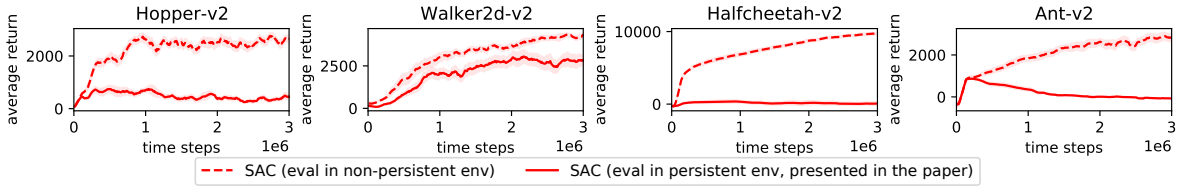
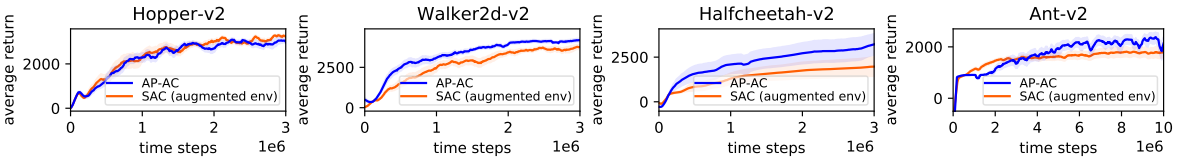
We thank all the reviewers for their valuable comments. We will reflect the comments in the final version of the paper.

[R2: results on SAC] Please let us correct the reviewer's misunderstanding of our experimental results on SAC. In Figure 5 of the paper, the baseline SAC agent is trained on the standard **non-persistent** environments while being evaluated on **c-persistent** environments where the action-persistence is enforced. As shown in the following figure[1], the performance of SAC consistently improves in the non-persistent environment that the agent is trained on, but its naïve projection into a c-persistent policy completely fails since the agent *never* considers the action-persistence during training. We would like to emphasize this is natural and **not** a broken result.



[R3: alternative baseline] Your understanding is correct. The environment modification method, which includes the 'last action' and '$t \bmod L$' in the augmented state, will also enjoy a linear complexity with respect to L. However, this alternative baseline still has some drawbacks compared to our proposed method. First, it is unable to exploit every transition sample to update every timestep's actor and critic, while AP-AC is capable of doing it in Eq. (9-10). Second, there exists a redundancy in the representation of Q-function, i.e. $Q(t, \bar{a}_{\text{last}}, s, a)$ is not succinct compared to ours of $Q(t, s, a)$. This incurs a factor of $|\mathcal{A}|^2$ increase in time complexity of policy evaluation in tabular FA-MDPs. Still, we conducted additional experiments that compare the proposed baseline (i.e. training vanilla SAC agent in the augmented environment that includes $\bar{a}_{\text{last}}$ and $t \bmod L$ in the observation) to AP-AC. As the following figure demonstrates, AP-AC still performs better than or on par with the alternative baseline.

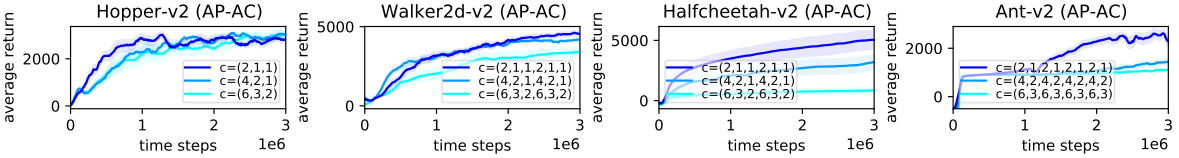


[R3: experiments on standard benchmarks] To the best of our knowledge, there is no standard benchmark to evaluate RL algorithms with multiple action persistence. Adopting our method to real-world tasks remains as future work.

[R3,R4: related works] Thanks for your suggestion. If every action variable's control frequency is same, c-persistent action can be understood as a particular instance of an option in semi-MDP framework, which always lasts c time steps. We will add more discussions on related works such as semi-MDPs, temporally abstract actions, and action-repeats in the final version of the paper.

[R4: comparison to Persistent FQI (PFQI)] Our algorithm, AP-AC, is a non-trivial extension of PFQI. First, PFQI is only applicable to *single* action-persistence and finite action space, while AP-AC can deal with arbitrarily *multiple* action-persistence and both finite and continuous action spaces. Second, PFQI maintains only one Q-function and performs Bellman optimality backup followed by action-persistence backup k times. Each optimality (or persistence) backup operation requires to solve a regression problem until convergence. As a consequence, it can only work in the batch RL (a.k.a. offline RL) setting. Also, as long as PFQI maintains single Q-function, its extension to online RL is not straightforward. In contrast, AP-AC uses L actors and L critics and simultaneously updates all of them via exploiting their recursive relationship, which enables online learning of the agent as well.

[R4: more comprehensive experiments] The goal of this work is to provide an efficient solution method for the *given* action-persistence c, not finding a proper c to speed up learning. Still, we conducted additional experiments to present the effects on the resulting policy of varying c. As the following figure shows, larger action persistence yields more degradation of asymptotic performance due to a limited degree of freedom of control. AP-AC consistently works well for various c's. We will also include qualitative examples (e.g. videos) in the supplementary material of the final paper.



We will also place the Broader Impact section into the main text, which is currently in the supplementary material.

[1] A performance gap exists compared to those reported in the original SAC paper, due to usage of different hyperparameters such as the number of hidden units per layer, i.e. 100 (ours) / 256 (original SAC paper).