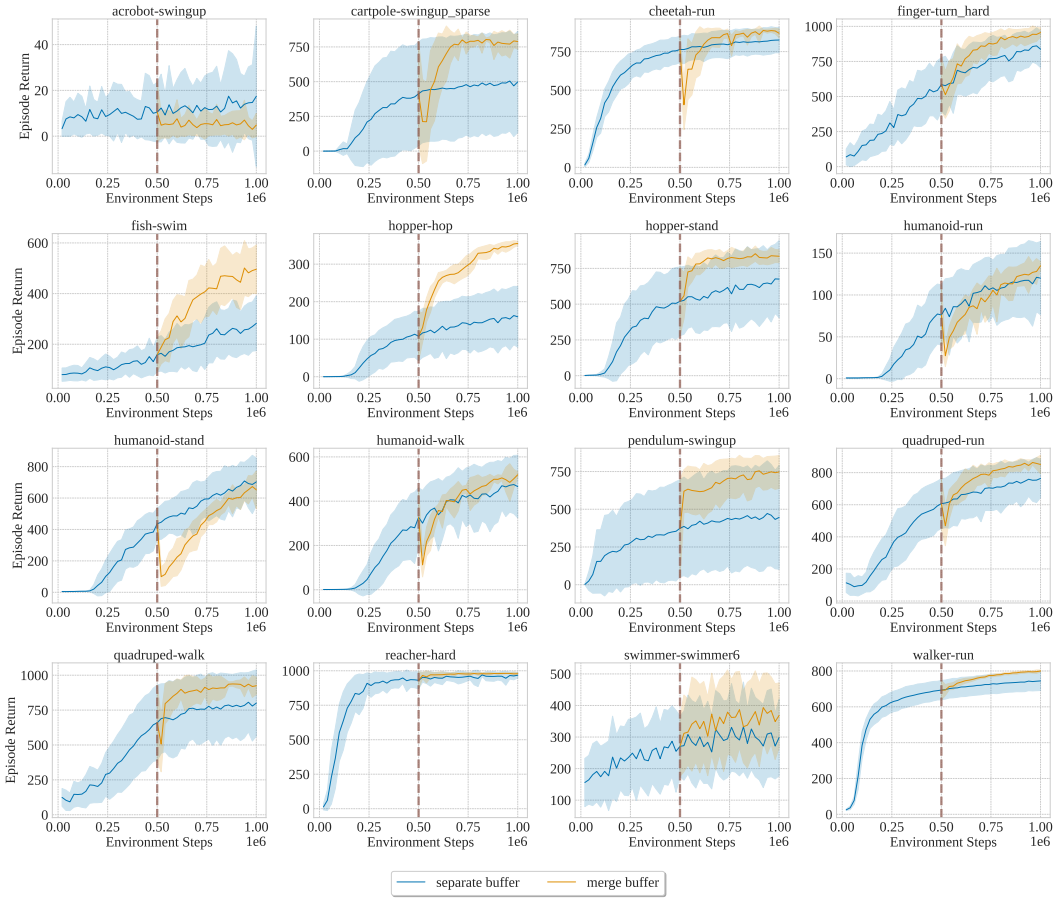

Figure 14: Per-environment results for the buffer merging experiment with replay ratio 1

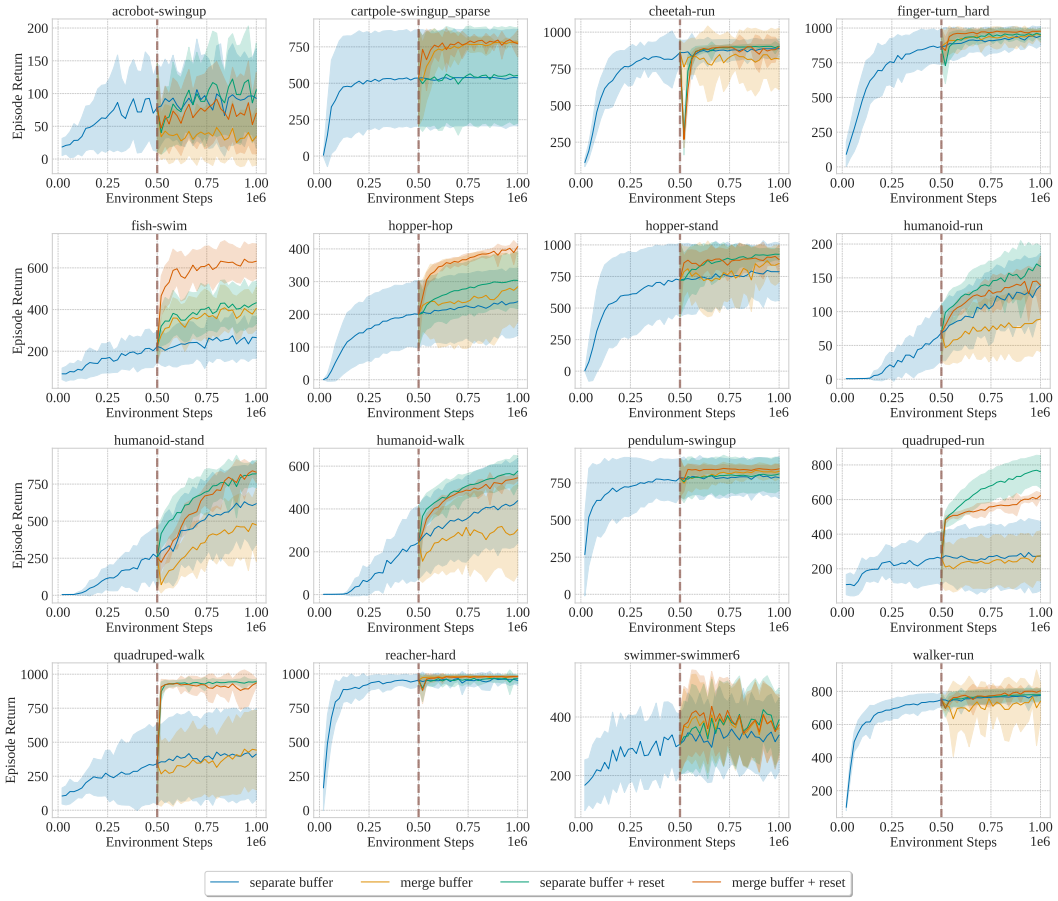

Figure 15: Per-environment results for the buffer merging experiment with replay ratio 16