Accountable Batched Control with Decision Corpus

Appendix: Table of Contents

A Missing Proofs

A.1 Proof of Proposition 3.8 (See page 4)

Proposition 3.8 (Estimation Error Bound for $v(a_t)$)**.** *Consider the belief variable $b_t = \boldsymbol{b}(o_t, a_t, h_t)$ and $\hat{b}$ the optimizer of Equation (6), the estimated value residual between $\boldsymbol{l}(b_t)$ and $\boldsymbol{l}(\hat{b})$ is controlled by the corpus residual:*

$$||\boldsymbol{l}(\hat{b}) - \boldsymbol{l}(b_t)||_{\mathcal{V}} \leq ||\boldsymbol{l}||_{\text{op}} \cdot r_{\mathcal{C}}(b_t) \tag{7}$$

where $||\cdot||_{\mathcal{V}}$ is a norm on $\mathcal{V}$ and $||\boldsymbol{l}||_{\text{op}} = \inf\{\lambda \in \mathbb{R}^+ : ||\boldsymbol{l}(b)||_{\mathcal{V}} \leq \lambda ||b||_{\mathcal{B}}\}$ is the operator norm for the linear mapping.

Proof. Leveraging the linearity of operator $\boldsymbol{l}$, the definition of the operator norm $||\cdot||_{\text{op}}$, and Definition 3.7, we have:

$$\begin{aligned} ||\boldsymbol{l}(\hat{b}) - \boldsymbol{l}(b_t)||_{\mathcal{V}} &= ||\boldsymbol{l}(\hat{b} - b_t)||_{\mathcal{V}} \\ &\leq ||\boldsymbol{l}||_{\text{op}} \cdot ||\hat{b} - b_t||_{\mathcal{V}} \\ &= ||\boldsymbol{l}||_{\text{op}} \cdot r_{\mathcal{C}}(b_t) \end{aligned} \tag{12}$$

□

A.2 Proof of Proposition 3.10 (See page 4)

Lemma A.1 (Affine Independence of $\tilde{\mathcal{C}}(b_t)$)**.** *The elements in the belief corpus built on top of $\tilde{\mathcal{C}}(b_t)$, as the corpus subset: $b^c \in \boldsymbol{b}(\tilde{\mathcal{C}}(b_t)) \subset \mathcal{B}$ must be affinely independent, that is*

$$\sum_{c=1}^{C} \lambda^c b^c = 0 \wedge \sum_{c=1}^{C} \lambda^c = 0 \Longrightarrow \lambda^c = 0, \forall c \in [C] \tag{13}$$

Proof. The proof is based on contradiction. Note that by definition, there are $d_b + 1$ elements in the belief corpus set. If those elements are not affinely independent, it basically means that b_t can be expressed in a lower dimensional space. In this case, the composition of the convex set is redundant. Without loss of generality, we use b^1 to denote the redundant element

$$b^1 = -\sum_{c=2}^{C} \frac{\lambda^c}{\lambda^1} b^c, \tag{14}$$

indicating that without b^1, we still have $b_t \in \mathcal{CB}(\mathcal{C}')$, where $\mathcal{C}' = \tilde{\mathcal{C}}(b_t) \backslash (o^1, a^1, h^1)$. This contradicts that $\tilde{\mathcal{C}}(b_t)$ is the minimal hull that contains b_t. □

Proposition 3.10 (Existence and Uniqueness)**.** *Consider the belief variable $b_t = \boldsymbol{b}(o_t, a_t, h_t)$, if $r_{\mathcal{C}}(b_t) \leq 0$ holds for $\mathcal{C} = \mathcal{D}$, then $\tilde{\mathcal{C}}(b_t)$ exists and the decomposition on the corresponding minimal convex hull exists and is unique.*

Proof. Following the assumption, there is at least one trivial corpus subset that exists for b_t — the offline dataset $\mathcal{D}$ as the corpus subset — such that a zero belief corpus residual can be achieved. [1] The existence of $\tilde{\mathcal{C}}(b_t)$ follows the fact the cardinality of the $\mathcal{D}$ is a finite number. The existence of decomposition on $\tilde{\mathcal{C}}(b_t)$ is then a consequence of the definition of the minimal hull.

Based on Lemma A.1, the elements in $\tilde{\mathcal{C}}(b_t)$ construct the minimal hull and are affinely independent. The uniqueness of the decomposition can be shown by contradiction:

Assume that there are two different decompositions for composing the belief state b_t:

$$b_t = \sum_{c=1}^{C} \omega^c b^c = \sum_{c=1}^{C} \tilde{\omega}^c b^c, \tag{15}$$

[1] otherwise, it falls into an out-of-distribution prediction problem and is beyond the scope of this work. Nonetheless, we note that ABC is able to perform OOD detection. See section F.5

where $\omega^c, \tilde{\omega}^c \in [0,1], \sum_{c=1}^{C} \omega^c = 1, \sum_{c=1}^{C} \tilde{\omega}^c = 1$. In this case, we have

$$
\begin{aligned}
0 &= \sum_{c=1}^{C} \omega^c b^c - \sum_{c=1}^{C} \tilde{\omega}^c b^c \\
&= \sum_{c=1}^{C} (\omega^c - \tilde{\omega}^c) b^c \\
&= \sum_{c=1}^{C} \lambda^c b^c, \text{where } \lambda^c \equiv \omega^c - \tilde{\omega}^c.
\end{aligned} \tag{16}
$$

However, $\sum_{c=1}^{C} \lambda^c = \sum_{c=1}^{C} (\omega^c - \tilde{\omega}^c) = \sum_{c=1}^{C} \omega^c - \sum_{c=1}^{C} \tilde{\omega}^c = 1 - 1 = 0$ contradicts with the fact that the elements are affinely independent. Hence the proof is completed. □

B Extended Related Work

B.1 Offline RL

Offline RL [7–15] has gained increasing attention in recent years due to its potential for solving practical problems, such as robotics control and game playing, where collecting new data can be expensive or time-consuming. However, it also presents several challenges, such as distribution shift and overfitting, which can lead to poor performance when deploying the learned policy to the actual environment. There are several model-free approaches to addressing these challenges in Offline RL. Such as distributional matching [7, 28], regularization techniques to prevent overfitting [29], conservation [11] or adding noise [14] to the policy or using adversarial training [30]. A third approach is to incorporate uncertainty estimation to evaluate the performance of the learned policy on unseen data [12].

On the other hand, model-based RL tackles the problem by first learning world models and then performing planning algorithms on the learned model [21, 22]. In either model-based or model-free approaches, black-box approximators are used; hence, the decision is not transparent.

B.2 Episodic Control and Nearest Neighbor Control

Episodic Memory and Episodic Control [16–18], inspired by biological learning mechanism, are studied as an alternative way of policy learning [31]. Follow-up works introduce various modifications and extend episodic control to the continuous domains. e.g., Hu et al. [32] introduces Generalized Episodic Memory (GEM) which effectively organizes the state-action values of episodic memory in a generalizable manner and supports implicit planning on memorized trajectories. Ma et al. [33] leverages episodic memory in offline RL setting with a pessimistically estimated value function. Li et al. [34] proposing a novel state-abstractor framework for episodic control and improving the learning efficiency in continuous control benchmarks. In all those works, a value function resembling the hippocampus episodic memorization mechanism is introduced as an alternative to Q-learning [35, 36] that updates the state-action values with temporal difference learning or Monte-Carlo estimation [37–39].

In the continuous control domains, policy gradient methods [40, 41] or supervised learning-based methods [42, 9, 43] are then applied for the policy improvement. All of those approaches are limited to black-box value-based learning and require additional black-box policy networks in the continuous control domain, whereas our proposed method performs a transparent decision-making process without explicit policy learning.

Explicit nearest neighbor methods that perform decision-making according to training-time similar trajectories have been studied theoretically [19] and empirically [20]. Although those methods also enjoy transparency, they suffer from the problem of the curse of dimensionality. Moreover, defining the nearest neighbor with a heuristically determined Euclidean metric also suffers the problem of aggressive extrapolation and thus is not suitable for the offline setting [7, 11].

B.3 Explainable RL

Understanding the decisions made by RL agents is a key issue in many high-stake real-world domains such as finance [44] and healthcare [45, 27]. Previous Explainable-RL (XRL) literature can be broadly classified with a taxonomy of three classes [46]: (1) Feature importance, that includes learning policy through an explainable policy class [47–49], converting black-box models into an interpretable format [50–54], and natural language [55–57] or saliency map based explanations [58–60]; (2) Transparent Learning Process that reveals the influences of MDP ingredients during the learning process, including methods that learn to predict the counterfactual outcomes for decision-making [61–63], decompose the learning objective [64–67], and identifying the crucial training datum [68, 69]; (3) Policy Level, which illustrates the long-term behavior of the agent [70–72]. For more extensive discussions on XRL literature, we refer the readers also to [73, 74]. Different from previous approaches, our work introduces the first example-based explanation for policy learning. Supported by training dataset examples, the execution of our control algorithm is accountable.

C The Strictly Batched Imitation Setting

The instant reward may not be contained in the offline dataset in strictly batched imitation (SBI) learning settings such as clinical treatment and healthcare scenarios. In such cases, the value of actions can not be estimated through Monte Carlo, and it is generally impossible to learn a belief state based on the value prediction. In such a case, we need to adapt Accountable Batched Controller accordingly.

In such a setting, the dataset $\mathcal{D}_{\text{SBI}} = \{o_t^i, a_t^i, o_{t+1}^i\}_{t=1,\dots,T}^{i=1,\dots,N}$ contains only sequential observations $o_t^i, o_{t+1}^i, i \in [N], t \in [T]$, actions $a_t^i, i \in [N], t \in [T]$ performed by behavior policies, which is always an expert or near-expert controller, and transition histories $h_t^i, i \in [N], t \in [T]$ that can be composed of the former quantities.

The Accountable Batched Controller should be adapted to handle this setting. Specifically, we can still define the corpus subset as

Definition C.1 (SBI Corpus Subset)**.** A *SBI Corpus Subset* $\mathcal{C}_{\text{SBI}}$ is defined as a subset of an offline dataset $\mathcal{D}_{\text{SBI}}$, indexed by $[C] := \{1, 2, ..., C\}$ — the natural numbers between 1 and C.

$$\mathcal{C}_{\text{SBI}} = \left\{ (o^c, a^c, h^c) \in \mathcal{D}_{\text{SBI}} \middle| c \in [C] \right\}. \tag{17}$$

Property C.2. (SBI Linear Restricted Belief) The policy function $\boldsymbol{\pi} : \mathcal{O} \times \mathcal{H} \mapsto \mathcal{A}$ can be decomposed as $\boldsymbol{\pi} = \boldsymbol{l} \circ \boldsymbol{b}$, where $\boldsymbol{b} : \mathcal{O} \times \mathcal{H} \mapsto \mathcal{B} \subseteq \mathbb{R}^{d_b}$ maps the joint observation-history space to a d_b dimensional belief variable $b_t = \boldsymbol{b}(o_t, h_t)$ and $\boldsymbol{l} : \mathcal{B} \mapsto \mathcal{A}$ is a linear function that maps the belief b_t to an output $\boldsymbol{l}(b_t) \in \mathcal{A}$.

Then any control time behavior generated by policy π is accountable in the sense that it can be decomposed by the belief corpus, defined as

Definition C.3. (SBI Belief Corpus) A *SBI Belief Corpus* $\boldsymbol{b}(\mathcal{C}_{\text{SBI}})$ is defined by applying the belief function $\boldsymbol{b}$ to the corpus subset $\mathcal{C}_{\text{SBI}}$,

$$\boldsymbol{b}(\mathcal{C}_{\text{SBI}}) = \left\{ b^c = \boldsymbol{b}(o^c, h^c) \middle| (o^c, a^c, h^c) \in \mathcal{C}_{\text{SBI}} \right\} \subset \mathcal{B}, \tag{18}$$

on top of which we can define the Belief Corpus Hull in the SBI setting:

Definition C.4. (SBI Belief Corpus Hull) The *SBI Belief Corpus Convex Hull* spanned by a corpus subset $\mathcal{C}_{\text{SBI}}$ with the belief corpus $\boldsymbol{b}(\mathcal{C}_{\text{SBI}})$ is the convex set

$$\mathcal{CB}(\mathcal{C}_{\text{SBI}}) = \left\{ \sum_{c=1}^{C} w^c b^c \middle| w^c \in [0,1], b^c \in \boldsymbol{b}(\mathcal{C}_{\text{SBI}}), \forall c \in [C], \sum_{c=1}^{C} w^c = 1 \right\}, \tag{19}$$

followed by the concept of Minimal Hull in the SBI setting defined as

Definition C.5 (SBI Minimal Hull)**.** Denoting the decision corpora whose belief convex hull contain a belief variable b_t by

$$\mathcal{C}_{\text{SBI}}(b_t) = \left\{ \mathcal{C}_{\text{SBI}} \subseteq \mathcal{D}_{\text{SBI}} \middle| b_t \in \mathcal{CB}(\mathcal{C}_{\text{SBI}}) \right\}$$

Among those subsets, the one that contains $d_b + 1$ decision corpora and has the smallest hyper-volume forms the *minimal hull*, denoted by $\tilde{\mathcal{C}}_{\text{SBI}}(b_t)$:

$$\tilde{\mathcal{C}}_{\text{SBI}}(b_t) := \min_{\mathcal{C}_{\text{SBI}}} \int_{\mathcal{CB}(\mathcal{C}_{\text{SBI}})} dV$$

Similar to the property in ABC, in the SBI setting, the control time decisions can be decomposed with examples in the offline dataset that constructs such an SBI Minimal Hull,

$$\pi(a_t|o_t, h_t) = \boldsymbol{l} \circ \boldsymbol{b}(o_t, h_t) = \boldsymbol{l} \circ \sum_{c=1}^{C} w^c \cdot \boldsymbol{b}(o^c, h^c) = \sum_{c=1}^{C} w^c \cdot \boldsymbol{l} \circ \boldsymbol{b}(o^c, h^c) = \sum_{c=1}^{C} w^c \cdot a^c,$$

where $(o^c, a^c, h^c) \in \tilde{\mathcal{C}}_{\text{SBI}}(b_t)$.

D Further Implementation and Experiment Details

D.1 Reproduceability: Code

We elaborate on our implementation details in this section. Our code is available anonymously at `https://anonymous.4open.science/r/AccountableBatchedController-9AC7`.

D.2 Learning the Belief Space

To efficiently encode the information in historical transition, in our work, we employ the Gated Recurrent Units (GRU) [75] to map fixed-length transition histories into embedding vector variables, followed by 3 fully connected layers as the belief function $\boldsymbol{b}$. In principle, any other recurrent networks should also be able to process such context information. Table 4 presents the hyper-parameters we use in the belief learning process.

Table 4: Hyperparameters in learning the belief function.

Hyper-Param	Choice
Context Model	GRU
Hidden Unit Number	128
Hidden Recurrent Layer	1
Batch Size	500
Epochs	4000
Optimizer	Adam
Learning Rate	0.001
Memory Length	4
Embedding Dimension	20

D.3 Addressing the Scalability Issue: Finding the Minimal Convex Hull Efficiently

Rigorously searching for the minimal convex hull in the belief space is a combinatorial optimization problem. In our implementation, we leverage a heuristic search method that first reduces the search space by looking for k-nearest neighbors in the belief space, and then build the approximate minimal convex hull on top of those k-nearest neighbors of the control time belief b_t. With d_b-dimensional belief space, the minimal convex hull will contain at most $d_b + 1$ examples, hence we can set $k = 2(d_b + 1)$, and conduct the combinatorial optimization on those k examples, which is much easier than the original problem (reduce combinatorial problem *Select* $d_b + 1$ *out of* N into *Select* $d_b + 1$ *out of* k).

D.4 Data Generation Process: Heterogeneous Pendulum

The classical control task of Pendulum has the goal to swing up and balance a pendulum using a control input. In the control task, a policy can apply torque to the joint in order to swing the pendulum up and then maintain its upright position. The state of the system is defined by the pendulum's angle and angular velocity, and the action is the torque applied to the joint. The reward function typically provides a positive reward for keeping the pendulum upright and a negative reward for large torques or deviations from the upright position.

To manifest the potential heterogeneous outcome in healthcare and generalize the study into POMDP, we consider a heterogeneous variant of the original task. There are two contradictory Pendulum systems: the normal one and the converse one. While in the first system, adding torque will lead to a dynamical change according to the original physical design, in the converse system, the torque inputs will be sent to the system with a negation.

We train TD3 policies in each system and merge the collected dataset together as the offline dataset. In the Low-Data settings, 50000 transitions of each environment are collected, hence the dataset contains in total of 100000 transitions; in the Mid-Data regime, 150000 transitions of each environment are collected, hence the dataset contains in total of 300000 transitions; in the Rich-Data regime, 300000 transitions of each environment are collected, hence the dataset contains in total of 600000 transitions.

In order to achieve high performance, the agent must be able to identify the decision corpora that are collected from the same system dynamics as in the control time.

D.5 Hardware and Running Time

We experiment on a machine with 2 TITAN X GPUs and 32 Intel(R) E5-2640 CPUs. In general, the computational expense of model training in ABC is cheap, as the neural networks used in ABC are in general shallow and of small scale. Learning the belief space requires half the calculation of building a world model. However, we acknowledge the exact calculation of the minimal convex hull in a large-data regime can be computationally expensive. And we have discussed our practical solution (Appendix D.3). With our proposed solution, the convex hull decomposition takes less than 10 seconds with a uniform sampler that samples 100 actions randomly for every time step. Increasing the batch size will lead to a sub-linear increase in the computational time with parallelization.

D.6 Baseline Implementations

Benchmark Algorithms Except for standardized components that we will introduce below, we use the publicly available source code when constructing the benchmark algorithms; references are in the following:

- **kNN and 1NN**: https://scikit-learn.org/.../neighbors, Reference: [19].
- **BC**: Implementation is straightforward using supervised learning.
- **MFRL**: https://github.com/sfujim/TD3, Reference: [26].
- **MPC**: https://github.com/UM-ARM-Lab/pytorch-mppi/.../mppi.py, Reference: [76].

Neural Network Backbones In all baseline methods and our implementation for ABC, we use the same network structure: 3-layer MLP with a recurrent model that encodes the historical trajectory information, which is called as the *Context Variable* in the literature [77, 78]. Our implementation of the recurrent model is based on the open-sourced code of https://github.com/amazon-science/meta-q-learning. In all experiments, we use the same neural network architecture and match the hyper-parameters for a fair comparison, except stated otherwise (e.g., the Q-learning baseline requires a larger batch size to guarantee convergence and boost stability, which will be elaborated in the following section.).

D.7 The Model-Free RL Baseline and Its Stability Issue

Our implementation of the Q-learning baseline leverages the twin-delayed techniques [26] to stabilize training. We find that Q-learning requires a large batch size (i.e., 10240) and many optimization epochs to converge. In the experiment settings with more offline data, the convergence becomes even

harder: the rich-data regime containing $6\times$ more data takes $6\times$ more training epochs to converge. And the converged performance is always with high vibration, leading to worse performance.

Learning curves are shown in Figure 6, experiments are done with 8 seeds and both averaged learning curves and individual curves are plotted.

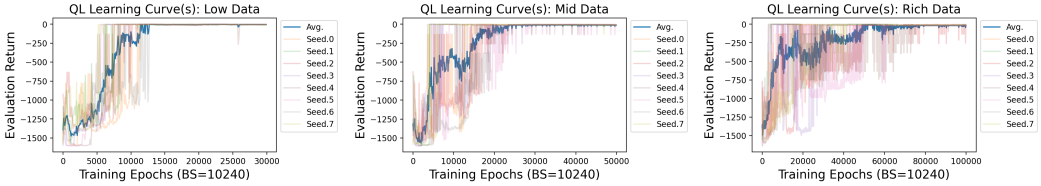

Figure 6: Q-Learning learning curves. When the size of offline data increases, more training epochs are needed for the convergence, and the stability of performance at convergence is reduced, leading to a larger variance and poorer performance in the rich-data regime.

E Additional Qualitative and Quantitative Results

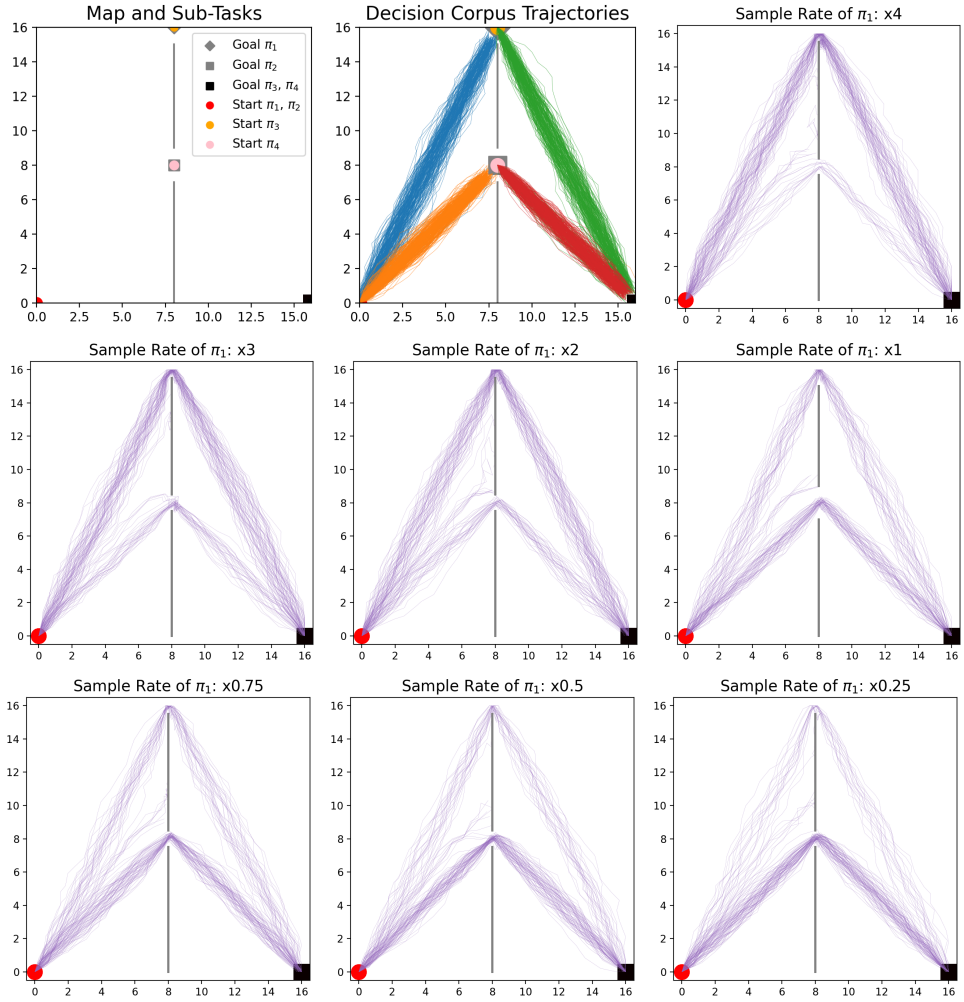

Figure 7: Visualization of control behaviors. The first two plots show the task and collected trajectories in the offline dataset. This is the same environment we have used in Section 5.3. The control time preferences can be controlled by changing the sub-sampling or re-sampling rate.

E.1 Adaptivity: Visualizing the Decisions of ABC and Quantitative Performance

We plot the control time trajectories in experiments of Section 5.4 in Figure 7. It is clearly shown that with a decreasing sampling rate of π_1's trajectories (passing through the upper gate) in the offline dataset, the control time behaviors tend to choose more actions following the behaviors of π_3 (passing through the lower gate).

Quantitatively, Table 5 shows that while changing the sampling rate of trajectories from π_1, the success rate does not change much while the proportions of strategies chosen by the control policy vary accordingly.

Table 5: Quantitative results in the re-sampling and sub-sampling experiments. In all settings, 100 trajectories in total are generated using the proposed method. The success rate of reaching the goal and choices of solutions are presented in the table.

Re/Sub-Sample	$\times 4$	$\times 3$	$\times 2$	$\times 1$	$\times 0.75$	$\times 0.5$	$\times 0.25$
Success Rate	0.91	0.97	0.90	0.92	0.92	0.93	0.92
Passing Upper Gate	76	69	61	42	31	24	17
Passing Lower Gate	15	28	29	50	61	69	75

E.2 More Visualization Results and Extended Discussion on the Healthcare Dataset

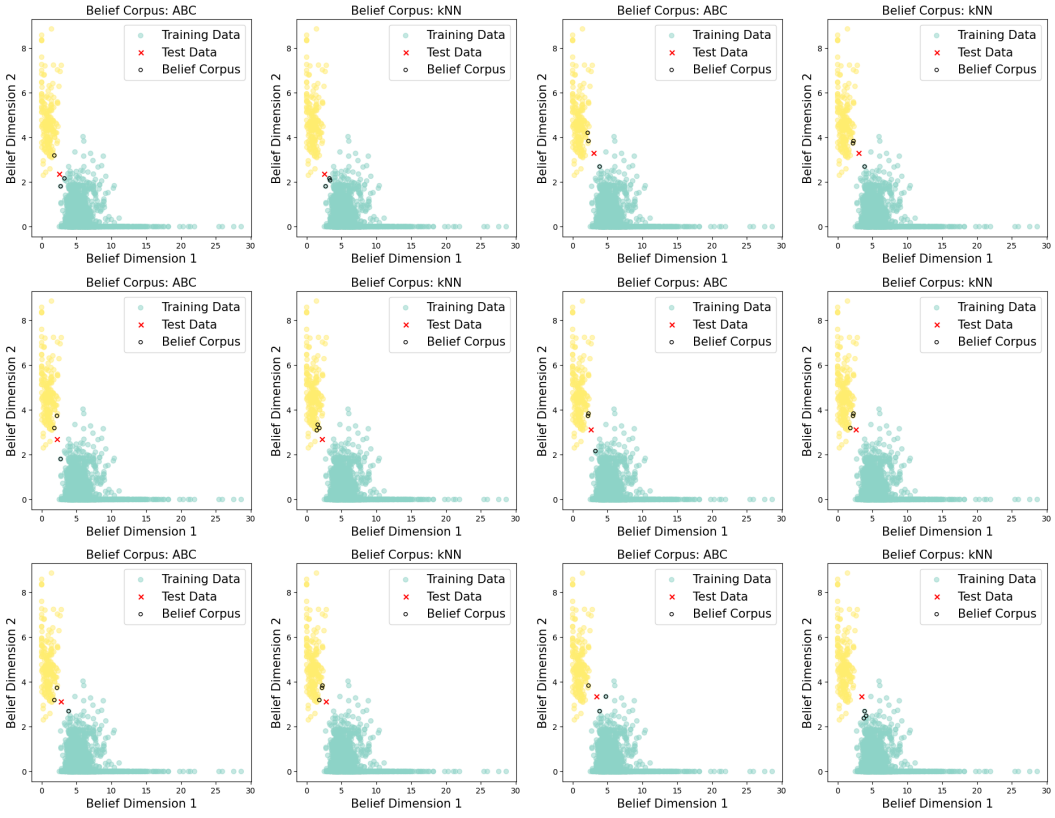

Figure 8: Visualization of the control time decision supports in the healthcare dataset. The two colors of the scatter plots denote different training time decisions. Test data is denoted by red cross marks, and the identified belief corpus subsets are marked with black circles.

We provide more qualitative results in Figure 8. In those figures, the colors yellow and blue indicate different treatments in the training data, separately.

In the realm of healthcare, treatment decisions involving atypical patients — those with rare or non-standard characteristics that do not closely resemble the majority of training examples — present a significant challenge. These outliers often reside on the boundaries of existing treatment records of patients, rendering the optimal course of action ambiguous even for domain experts.

Our proposed method ABC provides a critical advantage in these complex scenarios. ABC constructs a minimal convex hull around such patients, unearthing potential risks associated with incorrect treatment approaches. In cases where the decision corpus spans disparate treatment groups, ABC reveals the impossibility of establishing a more representative convex hull that both contains the patient and exclusively includes members of the same treatment class.

Conversely, the k-nearest neighbors (kNN) approach fails to identify these high-risk patients. The nearest neighbors for such boundary cases can all belong to the same treatment group, thus offering no warning signals that the patient might be atypical and warrant special attention.

Therefore, ABC outperforms the kNN method not merely in terms of higher accuracy in suggesting treatments, but also in its ability to flag patients whose decision basis exhibits heterogeneity at the test time. In these instances, doctors or other human experts can give particular consideration, thereby enhancing the trustworthiness of the decision system.

F Additional Experiments

F.1 Trade-Off between Accountability and Performance

Experiment Setting In principle, the minimal convex hull in the d-dimensional belief space contains $d + 1$ examples. However, searching for such a minimal convex hull is a combinatorial optimization problem and can be extremely hard in high-dimensional space with many samples. In our implementation, we leverage a heuristic search method that constrains the combinatorial search inside the k-nearest neighbors of a given control time belief state.

We change such a hyper-parameter — the maximal number of examples in the corpus subsets can be used in building the minimal belief space convex hull. We experiment with $k = 1, 10, 20, 100$ separately.

Table 6: The cardinality of the corpus subset is a hyper-parameter that trades off between accountability and performance. While in general using a larger number of corpus examples can better support the control decisions, it also has a risk of increasing the corpus residual in synthesizing the control time decision, hence hindering the performance of the control policy.

K	Performance
1	-670.51 ± 321.09
10	-510.39 ± 311.24
20	-1.25 ± 0.4
100	-2.06 ± 0.55

Results Table 6 shows the results. To make ABC work, the choice of k should at least roughly match the dimension of the latent space. Otherwise, the aggressive extrapolation will hinder the performance of ABC, including using only the nearest neighbor as an approximation. While such a choice naturally trades off between explainability and decision quality, our empirical study shows using a redundant set of the corpus will hinder the performance — this demonstrates the necessity of using the minimal convex hull in ABC.

Take-Away: *The minimal convex hull design in ABC is crucial for performance. We recommend the choice of size in constructing a minimal convex hull should match the dimension of the belief space.*

F.2 Visualizing Conservation of ABC

Experiment Setting To better illustrate the conservative behaviors introduced by the hyper-parameter ϵ, we conduct experiments on a Two Gates Maze environment to visualize the differences of using different choices of ϵ's.

In the Two Gates Maze task, an agent needs to navigate to a goal located at $(16, 8)$, the middle point of the right side wall, starting from $(0, 8)$, the middle point of the left side wall. Two gates open in a middle wall located at $x = 8$, hence the agent needs to pass one of those gates to reach the goal.

We generate an offline dataset from two behavior policies, each of which selects one of the two gates to pass the middle wall. The first two plots in Figure 9 show the map as well as the behavior trajectories as the dataset.

We then experiment with different choices of $\epsilon = [0.1, 0.3, 0.5, 0.7]$ in ABC to perform offline control.

Results Results are shown in the last 4 plots in Figure 9 (Purple lines denote the control time trajectories, ideal conservative policies' behaviors should be bounded by the behavior trajectories). In each setting, we roll out with 100 trajectories by ABC and visualize the behaviors, and report the averaged performance in Table 7. When a small ϵ is used, the control time behaviors show little extrapolation: trajectories are in general surrounded by the dataset behaviors. When ϵ becomes larger, more aggressive extrapolations emerge in control time behaviors, leading to poorer performances.

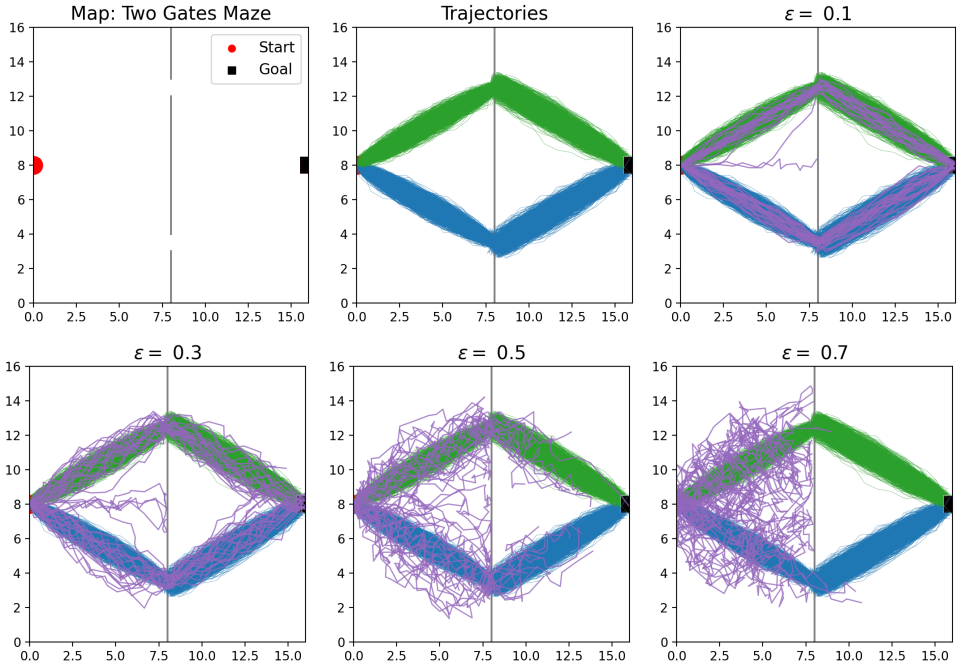

Figure 9: Visualizing the conservative behaviors controlled by the threshold controlled by a quantile number ϵ. Using a smaller ϵ leads to more conservative behaviors hence benefiting the offline control problems where aggressive extrapolations can be dangerous.

Table 7: The ϵ in Equation (11) contributes to the conservative behaviors of ABC.

ϵ	Performance
0.1	6.97 ± 2.57
0.3	2.58 ± 5.13
0.5	-3.10 ± 1.00
0.7	-3.20 ± 0.00

Take-Away: *Using a small ϵ leads to more conservative behaviors in ABC. ABC performs offline control avoiding aggressive extrapolations by constructing the Minimal Hull and performing control with the decision corpora that have minimized residuals.*

F.3 Additional Environment: LunarLanderContinuous

Experiment Setting We additionally experiment on the LunarLanderContinuous-v2 environment [79] for the general interests of the RL community. The LunarLanderContinuous-v2 environment is a physics-based simulation game in which the agent must control a lunar lander to successfully land on a designated landing pad. It has a 2-dim action space and a 8-dim state space, both of which are continuous.

We compare ABC with the nearest-neighbor controller **(1NN)** [19] and its variant that using k neighbors **(kNN)**, model-based RL with Mode Predictive Control (**MPC**) [23], model-free RL **(MFRL)** [26], and behavior clone **(BC)** [24]. We change the size of the dataset to showcase the performance difference of various methods under different settings. We experiment with different offline data availability, varying from 0.1M to 1M.

Table 8: Performance comparison on the LunarLanderContinuous-v2 environment under different settings (availability of offline data). The episodic cumulative reward of each method is reported. The last row (Data) reports the performance of trajectories in the offline dataset. **Higher is better.**

	0.1M	0.2M	0.3M	0.5M	1M
ABC	$\mathbf{100.77 \pm 174.5}$	$\mathbf{184.83 \pm 88.28}$	$\mathbf{240.81 \pm 44.47}$	$\mathbf{252.38 \pm 45.66}$	$\mathbf{253.71 \pm 25.11}$
kNN	-86.07 ± 172.57	-43.87 ± 69.1	-3.04 ± 116.76	-59.63 ± 143.33	-31.47 ± 152.72
1NN	-42.2 ± 31.36	-98.27 ± 89.35	-14.93 ± 115.75	-57.37 ± 59.1	-64.41 ± 69.71
BC	-130.23 ± 33.23	-118.61 ± 42.18	-36.91 ± 57.32	18.08 ± 35.4	54.46 ± 69.16
MFRL	$\mathbf{92.0 \pm 82.05}$	$\mathbf{165.11 \pm 54.48}$	$\mathbf{221.3 \pm 19.63}$	$\mathbf{219.55 \pm 38.97}$	$\mathbf{254.55 \pm 24.42}$
MPC	-31.42 ± 32.09	-35.52 ± 43.09	-50.96 ± 22.96	-67.88 ± 55.49	-96.22 ± 85.01
Data	149.99 ± 133.45	207.92 ± 58.34	171.19 ± 100.43	169.42 ± 96.02	184.92 ± 69.59

Results We present the results in Table 8. In all settings, we find ABC is able to achieve an on-par performance of black-box decision-making algorithms, and outperforms the accountable baselines. In this environment, we find the model-based approach is not able to learn a well-performing reward function approximator. On the other hand, different from the Pendulum-Het settings where the stability of the balanced state is essential in achieving high performance, the stability of the model-free method in this environment is not an issue.

Take-Away: *The results on the LunarLanderContinuous environment again demonstrate the desired properties of ABC: while being accountable, it achieves similar performance as the black-box learning algorithms and is robust under different data availability.*

F.4 Black-Box Policy as More Efficient Sampler

Experiment Setting In the main text, we introduce the uniform sampling approach for control time execution. While such an approach is simple and effective in our accountability-sensitive control benchmark tasks, it can be inefficient in high-dimensional continuous control tasks. For the interest of the general continuous control community, we experiment with a higher dimensional task in this section to stress test the capability of ABC in more challenging tasks.

We experiment on the BipedalWalker-v3 environment that has 24-dim observational space and 4-dimensional action space. Different from previous experiments where a uniform sampler is applied, in this section, we use black-box models as the more efficient sampler and leverage ABC as a post-hoc interpreter for decision accountability. Specifically, we compare ABC with a uniform sampler **(ABC-Uniform)** and ABC with Behavior Clone policy as a black-box sampler **(ABC-BC)** against

the same baselines as previously, i.e., the nearest-neighbor controller **(1NN)** [19] and its variant that using k neighbors **(kNN)**, model-based RL with Mode Predictive Control (**MPC**) [23], model-free RL **(MFRL)** [26], and behavior clone **(BC)** [24]. To improve the black-box controller's performance and hence isolate the source of gain, we use the offline dataset of size 1M.

Results Results are reported in Table 9. In this high-dimensional control task, the uniform sampler is inefficient and fails to converge to a well-performing policy in control time. Among all black-box methods, BC achieves the best performance. And ABC with BC as its sampler achieves improved performance, while at the same time being accountable.

Table 9: Performance on the BipedalWalker-v3 environment. The episodic cumulative reward of each method is reported. **Higher is better.**

Method	Performance
kNN	-109.72 ± 5.85
1NN	-111.95 ± 8.25
ABC-Uniform	-90.44 ± 14.06
ABC-BC	$\mathbf{276.98 \pm 82.78}$
BC	$\mathbf{208.72 \pm 95.72}$
MFRL	18.51 ± 111.53
MPC	-96.82 ± 24.7
Data-Avg-Return	202.25 ± 102.61

Take-Away: *ABC can work both in isolation or combined with black-box policies. ABC can be used as a plug-in to add accountability to black-box controllers in a post-hoc manner. In high-dimensional control tasks, uniform sampling can be inefficient and black-box samplers can alleviate such a difficulty.*

F.5 Identify Control Time OOD Examples with ABC

Experiment Setting In this section, we show that ABC can be applied to OOD example identification during control time. Specifically, we conduct the experiment with the **BipedalWalker** environment. During control time, a black-box controller described in the last section is used, and we start to inject Gaussian noise into the control actions as disturbance after the 500-th timestep to create an OOD scenario.

Results We repeat the above process for 10 times and report the averaged step-wise instant reward curve and corpus residual curve during rollouts in Figure 10. We note that in the figure in the beginning steps, the walker starts from a static state to walk, thus leading to an increase in the instant reward curve. We then label the control steps between the 100-th and the 500-th step as the stable walking phase, during which the walker receives a nearly constant instant reward (around 0.4 per step), and the corpus residual during this period is stable and close to 0. The Gaussian noise is injected after the 500-th timestep, leading to a clear increase in the corpus residual curve and a sudden decrease in the instant reward curve. According to the corpus residual values' sudden increase, the control time OOD examples can be identified according to the *3-sigma rule of thumb*. (The 3-sigma intervals are marked with shaded areas in the figure).

Take-Away: *Control time OOD examples can be identified by ABC according to the sudden increases in the corpus residual value.*

G Limitations and Future Work

In this study, our exploration of accountable batch control primarily targets low-dimensional control tasks, drawing inspiration from healthcare applications where treatments are typically of low dimension. However, the uniform sampling approach we propose may not perform as well in high-dimensional control systems, which can be of interest in the field of robotics study. We do offer an

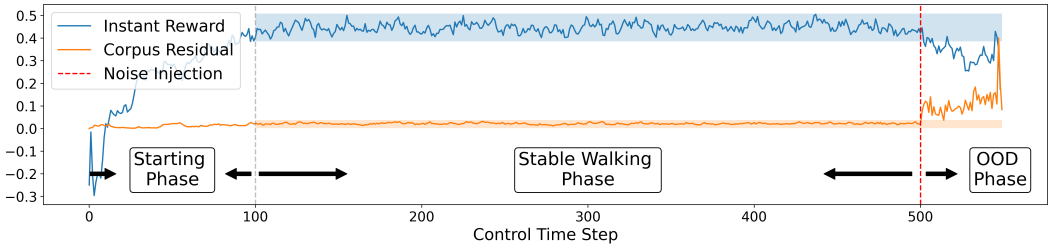

Figure 10: ABC can be used for OOD detection in control time. The corpus residual values can be used to detect large epistemic uncertainty on OOD examples. Shaded areas in the figure denote the **3-sigma interval**.

initial exploration of using black-box samplers in ABC within our appendix, but there remains ample room for further work in enhancing the efficiency of the sampling process. Additionally, there's potential for expanding ABC into online control settings by combining it with optimism in the face of uncertainty (OFU) explorers, to pursue accountable online control. This, however, lies beyond the scope of our current paper.

H Broader Impact

In this study, we examine the batch control problem, which holds significant potential for applications in costly, safety-sensitive, and critical domains such as healthcare and finance. While previous works have primarily focused on efficient learning in batch settings, the accountability of offline decisions remains largely unexplored despite its importance.

In critical domains like healthcare, it's vital that decisions are based on supportive evidence. For instance, when a patient is treated in a certain manner, it should be based on the successful outcomes of previous patients with comparable conditions who received the same treatment. The ability to trace the supportive basis of decisions enhances the process of policy reasoning and debugging, thereby improving the trustworthiness of decision-making systems.

However, we bring to light the potential risk associated with applying ABC to critical real-world decision-making systems. This risk stems from potential heterogeneous outcomes, i.e., the aleatoric uncertainty associated with decision outcomes. For example, similar patients undergoing the same treatment may experience different results. Therefore, when the variance of outcomes in the corpus subset is high, users should exercise caution regarding the potential heterogeneous outcomes of decisions.