A Proof of Theorem 1 for Safety Guarantee of CAPS

Theorem 1 *For any MDP M with optimal-cost variation ϵ and any κ-admissible policy π, we have that for any state s, time step t, and accumulated cost $c_{<t}$,*

$$V_t^{\pi,c}(s) \leq \max\{V_t^c(s), \kappa - c_{<t}\} + (T-t)\epsilon$$

Proof: We use proof by induction on the time step t with a base case of T.

Base Case $(t = T)$*:* The base case states that for any s and $c_{<t}$ $V_T^{\pi,c}(s) \leq \max\{V_T^c(s), \kappa - c_{<T}\}$. This follows directly from the facts that for all s, $V_T^{\pi,c}(s) = c(s)$ and that $V_T^c(s) = c(s)$.

Inductive Step: Here we assume the theorem holds for time step $t+1$ and show that it follows for time step t. Our inductive hypothesis states that for all s

$$V_{t+1}^{\pi,c}(s) \leq \max\{V_{t+1}^c(s), \kappa - c_{<t+1}\} + (T-t-1)\epsilon \tag{7}$$

Next expand the value function for t,

$$\begin{aligned}
V_t^{\pi,c}(s) &= c(s) + \sum_{s'} P(s, \pi_t(s, c_{<t}), s') V_{t+1}^{\pi,c}(s') \\
&\leq c(s) + \sum_{s'} P(s, \pi_t(s, c_{<t}), s') \cdot \max\{V_{t+1}^c(s'), \kappa - c_{<t} - c(s)\} \\
&\quad + (T-t-1)\epsilon \\
&= \sum_{s'} P(s, \pi_t(s, c_{<t}), s') \cdot \max\{V_{t+1}^c(s') + c(s), \kappa - c_{<t}\} \\
&\quad + (T-t-1)\epsilon
\end{aligned} \tag{8}$$

The first inequality follows from the inductive hypothesis (7) and the the fact that in the context of s, $c_{<t+1} = c_{<t} + c(s)$. The next equality follows by simply pushing c into the expectation noting that $c(s) = \sum_{s'} P(s, a, s') c(s)$ for any s and a.

We now use the assumption that π is κ-admissible, which implies that,

$$\begin{aligned}
Q_t^c(s, \pi_t(s, c_{<t})) &= c(s) + \sum_{s'} P(s, \pi_t(s, c< t), s') V_{t+1}^c(s') \\
&\leq \max\{V_t^c(s), \kappa - c_{<t}\}
\end{aligned}$$

which we rewrite to

$$\begin{aligned}
\sum_{s'} P(s, \pi_t(s, c< t), s') \cdot \big(V_{t+1}^c(s') + c(s)\big) \\
\leq \max\{V_t^c(s), \kappa - c_{<t}\}
\end{aligned}$$

We can now combine this with the assumption that our MDP has optimal-cost variation ϵ. Since the sum is an expectation this implies that for any s'

$$V_{t+1}^c(s') + c(s) \leq \max\{V_t^c(s), \kappa - c_{<t}\} + \epsilon \tag{9}$$

By combining (8) and (9) we conclude

$$\begin{aligned}
V_t^{\pi,c}(s) &\leq \sum_{s'} P(s, \pi_t(s, c_{<t}), s') \cdot \max\{V_t^c(s), \kappa - c_{<t}\} + \epsilon \\
&\quad + (T-t-1)\epsilon \\
&= \max\{V_t^c(s), \kappa - c_{<t}\} + (T-t)\epsilon
\end{aligned}$$

which completes the proof. □

A.1 Discussion on Assumptions for Safety Guarantees of CAPS

Assumptions 1 and 2 are essential for achieving safety guarantees under CAPS. Intuitively, Assumption 1 ensures that there exists at least one safe action at any intermediate step such that future costs can satisfy the cost constraint based on already observed costs. Having more heads theoretically ensures better coverage of the action space for each agent. Without such a safe action, no algorithm can guarantee safety. We include this assumption to generalize the theorem, making it applicable to other policies or strategies that meet the same condition, not just CAPS. This broadens the applicability of our theoretical analysis.

Assumption 2 helps define the types of Markov Decision Processes (MDPs) for which we can ensure safety guarantees. Without this assumption, we can easily create pathological counterexamples where guarantees would fail. In practice, most of our experimental benchmark environments comply with Assumption 2 with only a small epsilon value. This reflects realistic conditions where good policies demonstrate some degree of recoverability and avoid extreme scenarios such as falling into irrecoverable states or encountering unexpected high rewards.

These assumptions help frame the theoretical results, ensuring that CAPS is validated under reasonable conditions applicable to our benchmark environments.

B Ablation Results for CAPS

B.1 Results of CAPS with Varying Number of Heads

Table 3 compares the performance of the CAPS(IQL) instantiation with varying numbers of heads: 2, 4, and 8. As the number of heads increases, there is a slight improvement in rewards, but this comes with increased complexity and some potential risk. The 2-head configuration is the simplest and safest, consistently maintaining safety across most tasks while delivering reasonable rewards. The 4-head configuration strikes a good balance, offering slightly better rewards than the 2-head setup while still maintaining safety. In contrast, the 8-head configuration, although it achieves the highest rewards in some tasks, introduces greater complexity and sometimes exceeds the safety threshold, marking certain agents as unsafe. For example, in the "PointButton2" task, while the reward remains constant at 0.14 across all configurations, the cost exceeds the safety threshold (1.01) only in the 8-head configuration, highlighting the increased risk associated with more complex setups.

Table 4 compares the performance of the CAPS(SAC+BC) method across 2, 4, and 8-head configurations. The results indicate that while increasing the number of heads can lead to slight improvements in rewards, it often results in variations in cost, with some configurations occasionally exceeding the safety threshold.

The 2-head variant consistently achieves a strong balance between reward and safety, keeping costs well within the acceptable limits. Furthermore, the 2-head setup is less computationally demanding, making it more efficient and faster to train and deploy, solidifying it as a strong choice free from hyper-parameters.

B.2 Results for Shared Architecture Ablation

CAPS with shared backbone agents vs. separate agents. Table 5 compares the performance of SAC+BC and IQL instantiations of CAPS using separate versus shared backbones across various tasks. The results highlight the significant benefits of using a shared backbone, particularly for SAC+BC variant.

`CAPS(SAC+BC)`**.** SAC+BC agents with separate backbones often struggle, as seen in tasks such as CarGoal1 and AntVelocity. In CarGoal1, the reward jumps from 0.00 with separate agents to 0.28 with a shared backbone, while costs are well managed in both cases. Similarly, in AntVelocity, the reward increases dramatically from -0.78 with separate agents to 0.89 with a shared backbone, highlighting how separate backbones lead to poor performance. This is due to SAC's training process, where separate agents result in different Q-function updates, degrading the performance. However, sharing the backbone stabilizes these updates and significantly improves overall decisions.

`CAPS(IQL)`**.** For IQL, the impact of backbone sharing is less pronounced but still positive. Since IQL doesn't query actions outside the dataset during training, the Q-functions are the same whether agents are shared or not. For example, in CarGoal2, the reward slightly improves from 0.15 to 0.16 when using a shared backbone. Although the gains are modest, they suggest that even IQL agents benefit from the consistency of a shared backbone.

Sharing the backbone is crucial for SAC+BC agents, as it prevents performance degradation seen with separate agents. For IQL agents, while the improvements are smaller, shared backbones still offer a slight edge in performance. These findings emphasize the importance of considering backbone sharing to enhance agent performance, particularly in algorithms that rely on querying out-of-distribution actions.

Rollout of shared backbone agents vs. separate ones. The results validate our intuition. Using the IQL algorithm, we found that in 23 tasks, the shared IQL reward head incurred lower costs compared to a separate IQL reward agent. Similarly, the shared SAC+BC reward head incurred lower costs in 17 tasks compared to its separate counterpart. Furthermore, the SAC+BC shared reward head collected more rewards in 30 tasks, whereas the separate agent's disregard for cost information led to sub-optimal performance. This behavior was observed in only 11 tasks when using IQL, highlighting IQL's robustness compared to SAC+BC.

Additionally, in 23 tasks, the shared IQL cost head collected higher rewards compared to a separate IQL cost agent. Even more impressively, in 34 tasks, the shared SAC+BC cost head collected higher rewards compared to its separate counterpart. These findings underscore the effectiveness of a shared network in enhancing both cost and reward performance, confirming our hypothesis that a shared representation leads to superior outcomes in multi-objective learning scenarios. Results are shown in Tables 6, 7, 8, and 9.

B.3 Cost Limit Ablations

Table 10 compares the performance of CAPS(IQL) and CDT across different extra cost configurations $\{5, 10\}$, $\{15, 30\}$, and $\{30, 60\}$, revealing that CAPS(IQL) consistently outperforms CDT in maintaining safety while delivering competitive rewards. CAPS(IQL) effectively manages costs across all configurations, ensuring they remain within the safety threshold, even as costs decreases. In contrast, CDT, while occasionally achieving higher rewards, often does so at the expense of significantly higher costs, leading to many unsafe outcomes. Specifically, CAPS(IQL) achieves safe outcomes in 18, 31, and 33 instances for the three different cost settings, respectively, while CDT manages safety only 11, 19, and 22 times under the same conditions. This

Table 3: Complete evaluation results of normalized reward and cost across different head configurations (2 Heads, 4 Heads, 8 Heads) for the CAPS (IQL) instantiation. The cost threshold is 1. The ↑ symbol denotes that higher reward is better. The ↓ symbol denotes that lower cost (up to threshold 1) is better. Each value is averaged over 3 distinct cost thresholds, 20 evaluation episodes, and 3 random seeds. **Bold:** Safe agents whose normalized cost is smaller than 1. Gray: Unsafe agents.

CAPS(IQL)	2 Heads		4 Heads		8 Heads	
Task	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓
PointButton1	**0.03**	**0.5**	**0.06**	**0.5**	**0.08**	**0.57**
PointButton2	**0.14**	**0.75**	**0.14**	**0.82**	**0.14**	1.01
PointCircle1	**0.5**	**0.78**	**0.52**	**0.7**	**0.5**	**0.76**
PointCircle2	**0.51**	**0.8**	**0.51**	**0.7**	**0.5**	**0.63**
PointGoal1	**0.47**	**0.53**	**0.5**	**0.6**	**0.45**	**0.48**
PointGoal2	**0.33**	**0.8**	**0.35**	**0.83**	**0.36**	**0.85**
PointPush1	**0.19**	**0.29**	**0.17**	**0.29**	**0.15**	**0.46**
PointPush2	**0.13**	**0.64**	**0.13**	**0.57**	**0.12**	**0.62**
CarButton1	**-0.01**	**0.3**	**0**	**0.47**	**-0.01**	**0.25**
CarButton2	**-0.08**	**0.32**	**-0.06**	**0.33**	**-0.17**	**0.35**
CarCircle1	0.54	1.51	0.56	1.64	0.52	1.41
CarCircle2	0.5	1.55	**0.43**	**0.96**	0.52	2
CarGoal1	**0.33**	**0.38**	**0.35**	**0.42**	**0.35**	**0.45**
CarGoal2	**0.16**	**0.62**	**0.19**	**0.73**	**0.16**	**0.6**
CarPush1	**0.2**	**0.31**	**0.17**	**0.35**	**0.18**	**0.27**
CarPush2	**0.07**	**0.75**	**0.07**	**0.65**	**0.06**	**0.84**
SwimmerVelocity	0.43	1.58	0.38	2.62	**0.42**	**0.97**
HopperVelocity	**0.41**	**0.7**	**0.26**	**0.46**	**0.35**	**0.5**
HalfCheetahVelocity	**0.94**	**0.77**	**0.95**	**0.79**	**0.95**	**0.73**
Walker2dVelocity	**0.8**	**0.62**	**0.79**	**0.61**	**0.8**	**0.74**
AntVelocity	**0.95**	**0.64**	**0.96**	**0.5**	**0.97**	**0.54**
SafetyGym Avg	**0.36**	**0.72**	**0.35**	**0.74**	**0.35**	**0.72**
BallRun	**0.19**	**0.94**	**0.26**	**0.77**	0.23	1.13
CarRun	**0.97**	**0.25**	**0.98**	**0.83**	**0.97**	**0.26**
DroneRun	0.47	2.19	0.5	1.55	0.47	1.56
AntRun	**0.61**	**0.9**	**0.61**	**0.82**	**0.62**	**0.79**
BallCircle	**0.69**	**0.59**	**0.7**	**0.63**	**0.7**	**0.64**
CarCircle	**0.69**	**0.65**	**0.67**	**0.68**	**0.68**	**0.71**
DroneCircle	**0.55**	**0.67**	**0.57**	**0.7**	**0.57**	**0.7**
AntCircle	**0.41**	**0.15**	**0.42**	**0.15**	**0.42**	**0.14**
BulletGym Avg	**0.57**	**0.79**	**0.59**	**0.77**	**0.58**	**0.74**
easysparse	**0.11**	**0.34**	**0.16**	**0.21**	0.32	1.58
easymean	**0.01**	**0.2**	**0.23**	**0.5**	**0.03**	**0.09**
easydense	**0.1**	**0.19**	**0.22**	**0.51**	**0.18**	**0.53**
mediumsparse	**0.6**	**0.74**	**0.56**	**1**	**0.61**	**0.28**
mediummean	**0.66**	**0.94**	**0.77**	**0.47**	**0.44**	**0.47**
mediumdense	**0.69**	**0.56**	**0.27**	**0.29**	**0.29**	**0.3**
hardsparse	**0.45**	**0.72**	**0.34**	**0.74**	**0.24**	**0.85**
hardmean	**0.28**	**0.25**	**0.34**	**0.88**	**0.3**	**0.46**
harddense	**0.37**	**0.67**	**0.33**	**0.55**	**0.25**	**0.42**
MetaDrive Avg	**0.40**	**0.54**	**0.36**	**0.57**	**0.29**	**0.43**

demonstrates that CAPS(IQL) is a more reliable and robust method for balancing performance with safety, making it particularly suitable for environments where cost management and risk mitigation are critical. Table 11 presents the SAC+BC instantiation for the extra cost configurations.

Note that CDT requires specifying return and cost targets to generate trajectories. However, the authors do not provide a clear methodology for selecting these return targets; the values are hand-coded for each environment without any explanation of the rationale behind these choices. To address this, we used the provided return targets for the cost sets {10, 20, 40} and {20, 40,

Table 4: Complete evaluation results of normalized reward and cost across different head configurations (2 Heads, 4 Heads, 8 Heads) for the CAPS (SAC+BC) instantiation. The cost threshold is 1. The ↑ symbol denotes that higher reward is better. The ↓ symbol denotes that lower cost (up to threshold 1) is better. Each value is averaged over 3 distinct cost thresholds, 20 evaluation episodes, and 3 random seeds. **Bold**: Safe agents whose normalized cost ≤ 1. Gray: Unsafe agents.

CAPS(SAC+BC)	2 Heads		4 Heads		8 Heads	
Task	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓
PointButton1	0.1	1.23	**0.06**	**0.81**	**0.05**	**0.75**
PointButton2	0.21	1.54	**0.15**	**0.90**	**0.15**	**0.87**
PointCircle1	**0.53**	**0.36**	**0.56**	**0.46**	**0.56**	**0.58**
PointCircle2	0.61	1.38	0.61	1.35	**0.55**	**0.51**
PointGoal1	**0.17**	**0.11**	**0.11**	**0.06**	**0.07**	**0.06**
PointGoal2	**0.20**	**0.55**	**0.18**	**0.47**	**0.15**	**0.46**
PointPush1	**0.19**	**0.38**	**0.15**	**0.47**	**0.17**	**0.37**
PointPush2	**0.15**	**0.70**	**0.14**	**1.00**	**0.16**	**0.76**
CarButton1	0.01	1.17	**0.04**	**0.97**	**0.03**	**0.62**
CarButton2	-0.14	1.17	**-0.14**	**0.92**	**-0.15**	**0.97**
CarCircle1	0.64	3.24	0.66	3.64	0.66	3.81
CarCircle2	0.66	4.98	0.62	4.32	0.65	4.68
CarGoal1	**0.28**	**0.47**	**0.35**	**0.50**	**0.23**	**0.37**
CarGoal2	**0.21**	**0.75**	**0.18**	**0.67**	**0.15**	**0.52**
CarPush1	**0.22**	**0.34**	**0.21**	**0.61**	**0.22**	**0.60**
CarPush2	**0.14**	**0.82**	0.09	1.09	**0.06**	**0.97**
SwimmerVelocity	**0.41**	**0.51**	**0.40**	**0.13**	**0.39**	**0.22**
HopperVelocity	**0.62**	**0.71**	**0.80**	**0.32**	**0.40**	**0.45**
HalfCheetahVelocity	**0.95**	**0.21**	**0.95**	**0.35**	**0.95**	**0.20**
Walker2dVelocity	**0.80**	**0.04**	**0.73**	**0.00**	**0.80**	**0.26**
AntVelocity	**0.89**	**0.67**	**0.95**	**0.77**	**0.96**	**0.73**
SafetyGym Avg	0.37	1.02	**0.37**	**0.94**	**0.34**	**0.89**
BallRun	**0.24**	**0.60**	**0.25**	**0.41**	**0.16**	**0.38**
CarRun	**0.97**	**0.00**	**0.97**	**0.00**	**0.96**	**0.00**
DroneRun	**0.54**	**0.39**	**0.57**	**0.30**	**0.58**	**0.57**
AntRun	**0.53**	**0.25**	**0.38**	**0.03**	**0.26**	**0.12**
BallCircle	**0.59**	**0.19**	**0.68**	**0.31**	**0.69**	**0.35**
CarCircle	**0.61**	**0.29**	**0.67**	**0.42**	**0.63**	**0.80**
DroneCircle	**0.42**	**0.14**	**0.50**	**0.30**	**0.51**	**0.33**
AntCircle	**0.47**	**0.42**	**0.42**	**0.02**	**0.47**	**0.06**
BulletGym Avg	**0.55**	**0.29**	**0.56**	**0.22**	**0.53**	**0.32**
Easysparse	0.54	2.14	**0.09**	**0.58**	**0.09**	**0.49**
Easymean	**0.16**	**0.41**	**0.16**	**0.44**	0.47	1.94
Easydense	**0.11**	**0.15**	**0.20**	**0.71**	0.26	1.24
mediumsparse	**0.65**	**0.99**	**0.50**	**0.87**	**0.37**	**0.65**
mediummean	**0.13**	**0.23**	0.59	1.47	0.80	1.66
mediumdense	**0.69**	**0.80**	0.59	1.23	**0.41**	**0.62**
hardsparse	**0.10**	**0.18**	**0.13**	**0.25**	**0.30**	**0.48**
hardmean	**0.19**	**0.30**	**0.11**	**0.34**	**0.25**	**0.56**
harddense	**0.22**	**0.45**	**0.16**	**0.27**	**0.15**	**0.24**
MetaDrive Avg	**0.31**	**0.63**	**0.28**	**0.68**	**0.34**	**0.88**

80} to interpolate or extrapolate return targets for the additional cost sets {5, 15, 30} and {10, 30, 60}. This approach helps ensure consistency across different cost sets, although it may introduce some uncertainty in the resulting trajectories.

B.4 Results for Fitted-Q Evaluation (FQE) Variants of CAPS

The table in Appendix 12 showcases the ablation study on off-policy evaluation (OPE) by providing a detailed comparison of the performance of different Q-value function configurations within the CAPS framework across all environments. The table

Table 5: Comparison of evaluation results for normalized reward and cost across different environments of SAC+BC and IQL agents, examining the impact of using separate versus shared backbones. The cost threshold is set to 1. The ↑ symbol indicates that a higher reward is better, while the ↓ symbol indicates that a lower cost (up to the threshold of 1) is better. Each value is averaged over 3 distinct cost thresholds, 20 evaluation episodes, and 3 random seeds.

CAPS	SAC+BC Seperate Agents		SAC+BC Shared Backbone		IQL Seperate Agents		IQL Shared Backbone	
Task	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓
PointButton1	**-0.10**	**0.15**	0.10	1.23	**0.02**	**0.49**	**0.03**	**0.50**
PointButton2	**-0.03**	**0.46**	0.21	1.54	**0.14**	**0.97**	**0.14**	**0.75**
PointCircle1	**0.49**	**0.97**	**0.53**	**0.36**	0.43	1.32	**0.50**	**0.78**
PointCircle2	**0.14**	**0.34**	0.61	1.38	**0.51**	**0.80**	**0.51**	**0.80**
PointGoal1	**0.02**	**0.31**	**0.17**	**0.11**	**0.37**	**0.45**	**0.47**	**0.53**
PointGoal2	**-0.12**	**0.52**	**0.20**	**0.55**	**0.25**	**0.50**	**0.33**	**0.80**
PointPush1	**-0.07**	**0.07**	**0.19**	**0.38**	**0.15**	**0.38**	**0.19**	**0.29**
PointPush2	**-0.13**	**0.05**	**0.15**	**0.70**	**0.10**	**0.58**	**0.13**	**0.64**
CarButton1	**-0.09**	**0.36**	0.01	1.17	**-0.03**	**0.19**	**-0.01**	**0.30**
CarButton2	**-0.05**	**0.19**	-0.14	1.17	**-0.08**	**0.35**	**-0.08**	**0.32**
CarCircle1	-0.08	1.16	0.64	3.24	0.55	1.62	0.54	1.51
CarCircle2	**0.24**	**0.20**	0.66	4.98	0.49	1.62	0.50	1.55
CarGoal1	**0.00**	**0.20**	**0.28**	**0.47**	**0.31**	**0.40**	**0.33**	**0.38**
CarGoal2	**0.00**	**0.48**	**0.21**	**0.75**	**0.15**	**0.60**	**0.16**	**0.62**
CarPush1	**-0.20**	**0.02**	**0.22**	**0.34**	**0.19**	**0.36**	**0.20**	**0.31**
CarPush2	**-0.13**	**0.04**	**0.14**	**0.82**	**0.09**	**0.65**	**0.07**	**0.75**
SwimmerVelocity	**0.13**	**0.81**	**0.41**	**0.51**	0.43	2.08	0.46	1.95
HopperVelocity	**0.03**	**0.44**	**0.62**	**0.71**	**0.37**	**0.83**	**0.46**	**0.80**
HalfCheetahVelocity	**0.73**	**0.99**	**0.95**	**0.21**	**0.93**	**0.63**	**0.94**	**0.78**
Walker2dVelocity	**0.05**	**0.30**	**0.80**	**0.04**	**0.75**	**0.70**	**0.79**	**0.68**
AntVelocity	**-0.78**	**0.00**	**0.89**	**0.67**	**0.94**	**0.71**	**0.95**	**0.62**
SafetyGym Avg	**0.00**	**0.38**	0.37	1.02	**0.34**	**0.77**	**0.36**	**0.75**
BallRun	0.23	1.77	**0.24**	**0.60**	**0.16**	**0.90**	**0.19**	**0.94**
CarRun	**0.60**	**0.99**	**0.97**	**0.00**	**0.95**	**0.12**	**0.97**	**0.25**
DroneRun	0.46	3.49	**0.54**	**0.39**	0.47	2.47	0.47	2.19
AntRun	**0.07**	**0.02**	**0.53**	**0.25**	**0.61**	**0.85**	**0.61**	**0.90**
BallCircle	0.53	1.11	**0.59**	**0.19**	**0.67**	**0.72**	**0.69**	**0.59**
CarCircle	0.28	1.83	**0.61**	**0.29**	**0.70**	**0.70**	**0.69**	**0.65**
DroneCircle	**-0.21**	**0.67**	**0.42**	**0.14**	**0.55**	**0.76**	**0.55**	**0.67**
AntCircle	**0.01**	**0.04**	**0.47**	**0.42**	**0.39**	**0.19**	**0.41**	**0.15**
BulletGym Avg	0.25	1.24	**0.55**	**0.29**	**0.56**	**0.84**	**0.57**	**0.79**
easysparse	**-0.04**	**0.17**	0.54	2.14	**0.11**	**0.30**	**0.11**	**0.34**
easymean	**-0.06**	**0.08**	**0.16**	**0.41**	**0.01**	**0.20**	**0.01**	**0.20**
easydense	**-0.05**	**0.10**	**0.11**	**0.15**	**0.11**	**0.19**	**0.10**	**0.19**
mediumsparse	**-0.08**	**0.07**	**0.65**	**0.99**	**0.59**	**0.85**	**0.60**	**0.74**
mediummean	**-0.06**	**0.13**	**0.13**	**0.23**	**0.64**	**0.90**	**0.66**	**0.94**
mediumdense	**-0.07**	**0.10**	**0.69**	**0.80**	**0.66**	**0.56**	**0.69**	**0.56**
hardsparse	**0.03**	**0.42**	**0.10**	**0.18**	**0.43**	**0.72**	**0.45**	**0.72**
hardmean	**-0.03**	**0.11**	**0.19**	**0.30**	**0.23**	**0.11**	**0.28**	**0.25**
harddense	**-0.01**	**0.14**	**0.22**	**0.45**	**0.32**	**0.53**	**0.37**	**0.67**
MetaDrive Avg	**-0.04**	**0.15**	**0.31**	**0.63**	**0.34**	**0.48**	**0.36**	**0.51**

compares the original CAPS method, which uses offline RL-based Q^c and Q^r, with two FQE-based variants: 1) the reward-cost FQE approach, which employs FQE-based $\hat{Q}^r$ and $\hat{Q}^c$ for both reward and cost policies, and 2) the reward FQE method, which uses FQE-based $\hat{Q}^r$ for rewards while retaining Q^c from offline RL for costs. The results in the table show that the reward-cost FQE method often produces overly conservative policies, leading to lower rewards across various environments. This is because the $\hat{Q}^c$ functions frequently estimate that actions exceed the cost limit, resulting in the selection of low-cost but low-reward actions. In contrast, the reward FQE method, which uses FQE-based $\hat{Q}^r$ combined with the offline RL-based Q^c, achieves better performance by maintaining higher rewards while still adhering to safety constraints, as indicated by the blue-highlighted entries in the table. These findings support the ablation study's conclusion that while FQE can effectively differentiate between actions

Table 6: SAC+BC performance comparison of shared backbone cost head and separate cost agent across different environments.

	Shared Backbone Cost Head				Separate Cost Agent			
Environment	Reward	Normalized Reward	Cost	Episode Length	Reward Return	Normalized Reward	Cost	Episode Length
PointButton1	-0.01	0.00	18.25	1000.00	-1.57	-0.04	0.05	1000.00
PointButton2	5.89	0.14	29.85	1000.00	-2.32	-0.05	3.70	1000.00
PointCircle1	40.49	0.49	2.90	500.00	41.63	0.52	10.40	500.00
PointCircle2	38.82	0.55	9.90	500.00	35.12	0.44	0.00	500.00
PointGoal1	3.60	0.12	5.70	1000.00	-3.91	-0.13	0.00	1000.00
PointGoal2	4.39	0.16	13.65	1000.00	-3.64	-0.13	0.45	1000.00
PointPush1	2.97	0.18	17.60	1000.00	-15.80	-0.96	0.00	1000.00
PointPush2	1.76	0.12	24.00	1000.00	-0.50	-0.03	0.90	1000.00
CarButton1	0.84	0.02	72.55	1000.00	-3.06	-0.07	14.95	1000.00
CarButton2	-9.20	-0.22	23.20	1000.00	-1.25	-0.03	4.45	1000.00
CarCircle1	19.30	0.67	113.20	500.00	3.67	-0.26	0.00	500.00
CarCircle2	19.16	0.67	176.40	500.00	9.17	0.19	0.00	500.00
CarGoal1	13.06	0.33	21.50	1000.00	-3.94	-0.10	0.00	1000.00
CarGoal2	7.34	0.25	39.25	1000.00	-6.46	-0.22	0.00	1000.00
CarPush1	3.05	0.19	19.90	1000.00	-0.74	-0.05	0.00	1000.00
CarPush2	1.80	0.12	23.10	1000.00	-1.21	-0.08	23.15	1000.00
SwimmerVelocity	67.01	0.28	0.00	1000.00	-2.23	-0.01	0.00	1000.00
HopperVelocity	641.71	0.32	10.40	387.00	29.29	0.00	0.00	25.15
HalfCheetahVelocity	2397.03	0.85	0.00	1000.00	-260.81	-0.10	0.00	1000.00
Walker2dVelocity	2688.10	0.79	0.00	1000.00	-5.78	-0.01	0.00	7.00
AntVelocity	2047.74	0.69	0.00	1000.00	-1902.42	-0.64	0.00	1000.00
BallRun	140.58	0.09	0.00	100.00	387.43	0.28	0.00	100.00
CarRun	486.34	0.76	0.00	200.00	286.02	0.22	0.00	200.00
DroneRun	365.30	0.53	0.00	200.00	41.21	0.05	0.00	200.00
AntRun	425.31	0.45	0.05	200.00	31.50	0.03	0.00	200.00
BallCircle	475.28	0.54	2.30	200.00	0.43	0.00	0.00	200.00
CarCircle	279.19	0.52	0.00	300.00	166.77	0.31	0.00	300.00
DroneCircle	481.65	0.35	0.00	300.00	-23.62	-0.29	36.95	147.40
AntCircle	207.09	0.45	1.20	477.55	0.59	0.00	0.00	156.05
EasySparse	374.23	0.87	53.13	815.75	-4.16	-0.06	0.00	1000.00
EasyMean	99.72	0.19	2.24	605.95	-4.30	-0.07	1.00	18.00
EasyDense	10.01	-0.02	1.80	73.50	12.04	-0.02	1.00	85.75
MediumSparse	240.24	0.88	19.44	724.70	-3.48	-0.08	1.00	21.00
MediumMean	10.89	-0.02	1.00	119.80	0.49	-0.07	0.00	1000.00
MediumDense	182.84	0.66	6.83	717.25	-3.95	-0.07	1.00	134.00
HardSparse	9.80	-0.02	1.00	133.55	9.01	-0.02	1.00	92.00
HardMean	64.68	0.10	2.52	532.95	-0.13	-0.04	1.00	108.00
HardDense	109.53	0.20	3.57	837.05	-6.21	-0.05	1.00	311.00

in terms of rewards, it struggles with the precision required for cost estimation, which is critical for safety in decision-making. As such, the original CAPS method, which relies on offline RL-based Q^c for cost estimation, remains a robust approach for balancing reward maximization and safety.

C Experimental Details

C.1 Environment descriptions

The environments designed for testing safe offline reinforcement learning (RL) methods are built on different simulators, each tailored to specific tasks and agents.

Safety-Gymnasium (Ray, Achiam, and Amodei 2019; Ji et al. 2024): Developed using the Mujoco physics simulator, Safety-Gymnasium offers a variety of environments focused on safety-critical tasks. The Car agent has several tasks, including Button, Push, and Goal, each available in two difficulty levels, requiring the agent to navigate hazards while achieving specific objectives. For instance, in the Goal task, the agent moves towards multiple goal positions, with each new goal randomly reset upon completion. In the Push task, the agent must move a box to different goal positions, with new locations generated after each success. The Button task involves navigating to and pressing goal buttons scattered across the environment. Additionally, Safety-Gymnasium includes velocity constraint tasks for agents such as Ant, HalfCheetah, and Swimmer. In the Velocity task, the robot coordinates leg movement to move forward, while in the Run task, it starts in a random direction and speed to reach the opposite side of the map. The Circle task rewards agents for following a path along a green circle while avoiding the red region outside. Tasks are named by combining the agent, task, and difficulty level (e.g., CarPush1), reflecting the complexity and objectives of each scenario. Figure 2 illustrates these tasks.

Table 7: SAC+BC performance comparison of shared backbone reward head and separate reward agent across different environments.

	Shared Backbone Reward Head				Separate Reward Agent			
Environment	Reward	Normalized Reward	Cost	Episode Length	Reward Return	Normalized Reward	Cost	Episode Length
PointButton1	3.43	0.08	16.35	1000.00	3.28	0.08	26.65	1000.00
PointButton2	14.56	0.34	77.10	1000.00	0.59	0.01	107.50	1000.00
PointCircle1	47.99	0.67	189.50	500.00	49.60	0.71	193.75	500.00
PointCircle2	47.17	0.80	212.45	500.00	41.69	0.63	171.60	500.00
PointGoal1	20.11	0.67	31.45	1000.00	19.13	0.64	49.90	1000.00
PointGoal2	15.26	0.55	67.30	1000.00	-1.52	-0.05	72.65	1000.00
PointPush1	2.90	0.18	22.50	1000.00	0.32	0.02	98.40	1000.00
PointPush2	2.49	0.17	29.85	1000.00	-1.51	-0.10	28.75	1000.00
CarButton1	0.97	0.02	47.75	1000.00	4.43	0.10	276.95	1000.00
CarButton2	-4.37	-0.10	52.80	1000.00	4.38	0.10	297.45	1000.00
CarCircle1	19.94	0.70	207.10	500.00	14.15	0.36	51.95	500.00
CarCircle2	20.88	0.76	275.55	500.00	15.89	0.52	244.90	500.00
CarGoal1	15.67	0.39	35.65	1000.00	2.23	0.06	89.15	1000.00
CarGoal2	6.46	0.22	28.60	1000.00	-1.34	-0.05	69.85	1000.00
CarPush1	2.94	0.18	16.90	1000.00	0.50	0.03	0.00	1000.00
CarPush2	2.04	0.13	26.15	1000.00	0.22	0.01	139.55	1000.00
SwimmerVelocity	140.33	0.59	177.35	1000.00	72.80	0.30	95.45	1000.00
HopperVelocity	1753.94	0.92	530.80	1000.00	752.65	0.38	189.20	198.70
HalfCheetahVelocity	2744.72	0.98	86.10	1000.00	4488.50	1.60	975.70	1000.00
Walker2dVelocity	2697.91	0.79	0.00	1000.00	181.58	0.05	14.60	88.65
AntVelocity	2887.08	0.97	30.90	1000.00	-2319.29	-0.78	0.00	1000.00
BallCircle	786.73	0.89	63.95	200.00	765.98	0.87	60.85	200.00
CarCircle	430.84	0.81	93.65	300.00	455.02	0.85	90.15	300.00
DroneCircle	851.22	0.82	90.50	300.00	48.63	-0.20	13.35	65.40
AntCircle	376.99	0.82	139.65	462.05	15.13	0.03	14.10	307.70
BallRun	1419.25	1.07	86.50	100.00	1459.21	1.10	92.00	100.00
CarRun	567.43	0.98	0.50	200.00	576.56	1.01	79.35	200.00
DroneRun	487.25	0.71	100.60	200.00	570.95	0.83	145.40	189.75
AntRun	667.27	0.70	74.55	200.00	42.55	0.04	0.00	200.00
EasySparse	134.53	0.28	28.91	254.85	-4.17	-0.06	1.00	16.00
EasyMean	59.87	0.09	11.39	152.20	6.04	-0.04	3.25	39.00
EasyDense	59.50	0.10	8.94	228.05	-4.14	-0.06	1.00	16.00
MediumSparse	108.45	0.36	25.27	160.40	9.53	-0.03	4.02	42.00
MediumMean	8.46	-0.03	2.84	46.00	-2.64	-0.08	1.09	21.00
MediumDense	15.33	0.00	3.43	74.00	3.71	-0.04	2.65	37.00
HardSparse	7.15	-0.02	1.00	110.00	1.60	-0.03	2.10	35.00
HardMean	15.38	-0.01	3.37	67.00	-0.91	-0.04	1.35	25.00
HardDense	82.38	0.14	15.24	159.65	-2.65	-0.04	1.09	21.00

Bullet-Safety-Gym (Gronauer 2022): This environment suite, created using the PyBullet physics simulator, includes four agent types—Ball, Car, Drone, and Ant—along with two primary tasks: Circle and Run. In the Run task, agents navigate through a corridor bounded by safety lines that they can cross without physical collision, though doing so incurs a penalty. Additionally, if an agent exceeds a certain speed limit, it accrues extra penalties. In the Circle task, agents are required to move clockwise around a circular path, with rewards increasing as they maintain higher speeds closer to the boundary. Penalties are given if the agent strays outside a predefined safety zone. These environments are designed to evaluate offline reinforcement learning methods with a focus on safety, featuring shorter, more straightforward tasks when compared to those in Safety-Gymnasium. Figure 3 provides a visual representation of these tasks.

MetaDrive (Li et al. 2022): MetaDrive is a self-driving simulation environment based on the Panda3D game engine. It replicates realistic driving conditions with varying levels of road complexity (easy, medium, hard) and traffic density (sparse, mean, dense). Tasks are named according to the combination of road and vehicle conditions. This environment allows the assessment of offline RL algorithms in scenarios that mirror real-world driving challenges. Figure 4 visualizes these tasks.

Each of these environments offers unique challenges for testing offline safe RL methods, from driving simulations to tasks that require careful navigation and hazard avoidance.

C.2 Computation Time

The computation time comparison in Table 13 reveals that both IQL and SAC+BC instantiations of CAPS training are significantly more efficient than CDT training, tested on the HalfCheetah task. While CDT takes approximately 154 minutes, CAPS(IQL) completes the task in a much shorter time, ranging from 24 to 33 minutes, depending on the number of heads. Similarly, CAPS(SAC+BC), though slightly more time-intensive than IQL, still remains considerably faster than CDT, with computation

Table 8: IQL performance comparison of shared backbone cost head and separate cost agent across different environments.

	Shared Backbone Cost Head				Separate Cost Agent			
Environment	Reward	Normalized Reward	Cost	Episode Length	Reward Return	Normalized Reward	Cost	Episode Length
PointButton1	1.44	0.03	21.22	1000.00	-0.75	-0.02	10.43	1000.00
PointButton2	3.07	0.07	21.00	1000.00	4.53	0.11	17.73	1000.00
PointCircle1	24.33	0.10	11.37	500.00	28.70	0.21	15.60	500.00
PointCircle2	34.56	0.42	0.88	500.00	32.91	0.37	0.00	500.00
PointGoal1	5.90	0.20	9.18	1000.00	6.94	0.23	8.33	1000.00
PointGoal2	4.32	0.16	10.03	1000.00	4.45	0.16	12.78	1000.00
PointPush1	2.33	0.14	8.52	1000.00	1.34	0.08	14.20	1000.00
PointPush2	1.79	0.12	21.45	1000.00	1.34	0.09	12.85	1000.00
CarButton1	-0.57	-0.01	8.07	1000.00	-1.22	-0.03	12.62	1000.00
CarButton2	-3.28	-0.08	6.78	1000.00	-5.19	-0.12	15.62	1000.00
CarCircle1	15.69	0.45	34.85	500.00	15.41	0.44	37.63	500.00
CarCircle2	14.23	0.44	59.22	500.00	13.62	0.41	38.67	500.00
CarGoal1	14.21	0.36	14.68	1000.00	11.84	0.30	9.35	1000.00
CarGoal2	3.45	0.12	14.83	1000.00	2.11	0.07	7.18	1000.00
CarPush1	2.85	0.17	7.53	1000.00	2.21	0.13	5.90	1000.00
CarPush2	0.77	0.05	16.13	1000.00	0.62	0.04	15.00	1000.00
SwimmerVelocity	96.85	0.41	28.60	1000.00	78.59	0.33	75.28	1000.00
HopperVelocity	486.36	0.24	8.60	348.05	698.29	0.35	1.20	592.55
HalfCheetahVelocity	2413.98	0.86	0.42	1000.00	2378.76	0.85	0.12	1000.00
Walker2dVelocity	2674.55	0.78	0.07	1000.00	2682.83	0.78	2.97	1000.00
AntVelocity	2499.16	0.84	0.77	1000.00	2504.33	0.84	0.20	1000.00
BallCircle	223.10	0.25	0.50	200.00	135.04	0.15	5.07	200.00
CarCircle	50.83	0.09	0.00	300.00	27.68	0.05	0.00	300.00
DroneCircle	366.74	0.20	0.00	300.00	364.69	0.20	0.23	298.15
AntCircle	131.15	0.28	0.00	444.50	125.54	0.27	0.08	438.78
BallRun	110.83	0.06	0.00	100.00	169.66	0.11	0.00	100.00
CarRun	479.18	0.74	0.60	200.00	512.50	0.83	0.00	200.00
DroneRun	258.95	0.37	71.87	199.18	316.17	0.45	38.65	199.70
AntRun	439.12	0.46	6.05	200.00	523.04	0.55	10.28	200.00
EasySparse	46.93	0.07	3.20	376.07	115.36	0.23	3.17	782.12
EasyMean	23.90	0.00	3.66	123.93	35.04	0.03	3.14	172.10
EasyDense	61.58	0.10	1.69	452.87	98.31	0.19	0.84	794.32
MediumSparse	118.93	0.40	5.29	659.57	167.06	0.59	16.50	514.97
MediumMean	137.99	0.48	12.51	727.23	117.97	0.40	9.86	627.22
MediumDense	179.50	0.65	6.84	824.30	95.94	0.32	1.78	593.70
HardSparse	193.47	0.38	3.84	897.07	49.76	0.07	5.48	168.00
HardMean	107.38	0.19	0.54	928.90	82.52	0.14	4.73	373.23
HardDense	148.81	0.28	4.88	911.10	105.56	0.19	15.80	252.83

times ranging from 44 to 103 minutes. This demonstrates the training efficiency of both IQL and SAC+BC instantiations, particularly when compared to CDT. All experiments were performed on an A40 GPU and an AMD EPYC 7573X 32-Core Processor.

C.3 CAPS Hyperparameters

Table 14 presents the hyperparameters used across different environments (BulletGym, SafetyGym, and MetaDrive) for CAPS, detailing common configurations such as cost thresholds, training steps, and network specifications.

Table 9: IQL performance comparison of shared backbone reward head and separate reward agent across different environments.

	Shared Backbone Reward Head				Separate Reward Agent			
Environment	Reward	Normalized Reward	Cost	Episode Length	Reward Return	Normalized Reward	Cost	Episode Length
PointButton1	4.79	0.12	30.62	1000.00	5.18	0.13	33.47	1000.00
PointButton2	11.95	0.28	65.03	1000.00	12.32	0.29	69.48	1000.00
PointCircle1	50.38	0.73	176.30	500.00	51.06	0.74	192.32	500.00
PointCircle2	47.12	0.79	232.92	500.00	45.49	0.75	199.08	500.00
PointGoal1	20.28	0.67	38.15	1000.00	17.18	0.57	27.38	1000.00
PointGoal2	14.44	0.52	88.05	1000.00	13.70	0.49	73.12	1000.00
PointPush1	3.25	0.20	29.98	1000.00	3.01	0.18	30.65	1000.00
PointPush2	1.75	0.12	29.38	1000.00	1.73	0.12	40.10	1000.00
CarButton1	0.24	0.01	54.80	1000.00	2.97	0.07	43.73	1000.00
CarButton2	-3.71	-0.09	40.52	1000.00	-0.83	-0.02	58.40	1000.00
CarCircle1	20.38	0.73	190.42	500.00	20.47	0.74	197.78	500.00
CarCircle2	20.28	0.73	258.28	500.00	20.98	0.76	270.90	500.00
CarGoal1	16.47	0.41	19.22	1000.00	13.68	0.34	26.05	1000.00
CarGoal2	8.12	0.28	37.63	1000.00	6.29	0.22	35.67	1000.00
CarPush1	2.90	0.18	21.03	1000.00	3.50	0.21	17.03	1000.00
CarPush2	1.05	0.07	27.12	1000.00	1.78	0.12	28.52	1000.00
SwimmerVelocity	53.22	0.22	72.37	1000.00	48.98	0.20	56.58	1000.00
HopperVelocity	654.39	0.33	158.12	191.83	673.98	0.34	145.85	195.38
HalfCheetahVelocity	2793.41	1.00	250.60	1000.00	2797.74	1.00	187.90	1000.00
Walker2dVelocity	3078.16	0.90	206.02	959.28	3122.61	0.91	196.93	988.68
AntVelocity	2895.56	0.97	140.27	1000.00	2964.54	1.00	208.55	1000.00
BallCircle	828.74	0.94	59.08	200.00	838.51	0.95	60.42	200.00
CarCircle	513.94	0.96	92.77	300.00	508.34	0.95	91.67	300.00
AntCircle	309.75	0.67	95.95	451.78	357.56	0.78	124.58	486.78
DroneCircle	906.13	0.89	99.17	300.00	952.44	0.94	98.25	300.00
BallRun	1546.65	1.17	92.33	100.00	1568.23	1.19	92.33	100.00
CarRun	564.23	0.97	5.12	200.00	564.44	0.97	9.20	200.00
DroneRun	391.63	0.57	31.57	200.00	422.06	0.61	22.83	200.00
AntRun	925.83	0.97	146.22	199.58	961.77	1.01	159.13	200.00
EasySparse	119.32	0.24	25.35	230.28	408.73	0.96	69.40	744.73
EasyMean	40.32	0.05	10.02	84.27	303.79	0.70	52.96	728.45
EasyDense	88.30	0.17	11.30	348.38	257.20	0.58	41.75	750.50
MediumSparse	136.17	0.47	15.16	393.50	246.68	0.91	35.13	574.83
MediumMean	213.06	0.78	33.24	490.97	196.74	0.71	28.82	479.43
MediumDense	215.81	0.79	26.60	583.42	195.87	0.71	32.88	352.27
HardSparse	225.52	0.44	29.73	576.53	231.79	0.46	30.85	667.65
HardMean	158.70	0.30	11.33	715.70	208.21	0.40	25.68	605.25
HardDense	176.90	0.34	19.76	554.32	221.44	0.44	31.67	561.67

Table 10: Performance across different extra cost configurations for CAPS(IQL) and CDT. The cost threshold is 1. The ↑ symbol denotes that a higher reward is better. The ↓ symbol denotes that a lower cost (up to threshold 1) is better. Each value is averaged over 20 evaluation episodes, and 3 random seeds. **Bold**: Safe agents whose normalized cost ≤ 1. Gray: Unsafe agents.

Cost Configuration	{5, 10}				{15, 30}				{30, 60}			
	CAPS IQL		**CDT**		**CAPS IQL**		**CDT**		**CAPS IQL**		**CDT**	
Environment	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓
PointButton1	0.02	1.36	0.51	9.56	**0.02**	**0.35**	0.53	3.60	**0.07**	**0.33**	0.54	1.93
PointButton2	0.14	2.84	0.42	9.33	0.15	1.09	0.47	2.95	**0.13**	**0.52**	0.43	1.62
PointCircle1	0.27	1.27	**0.54**	**0.58**	**0.49**	**0.73**	**0.56**	**0.77**	**0.53**	**0.64**	**0.59**	**0.92**
PointCircle2	**0.42**	**0.11**	0.60	1.93	**0.44**	**0.58**	0.62	1.46	0.59	1.01	0.64	1.16
PointGoal1	0.32	1.54	0.69	3.17	**0.46**	**0.75**	0.69	1.30	**0.57**	**0.50**	**0.73**	**0.62**
PointGoal2	0.22	1.12	0.43	5.30	**0.27**	**0.72**	0.49	2.12	**0.37**	**0.62**	0.59	1.39
PointPush1	**0.12**	**0.52**	0.23	3.02	**0.16**	**0.54**	0.25	1.19	**0.2**	**0.35**	**0.21**	**0.45**
PointPush2	0.09	2.08	0.20	4.54	**0.10**	**0.90**	0.20	1.45	**0.15**	**0.45**	0.23	1.02
CarButton1	**-0.04**	**0.62**	0.20	9.04	**-0.02**	**0.24**	0.20	3.32	**0.01**	**0.24**	0.18	1.67
CarButton2	-0.12	1.13	0.25	14.47	**-0.11**	**0.36**	0.24	4.93	**-0.08**	**0.20**	0.13	1.95
CarCircle1	0.46	4.21	0.54	8.50	0.49	1.55	0.54	3.19	0.59	1.30	0.65	2.33
CarCircle2	0.45	4.94	0.63	13.20	0.47	1.53	0.65	4.89	0.54	1.21	0.69	3.11
CarGoal1	0.29	1.24	0.62	3.27	**0.32**	**0.47**	0.66	1.27	**0.31**	**0.27**	**0.67**	**0.75**
CarGoal2	0.12	2.52	0.43	7.01	**0.13**	**0.75**	0.44	2.20	**0.19**	**0.45**	0.57	1.43
CarPush1	**0.17**	**0.48**	0.28	2.23	**0.19**	**0.25**	**0.29**	**0.78**	**0.19**	**0.13**	**0.33**	**0.54**
CarPush2	0.07	2.21	0.15	6.10	**0.10**	**0.85**	0.15	2.11	**0.06**	**0.37**	0.20	1.21
SwimmerVelocity	0.45	3.35	**0.66**	**0.98**	0.49	2.47	**0.68**	**0.96**	0.44	1.47	**0.69**	**0.95**
HopperVelocity	0.26	1.40	0.69	1.16	**0.34**	**0.70**	**0.75**	**0.79**	**0.51**	**0.68**	**0.71**	**0.73**
HalfCheetahVelocity	**0.87**	**0.31**	**0.98**	**0.68**	**0.92**	**0.77**	**0.98**	**0.18**	**0.97**	**0.84**	**0.98**	**0.11**
Walker2dVelocity	**0.79**	**0.09**	**0.80**	**0.16**	**0.8**	**0.54**	**0.78**	**0.15**	**0.81**	**0.86**	**0.78**	**0.14**
AntVelocity	**0.87**	**0.22**	**0.98**	**0.71**	**0.94**	**0.60**	**0.99**	**0.52**	**0.99**	**0.72**	**0.99**	**0.46**
BallCircle	**0.31**	**0.00**	0.70	2.25	**0.65**	**0.53**	0.75	1.28	**0.76**	**0.77**	**0.81**	**0.97**
CarCircle	**0.41**	**0.02**	0.72	2.01	**0.7**	**0.62**	**0.72**	**0.87**	**0.72**	**0.81**	**0.77**	**0.96**
DroneCircle	**0.33**	**0.00**	0.55	1.39	**0.54**	**0.64**	0.59	1.14	**0.58**	**0.81**	0.64	1.07
AntCircle	**0.31**	**0.11**	0.43	6.21	**0.34**	**0.08**	0.45	2.76	**0.45**	**0.20**	0.52	1.60
BallRun	**0.07**	**0.00**	0.32	2.26	0.11	1.52	**0.32**	**0.84**	0.22	1.19	**0.38**	**0.95**
CarRun	**0.97**	**0.41**	**0.99**	**0.94**	**0.97**	**0.26**	**0.99**	**0.66**	**0.98**	**0.23**	**0.99**	**0.63**
DroneRun	0.4	12.06	**0.51**	**0.54**	0.41	3.99	**0.59**	**0.49**	**0.51**	**0.45**	0.62	1.32
AntRun	0.47	1.39	0.70	1.69	**0.59**	**0.90**	**0.72**	**0.85**	**0.69**	**0.90**	**0.73**	**0.79**
EasySparse	**0.09**	**0.77**	**0.07**	**0.00**	**0.10**	**0.32**	**0.18**	**0.00**	**0.19**	**0.54**	**0.52**	**0.74**
EasyMean	**0.00**	**0.67**	0.40	3.69	**0.00**	**0.22**	0.45	1.31	**0.01**	**0.11**	**0.49**	**0.71**
EasyDense	**0.10**	**0.53**	**0.29**	**0.21**	**0.11**	**0.18**	**0.42**	**0.25**	**0.10**	**0.12**	0.44	1.19
MediumSparse	0.51	1.95	0.44	1.16	**0.54**	**0.78**	0.54	1.22	**0.63**	**0.50**	0.65	1.05
MediumMean	0.53	2.55	0.55	1.57	0.64	1.04	**0.35**	**0.99**	**0.73**	**0.64**	**0.36**	**0.61**
MediumDense	0.66	1.66	**0.12**	**0.17**	**0.62**	**0.59**	**0.15**	**0.08**	**0.75**	**0.51**	**0.31**	**0.49**
hardsparse	0.41	2.15	0.24	1.85	**0.45**	**0.86**	**0.20**	**0.20**	**0.48**	**0.53**	**0.30**	**0.89**
hardmean	**0.20**	**0.33**	**0.07**	**0.07**	**0.25**	**0.22**	**0.11**	**0.20**	**0.31**	**0.22**	**0.27**	**0.81**
harddense	**0.23**	**0.85**	0.26	2.77	**0.34**	**0.66**	**0.28**	**0.83**	**0.45**	**0.66**	**0.31**	**0.92**
# **safe**	**18**		**11**		**31**		**19**		**33**		**22**	

Table 11: Performance across different extra cost configurations for CAPS(SAC+BC). The cost threshold is 1. The ↑ symbol denotes that a higher reward is better. The ↓ symbol denotes that a lower cost (up to threshold 1) is better. Each value is averaged over 20 evaluation episodes, and 3 random seeds. **Bold**: Safe agents whose normalized cost ≤ 1. Gray: Unsafe agents.

Cost Configuration	{5, 10}		{15, 30}		{30, 60}	
	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓
PointCircle1	**0.51**	**0.35**	**0.52**	**0.22**	**0.55**	**0.51**
PointCircle2	0.59	3.88	0.60	1.37	0.61	1.05
SwimmerVelocity	**0.32**	**0.14**	**0.37**	**0.47**	**0.49**	**0.63**
HopperVelocity	0.61	2.35	**0.58**	**0.75**	**0.59**	**0.39**
HalfCheetahVelocity	**0.88**	**0.11**	**0.94**	**0.19**	**0.97**	**0.28**
AntVelocity	**0.73**	**0.35**	**0.86**	**0.69**	**0.96**	**0.73**
CarCircle	**0.56**	**0.00**	**0.58**	**0.02**	**0.66**	**0.51**
BallRun	0.25	1.56	**0.25**	**0.67**	**0.26**	**0.68**
AntRun	0.41	0.01	**0.42**	**0.01**	**0.64**	**0.47**
EasyDense	**0.08**	**0.39**	**0.06**	**0.12**	**0.09**	**0.15**
MediumMean	**0.11**	**0.66**	**0.07**	**0.10**	**0.10**	**0.10**
hardsparse	**0.10**	**0.38**	**0.10**	**0.20**	**0.11**	**0.18**

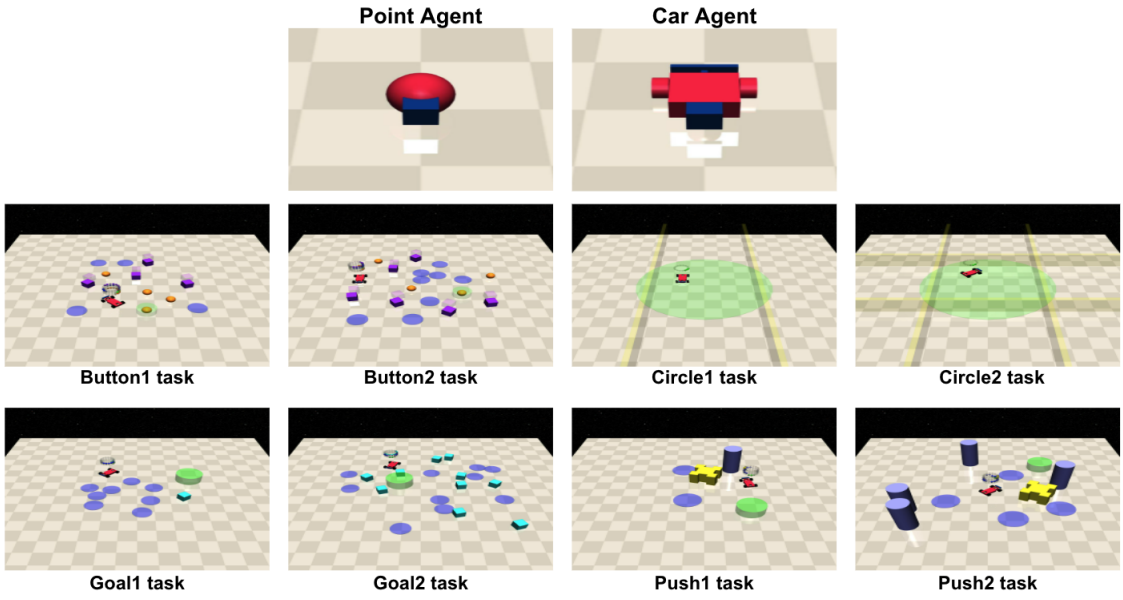

Figure 2: Visualization of the Safety-Gymnasium environments.

Table 12: Performance comparison of FQE based Q-value functions. The cost threshold is 1. The ↑ symbol denotes that a higher reward is better. The ↓ symbol denotes that a lower cost (up to threshold 1) is better. Each value is averaged over 3 distinct cost thresholds, 20 evaluation episodes, and 3 random seeds. **Bold:** Safe agents whose normalized cost is smaller than 1. Gray: Unsafe agents. Blue: Safe agent with the highest reward.

	FQE $\hat{Q}^r$ & $\hat{Q}^c$		Q^c + FQE $\hat{Q}^r$		Q^c & Q^r	
Task	reward ↑	cost ↓	reward ↑	cost ↓	reward ↑	cost ↓
Bullet Safety Gym						
AntCircle	**0.32**	**0.01**	**0.39**	**0.14**	**0.41**	**0.15**
AntRun	**0.43**	**0.36**	**0.6**	**0.88**	**0.61**	**0.90**
CarCircle	**0.18**	**0.14**	**0.7**	**0.66**	**0.69**	**0.65**
DroneCircle	**0.19**	**0.01**	**0.55**	**0.69**	**0.55**	**0.67**
DroneRun	0.40	3.58	0.44	2.97	0.47	2.19
BallCircle	**0.41**	**0.17**	**0.69**	**0.59**	**0.69**	**0.59**
BallRun	**0.06**	**0**	**0.13**	**0.58**	**0.19**	**0.94**
CarRun	**0.94**	**0.16**	**0.95**	**0.12**	**0.97**	**0.25**
Safety Gymnasium Car						
CarButton1	**-0.01**	**0.44**	**-0.03**	**0.18**	**-0.01**	**0.30**
CarButton2	**-0.08**	**0.36**	**-0.09**	**0.34**	**-0.08**	**0.32**
CarCircle1	0.47	1.09	0.54	1.55	0.54	1.51
CarCircle2	0.43	1.41	0.49	1.65	0.50	1.55
CarGoal1	**0.25**	**0.31**	**0.29**	**0.42**	**0.33**	**0.38**
CarGoal2	**0.14**	**0.50**	**0.15**	**0.59**	**0.16**	**0.62**
CarPush1	**0.17**	**0.29**	**0.20**	**0.37**	**0.20**	**0.31**
CarPush2	**0.03**	**0.48**	**0.09**	**0.67**	**0.07**	**0.75**
Safety Gymnasium Point						
PointButton1	**0.01**	**0.38**	**0.02**	**0.40**	**0.03**	**0.45**
PointButton2	**0.09**	**0.62**	**0.13**	**0.94**	**0.14**	**0.75**
PointCircle1	**0.23**	**0.37**	0.44	1.22	**0.50**	**0.78**
PointCircle2	**0.40**	**0.21**	**0.50**	**0.77**	**0.51**	**0.80**
PointGoal1	**0.25**	**0.36**	**0.37**	**0.42**	**0.47**	**0.53**
PointGoal2	**0.14**	**0.38**	**0.25**	**0.53**	**0.33**	**0.80**
PointPush1	**0.14**	**0.21**	**0.16**	**0.38**	**0.19**	**0.29**
PointPush2	**0.11**	**0.56**	**0.12**	**0.60**	**0.13**	**0.64**
Safety Gymnasium Velocity						
AntVelocity	**0.87**	**0.18**	**0.95**	**0.7**	**0.95**	**0.64**
HalfCheetahVelocity	**0.87**	**0.18**	**0.94**	**0.66**	**0.94**	**0.77**
HopperVelocity	**0.24**	**0.25**	**0.43**	**0.72**	**0.41**	**0.70**
SwimmerVelocity	**0.4**	**0.68**	0.43	2.02	0.43	1.58
Walker2dVelocity	**0.76**	**0.06**	**0.71**	**0.72**	**0.80**	**0.62**
Metadrive environments						
easysparse	0.24	1.39	**0.11**	**0.33**	**0.11**	**0.34**
easymean	**0**	**0.16**	**0.01**	**0.20**	**0.01**	**0.20**
easydense	**0.12**	**0.11**	**0.12**	**0.18**	**0.10**	**0.19**
mediumsparse	**0.43**	**0.34**	**0.56**	**0.73**	**0.60**	**0.74**
mediummean	**0.55**	**0.83**	**0.66**	**0.86**	**0.66**	**0.94**
mediumdense	**0.76**	**0.60**	**0.67**	**0.61**	**0.69**	**0.56**
hardsparse	**0.42**	**0.51**	**0.45**	**0.76**	**0.45**	**0.72**
hardmean	**0.20**	**0.06**	**0.23**	**0.06**	**0.28**	**0.25**
harddense	**0.28**	**0.39**	**0.30**	**0.53**	**0.37**	**0.67**

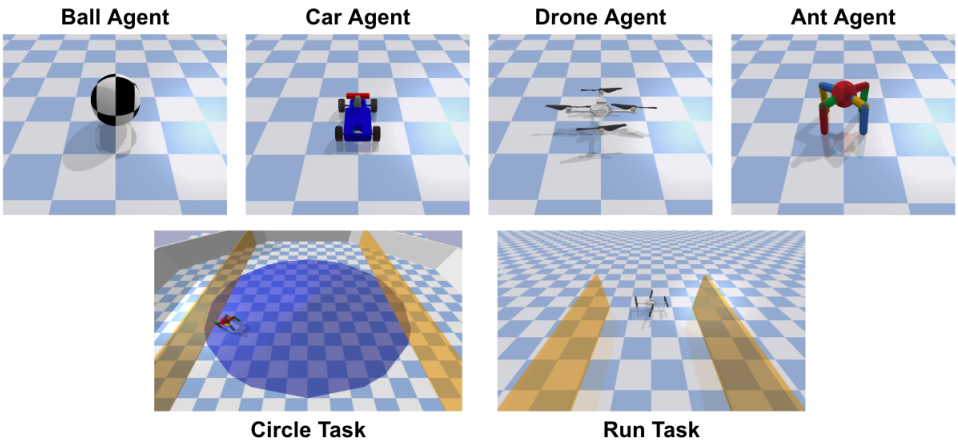

Figure 3: Visualization of the Bullet-Safety-Gym environments.

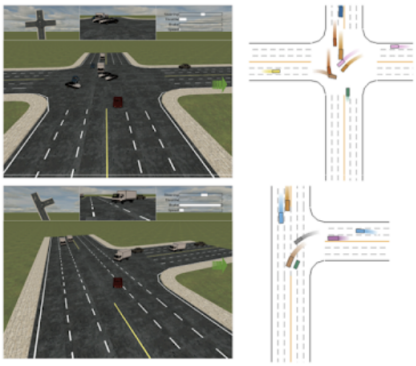

Figure 4: Visualization of the MetaDrive environments.

Table 13: Approximate computation time comparison on Halfcheetah task (NVIDIA A40)

Method	**Time (min)**
CDT	$\approx$ 154
CAPS IQL	
2 heads	$\approx$ 24
4 heads	$\approx$ 26
8 heads	$\approx$ 33
CAPS SAC+BC	
2 heads	$\approx$ 44
4 heads	$\approx$ 62
8 heads	$\approx$ 103

Table 14: CAPS hyperparameters.

Common Parameters	**BulletGym**	**SafetyGym**	**MetaDrive**
Cost thresholds (for evaluation)	{10, 20, 40}	{20, 40, 80}	{10, 20, 40}
Training steps	100000		200000
discount γ	0.99		
Batch size	512		
Optimizer	Adam		
Actor, Critic, and Value Networks hidden size	{512, 512}		
λ_k	$k/((\text{heads\_count} - 1)/2)$		
seeds	{0, 10, 20}		
IQL instantiation			
IQL Expectile τ $\{V_r, V_c\}$	{0.7, 0.7}		{0.5, 0.5}
IQL Inverse Temperature β	3		
Actor, Critic, and Value Networks learning rate	3e-4		
SAC+BC instantiation			
Actor and learning rate	1e-4		
Critic learning rate	1e-3		