1 Appendix

1.1 Comparison to Offline DAU

Within this paper we focus on mixing smaller frequencies together with larger frequencies. Prior work has identified that training deep q-learning on even one frequency can be challenging as frequencies become smaller [1]. To understand what portion of the mixing challenge is due to instability at smaller frequencies, as opposed to frequency inconsistencies, we implement an offline variant of Deep Advantage Updating (DAU). We replace the learned Q-values with advantage functions that have a limit as $\delta t \to 0$. We try both the default DAU implementation and also an offline modification of DAU where we add all of the additional CQL losses applied to the advantage function with a CQL alpha value of 5. On the FrankaKitchen environment, both DAU variants achieve a final performance of 0.0.

The failure of DAU is likely explained by its incompatibility with the CQL objective. Recall that in the approximate advantage formulation of DAU, the value and advantage networks are updated with the following loss:

$$
\begin{aligned}
Q^i &\leftarrow V_\theta(s_i) + \delta t(\bar{A}_\psi(s^i, a^i) - \max_a \bar{A}_\psi(s^i, a)) \\
\tilde{Q}^i &\leftarrow r^i \delta_t + (1 - d^i)\gamma^{\delta t} V_\theta(s^{i+1}) \\
\mathcal{L} &= ||Q^i - \tilde{Q}^i||_2^2
\end{aligned}
$$

In practice $\tilde{Q}^i$ is a target network updated with Polyak averaging, so the effect of this loss in an offline setting is to minimize $\bar{A}_\psi(s^i, a^i)$ and maximize $\max_a \bar{A}_\psi(s^i, a))$. This directly contradicts the CQL objective, which enforces pessimism on actions that maximize Q-values and optimism on actions within the replay buffer. We believe this conflict is why DAU fails. Further research is necessary to come up with a robust version of offline DAU. We'll include these comparisons in the final version of the paper and discuss the applicability of DAU to the offline RL setting.

1.2 Interpolation and Extrapolation of Learned Policies

To study the robustness of the learned policies across a range of discretizations, we perform inference across a range of unseen discretizations for both the Kitchen and Pendulum environments. Table 1 shows that Adaptive N-step achieves reasonable performance when evaluated on new δt.

Env Name	In Training Data?	δt	Adaptive N-step
pendulum	Yes	10	89.5 ± 8.1
	No	9	86.7 ± 10.3
	No	8	86.7 ± 10.3
	No	7	100.0 ± 0.0
	No	6	86.7 ± 10.3
	Yes	5	92.9 ± 5.7
	No	4	66.7 ± 25.8
	No	3	100.0 ± 0.0
	Yes	2	100.0 ± 0.0
	Yes	1	87.5 ± 7.5
kitchen-complete-v0	No	45	15 ± 14.8
	Yes	40	34.6 ± 8.7
	No	35	28 ± 11.8
	Yes	30	19.9 ± 7.2
	No	25	2.2 ± 1.96

Table 1: Adaptive N-Step maintains strong performance when interpolating across unseen δt and even exhibits some degree of extrapolation to δt outside the range of the training data.

Env Name	Policy Conditioning	δt	Adaptive N-step
kitchen-complete-v0	$\pi_{\delta t=40}(a\|s)$	40	48.3 ± 3.3
	$\pi_{\delta t=30}(a\|s)$	40	21.1 ± 6.5
	$\pi_{\delta t=30}(a\|s)$	30	16.6 ± 8.6
	$\pi_{\delta t=40}(a\|s)$	30	11.7 ± 14.2

Table 2: When a policy is conditioned on a different frequency than is observed during evaluation, performance drops. This suggests that Adaptive N-Step is able to learn to specialize to different frequencies.

1.3 Learning Different Policies for Different Frequencies

One challenge with learning optimal policies for different frequencies is that that optimal policy for one frequency may not be optimal at another frequency. To understand if Adaptive N-Step differentiates between policies at different frequencies, we conduct an experiment where we condition on a different frequency than is observed during evaluation. Table 2 shows that performance drops for policies evaluated outside of the training frequency, suggesting that Adaptive N-Step is able to specialize to different frequencies with δt conditioning.

1.4 Performance Curves by δt

In this section, we provide performance curves broken down by δt for each task to understand how Adaptive N-Step impacts discretizations that have better data or are less stable to train.

Pendulum. We plot performance by discretization on the Pendulum task in Figure 1. For every δt, Adaptive N-Step provides substantial improvement over naïve mixing. Because pendulum is a sparse reward task with a simple state-action space, larger values of N should propagate value more quickly than Adaptive N-Step Returns. For example, the Adaptive N for $\delta t = 0.005, 0.01$, and 0.02 is $N = 4, 2$, and 1, respectively. However, for $\delta t = 0.005$, performance declines with max N even though the value of N used for that discretization is unchanged. Likewise for $\delta t = 0.01$ and $\delta t = 0.02$, the value of N is not correlated with performance in that discretization. These results suggest that more than the absolute choice of N matters for performance.

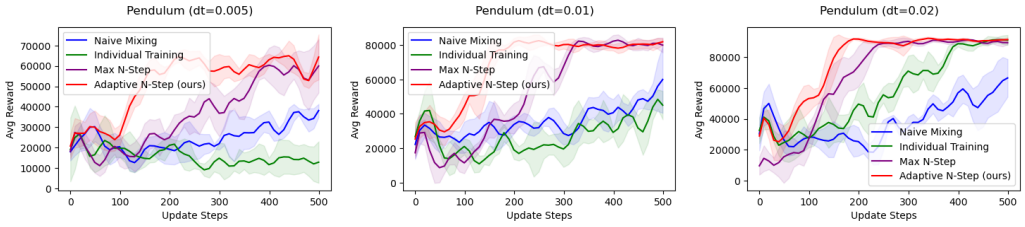


Figure 1: Pendulum performance during training for each δt.

Meta-World Door-Open. We plot performance by discretization on the Meta-World task in Figure 2. Within the door-open task, Adaptive N-Step returns improves the training stability on the smallest discretization $\delta t = 1$ without compromising final performance on coarser discretizations. In this setting naïve mixing fails to stably learn a well-performing policy at the smallest discretization.

Kitchen. We plot performance by discretization on the four FrankaKitchen tasks in Figure 3. Similar to Pendulum, the performance on FrankaKitchen is not fully explained by the absolute value of N chosen for each discretization. For $\delta t = 30$, the same value of N is used for Adaptive N-Step Returns and Max N-Step Returns, but Adaptive N still sees a performance gain. Using any N-step return formulation sees a significant improvement over Naïve Mixing.

1.5 Visualizing the Learned Q-Values

We visualize how Adaptive N-Step Returns affect the quality of the learned Q-function.

Pendulum. In Figure 4, we plot the Q-value over the complete state space of the Pendulum environment for each δt at the end of training. The x-axis is the angle and the y-axis is the velocity of

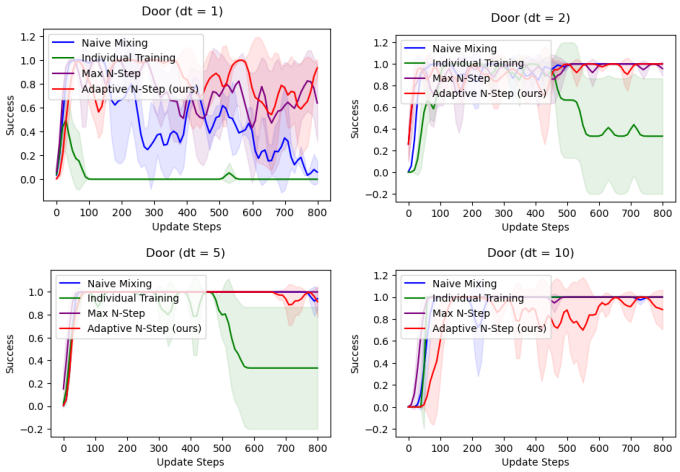


Figure 2: Door-open performance during training for each δt.

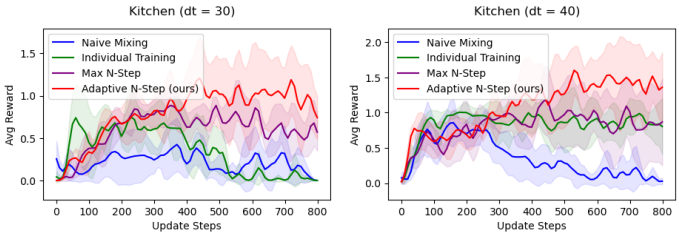


Figure 3: Kitchen performance during training for each δt.

the pendulum. For each state, we measure the Q-value on the action predicted by the final policy. Intuitively, we expect a policy that attains high reward to predict larger Q-values along the lines $y = x$ and $y = -x$, since the velocity of the pendulum should be proportional to its distance from the balance position.

Both Naïve Mixing and Adaptive N-Step converge to similar average Q-values by the end of training, but Adaptive N-Step learns a cleaner Q-value over the state space. This suggests that even though the final Q-values are similar across discretizations with and without adaptive N-step, mixing data without N-step corrodes the quality of the learned Q-value

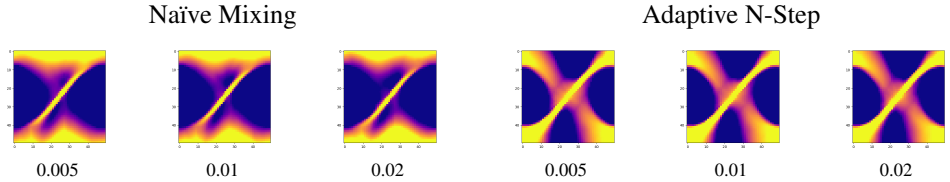
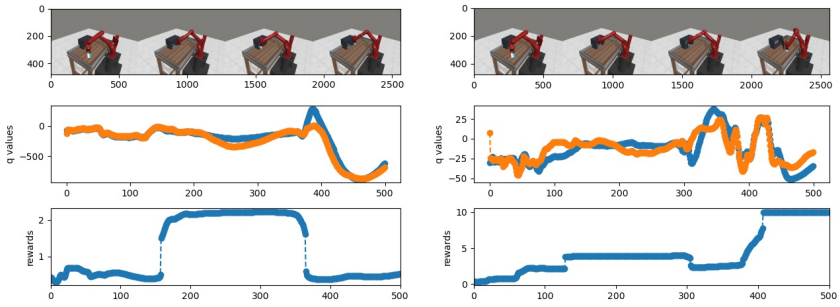


Figure 4: We visualize the Q-values across the entire state-action space for each δt at the end of training. The x-axis is the angle of the pendulum and the y-axis is the angular velocity. The quality of the learned value function is corroded with Naïve Mixing, but not with Adaptive N-step.

Meta-World Door-Open. In Figure 5, we plot the Q-value (middle) for the Meta-World door-open task at a discretization of 1 alongside images taken at regular intervals from the trajectory (top) and the reward attained by the agent (bottom). The Q-values learned with Naïve mixing (left) experience a slight dip with task progress before diverging towards a large negative number. The Q-values learning with Adaptive N-Step (right) progress more closely with task progress and the policy is able to complete the task successfully.

Kitchen. In Figure 6, we plot the Q-value (middle) for the Kitchen environment at a discretization of 40 alongside images taken at regular intervals from the trajectory (top) and the reward attained



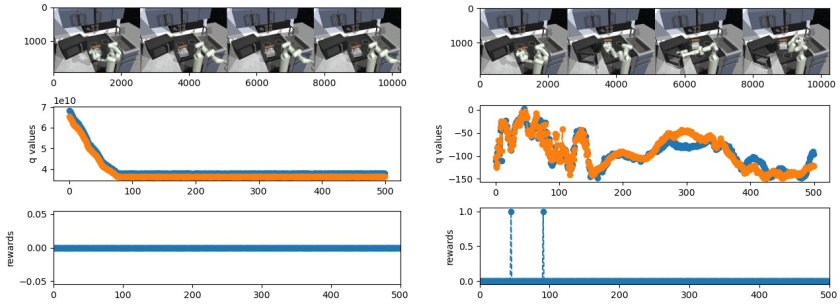
(a) (b)

Figure 5: We visualize both Q-values (blue and orange) for each step of the trajectory for the door open task trained without (left) and with (right) adaptive n-step.

by the agent (bottom). On the left-hand side, we see that at the starting state, the Q-value trained with Naïve Mixing predicts large values at the initial state and then diminishes as the robot moves farther from the interactive task elements. On the right-hand side, we see that, although the Q-value is slightly negative, it correlates well with attained reward. Towards the middle of the trajectory, it still attains a high reward as it moves closer to interactive task elements (i.e., back to the microwave) and continues to drop as the robot moves away from these elements at the end of the trajectory.

1.6 Training and Evaluation Details

All plots and performance numbers are averaged over 5 evaluation trajectories. All Naïve Mixing and Adaptive N-Steps results are averaged over 5 seeds. All Individual Training and Max N-Step Results are averaged over 3 seeds. The plots in the appendix are presented with 95% confidence intervals and smoothed with a 1-D Gaussian filter with a standard deviation of 1.



(a) (b)

Figure 6: We visualize both Q-values (blue and orange) for each step of the trajectory for the FrankaKitchen environment trained without (left) and with (right) adaptive n-step.

References

[1] C. Tallec, L. Blier, and Y. Ollivier. Making deep q-learning methods robust to time discretization. In *International Conference on Machine Learning*, pages 6096–6104. PMLR, 2019.