# Double Check Your State Before Trusting It: Confidence-Aware Bidirectional Offline Model-Based Imagination

**Jiafei Lyu**[1], **Xiu Li**[1,*], **Zongqing Lu**[2*]
[1]Tsinghua Shenzhen International Graduate School, Tsinghua University
[2]School of Computer Science, Peking University
lvjf20@mails.tsinghua.edu.cn,
li.xiu@sz.tsinghua.edu.cn, zongqing.lu@pku.edu.cn

## Abstract

The learned policy of model-free offline reinforcement learning (RL) methods is often constrained to stay within the support of datasets to avoid possible dangerous out-of-distribution actions or states, making it challenging to handle out-of-support region. Model-based RL methods offer a richer dataset and benefit generalization by generating imaginary trajectories with either trained forward or reverse dynamics model. However, the imagined transitions may be inaccurate, thus downgrading the performance of the underlying offline RL method. In this paper, we propose to augment the offline dataset by using trained bidirectional dynamics models and rollout policies with *double check*. We introduce conservatism by trusting samples that the forward model and backward model agree on. Our method, *confidence-aware bidirectional offline model-based imagination*, generates reliable samples and can be combined with any model-free offline RL method. Experimental results on the D4RL benchmarks demonstrate that our method significantly boosts the performance of existing model-free offline RL algorithms and achieves competitive or better scores against baseline methods.

## 1 Introduction

Offline reinforcement learning (offline RL), also known as batch RL [34], aims at learning from a static dataset that was previously gathered by an unknown behavioral policy. Offline RL is deemed to be promising [16, 14] as online learning requires the agent to continuously interact with the environment, which however may be costly, time-consuming, or even dangerous. The progress in offline RL will undoubtedly scale RL methods to being widely applied in real-world applications, considering the impressive success in computer vision or natural language processing by adopting large-scale offline datasets [9, 6].

Prior off-policy online RL methods [17, 21, 48] are known to fail on fixed offline datasets, even on expert demonstrations [14], due to extrapolation errors [16]. In the offline setting, the agent can overgeneralize from the static dataset, resulting in arbitrarily wrong estimates upon out-of-distribution (OOD) state-action pairs and dangerous action execution. To address this issue, recent model-free offline RL algorithms compel the learned policy to stay close to the behavioral policy [16, 31, 72], or incorporate some penalties into the critic [50, 32, 29]. However, such approaches often suffer from loss of generalization capability [73, 69], since they purposely avoid OOD states or actions.

Model-based offline RL methods, instead, enrich the logged dataset by generating synthetic samples with the trained forward or reverse (backward) dynamics model [26, 75, 69, 74]. These methods

*Corresponding Authors

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

benefit from better generalization thanks to richer transition samples. Intuitively, the performance of the agent is largely confined by the quality of the model-generated data, i.e., learning on bad states or actions will negatively affect the policy via backpropagation. Unfortunately, there is no guarantee that reliable transitions can be generated by the trained forward or backward dynamics model [2].

In this paper, we aim to generate reliable transitions for offline RL via a *double check mechanism*. The intuition behind this lies in the fact that humans often do double check when they are uncertain and need to be cautious. Besides the forward model, we train the backward model to generate simulated rollouts backward and use one to check whether the synthetic samples the other generated are credible. To be specific, we train bidirectional dynamics models along with bidirectional rollout policies. Instead of injecting pessimism into value estimation, we introduce *conservatism into transition*, i.e., only samples that the forward model and reverse model agree on are trusted.

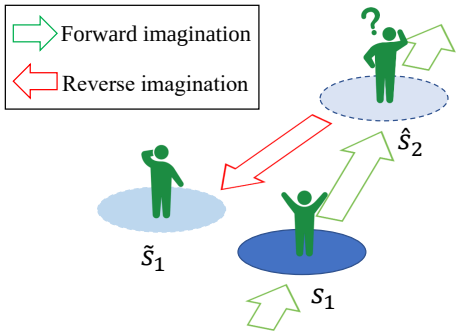



Figure 1: Illustration of the basic idea of confidence-aware bidirectional offline model-based imagination. To ensure that the forward imagination $\hat{s}_2$ from $s_1$ is valid and reasonable, one needs to 'look back' to check whether the imagined previous state $\tilde{s}_1$ based on $\hat{s}_2$ is similar to $s_1$. We trust $\hat{s}_2$ if the deviation between $s_1$ and $\tilde{s}_1$ is small.

We use Figure 1 to further illustrate our insight, where we take forward transition generation as an example, of which the process is identical to the reverse setting. Starting from $s_1$, the forward model predicts next state $\hat{s}_2$. However, it is hard for the agent to decide whether $\hat{s}_2$ is trustworthy. One natural solution, which follows human's way of reasoning [22], is backtracking where it comes from, i.e., looking backward to trace previous state $\tilde{s}_1$, and check whether the imagined state $\tilde{s}_1$ based on $\hat{s}_2$ is different from the true state $s_1$. We are confident to $\hat{s}_2$ if $\tilde{s}_1$ is similar to $s_1$ and vice versa.

To this end, we propose Confidence-Aware Bidirectional Offline Model-Based Imagination (CABI), which is a simple yet effective data augmentation method. CABI generally guarantees the reliability of the generated samples via the double check mechanism, and can be combined with any off-the-shelf model-free offline RL methods, e.g., BCQ [16] and TD3_BC [15], to enjoy better generalization in a conservative manner. Extensive experimental results on the D4RL benchmarks [14] show that CABI significantly boosts the performance of the base model-free offline RL methods, and achieves competitive or even better scores against recent model-free and model-based offline RL methods.

## 2 Related Work

In this paper, we consider offline reinforcement learning [34, 38], which defines the task of learning from a static dataset that was collected by an unknown behavior policy. Applications of offline RL include robotics [45, 57, 54], healthcare [19, 70], recommendation system [59, 64], etc.

**Model-free offline RL.** Since it is risky to execute out-of-support actions, existing offline RL algorithms are often designed to constrain the policy search within the support of the static offline dataset. They realize it via importance sampling [53, 63, 39, 49, 18], explicit or implicit policy constraints [16, 31, 72, 35, 40, 78], learning conservative critics [32, 44, 30, 43, 29, 42], and quantifying estimation uncertainty [73, 76, 10]. Recently, sequential modeling is also explored in the offline RL setting [7, 24, 47]. Despite these advances, model-free offline RL methods suffer from loss of generalization beyond the dataset [69], and CABI is proposed to mitigate it.

**Model-based offline RL.** Model-based offline RL algorithms benefit from better generalization as the static dataset is extended by the synthetic samples generated from the trained forward [56, 13, 1] or reverse dynamics model [69]. These methods heavily rely on uncertainty quantification [52, 75, 26, 11], compelling the policy towards the behavior policy [65, 46], representation learning [36, 54], and penalizing Q-values [74]. However, it is hard to judge whether the transitions generated by the trained dynamics model are reliable, and poor imagined samples will negatively affect the performance of the agent. Recently, [75, 41] explore how inaccurate rollouts that are well penalized can still be useful for model-based training. However, they work only for model-based offline RL methods.

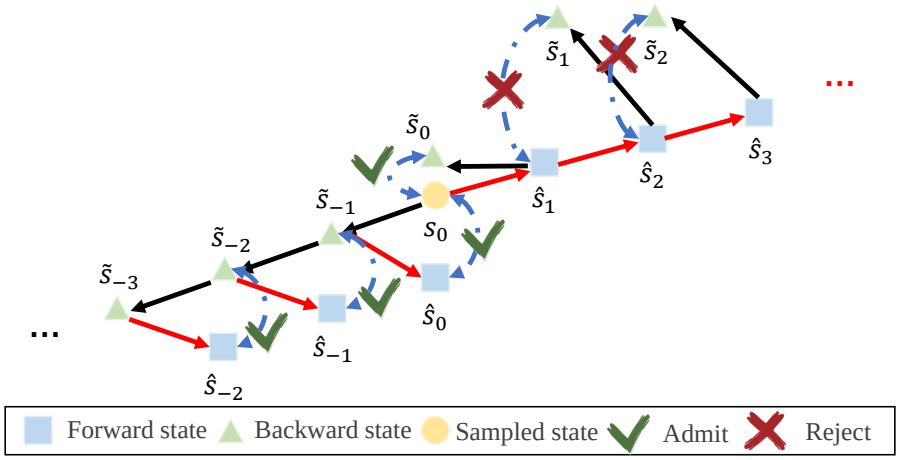



Figure 2: Illustration of the double check mechanism in bidirectional modeling. We use the trained opposite dynamics model to check whether the synthetic transitions generated by the current dynamics model are reliable. We are confident about the transitions that can be traced back to the starting state with small errors (green check) and reject those with large disagreements (red cross).

There are also some studies that focus on trajectory pruning [26, 77], while they involve uncertainty measurement. CABI, instead, ensures reliable imaginations by conducting double check with the forward and backward models, which fully exploits the advantages of bidirectional modeling.

**Model-based online RL.** Model-based online RL methods achieve superior sample efficiency [61, 25, 4, 23] by learning a dynamics model of the environment and planning with the model [62, 68, 71]. Learning a backward dynamics model that produces traces towards the aimed state is also widely explored [12, 37, 20, 33]. Among them, most similar to our work is [33], which leverages bidirectional model rollouts for reduced compounding error in the online setting. However, the main differences are: (1) CABI is proposed to augment the fixed dataset instead of performing policy optimization in a model-based way; (2) CABI interpolates a double check mechanism for reliable imaginations; (3) Model predictive control (MPC) [5] is not involved in CABI.

## 3 Preliminaries

We study RL under Markov Decision Process (MDP) specified by a tuple $\langle \mathcal{S}, \mathcal{A}, \rho_0, p, r, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\rho_0$ denotes initial state distribution, $p(s'|s,a)$ is the stochastic transition dynamics, $r(s,a) : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the reward function, and $\gamma \in [0,1)$ is the discount factor. The policy $\pi(a|s) : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}_+$ is a mapping from states to a probability distribution over actions. The goal of RL is to obtain a policy $\pi$ such that the expected discounted cumulative rewards can be maximized, $\max_\pi J_{\rho_0}(\pi) := \mathbb{E}_{s\sim\rho_0, a_t\sim\pi(\cdot|s_t), s_{t+1}\sim p(\cdot|s_t,a_t)} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$. In online RL, the agent learns from the experience collected from the interactions with the environment. However, in the offline setting, both interaction and exploration are infeasible, and the agent can only get access to the logged static dataset $\mathcal{D}_{\text{env}} = \{(s,a,r,s')\}$, which was gathered in advance by the unknown behavior policy. Since the fixed dataset $\mathcal{D}_{\text{env}}$ is typically a subset of full space $\mathcal{S} \times \mathcal{A}$, the generalization beyond the raw dataset becomes challenging. Model-based RL mitigates this issue by learning a dynamics model $\hat{p}(s'|s,a)$ and reward function $\hat{r}(s,a)$, and generating synthetic transitions to augment the dataset. However, there is no guarantee that the generated samples are reliable (see Section 4.1), and we focus on addressing this issue in this paper.

## 4 Confidence-Aware Bidirectional Offline Model-Based Imagination

In this section, we first use a toy example to illustrate the necessity of training bidirectional models with the double check mechanism. Then, we give the detailed framework of our method, Confidence-Aware Bidirectional Offline Model-Based Imagination (CABI).

### 4.1 You Need to Double Check Your State

Many model-free offline RL algorithms suffer from poor generalization as they are trained on a fixed dataset with limited samples. Model-based methods extend the logged dataset by generating synthetic transitions from the trained dynamics model. Despite such an advantage, they lack a mechanism for checking the *reliability* of the generated samples. If the model is inaccurate, poor transition samples that lie in the out-of-support region of the dataset can be generated, which may downgrade the performance of offline RL algorithms.

Human beings tend to conduct *double check* when they are uncertain about the outcome, e.g., clinical medicine research [60], autonomous driving [67], etc. Inspired by this nature, we propose to train bidirectional dynamics models and admit the samples where the forward model and backward model have few disagreements instead of roughly trusting all generated samples. In this way, we introduce conservatism into the transition itself instead of the critic or the actor networks. We give the illustration of double check mechanism in Figure 2.

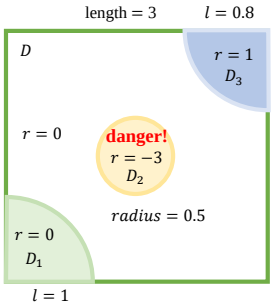
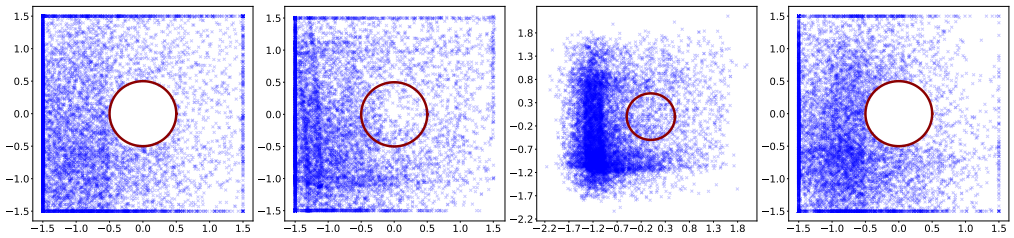



Figure 3: RiskWorld.

We argue that either forward dynamics model or backward dynamics model is unreliable, and bidirectional modeling in conjunction with the double check mechanism is critical for trustworthy sample generation. We verify this by designing a toy task, 2-dimensional environment with continuous state space and action space, namely RiskWorld, as shown in Figure 3. The central point of the square region in RiskWorld is $(0, 0)$, and the state space gives $D := [-1.5, 1.5] \times [-1.5, 1.5]$. Each episode, the agent randomly starts at the region $D_1 := \{(x, y)|(x+1.5)^2+(y+1.5)^2 \leq 1, x < 0, y < 0\}$ and takes actions $a \in [-0.5, 0.5]$. There is a danger zone $D_2 := \{(x, y)|x^2 + y^2 \leq 0.5^2\}$, and the done flag would turn into true if the agent steps into this region, along with a reward of $-3$. The agent will receive a reward of $+1$ if it lies in $D_3 := \{(x, y)|(x - 1.5)^2 + (y - 1.5)^2 \leq 0.8^2, x < 1.5, y < 1.5\}$, and 0 if it locates at $D \backslash (D_2 \cup D_3)$.

We run a random policy in RiskWorld for $10^4$ timesteps to collect an offline dataset. Figure 4(a) shows the state distribution (blue cross) of the dataset, where there are no transitions in $D_2$ (red circle area) as the episode terminates if the agent steps into $D_2$. To compare different ways of imagination, we train a forward model, a backward model, and a bidirectional model with the double check mechanism on this dataset. The training epoch is set to be 100, and the rollout horizon is set to be 3 for all of them. The detailed experimental setup is available in Appendix A.



(a) Random policy (b) Forward (c) Backward (d) Bidirectional

Figure 4: Visualizations of the offline dataset collected by a random policy (a), and synthetic transitions generated by a forward model (b), a backward model (c), a bidirectional model with the double check mechanism (d), based on the offline dataset in (a).

We use the trained forward model, reverse model, and bidirectional model to generate $10^4$ transition samples, and plot the state distributions of their generated samples respectively. As shown in Figure 4(b), the forward model generates many samples in the danger zone $D_2$ (red circle area). Figure 4(c) reveals that the reverse model generates a lot of illegal samples that lie out of the state space $D$, and also many transitions that lie in the dangerous area $D_2$. These all show evidence that both the forward and backward dynamics model fail to output reliable transitions. However, we observe in Figure 4(d) that bidirectional modeling with the double check mechanism successfully produces reliable and conservative synthetic samples, i.e., out-of-support or dangerous samples are not included, because

**Algorithm 1** CABI (Model Training)

**Input:** Dataset $\mathcal{D}_{\text{env}}$, iterations $N_1, N_2$, learning rate $\alpha_\psi$, $\alpha_\phi$, $\alpha_\theta$, $\alpha_\omega$

Randomly initialize forward model parameters $\psi$ and reverse model parameters $\phi$
**for** $i$ = 1 to $N_1$ **do**
  Compute $\mathcal{L}_\psi^{\text{fwd}}$ and $\mathcal{L}_\phi^{\text{bwd}}$ via (1) and (2)
  Update model parameters: $\psi \leftarrow \psi - \alpha_\psi \nabla_\psi \mathcal{L}_\psi^{\text{fwd}}$, $\phi \leftarrow \phi - \alpha_\phi \nabla_\phi \mathcal{L}_\phi^{\text{bwd}}$
**end for**
Randomly initialize forward and backward rollout policy parameters $\theta$, $\omega$
**for** $i$ = 1 to $N_2$ **do**
  Compute $\mathcal{L}_{\text{fvae}}$ and $\mathcal{L}_{\text{bvae}}$ via (3) and (4)
  Update forward and reverse rollout policy: $\theta \leftarrow \theta - \alpha_\theta \nabla_\theta \mathcal{L}_{\text{fvae}}$, $\omega \leftarrow \omega - \alpha_\omega \nabla_\omega \mathcal{L}_{\text{bvae}}$
**end for**

the forward model and backward model have large disagreements at those states. We hence argue that bidirectional modeling with double check is necessary for reliable data generation in offline RL.

### 4.2 Bidirectional Models Learning in CABI

**Bidirectional dynamics models training.** Our bidirectional modeling models transition probability and reward function simultaneously, i.e., the forward model $\hat{p}_\psi(s', r|s, a)$ and the reverse model $\hat{p}_\phi(s, r|s', a)$ parameterized by $\psi$ and $\phi$ respectively. The forward model $\hat{p}_\psi(s', r|s, a)$ represents the probability of the next state and corresponding reward given the current state and action, and the backward model $\hat{p}_\phi(s, r|s', a)$ outputs the probability of the current state and reward using the next state and action as input. We assume that the predicted reward function $\hat{r}(s, a)$ only depends on the current state $s$ and action $a$, then the unified model can be decomposed as $\hat{p}_\psi(s', r|s, a) = \hat{p}(s'|s, a)\hat{p}(r|s, a)$ and $\hat{p}_\phi(s, r|s', a) = \hat{p}(s|s', a)\hat{p}(r|s, a)$. We denote the loss functions for the forward model and backward model as $\mathcal{L}_\psi^{\text{fwd}}$ and $\mathcal{L}_\phi^{\text{bwd}}$ respectively, and optimize them by maximizing the log-likelihood via (1) and (2), where $\mathcal{D}_{\text{env}}$ is the raw static dataset.

$$\mathcal{L}_\psi^{\text{fwd}} = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}_{\text{env}}} \left[-\log \hat{p}_\psi(s', r|s, a)\right], \tag{1}$$

$$\mathcal{L}_\phi^{\text{bwd}} = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}_{\text{env}}} \left[-\log \hat{p}_\phi(s, r|s', a)\right]. \tag{2}$$

Following prior work [75, 26], we train an ensemble of bootstrapped probabilistic dynamics models, which has been widely demonstrated to be effective in model-based RL [8, 23]. Each model in the ensemble is parameterized by a multi-layer neural network, which outputs a Gaussian distribution $\mathcal{N}(\mu, \Sigma)$. Detailed hyperparameter setup for dynamics models training is deferred to Appendix C.

**Bidirectional rollout policies training.** We additionally train bidirectional generative models, which serve as rollout policies, and are used to generate actions to augment the static dataset. We model the rollout policy with a conditional variational autoencoder (CVAE) [27, 58, 16], which offers diverse actions while staying within the span of the dataset. CVAE is made up of an encoder $E$ that outputs the latent variable $z$ under the Gaussian distribution, and a decoder $D$ that maps $z$ to the desired space. We denote the forward rollout policy as $G_\theta^{\text{fwd}}(s)$ parameterized by $\theta = \{\xi_1, \nu_1\}$ where $\xi_1$ is the parameter of the encoder $E_{\xi_1}^{\text{fwd}}(s, a)$ and $\nu_1$ is the parameter of the decoder $D_{\nu_1}^{\text{fwd}}(s, z)$. The forward rollout policy is then trained by maximizing its variational lower bound, which is equivalent to minimizing the following loss:

$$\mathcal{L}_{\text{fvae}}(\theta) = \mathop{\mathbb{E}}_{(s,a,r,s')\sim\mathcal{D}_{\text{env}}, z\sim E_{\xi_1}^{\text{fwd}}(s,a)} \left[\left(a - D_{\nu_1}^{\text{fwd}}(s, z)\right)^2 + D_{\text{KL}}\left(E_{\xi_1}^{\text{fwd}}(s, a) \| \mathcal{N}(0, \mathbf{I})\right)\right], \tag{3}$$

where $D_{\text{KL}}(\cdot\|\cdot)$ denotes the KL-divergence, and $\mathbf{I}$ is an identity matrix. The first term of RHS of (3) represents the reconstruction loss where we want the decoded action to approximate the real action. Then for action generation, we first sample latent vector $z$ from the multivariate Gaussian distribution $\mathcal{N}(0, \mathbf{I})$, and then pass it with the current state $s$ into the decoder $D_{\nu_1}^{\text{fwd}}(s, z)$ to output the action.

Similarly, the backward rollout policy $G_\omega^{\text{bwd}}(s')$ parameterized by $\omega$ contains an encoder $E_{\xi_2}^{\text{bwd}}(s', a)$ and a decoder $D_{\nu_2}^{\text{bwd}}(s', z)$, $\omega = \{\xi_2, \nu_2\}$. The loss function of the backward rollout policy gives:

$$\mathcal{L}_{\text{bvae}}(\omega) = \mathop{\mathbb{E}}_{(s,a,r,s')\sim\mathcal{D}_{\text{env}}, z\sim E_{\xi_2}^{\text{bwd}}(s',a)} \left[\left(a - D_{\nu_2}^{\text{bwd}}(s', z)\right)^2 + D_{\text{KL}}\left(E_{\xi_2}^{\text{bwd}}(s', a) \| \mathcal{N}(0, \mathbf{I})\right)\right]. \tag{4}$$

**Algorithm 2** CABI (Data Generation)

**Input:** Offline dataset $\mathcal{D}_{\text{env}}$, horizon $H$, iteration $N$

Initialize model replay buffer $\mathcal{D}_{\text{model}} \leftarrow \emptyset$
**for** $i$ = 1 to $N$ **do**
  Sample state $s_t$ and next state $s_{t+1}$ from $\mathcal{D}_{\text{env}}$
  **for** $j$ = 0 to $H-1$ **do**
    Obtain forward rollout $\{s_{t+j}, a_{t+j}, r_{t+j}, s_{t+1+j}\}$ from $s_{t+j}$ by drawing samples from the forward model $\hat{p}_\psi$ and forward rollout policy $G_\theta^{\text{fwd}}$
    Generate backward state $\tilde{s}_{t+j}$ from $s_{t+1+j}$, and evaluate the deviation of $\tilde{s}_{t+j}$ from $s_{t+j}$
    Get backward rollout $\{s_{t-j}, a_{t-j}, r_{t-j}, s_{t+1-j}\}$ from $s_{t+1-j}$ by drawing samples from the backward model $\hat{p}_\phi$ and backward rollout policy $G_\omega^{\text{bwd}}$
    Generate forward state $\hat{s}_{t+1-j}$ from $s_{t-j}$. Evaluate the deviation of $\hat{s}_{t+1-j}$ from $s_{t+1-j}$
    Add selected imaginations into $\mathcal{D}_{\text{model}}$
  **end for**
**end for**
Get composite dataset $\mathcal{D}_{\text{total}} \leftarrow \mathcal{D}_{\text{env}} \bigcup \mathcal{D}_{\text{model}}$
Get the final policy $\pi_{\text{out}}$ with *any* model-free offline RL algorithm based on the dataset $\mathcal{D}_{\text{total}}$

We then draw $z$ from the Gaussian distribution $\mathcal{N}(0, \mathbf{I})$, and draw the action from the action decoder $D_{\nu_2}^{\text{bwd}}(s', z)$ with the next state $s'$ and latent variable $z$ as input.

We present the detailed procedure for the model training part of CABI in Algorithm 1.

### 4.3 Conservative Data Augmentation with CABI

After the bidirectional dynamics models and bidirectional rollout policies are well trained, we utilize them to generate imaginary samples. Each time, we sample a state $s_t$ from the raw dataset $\mathcal{D}_{\text{env}}$ to produce imagined forward trajectory $\hat{\tau}^{\text{fwd}} = \{s_{t+j}, a_{t+j}, r_{t+j}, s_{t+1+j}\}_{j=0}^{H-1}$ with the forward dynamics model $\hat{p}_\psi$ and forward rollout policy $G_\theta^{\text{fwd}}$, and sample the next state $s_{t+1}$ from $\mathcal{D}_{\text{env}}$ to generate imagined reverse trajectory $\hat{\tau}^{\text{bwd}} = \{s_{t-j}, a_{t-j}, r_{t-j}, s_{t+1-j}\}_{j=0}^{H-1}$ with the reverse dynamics model $\hat{p}_\phi$ and reverse rollout policy $G_\omega^{\text{bwd}}$. For each step in the rollout horizon $H$, we do double check and reject those badly imagined synthetic transitions.

To be specific, when performing forward imagination from $s_t$ and generating synthetic next state $\hat{s}_{t+1}$, we trace back from $\hat{s}_{t+1}$ with the reverse model, and get the backward state $\tilde{s}_t$. We evaluate the deviation of $\tilde{s}_t$ from $s_t$, and trust $\hat{s}_{t+1}$ if the deviation is small. Similarly, starting from the state $s_{t+1}$, we backtrack its previous state $\tilde{s}_t$ with the backward dynamics model, and then look forward from $\tilde{s}_t$ to get $\hat{s}_{t+1}$ with the forward dynamics model. We trust $\tilde{s}_t$ if the deviation of $\hat{s}_{t+1}$ from $s_{t+1}$ is small.

We keep those trustworthy rollouts and gather them to get the model buffer $\mathcal{D}_{\text{model}}$. We combine the synthetic model buffer $\mathcal{D}_{\text{model}}$ with $\mathcal{D}_{\text{env}}$ to obtain the final buffer $\mathcal{D}_{\text{total}}$, i.e., $\mathcal{D}_{\text{total}} = \mathcal{D}_{\text{env}} \cup \mathcal{D}_{\text{env}}$. We then can train *any* model-free offline RL algorithms based on the composite dataset.

One naïve way for implementing double check mechanism is to set a threshold $\delta$, and admit the transition if $\|s_t - \tilde{s}_t\|_2 \leq \delta$ for forward imagination, or $\|s_{t+1} - \hat{s}_{t+1}\|_2 \leq \delta$ for backward imagination. However, such a method lacks flexibility, and one may need to carefully tune $\delta$ per dataset based on the strong prior knowledge about the dataset, which impedes the application of double check mechanism. We resort to sorting the transitions in a mini-batch by the state deviation from small to large and keep the top $k\%$ of them that have the smallest deviation. We keep 20% transitions that have the smallest deviation for all of our experiments in Section 5 (empirical study on $k$ is available in Appendix C).

Our method is confidence-aware and conservative as we only admit the transitions that the forward model and backward model agree on, thus excluding those poor transitions from the model buffer $\mathcal{D}_{\text{model}}$. The full procedure for the data generation part of CABI is available in Algorithm 2.

Table 1: Normalized average score comparison of CABI+BCQ against different baselines on the Adroit "-v0" tasks, where score 0 represents the performance of a random policy and 100 corresponds to an expert policy performance. The results are averaged over the final 10 evaluations and 5 different random seeds. The highest mean scores are in **bold**.

| Task Name | CABI+BCQ | BCQ | UWAC | BEAR | BC | AWR | CQL | MOPO | COMBO |
|---|---|---|---|---|---|---|---|---|---|
| pen-cloned | 54.7±2.0 | 44.0 | 33.1 | 26.5 | **56.9** | 28.0 | 39.2 | -2.1 | -2.4 |
| pen-human | **75.1**±1.5 | 68.9 | 21.7 | -1.0 | 34.4 | 12.3 | 37.5 | 9.7 | 27.7 |
| pen-expert | **127.6**±2.0 | 114.9 | 111.9 | 105.9 | 85.1 | 111.0 | 107.0 | -0.6 | 11.5 |
| door-cloned | **0.5**±0.2 | 0.0 | 0.0 | -0.1 | -0.1 | 0.0 | 0.4 | -0.1 | 0.0 |
| door-human | 1.7±0.1 | 0.0 | 2.1 | -0.3 | 0.5 | 0.4 | **9.9** | -0.2 | -0.3 |
| door-expert | **105.3**±0.5 | 99.0 | 104.1 | 103.4 | 34.9 | 102.9 | 101.5 | -0.2 | 4.9 |
| relocate-cloned | -0.2±0.0 | -0.3 | -0.3 | -0.3 | **-0.1** | -0.2 | **-0.1** | -0.3 | **-0.1** |
| relocate-human | 0.1±0.1 | **0.5** | **0.5** | -0.3 | 0.0 | 0.0 | 0.2 | -0.3 | -0.3 |
| relocate-expert | **105.9**±1.0 | 41.6 | 105.6 | 98.6 | 101.3 | 91.5 | 95.0 | -0.2 | 17.2 |
| hammer-cloned | **4.3** ±1.6 | 0.4 | 0.4 | 0.3 | 0.8 | 0.4 | 2.1 | 0.2 | 0.4 |
| hammer-human | 3.1±2.2 | 0.5 | 1.1 | 0.3 | 1.5 | 1.2 | **4.4** | 0.2 | 0.2 |
| hammer-expert | **128.9**±0.9 | 107.2 | 110.6 | 127.3 | 125.6 | 39.0 | 86.7 | 0.3 | 0.3 |
| Total Score | **607.0** | 476.7 | 490.8 | 460.3 | 440.8 | 386.5 | 483.8 | 6.4 | 59.1 |

## 5 Experiments

In this section, we combine CABI with off-the-shelf model-free offline RL algorithms and conduct extensive experiments on the D4RL benchmarks [14]. In Section 5.1, we combine CABI with BCQ [16], and evaluate it on the challenging Adroit dataset to show the effectiveness of conservative data augmentation with CABI. We present a detailed ablation study in Section 5.2, where we aim to answer the following questions: (1) Is the double check mechanism a critical component for CABI? (2) How does CABI compare with the forward/reverse imagination? (3) How does CABI compare against other augmentation methods, e.g., random selection? Furthermore, we incorporate CABI with another recent model-free offline RL method, TD3_BC [15], and evaluate it on the MuJoCo datasets, to show the generality and advantages of CABI. We additionally combine CABI with IQL [28] and evaluate the performance of CABI+IQL on both Adroit tasks and MuJoCo tasks. Due to the space limit, the results are deferred to Appendix G.

### 5.1 Performance on Challenging Adroit Dataset

We demonstrate the benefits of CABI by combining it with BCQ and evaluating it on the challenging Adroit dataset [55]. Adroit dataset involves controlling a 24-DoF simulated robotic hand that aims at hammering a nail, opening a door, twirling a pen, or picking/moving a ball. It contains three types of datasets for each task (*human*, *cloned*, and *expert*), yielding a total of 12 datasets. This domain is very challenging for prior methods to learn from because the dataset is made up of narrow human demonstrations on a sparse reward, high-dimensional robotic manipulation task.

We summarize the overall results in Table 1, where we compare CABI+BCQ against recent model-free offline RL methods, such as UWAC [73], CQL [32], BCQ [16], and model-based offline RL methods, such as MOPO [75], and COMBO [74]. We run MOReL and COMBO on these datasets with our reproduced code. Results of MOPO and UWAC on the Adroit domain are acquired by running their official codebases, and the results of the rest baselines are taken directly from [14]. All methods are run over 5 different random seeds and normalized average scores are reported in Table 1. We only report the standard deviation for CABI+BCQ, and the full table is deferred to Appendix H.

As shown, CABI significantly boosts the performance of vanilla BCQ on almost all datasets, achieving a total score of **607.0** vs. 476.7 of BCQ. CABI+BCQ also surpasses the baseline model-free and model-based offline RL methods on 7 out of 12 datasets and achieves the highest total score.

It is worth noting that model-based offline RL methods generally fail on the Adroit tasks, because (1) the dataset distribution is narrow and high-dimensional, making it challenging for the trained forward dynamics model to generate accurate and reliable transitions; (2) the actions in the synthetic transitions are generated by the actor during the training process, thus the error may accumulate if

Table 2: Normalized average score comparison on the Adroit tasks between CABI+BCQ, forward imagination+BCQ, backward imagination+BCQ, and BOMI+BCQ, where $\pm$ captures standard deviation. The results are averaged over the final 10 evaluations and 5 different random seeds. The highest mean scores are in **bold**.

| Task name | BCQ | +Forward | +Backward | +BOMI | +CABI |
|---|---|---|---|---|---|
| pen-cloned | 44.0 | 41.2±1.1 | 36.8±6.6 | 43.4±6.1 | **54.7**±2.0 |
| pen-human | 68.9 | 57.8±9.3 | 60.9±5.6 | 49.6±1.4 | **75.1**±1.5 |
| pen-expert | 114.9 | 114.4±5.4 | 118.5±4.7 | 121.8±1.2 | **127.6**±2.0 |
| door-cloned | 0.0 | 0.0±0.0 | 0.0±0.0 | 0.0±0.0 | **0.5**±0.2 |
| door-human | 0.0 | -0.1±0.1 | 0.0±0.1 | 0.0±0.1 | **1.7**±0.1 |
| door-expert | 99.0 | 104.2±0.3 | 103.7±0.2 | 102.5±1.2 | **105.3**±0.5 |
| relocate-cloned | -0.3 | -0.3±0.0 | -0.3±0.0 | **-0.2**±0.0 | **-0.2**±0.0 |
| relocate-human | **0.5** | 0.0±0.0 | 0.0±0.0 | 0.0±0.1 | 0.1±0.1 |
| relocate-expert | 41.6 | 72.9±2.0 | 76.8±6.8 | 80.1±9.3 | **105.9**±1.0 |
| hammer-cloned | 0.4 | 1.7±0.1 | 0.4±0.1 | 3.1±3.8 | **4.3**±1.6 |
| hammer-human | 0.5 | 2.0±0.2 | 2.8±0.5 | 2.1±0.7 | **3.1**±2.2 |
| hammer-expert | 107.2 | 126.8±1.0 | 126.9±1.0 | 126.8±1.3 | **128.9**±0.9 |
| Total score | 476.7 | 520.6 | 526.5 | 529.2 | **607.0** |

the actor is updated towards a wrong direction. CABI, instead, alleviates the underlying issues via adopting the CVAE for action generation and conducting double check on state prediction.

### 5.2 Ablation Study

**Is the double check mechanism critical?** To answer this question, we exclude the double check mechanism in CABI and admit all generated synthetic samples from bidirectional models, which gives rise to Bidirectional Offline Model-based Imagination (BOMI). We evaluate CABI+BCQ and BOMI+BCQ on the Adroit tasks with identical parameter configuration over 5 different random seeds and show the average normalized scores in Table 2. It can be seen that BOMI brings some performance improvement on most of the tasks via data augmentation with bidirectional models and rollout policies. However, the generated data may be unreliable (we observe a performance drop in *pen-cloned, pen-human*), which impedes the benefits of bidirectional data augmentation. Such concern can be alleviated with the aid of the double check mechanism. As illustrated in Table 2, CABI+BCQ outperforms BOMI+BCQ on most tasks and incurs a much better total score.

**CABI against forward/backward imagination.** We incorporate BCQ with the pure forward model, backward model, and CABI, and conduct extensive experiments on the Adroit tasks over 5 different random seeds. The forward model and reverse model are trained with the same configuration as CABI. The results are summarized in Table 2. It can be seen that either the forward or reverse model results in limited improvement, which is consistent with the results of BOMI. As previously discussed, the forward model and reverse model may generate unreliable transitions. We see such evidence as the performance of BCQ falls on some of the tasks (e.g., *pen-cloned*) if trained on mere forward or reverse imagination. BCQ+CABI, instead, outperforms BCQ+Forward and BCQ+Backward on all tasks. Hence, we conclude that CABI guarantees trustworthy transitions for training, and brings improvement on almost all of the tasks.

**CABI against other augmentation methods.** We further compare CABI against three data augmentation methods: (1) CABI-random where we replace the CVAE with the random policy as the rollout policy in CABI; (2) R-20 where we *randomly* select 20% synthetic transitions for bidirectional imagination; (3) EV-20 where we select 20% transitions with the smallest *ensemble variance* for bidirectional imagination, i.e., we evaluate the variance of the output of the ensemble of the forward and backward dynamics models and reject those with large variance. We use BCQ as the base algorithm and run experiments on four Adroit tasks for these augmentation methods with identical parameter setup as CABI (e.g., real data ratio). The results in Table 4 show that CABI performs consistently better than these methods. Since the data augmentation process of CABI is isolated from the policy optimization, we cannot leverage a random rollout policy because the generated actions of a random policy may possibly lie out of the span of the dataset, which can negatively affect the performance of the agent. Hence, CVAE is critical to ensure a safe and conservative data

Table 3: Normalized average score comparison of CABI+TD3_BC vs. baseline methods on the D4RL MuJoCo "-v0" dataset, where score 0 corresponds to a random policy performance and 100 corresponds to an expert policy performance. The results are averaged over the final 10 evaluations and 5 different random seeds. The highest mean scores are in **bold**.

| Task Name | CABI+TD3_BC | TD3_BC | UWAC | MOPO | BCQ | BC | CQL | FisherBRC |
|---|---|---|---|---|---|---|---|---|
| halfcheetah-random | 15.1±0.4 | 10.2 | 2.3 | **35.4** | 2.2 | 2.0 | 21.7 | 32.2 |
| hopper-random | **11.9**±0.1 | 11.0 | 9.8 | 11.7 | 10.6 | 9.5 | 10.7 | 11.4 |
| walker2d-random | 6.4±1.5 | 1.4 | 3.8 | **13.6** | 4.9 | 1.2 | 2.7 | 0.6 |
| halfcheetah-medium-replay | 44.4±0.2 | 43.3 | 38.9 | **53.1** | 38.2 | 34.7 | 41.9 | 43.3 |
| hopper-medium-replay | 31.3±0.7 | 31.4 | 18.0 | **67.5** | 33.1 | 19.7 | 28.6 | 35.6 |
| walker2d-medium-replay | 29.4±1.3 | 25.2 | 8.4 | 39.0 | 15.0 | 8.3 | 15.8 | **42.6** |
| halfcheetah-medium | **45.1**±0.1 | 42.8 | 37.4 | 42.3 | 40.7 | 36.6 | 37.2 | 41.3 |
| hopper-medium | **100.4**±0.3 | 99.5 | 30.3 | 28.0 | 54.5 | 30.0 | 44.2 | 99.4 |
| walker2d-medium | **82.0**±0.4 | 79.7 | 17.4 | 17.8 | 53.1 | 11.4 | 57.5 | 79.5 |
| halfcheetah-medium-expert | **105.0**±0.2 | 97.9 | 40.6 | 63.3 | 64.7 | 67.6 | 27.1 | 96.1 |
| hopper-medium-expert | **112.7**±0.0 | 112.2 | 95.4 | 23.7 | 110.9 | 89.6 | 111.4 | 90.6 |
| walker2d-medium-expert | **108.4**±1.3 | 101.1 | 14.8 | 44.6 | 57.5 | 12.0 | 68.1 | 103.6 |
| halfcheetah-expert | **107.6**±0.9 | 105.7 | 104.0 | - | 89.9 | 105.2 | 82.4 | 106.8 |
| hopper-expert | **112.4**±0.1 | 112.2 | 109.1 | - | 107.0 | 111.5 | 111.2 | 112.3 |
| walker2d-expert | **108.6**±1.5 | 105.7 | 88.4 | - | 102.3 | 56.0 | 103.8 | 79.9 |
| Total Score | **1020.7** | 979.3 | 618.6 | - | 784.6 | 595.3 | 764.3 | 974.6 |

augmentation. Meanwhile, relying on the ensemble variance for data selection is not trustworthy as the models in the ensemble are trained on the identical data and may all incur wrong predictions but small variance.

Table 4: Normalized average score comparison on four Adroit tasks. The results are averaged over the final 10 evaluations and 5 different random seeds. CABI-random denotes the rollout policy in CABI is a random policy. R-20 denotes **R**andomly keep 20% transitions, EV-20 denotes keep 20% samples that have the smallest **E**nsemble **V**ariance.

| Task Name | BCQ | +CABI | +R-20 | +EV-20 | +CABI-random |
|---|---|---|---|---|---|
| pen-cloned | 44.0 | **54.7**±2.0 | 41.2±3.0 | 40.4±2.0 | 37.6±7.8 |
| pen-expert | 114.9 | **127.6**±2.0 | 112.6±5.6 | 118.8±2.5 | 106.3±3.7 |
| hammer-cloned | 0.4 | **4.3**±1.6 | 0.9±0.6 | 0.4±0.1 | 0.3±0.0 |
| hammer-expert | 107.2 | **128.9**±0.9 | 104.2±24.6 | 125.5±5.5 | 103.8±1.5 |

### 5.3 Broad Results on MuJoCo Dataset

To show the generality of our method, we integrate CABI with another recent model-free offline RL method, TD3_BC [15], and conduct experiments on 15 MuJoCo datasets. We widely compare CABI+TD3_BC against other recent model-free offline RL methods, such as FisherBRC [29], UWAC [73], CQL [32], and model-based batch RL method, MOPO [75]. We run CABI+TD3_BC over 5 different random seeds. We also run UWAC using the official codebase on the MuJoCo datasets over 5 different random seeds. The results of TD3_BC, BC, CQL, FisherBRC are taken directly from [15], and the results of other baseline methods are taken from [73].

The experimental results in Table 3 reveal that our approach exceeds all baseline methods on 10 out of 15 datasets, and is the strongest in terms of the total score. On almost all of the tasks, we observe performance improvement with CABI over the base TD3_BC algorithm. Unfortunately, with the existence of behavioral cloning term, the performance improvement upon TD3_BC is limited. Still, the experimental results in Table 1 and 3 show that CABI is a powerful data augmentation method and can boost the performance of the base model-free offline RL methods.

## 6 Conclusion and Limitations

In this paper, we follow human nature and propose to do *double check* during synthetic transition generation to ensure that the imagined samples are conservative and accurate. We admit samples that the forward model and reverse model agree on. Our method, CABI, involves training bidirectional dynamics models and rollout policies and can be combined with *any* off-the-shelf model-free offline RL algorithms. Extensive experiments on the D4RL benchmarks show that our method significantly boosts the performance of the base model-free offline RL method, and can achieve competitive or better performance against recent baseline methods. For future work, it is interesting to evaluate CABI in the online setting and investigate whether it can benefit model-based online RL as well.

The major limitation of our proposed method lies in the computation cost as we train bidirectional dynamics models and rollout policies. However, since CABI is isolated from policy optimization, we can enhance the dataset beforehand.

## Acknowledgments and Disclosure of Funding

This work was supported in part by the Science and Technology Innovation 2030-Key Project under Grant 2021ZD0201404, in part by the NSF China under Grant 61872009. The authors would like to thank the anonymous reviewers for their valuable comments and advice.

## References

[1] A. Argenson and G. Dulac-Arnold. Model-Based Offline Planning. *ArXiv*, abs/2008.05556, 2021.

[2] K. Asadi, E. Cater, D. K. Misra, and M. L. Littman. Towards a Simple Approach to Multi-step Model-based Reinforcement Learning. *ArXiv*, abs/1811.00128, 2018.

[3] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI Gym. *ArXiv*, abs/1606.01540, 2016.

[4] J. Buckman, D. Hafner, G. Tucker, E. Brevdo, and H. Lee. Sample-Efficient Reinforcement Learning with Stochastic Ensemble Value Expansion. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[5] E. F. Camacho and C. B. Alba. *Model Predictive Control*. Springer Science & Business Media, 2013.

[6] C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. T. Koehn, and T. Robinson. One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling. *ArXiv*, abs/1312.3005, 2014.

[7] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision Transformer: Reinforcement Learning via Sequence Modeling. *ArXiv*, abs/2106.01345, 2021.

[8] K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[10] Z. Deng, Z. Fu, L. Wang, Z. Yang, C. Bai, Z. Wang, and J. Jiang. SCORE: Spurious COrrelation REduction for Offline Reinforcement Learning. *ArXiv*, abs/2110.12468, 2021.

[11] C. P. Diehl, T. Sievernich, M. Krüger, F. Hoffmann, and T. Bertram. UMBRELLA: Uncertainty-Aware Model-Based Offline Reinforcement Learning Leveraging Planning. *ArXiv*, abs/2111.11097, 2021.

[12] A. D. Edwards, L. Downs, and J. C. Davidson. Forward-Backward Reinforcement Learning. *ArXiv*, abs/1803.10227, 2018.

[13] C. Finn and S. Levine. Deep Visual Foresight for Planning Robot Motion. In *International Conference on Robotics and Automation (ICRA)*, 2017.

[14] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4RL: Datasets for Deep Data-Driven Reinforcement Learning. *ArXiv*, abs/2004.07219, 2020.

[15] S. Fujimoto and S. S. Gu. A Minimalist Approach to Offline Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[16] S. Fujimoto, D. Meger, and D. Precup. Off-Policy Deep Reinforcement Learning without Exploration. In *International Conference on Machine Learning (ICML)*, 2019.

[17] S. Fujimoto, H. van Hoof, and D. Meger. Addressing Function Approximation Error in Actor-Critic Methods. In *International Conference on Machine Learning (ICML)*, 2018.

[18] C. Gelada and M. G. Bellemare. Off-Policy Deep Reinforcement Learning by Bootstrapping the Covariate Shift. In *AAAI Conference on Artificial Intelligence*, volume 33, 2019.

[19] O. Gottesman, F. D. Johansson, J. Meier, J. Dent, D. Lee, S. Srinivasan, L. Zhang, Y. Ding, D. Wihl, X. Peng, J. Yao, I. Lage, C. Mosch, L. wei H. Lehman, M. Komorowski, A. A. Faisal, L. A. Celi, D. A. Sontag, and F. Doshi-Velez. Evaluating Reinforcement Learning Algorithms in Observational Health Settings. *ArXiv*, abs/1805.12298, 2018.

[20] A. Goyal, P. Brakel, W. Fedus, S. Singhal, T. Lillicrap, S. Levine, H. Larochelle, and Y. Bengio. Recall Traces: Backtracking Models for Efficient Reinforcement Learning. In *International Conference on Learning Representations (ICLR)*, 2019.

[21] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *International Conference on Machine Learning (ICML)*, 2018.

[22] K. J. Holyoak and D. Simon. Bidirectional Reasoning in Decision Making by Constraint Satisfaction. *Journal of Experimental Psychology: General*, 128:3–31, 1999.

[23] M. Janner, J. Fu, M. Zhang, and S. Levine. When to Trust Your Model: Model-Based Policy Optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[24] M. Janner, Q. Li, and S. Levine. Offline Reinforcement Learning as One Big Sequence Modeling Problem. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[25] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[26] R. Kidambi, A. Rajeswaran, P. Netrapalli, and T. Joachims. MOReL: Model-Based Offline Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[27] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. *ArXiv*, abs/1312.6114, 2013.

[28] I. Kostrikov, A. Nair, and S. Levine. Offline Reinforcement Learning with Implicit Q-Learning. In *International Conference on Learning Representations*, 2022.

[29] I. Kostrikov, J. Tompson, R. Fergus, and O. Nachum. Offline Reinforcement Learning with Fisher Divergence Critic Regularization. In *International Conference on Machine Learning (ICML)*, 2021.

[30] A. Kumar, R. Agarwal, T. Ma, A. C. Courville, G. Tucker, and S. Levine. DR3: Value-Based Deep Reinforcement Learning Requires Explicit Regularization. *ArXiv*, abs/2112.04716, 2021.

[31] A. Kumar, J. Fu, G. Tucker, and S. Levine. Stabilizing Off-Policy Q-Learning via Bootstrapping Error Reduction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[32] A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative Q-Learning for Offline Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[33] H. Lai, J. Shen, W. Zhang, and Y. Yu. Bidirectional Model-based Policy Optimization. In *International Conference on Machine Learning (ICML)*, 2020.

[34] S. Lange, T. Gabel, and M. A. Riedmiller. Batch Reinforcement Learning. In *Reinforcement Learning*, 2012.

[35] R. Laroche and P. Trichelair. Safe Policy Improvement with Baseline Bootstrapping. In *International Conference on Machine Learning (ICML)*, 2019.

[36] B.-J. Lee, J. Lee, and K.-E. Kim. Representation Balancing Offline Model-based Reinforcement Learning. In *International Conference on Learning Representations (ICLR)*, 2021.

[37] K. Lee, Y. Seo, S. Lee, H. Lee, and J. Shin. Context-aware Dynamics Model for Generalization in Model-Based Reinforcement Learning. In *International Conference on Machine Learning (ICML)*, 2020.

[38] S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. *ArXiv*, abs/2005.01643, 2020.

[39] Y. Liu, A. Swaminathan, A. Agarwal, and E. Brunskill. Off-Policy Policy Gradient with State Distribution Correction. *ArXiv*, abs/1904.08473, 2019.

[40] Y. Liu, A. Swaminathan, A. Agarwal, and E. Brunskill. Provably Good Batch Reinforcement Learning Without Great Exploration. *ArXiv*, abs/2007.08202, 2020.

[41] C. Lu, P. J. Ball, J. Parker-Holder, M. A. Osborne, and S. J. Roberts. Revisiting Design Choices in Offline Model Based Reinforcement Learning. In *International Conference on Learning Representation*, 2022.

[42] J. Lyu, X. Ma, X. Li, and Z. Lu. Mildly Conservative Q-learning for Offline Reinforcement Learning. In *Thirty-sixth Conference on Neural Information Processing Systems*, 2022.

[43] X. Ma, Y. Yang, H. Hu, Q. Liu, J. Yang, C. Zhang, Q. Zhao, and B. Liang. Offline Reinforcement Learning with Value-based Episodic Memory. *ArXiv*, abs/2110.09796, 2021.

[44] Y. J. Ma, D. Jayaraman, and O. Bastani. Conservative Offline Distributional Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.

[45] A. Mandlekar, F. Ramos, B. Boots, L. Fei-Fei, A. Garg, and D. Fox. IRIS: Implicit Reinforcement without Interaction at Scale for Learning Control from Offline Robot Manipulation Data. In *International Conference on Robotics and Automation (ICRA)*, 2020.

[46] T. Matsushima, H. Furuta, Y. Matsuo, O. Nachum, and S. Gu. Deployment-Efficient Reinforcement Learning via Model-Based Offline Optimization. In *International Conference on Learning Representations (ICLR)*, 2021.

[47] L. Meng, M. Wen, Y. Yang, C. Le, X. Li, W. Zhang, Y. Wen, H. Zhang, J. Wang, and B. Xu. Offline Pre-trained Multi-Agent Decision Transformer: One Big Sequence Model Tackles All SMAC Tasks. *ArXiv*, abs/2112.02845, 2021.

[48] R. Munos, T. Stepleton, A. Harutyunyan, and M. G. Bellemare. Safe and Efficient Off-Policy Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[49] O. Nachum, Y. Chow, B. Dai, and L. Li. DualDICE: Behavior-Agnostic Estimation of Discounted Stationary Distribution Corrections. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[50] O. Nachum, B. Dai, I. Kostrikov, Y. Chow, L. Li, and D. Schuurmans. AlgaeDICE: Policy Gradient from Arbitrary Experience. *ArXiv*, abs/1912.02074, 2019.

[51] A. Nair, M. Dalal, A. Gupta, and S. Levine. Accelerating Online Reinforcement Learning with Offline Datasets. *ArXiv*, abs/2006.09359, 2020.

[52] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. V. Dillon, B. Lakshminarayanan, and J. Snoek. Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[53] D. Precup, R. S. Sutton, and S. Dasgupta. Off-Policy Temporal Difference Learning with Function Approximation. In *International Conference on Machine Learning (ICML)*, 2001.

[54] R. Rafailov, T. Yu, A. Rajeswaran, and C. Finn. Offline Reinforcement Learning from Images with Latent Space Models. In *Learning for Dynamics and Control (L4DC)*, 2021.

[55] A. Rajeswaran, V. Kumar, A. Gupta, J. Schulman, E. Todorov, and S. Levine. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. *ArXiv*, abs/1709.10087, 2018.

[56] S. Ross and J. A. Bagnell. Agnostic System Identification for Model-Based Reinforcement Learning. In *International Conference on Machine Learning (ICML)*, 2012.

[57] A. Singh, A. Yu, J. Yang, J. Zhang, A. Kumar, and S. Levine. COG: Connecting New Skills to Past Experience with Offline Reinforcement Learning. *ArXiv*, abs/2010.14500, 2020.

[58] K. Sohn, H. Lee, and X. Yan. Learning Structured Output Representation using Deep Conditional Generative Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

[59] A. L. Strehl, J. Langford, L. Li, and S. M. Kakade. Learning from Logged Implicit Exploration Data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2010.

[60] R. Subramanyam, M. A. Mahmoud, D. Buck, and A. M. Varughese. Infusion Medication Error Reduction by Two-Person Verification: A Quality Improvement Initiative. *Pediatrics*, 138, 2016.

[61] R. S. Sutton. Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming. In *International Conference on Machine Learning (ICML)*, 1990.

[62] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[63] R. S. Sutton, A. R. Mahmood, and M. White. An Emphatic Approach to the Problem of Off-policy Temporal-Difference Learning. *Journal of Machine Learning Research*, 17:2603–2631, 2016.

[64] A. Swaminathan and T. Joachims. Batch Learning from Logged Bandit Feedback through Counterfactual Risk Minimization. *J. Mach. Learn. Res.*, 16:1731–1755, 2015.

[65] P. Swazinna, S. Udluft, and T. A. Runkler. Overcoming Model Bias for Robust Offline Deep Reinforcement Learning. *Engineering Applications of Artificial Intelligence*, 104:104366, 2021.

[66] E. Todorov, T. Erez, and Y. Tassa. MuJoCo: A Physics Engine for Model-based Control. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[67] D. N. Van, L. Kuhnert, S. Thamke, J. Schlemper, and K.-D. Kuhnert. A Novel Approach for A Double-Check of Passable Vegetation Detection in Autonomous Ground Vehicles. In *IEEE Conference on Intelligent Transportation Systems*, 2012.

[68] T. J. Walsh, S. Goschin, and M. L. Littman. Integrating Sample-Based Planning and Model-Based Reinforcement Learning. In *AAAI Conference on Artificial Intelligence*, 2010.

[69] J. Wang, W. Li, H. Jiang, G. Zhu, S. Li, and C. Zhang. Offline Reinforcement Learning with Reverse Model-based Imagination. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[70] L. Wang, W. Zhang, X. He, and H. Zha. Supervised Reinforcement Learning with Recurrent Neural Network for Dynamic Treatment Recommendation. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.

[71] T. Wang and J. Ba. Exploring Model-based Planning with Policy Networks. In *International Conference on Learning Representations (ICLR)*, 2020.

[72] Y. Wu, G. Tucker, and O. Nachum. Behavior Regularized Offline Reinforcement Learning. *ArXiv*, abs/1911.11361, 2019.

[73] Y. Wu, S. Zhai, N. Srivastava, J. M. Susskind, J. Zhang, R. Salakhutdinov, and H. Goh. Uncertainty Weighted Actor-Critic for Offline Reinforcement Learning. In *International Conference on Machine Learning (ICML)*, 2021.

[74] T. Yu, A. Kumar, R. Rafailov, A. Rajeswaran, S. Levine, and C. Finn. COMBO: Conservative Offline Model-Based Policy Optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[75] T. Yu, G. Thomas, L. Yu, S. Ermon, J. Y. Zou, S. Levine, C. Finn, and T. Ma. MOPO: Model-based Offline Policy Optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[76] A. Zanette, M. J. Wainwright, and E. Brunskill. Provable Benefits of Actor-Critic Methods for Offline Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[77] X. Zhan, X. Zhu, and H. Xu. Model-Based Offline Planning with Trajectory Pruning. *ArXiv*, abs/2105.07351, 2021.

[78] W. Zhou, S. Bajracharya, and D. Held. PLAS: Latent Action Space for Offline Reinforcement Learning. In *Conference on Robot Learning (CoRL)*, 2020.

## Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

# A Experimental Setup of Toy RiskWorld Task

In this section, we give the detailed experimental setup of our toy RiskWorld task. RiskWorld is a 2-dimensional, continuous state space, continuous action space environment as shown in Figure 3. We suppose the central point of RiskWorld is $(0, 0)$, and the permitted range of RiskWorld gives $D := [-1.5, 1.5] \times [-1.5, 1.5]$, i.e., the length of RiskWorld is 3. The state information in RiskWorld is composed of the coordinates of the agent, i.e., $s = (x, y), x, y \in [-1.5, 1.5]$. There is a dangerous area $D_2$ locating at the central point with radius 0.5, i.e., $D_2 := \{(x, y)|x^2 + y^2 \leq 0.5^2\}$. The agent randomly starts at a point in $D_1 := \{(x, y)|(x + 1.5)^2 + (y + 1.5)^2 \leq 1, x < 0, y < 0\}$, and can take actions $a \in [-0.5, 0.5]$. There is also a high reward zone locating at $D_3 := \{(x, y)|(x - 1.5)^2 + (y - 1.5)^2 \leq 0.8^2, x < 1.5, y < 1.5\}$. The reward function $r(s, a)$ is defined in (5).

$$r(s,a) = \begin{cases} -3, & \text{if } s \in D_2 := \{(x,y)|x^2+y^2 \leq 0.5^2\}, \\ 1, & \text{if } s \in D_3 := \{(x,y)|(x-1.5)^2+(y-1.5)^2 \leq 0.8^2, x < 1.5, y < 1.5\}, \\ 0, & \text{if } s \in D \backslash (D_2 \cup D_3). \end{cases} \tag{5}$$

The agent will receive a large minus reward of $-3$ if it steps into the dangerous zone $D_2$, and the done flag will turn into *true*. The agent is not allowed to step out of the legal region $D$. The episode length for RiskWorld is set to be 300. The RiskWorld is intrinsically a sparse reward environment. We run a random policy on RiskWorld for $10^4$ timesteps and log the transition data it collected during interactions to form a static dataset *RiskWorld-random*.

Model-based reinforcement learning (RL) learns either forward dynamics or reverse dynamics of the environment [62, 20], and can produce imaginary transitions for training, which has been widely demonstrated to be effective in improving the sample efficiency of RL in the online setting [4, 23]. The forward dynamics model $\hat{p}_\psi(s', r|s, a)$ predicts the next state and the corresponding reward function given the current state and action, and the reverse dynamics model $\hat{p}_\phi(s, r|s', a)$ outputs the previous state and reward signal given action and the next state. Bidirectional modeling combines both forward dynamics model and backward dynamics model.

To compare different ways of imagination, i.e., the forward imagination, reverse imagination, and bidirectional imagination with the double check mechanism, we train a forward dynamics model, a backward dynamics model, and a bidirectional dynamics model on RiskWorld-random dataset, respectively. We represent the forward and reverse dynamics model by training a probabilistic neural network. The forward model $\hat{p}_\psi(s', r|s, a)$ parameterized by $\psi$ receives the current state and action as input, and outputs a multivariate Gaussian distribution that predicts the next state and reward as shown in (6).

$$\hat{p}_\psi(s', r|s, a) = \mathcal{N}(\mu_\psi(s, a), \mathbf{\Sigma}_\psi(s, a)), \tag{6}$$

where $\mu_\psi$ and $\mathbf{\Sigma}_\psi$ represent the mean and variance of the forward model $\hat{p}_\psi(s', r|s, a)$, respectively.

Similarly, for the backward model $\hat{p}_\phi(s, a|s', a)$ parameterized by $\phi$, it adopts the next state and action as input and outputs a multivariate normal distribution predicting reward signal and the previous state (see (7)).

$$\hat{p}_\phi(s, a|s', a) = \mathcal{N}(\mu_\phi(s', a), \mathbf{\Sigma}_\phi(s', a)), \tag{7}$$

where $\mu_\phi$ and $\mathbf{\Sigma}_\phi$ denote the mean and variance of the backward model $\hat{p}_\phi(s, r|s', a)$, respectively.

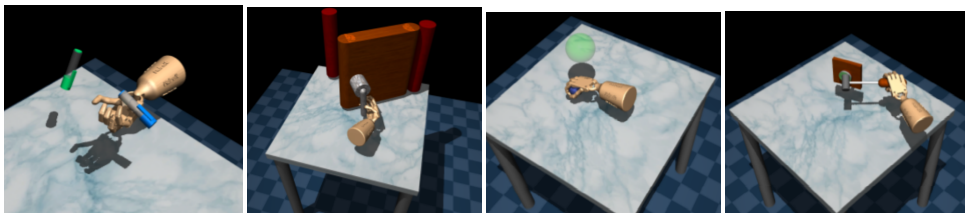
The probabilistic neural network is modeled by a multi-layer neural network that consists of 4 feedforward layers with 400 hidden units. We adopt *swish* activation for each intermediate layer. Following prior works [23, 75, 26], we train an ensemble of seven such probability neural networks for both the forward and backward model. We use a hold-out set made up of 1000 transitions to validate the performance of the trained dynamics, and select the five models that have the best performance accordingly. When performing forward or reverse imagination, we randomly pick one model out of the five best model candidates to generate synthetic trajectories per step. Considering the simplicity of the toy RiskWorld task, we train both the forward dynamics model and reverse dynamics model for 100 epochs, and the rollout length (horizon) is set to be 3 for the forward model, reverse model, and bidirectional model. We use the trained dynamics model (forward, reverse, bidirectional) to generate $10^4$ imaginary transition samples, and log their model buffer respectively. We plot in Figure 4 the model buffer of these dynamics models and the raw static dataset obtained by running the random policy.

# B Datasets and Evaluation Setting on the D4RL Benchmarks

In this section, we give a detailed description of the datasets we used in this paper, and also describe the evaluation setting that is adopted on the D4RL benchmarks [14]. D4RL is specially designed for evaluating offline RL (or batch RL) algorithms, which covers the dimensions that offline RL may encounter in practical applications, such as passively logged data, human demonstrations, etc.

### B.1 Adroit datasets and MuJoCo datasets

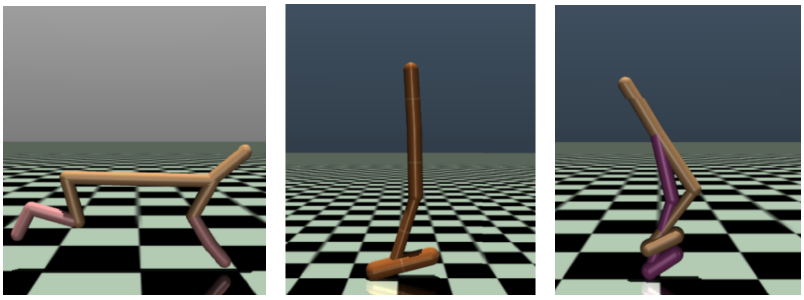
The Adroit dataset involves controlling a 24-DoF simulated Shadow Hand robot to perform tasks like hammering a nail, opening a door, twirling a pen, and picking/moving a ball, as shown in Figure 5. The Adroit domain is super challenging for even online RL algorithms because: (1) the dataset contains narrow human demonstrations; (2) this domain solves sparse reward, high-dimensional robotic manipulation tasks. There are four tasks in the dataset, and there are three types of datasets for each task, *cloned, human, expert*. **human:** a small number of demonstrations operated by a human (25 trajectories per task). **expert:** a large amount of expert data from a fine-tuned RL policy. **cloned:** a large amount of data generated by performing imitation learning on the human demonstrations, running the policy, and mixing the data at a 50-50 ratio with the demonstrations. Dataset mixing is involved for *cloned* as the cloned policies themselves fail on the tasks, making the dataset otherwise hard to learn from.



(a) Pen (b) Door (c) Relocate (d) Hammer

Figure 5: Adroit datasets. There are four tasks, namely pen, door, relocate, and hammer.

The MuJoCo dataset is collected during the interactions with the continuous action environments in Gym [3] simulated by MuJoCo [66]. We adopt three tasks in this dataset, *halfcheetah, hopper, walker2d* as illustrated in Figure 6. Each task in the MuJoCo dataset contains five types of datasets, *random, medium, medium-replay, medium-expert, expert*. **random:** a large amount of data from a random policy. **medium:** experiences collected from an early-stopped SAC policy for 1M steps. **medium-replay:** replay buffer of a policy trained up to the performance of the medium agent. **expert:** a large amount of data gathered by the SAC policy that is trained to completion. **medium-expert:** a large amount of data by mixing the medium data and expert data at a 50-50 ratio.



(a) HalfCheetah (b) Hopper (c) Walker2d

Figure 6: MuJoCo datasets. We conduct experiments on halfcheetah, hopper, and walker2d tasks.

Note that the Adroit dataset is qualitatively different from the MuJoCo Gym dataset because (1) there do not exist human demonstrations in the MuJoCo dataset; (2) the reward is dense in MuJoCo, making it less challenging to learn from; (3) the dimension of transitions in MuJoCo is low compared with Adroit. It is hard for even online RL algorithms to learn useful policies on the Adroit tasks, while it is easy for online RL methods to achieve superior performance on the MuJoCo environments.

### B.2 Evaluation setting in D4RL

D4RL suggests using the normalized score metric to evaluate the performance of the offline RL algorithms [14]. Denote the expected return of a random policy on the dataset as $C_r$ (reference min score), and the expected return of an expert policy as $C_e$ (reference max score). Suppose that an offline RL algorithm achieves an expected

return of $C$ after training on the given dataset. Then the normalized score $\tilde{C}$ is given by (8).

$$\tilde{C} = \frac{C - C_r}{C_e - C_r} \times 100 = \frac{C - \text{performance of random policy}}{\text{performance of expert policy} - \text{performance of random policy}} \times 100. \quad (8)$$

The normalized score ranges roughly from 0 to 100, where 0 corresponds to the performance of a random policy and 100 corresponds to the performance of an expert policy. We give the detailed reference min score $C_r$ and reference max score $C_e$ in Table 5, where all of the tasks share the same reference min score and reference max score across different types of datasets.

Table 5: The referenced min score and max score for the Adroit dataset and MuJoCo dataset in D4RL.

| Domain | Task Name | Reference min score $C_r$ | Reference max score $C_e$ |
|---|---|---|---|
| Adroit | pen | 96.26 | 3076.83 |
| Adroit | door | −56.51 | 2880.57 |
| Adroit | relocate | −6.43 | 4233.88 |
| Adroit | hammer | −274.86 | 12794.13 |
| MuJoCo | halfcheetah | −280.18 | 12135.0 |
| MuJoCo | hopper | −20.27 | 3234.3 |
| MuJoCo | Walker2d | 1.63 | 4592.3 |

# C Implementation Details and Hyperparameters

## C.1 Implementation details

In this section, we give implementation details and hyperparameters for Confidence-Aware Bidirectional Offline Model-Based Imagination (CABI). We represent the approximated forward dynamics and reward model, and backward dynamics and reward model by training a probabilistic neural network. The configuration of the probabilistic neural network is identical to Appendix A. That is, the forward model and reverse model are modeled as a multivariate Gaussian distribution with mean $\mu$ and variance $\mathbf{\Sigma}$. For the forward model $\hat{p}_\psi(s', r|s, a)$ parameterized by $\psi$, it accepts the current state and action as input and generates the next state and reward. The backward model $\hat{p}_\phi(s, r|s', a)$ parameterized by $\phi$ receives the next state and current action as input and outputs the former state and scalar reward. The probabilistic neural network is modeled by a multi-layer neural network that contains 4 feedforward layers, with 400 hidden units in each layer, and a *swish* activation in each intermediate layer. We train an ensemble of seven such probabilistic neural networks and select the best five models based on their performance on a hold-out set made up of 1000 transitions from the offline dataset.

As a data augmentation method, CABI does not actively generate actions during the training process, i.e., the policy optimization process is isolated from the data generation process. We use a conditional variational autoencoder (CVAE) to approximate the behavior policy in the static dataset. We give brief introduction to the VAE in Appendix E. The CVAE $G$ contains an encoder $E$ and a decoder $D$. Both the encoder and the decoder in the forward rollout policy and backward rollout policy contain two intermediate layers with 750 hidden units each layer. We adopt *relu* activation for each intermediate layer. Specifically, we train a forward rollout policy with a CVAE $G_\theta^{\text{fwd}}(s)$, which contains an encoder $E_{\xi_1}(s, a)$ and a decoder $D_{\nu_1}(s, z)$, $\theta = \{\xi_1, \nu_1\}$, and a reverse rollout policy $G_\omega^{\text{bwd}}(s') = \{E_{\xi_2}(s', a), D_{\nu_2}(s', z)\}$, where $\omega = \{\xi_2, \nu_2\}$. Note that the forward rollout policy $G_\theta^{\text{fwd}}(s)$ and reverse rollout policy $G_\omega^{\text{bwd}}(s')$ sample actions using stochastic inference from an underlying latent space, so as to increase diversity in the generated actions.

Intrinsically, CABI can be combined with *any* off-the-shelf model-free offline RL algorithms. In this work, we incorporate CABI with BCQ [16] and TD3_BC [15], and conduct extensive experiments on the Adroit dataset and MuJoCo dataset on the D4RL benchmarks, respectively. There are generally three steps when combining CABI with model-free offline RL methods: (1) **Model training.** We first train bidirectional dynamics models and bidirectional rollout policies using the raw static offline dataset $\mathcal{D}_{\text{env}}$; (2) **Data generation.** After the bidirectional models and rollout policies are well trained, we utilize them to generate imaginary trajectories, while conducting double check and admitting high-confidence transitions simultaneously. This will induce the model generated dataset $\mathcal{D}_{\text{model}}$; (3) **Policy optimization.** We then merge the real dataset $\mathcal{D}_{\text{env}}$ with the imagined dataset $\mathcal{D}_{\text{model}}$ to form a composite dataset $\mathcal{D}_{\text{total}}$, i.e., $\mathcal{D}_{\text{total}} = \mathcal{D}_{\text{env}} \cup \mathcal{D}_{\text{model}}$. The mini-batch samples used for training the model-free offline RL algorithms come from $\mathcal{D}_{\text{env}}$ and $\mathcal{D}_{\text{model}}$. To be specific, we define the ratio of data come from $\mathcal{D}_{\text{env}}$ (real data) as $\eta$. Suppose we use a mini-batch size of $N$ for training the algorithm. Then for each optimization step, we sample $\eta N$ samples from $\mathcal{D}_{\text{env}}$, and sample $(1 - \eta)N$ samples from $\mathcal{D}_{\text{model}}$. We pick the optimal real data ratio $\eta$ among $\{0.1, 0.3, 0.5, 0.7, 0.9\}$ using grid search.

**Influence of $k$.** For the double check mechanism, we keep the top 20% of samples in a mini-batch that the forward model and backward model have the smallest disagreements. As explained in Section 4.3, we do not set a threshold and then admit samples that the deviation of the forward model and backward model on them are smaller than the threshold, because it requires human knowledge and the threshold is task-specific. However, in real-world problems, we cannot always have full knowledge about the system. For better flexibility and generality, we choose to keep top $k\%$ samples. Note that if we use $k = 0$, then the influence of data augmentation will be excluded. If we use $k = 100$, then CABI will degenerate into Bidirectional Model-based Offline Imagination (BOMI), where there is no double check procedure. Intuitively, if $k$ is small, few samples can be left, which may negatively affect the advantages of data augmentation with a model. While if $k$ is large, some poorly imagined transitions may be included in the model buffer. We simply set $k = 20$ by default, and use it throughout our experiments on the Adroit dataset and MuJoCo dataset. We conduct experiments on two types of MuJoCo datasets, *random, medium*, with varied $k$ in $\{0, 10, 20, 50, 100\}$ over 5 different random seeds. The results are shown in Table 6, where we observe performance drop for both large $k$ and small $k$. We hence set $k = 20$ for all of our experiments.

Table 6: Normalized average score of CABI+TD3_BC on two datasets, random and medium, from MuJoCo dataset with different values of $k$. CABI+TD3_BC with $k = 0$ degenerates into vanilla TD3_BC, and CABI+TD3_BC with $k = 100$ turns into BOMI+TD3_BC. We **bold** the highest mean.

| Task Name | $k = 0$ | $k = 10$ | $k = 20$ | $k = 50$ | $k = 100$ |
|---|---|---|---|---|---|
| halfcheetah-random | 10.2 | 14.8 | **15.1** | 14.0 | 11.4 |
| hopper-random | 11.0 | 10.3 | **11.9** | 10.6 | 9.3 |
| walker2d-random | 1.4 | 4.9 | **6.4** | 5.8 | 4.3 |
| halfcheetah-medium | 42.8 | 44.6 | **45.1** | 44.4 | 44.3 |
| hopper-medium | 99.5 | 99.7 | **100.4** | 32.3 | 3.1 |
| walker2d-medium | 79.7 | 78.7 | **82.0** | 79.8 | 78.6 |

**Computation time and compute infrastructure.** The computation time for CABI ranges from 4 to 14 hours on the MuJoCo and Adroit tasks. The model training time differs on different types of datasets, e.g., it takes much less time to train our bidirectional models and rollout policies on *medium-replay* and *human* datasets (about 40 minutes), while it takes comparatively longer time to train on other types of datasets (about 2-6 hours). TD3_BC consumes about 3 hours to run on all MuJoCo datasets, and BCQ takes about 6-8 hours to train on the Adroit tasks. We run both CABI+BCQ and CABI+TD3_BC for $1 \times 10^6$ timesteps. We additionally run IQL [28], and the results are presented in Section G. IQL takes about 3-7 hours to run on all tasks with $1 \times 10^6$ timesteps. We give detailed compute infrastructure in Appendix F.

**Discussion on CABI and ROMI [69].** A recent work, Reverse Offline Model-based Imagination (ROMI) [69], explores the data augmentation in offline RL via training a reverse dynamics model. It is worth noting that we do not directly compare with ROMI+BCQ as there are *many secondary components* in the codebase of ROMI (https://github.com/wenzhe-li/romi), e.g., prioritized experience replay, modifying state information, adopting varied rollout policies for different domains, etc. CABI represents the rollout policy using only conditional variational autoencoder (CVAE). Also, ROMI assumes that the termination functions are known. However, CABI does not include any prior knowledge about termination conditions, even on simple MuJoCo tasks. That generally follows the key claims in the recent work of [15]. For a fair comparison, we disable the forward model as well as the double check mechanism in CABI to get our reverse model. As for the comparison with forward imagination, we disable the double check mechanism and the reverse dynamics part of CABI to get our pure forward model.

**Ensemble variance.** In the main text, we compare our CABI against EV-20, which rejects transitions with large ensemble variance. The ensemble variance is calculated in the following way where we take the forward imagination as an example: suppose we have an ensemble of forward dynamics models $f_i(\cdot|s, a), i = 1, \ldots, N$ with the ensemble size $N$. Then each model in the ensemble predicts a next state $s'_i$. We then have a collection of next state $\{s'_i\}, i = 1, \ldots, N$. We then randomly pick one next state while recording the variance in the ensemble at the same time. We reject the generated next state if the variance in the ensemble is large. That is, we evaluate the variance of $\{s'_i\}, i = 1, \ldots, N$. We sort the transitions in a batch by their calculated variance, and only trust the top 20% that have the smallest ensemble variance.

**Other implementation details.** On the Adroit tasks, we combine CABI with BCQ, and compare against recent state-of-the-art methods, including vanilla BCQ [16], UWAC [73], CQL [32], MOPO [75], and COMBO [74], etc. On the MuJoCo domain, we incorporate CABI with TD3_BC [15], and compare with baseline methods like FisherBRC [29], UWAC, CQL, BEAR [31], MOPO, etc. Note that we omit some baseline methods, such as AWAC [51] and BRAC [72] in the MuJoCo domain, as they do not obtain good enough performance for comparison. We run UWAC with the official codebase (https://github.com/apple/ml-uwac), and so is MOPO (https://github.com/tianheyu927/mopo). We run MOPO on the Adroit tasks as those results are not reported

in the original paper, and we take the results of MOPO on the MuJoCo datasets from [73] directly. We re-run UWAC on the Adroit domain and MuJoCo domain because, unfortunately, we cannot reproduce the results reported in its original paper. All baseline methods are run for $1 \times 10^6$ timesteps over 5 different random seeds.

### C.2 Hyperparameters

In this subsection, we give the detailed hyperparameter setup for our experiments in Table 7. We keep the top 20% samples in a mini-batch for all tasks. For simplicity, we use identical rollout length for the forward model and backward model. For all of the MuJoCo tasks and most of the Adroit tasks, the rollout length for both the forward model and backward model is set to be 3, which yields a total horizon of 6. On datasets that the model disagreement are comparatively large for long horizons (e.g., *pen-expert*, see Table 9), we set the forward and backward horizon as 1, which leads to a total horizon of 2. We use the forward and backward horizon 5 for *hammer-human* as we experimentally find that it performs better. On other tasks, the forward and backward horizon is set to be 3 by default. Note that the model disagreement on MuJoCo datasets are smaller than 0.1 for all horizons, and the trained forward and backward model well fits MuJoCo datasets. We therefore adopt the forward and backward horizon of 3 for all of these tasks.

Table 7: Hyperparameters setup in our experiments with CABI on the Adroit dataset and MuJoCo dataset. ForH = Forward Horizon, BackH = Backward Horizon.

| Domain | Dataset Type | Task Name | ForH | BackH | Real Data Ratio $\eta$ |
|---|---|---|---|---|---|
| Adroit | human | pen | 1 | 1 | 0.7 (BCQ), 0.5 (IQL) |
| Adroit | human | door | 3 | 3 | 0.7 (BCQ), 0.5 (IQL) |
| Adroit | human | relocate | 3 | 3 | 0.9 (BCQ), 0.5 (IQL) |
| Adroit | human | hammer | 5 | 5 | 0.5 (BCQ), 0.7 (IQL) |
| Adroit | cloned | pen | 1 | 1 | 0.5 (BCQ), 0.5 (IQL) |
| Adroit | cloned | door | 3 | 3 | 0.5 (BCQ), 0.7 (IQL) |
| Adroit | cloned | relocate | 3 | 3 | 0.3 (BCQ), 0.5 (IQL) |
| Adroit | cloned | hammer | 3 | 3 | 0.5 (BCQ), 0.7 (IQL) |
| Adroit | expert | pen | 1 | 1 | 0.7 (BCQ), 0.9 (IQL) |
| Adroit | expert | door | 3 | 3 | 0.7 (BCQ), 0.9 (IQL) |
| Adroit | expert | relocate | 3 | 3 | 0.9 (BCQ), 0.7 (IQL) |
| Adroit | expert | hammer | 1 | 1 | 0.9 (BCQ), 0.5 (IQL) |
| MuJoCo | random | halfcheetah | 3 | 3 | 0.7 (TD3_BC), 0.7 (IQL) |
| MuJoCo | random | hopper | 3 | 3 | 0.1 (TD3_BC), 0.7 (IQL) |
| MuJoCo | random | walker2d | 3 | 3 | 0.1 (TD3_BC), 0.7 (IQL) |
| MuJoCo | medium | halfcheetah | 3 | 3 | 0.7 (TD3_BC), 0.7 (IQL) |
| MuJoCo | medium | hopper | 3 | 3 | 0.9 (TD3_BC), 0.7 (IQL) |
| MuJoCo | medium | walker2d | 3 | 3 | 0.7 (TD3_BC), 0.9 (IQL) |
| MuJoCo | medium-replay | halfcheetah | 3 | 3 | 0.5 (TD3_BC), 0.7 (IQL) |
| MuJoCo | medium-replay | hopper | 3 | 3 | 0.7 (TD3_BC), 0.7 (IQL) |
| MuJoCo | medium-replay | walker2d | 3 | 3 | 0.5 (TD3_BC), 0.9 (IQL) |
| MuJoCo | medium-expert | halfcheetah | 3 | 3 | 0.7 (TD3_BC), 0.9 (IQL) |
| MuJoCo | medium-expert | hopper | 3 | 3 | 0.9 (TD3_BC), 0.9 (IQL) |
| MuJoCo | medium-expert | walker2d | 3 | 3 | 0.7 (TD3_BC), 0.9 (IQL) |
| MuJoCo | expert | halfcheetah | 3 | 3 | 0.7 (TD3_BC), 0.9 (IQL) |
| MuJoCo | expert | hopper | 3 | 3 | 0.9 (TD3_BC), 0.9 (IQL) |
| MuJoCo | expert | walker2d | 3 | 3 | 0.7 (TD3_BC), 0.9 (IQL) |

We search for the best $\eta$ over $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. We find that the real data ratio $\eta = 0.7$ and $\eta = 0.9$ are generally effective for CABI. The best ratio $\eta$ strongly depends on the dataset and may need to be tuned manually. For example, *random* dataset in the MuJoCo domain and *cloned* dataset in the Adroit domain are poor for training naturally, and small $\eta$ is therefore needed. While for *expert* dataset or *medium* dataset, a comparatively large $\eta$ is better.

## D Model Prediction Error and Model Disagreement

In this section, we are interested in exploring (1) can CABI generate more trustworthy transitions in complex environments (2) the model disagreement of the forward and backward models in CABI under different horizons,

Table 8: Comparison of one-step model prediction error of the forward model, reverse model, and bidirectional model with the double check mechanism on the Adroit tasks.

| Task Name | Unidirectional | | Bidirectional (CABI) | |
|---|---|---|---|---|
| | $\epsilon_{\text{fwd}}$ | $\epsilon_{\text{bwd}}$ | $\epsilon_{\text{fwd}}$ | $\epsilon_{\text{bwd}}$ |
| pen-cloned | 837.5 | 777.4 | **751.5** | **603.0** |
| pen-human | 195.0 | 177.8 | **107.5** | **97.8** |
| pen-expert | 169.01 | 179.8 | **143.58** | **149.8** |
| door-cloned | 24.7 | 27.7 | **0.05** | **0.01** |
| door-human | 18.2 | 20.2 | **4.4** | **6.0** |
| door-expert | 4.3 | 10.5 | **1.8** | **6.3** |
| relocate-cloned | 351.9 | 1271.4 | **0.0** | **0.9** |
| relocate-human | 229.5 | 267.4 | **178.6** | **205.1** |
| relocate-expert | 201.5 | 48.3 | **167.5** | **37.9** |
| hammer-cloned | 1330.8 | 1984.3 | **72.3** | **1602.2** |
| hammer-human | 577.9 | 596.4 | **480.9** | **477.8** |
| hammer-expert | 601.1 | 561.4 | **557.2** | **503.4** |

aiming at checking whether the model disagrees with each other more with the increment of the horizon. To begin with, we define the one-step model prediction error to check whether CABI admits more accurate transitions.

**Definition D.1** (Model Prediction Error)**.** Given the static offline dataset $\mathcal{D}_{\text{env}}$, we define one-step model prediction error for forward model $\epsilon_{\text{fwd}}$ and reverse model $\epsilon_{\text{bwd}}$ as:

$$\epsilon_{\text{fwd}} = \mathbb{E}_{\substack{(s,a,r,s')\sim\mathcal{D}_{\text{env}} \\ \hat{s},\hat{r}\sim\hat{p}_{\psi}(\cdot|s,a)}} \left[\|s' - \hat{s}\|_2^2 + (r - \hat{r})^2\right],$$

$$\epsilon_{\text{bwd}} = \mathbb{E}_{\substack{(s,a,r,s')\sim\mathcal{D}_{\text{env}} \\ \tilde{s},\tilde{r}\sim\hat{p}_{\phi}(\cdot|s',a)}} \left[\|s - \tilde{s}\|_2^2 + (r - \tilde{r})^2\right].$$

$\epsilon_{\text{fwd}}$ and $\epsilon_{\text{bwd}}$ generally capture the accuracy of the trained dynamics models, i.e., smaller $\epsilon_{\text{fwd}}$ and $\epsilon_{\text{bwd}}$ indicate better forward and backward dynamics model fitting. Intuitively, the one-step model prediction error of admitted samples in CABI should be smaller than that of the mere forward dynamics model or reverse dynamics model, as only transitions that the forward model and backward model are all confident about are admitted. We verify this by comparing the one-step model error in the forward model, backward model, and CABI, where we keep the top 20% imagined samples for CABI. The results are presented in Table 8, where we observe CABI leads to significant error drop for both forward and reverse models on all of the tasks. For example, the forward error in *door-cloned* drops from 24.7 to **0.05** and the backward error drops from 27.7 to **0.01**, which reveals that CABI can select reliable and conservative imaginations that well fit the dataset for training.

We then define the model disagreement of forward model and backward model in the following.

**Definition D.2** (Bidirectional Model Disagreement)**.** For a sampled current state $s$ and reward $r$ from a given static offline dataset $\mathcal{D}$, a series of forward states $\hat{s}_i$ and reward signals $\hat{r}_i$ can be generated by utilizing the forward model, $i = 1, \ldots, H$. Denote the imagined backward state and reward based on $\hat{s}_i$ as $\tilde{s}_{i-1}$ and $\tilde{r}_{i-1}$, respectively, $i = 1, \ldots, H$. Then the forward model disagreement is defined as:

$$\epsilon_i^{\text{fwd}} = \begin{cases} \mathbb{E}\left[\|\tilde{s}_0 - s\|_2^2 + (\tilde{r}_0 - r)^2\right], & \text{if } i = 1, \\ \mathbb{E}\left[\|\tilde{s}_{i-1} - \hat{s}_{i-1}\|_2^2 + (\tilde{r}_{i-1} - \hat{r}_{i-1})^2\right], & \text{if } i \geq 2. \end{cases} \tag{9}$$

Similarly, for a sampled next state $s'$ and reward $r$ from the offline dataset, an imaginary trajectory $\hat{\tau}_{\text{bwd}}$ containing the backward states $\tilde{s}_{-i}$ and rewards $\tilde{r}_{-i}$, $i = 1, \ldots, H$, can be generated with the aid of the backward dynamics model. For each imagined state $\tilde{s}_{-i}$ in $\hat{\tau}_{\text{bwd}}$, its previous state $\hat{s}_{-i+1}$ and reward $\hat{r}_{-i+1}$ are generated by the forward model, $i = 1, \ldots, H$. Then the backward model disagreement is defined as:

$$\epsilon_i^{\text{bwd}} = \begin{cases} \mathbb{E}\left[\|\hat{s}_0 - s'\|_2^2 + (\hat{r}_0 - r)^2\right], & \text{if } i = 1, \\ \mathbb{E}\left[\|\tilde{s}_{-i+1} - \hat{s}_{-i+1}\|_2^2 + (\tilde{r}_{-i+1} - \hat{r}_{-i+1})^2\right], & \text{if } i \geq 2. \end{cases} \tag{10}$$

**Remark:** The above definition generally capture the disagreement between the forward model and the backward model. Note that the model disagreement is different from the model prediction error defined above even if the rollout length is set as 1. The model prediction error measure how well the forward or backward model fits the transition data, while the model disagreement measures how the forward model and backward model disagree on the transition. We take the forward setting as an example. The forward model prediction error is the deviation between the forward imagined state and reward against the real next state and reward signal, while the forward model disagreement is the deviation between the real *current state* and scalar reward with the backward imagined *current state* and reward based on the forward imagination.

Table 9 details model disagreement comparison of CABI against CABI without double check mechanism, which turns into BOMI, i.e., bidirectional modeling without double check, under different horizons. We perform experiments on 12 Adroit tasks and the sampled mini-batch size is set to be $5 \times 10^4$. As demonstrated in the table, the model disagreement of CABI is significantly smaller than that of BOMI under different rollout steps. It is worth noting that the model disagreement for both CABI and BOMI is irrelevant to the rollout length. The model disagreement generally is small when performing one-step model rollout, and increases if longer horizon imaginations are generated (some datasets like *door-human* are exceptions). We observe that the model disagreement in CABI is much more controllable than BOMI, e.g., on some expert datasets.

Table 9: The model disagreement comparison of **CABI** and **BOMI** under different rollout length. The superscript $\mathrm{fwd}$ denotes forward, and $\mathrm{bwd}$ denotes backward. The subscript denotes the imagined horizon, e.g., $\epsilon_1^{\mathrm{fwd}}$ represents the forward model disagreement under horizon 1. The best results are in **bold** (smaller is better).

| Task Name | $\epsilon_1^{\mathrm{fwd}}$ | | $\epsilon_2^{\mathrm{fwd}}$ | | $\epsilon_3^{\mathrm{fwd}}$ | | $\epsilon_1^{\mathrm{bwd}}$ | | $\epsilon_2^{\mathrm{bwd}}$ | | $\epsilon_3^{\mathrm{bwd}}$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CABI | BOMI | CABI | BOMI | CABI | BOMI | CABI | BOMI | CABI | BOMI | CABI | BOMI |
| pen-human | **0.23** | 1829.21 | **20.63** | 1608.66 | **21.24** | 1608.94 | **0.23** | 1857.14 | **19.93** | 1693.70 | **19.25** | 1686.21 |
| door-human | **0.53** | 36.03 | **0.21** | 28.06 | **0.21** | 29.04 | **0.53** | 36.19 | **0.40** | 29.38 | **0.38** | 30.07 |
| relocate-human | **0.00** | 268.37 | **5.32** | 237.03 | **5.45** | 236.64 | **0.00** | 265.34 | **4.68** | 193.56 | **4.72** | 195.08 |
| hammer-human | **0.67** | 401.58 | **5.56** | 261.43 | **5.53** | 259.86 | **0.69** | 402.67 | **5.63** | 260.93 | **5.43** | 258.86 |
| pen-cloned | **159.58** | 12048.06 | **373.10** | 22506.36 | **359.56** | 22440.26 | **136.89** | 11624.98 | **359.91** | 22350.11 | **331.18** | 21939.83 |
| door-cloned | **0.00** | 46.35 | **0.00** | 63.75 | **0.00** | 62.89 | **0.00** | 30.68 | **0.00** | 61.03 | **0.00** | 62.96 |
| relocate-cloned | **0.00** | 745.77 | **0.00** | 938.51 | **0.00** | 929.95 | **0.00** | 269.66 | **0.00** | 909.64 | **0.00** | 889.79 |
| hammer-cloned | **0.0** | 3791.10 | **0.01** | 4509.45 | **0.01** | 4395.46 | **0.0** | 1021.85 | **0.02** | 4443.90 | **0.03** | 4447.14 |
| pen-expert | **0.42** | 1965.76 | **21.92** | 9642.54 | **26.05** | 9691.13 | **0.41** | 1953.54 | **56.12** | 9569.12 | **52.69** | 9399.45 |
| door-expert | **0.95** | 257.49 | **0.94** | 261.84 | **1.04** | 263.52 | **1.01** | 258.51 | **0.79** | 276.60 | **0.69** | 276.12 |
| relocate-expert | **0.05** | 649.59 | **1.47** | 770.99 | **2.18** | 784.79 | **0.06** | 652.63 | **0.06** | 631.83 | **0.05** | 631.74 |
| hammer-expert | **1.03** | 6135.13 | **58.07** | 82422.41 | **60.78** | 82633.47 | **1.02** | 6198.06 | **11.24** | 92882.87 | **10.70** | 92102.51 |

The double check mechanism we introduced in the main text selects trustworthy synthetic samples based on the deviation between states, i.e., transition samples with small state deviation will be kept in the model buffer. While we can also trust the transition samples via the model disagreement, i.e., keep transition samples with small model disagreement. We experimentally find that evaluating the deviation between states brings almost the same performance as evaluating the model disagreement under the identical hyperparameter setup. We choose to select transitions according to the deviation between states alone as shown in Algorithm 2 for both space and time saving during data generation process of CABI.

# E Omitted Background for VAE

In this section, we provide a brief introduction to the variational autoencoder (VAE) [27]. Given a dataset $X = \left\{x^{(i)}\right\}_{i=1}^{N}$, the VAE is trained to generate samples that come from the same distribution as the data points. That is to say, the goal of a VAE is to maximize $p_\theta(X) = \prod_{i=1}^{N} p_\theta\left(x^{(i)}\right)$, where $\theta$ is the parameter of the approximate maximum-likelihood (ML) or maximum a posterior (MAP) estimation. To reach this goal, a latent variable $z$ sampled from its posterior distribution $p(z)$ is introduced, and we model a decoder $p_\theta(X|z)$ parameterized by $\theta$. However, directly optimizing the marginal likelihood $p_\theta(X) = \int p(z)p_\theta(X|z)dz$ is intractable. Instead, VAE approximates the true posterior $p_\theta(z|X)$ via training an encoder $q_\phi(z|X)$, and we resort to optimizing the evidence lower bound (ELBO) on the log-likelihood of the data as shown in (11).

$$\max_{\theta,\phi} \log p_\theta(X) \geq \max_{\theta,\phi} \mathbb{E}_{q_\phi(z|X)}[\log p_\theta(X|z)] - D_{\mathrm{KL}}(q_\phi(z|X)\|p_\theta(z)). \tag{11}$$

The first term in the right-hand-side of (11) denotes the reconstruction loss, where $z$ is sampled from $q_\phi(z|X)$. The second term represents the KL-divergence between the learned encoder of $z$ and its true prior. The encoder $q_\phi(z|X)$ is usually set to be a multivariate Gaussian distribution with mean $\mu_\phi$ and variance $\mathbf{\Sigma}_\phi$. The prior of the latent variable $z$ is set to be a standard multivariate Gaussian distribution. Optimizing the lower bound in (11) enables the trained model to generate samples similar to the data distribution. After the VAE is well trained, we sample $z$ from the encoder $q_\phi(z|X)$ and pass it through the decoder $p_\theta(X|z)$ to obtain samples.

In this work, we use the conditional variational autoencoder (CVAE) [16] to model the behavior policy in the dataset. CVAE is a variant of the vanilla VAE, which aims to model $p_\theta(X|Y)$. Similar to the original ELBO of VAE, CVAE optimizes the conditional lower bound as shown in (12).

$$\max_{\theta,\phi} \log p_\theta(X|Y) \geq \max_{\theta,\phi} \mathbb{E}_{q_\phi(z|X,Y)}[\log p_\theta(X|z,Y)] - D_{\mathrm{KL}}(q_\phi(z|X,Y)\|p_\theta(z|Y)). \tag{12}$$

## F Compute Infrastructure

In Table 10, we list the compute infrastructure that we use to run all of the baseline algorithms and experiments.

Table 10: Compute infrastructure.

| CPU | GPU | Memory |
|---|---|---|
| AMD EPYC 7452 | RTX3090×8 | 288GB |

## G Experimental Results of CABI+IQL

In this section, we additionally incorporate CABI with a recently proposed offline RL method, IQL [28]. IQL learns without querying OOD samples. Such a learning paradigm ensures that the whole learning process is conducted under the support of the dataset, and a safe policy can be learned. However, as we explained in the main text, the datasets often cannot contain all possible transitions. Hence, the generalization capability of IQL is actually limited. With the aid of CABI, such concern can be mitigated to some extent. We conduct experiments on 12 Adroit datasets and 15 MuJoCo datasets over 5 different random seeds. For IQL, we use its official codebase (https://github.com/ikostrikov/implicit_q_learning) to run on all 27 datasets over 5 random seeds with the hyperparameters suggested by the authors. We incorporate CABI with IQL and run CABI+IQL over 5 different random seeds. The forward and backward horizons for CABI+IQL are identical to CABI+BCQ on Adroit tasks and CABI+TD3_BC on MuJoCo datasets. We summarize the results in Table 11 and Table 12.

As shown, CABI boosts the performance of IQL on all 27 datasets of Adroit and MuJoCo. CABI+IQL outperforms baseline methods on 10 out of 12 datasets. While on MuJoCo datasets, CABI+IQL only surpasses baseline methods on 5 out of 15 datasets, due to the fact that the base method IQL itself has poor performance on "-v0" datasets. Nevertheless, CABI+IQL has a total score of 604.1 on Adroit, surpassing the total score 562.5 of the vanilla IQL. CABI+IQL achieves a total score of 909.3 on MuJoCo datasets, while vanilla IQL only has a total score of 860.7. We want to emphasize here that we do not aim to beat the most recent strong baseline methods in this paper, the key point we want to carry here is the conservative data augmentation with CABI is effective and beneficial for the performance improvement over the base offline RL algorithms. The empirical experiments work as the evidence to validate our claim.

## H Omitted Full Comparison of CABI against Baselines

In this section, we provide the full comparison of CABI against baseline methods as we omit standard deviation for baselines in the main text due to space limitation. We show in Table 13 the full performance comparison of CABI+BCQ against BCQ [16], UWAC [73], CQL [32], MOPO [75], COMBO [74], etc. We additionally compare against SAC. As MOPO and COMBO do not report the performance on the Adroit dataset in their original paper, we run COMBO on the Adroit tasks over 5 different random seeds with our reproduced code, and run MOPO and UWAC with their official codebases on the Adroit tasks over 5 different random seeds, respectively. The results of the rest of the baselines are taken directly from [14].

Table 14 gives the full comparison of CABI+TD3_BC against TD3_BC [15], BCQ [16], UWAC [73], FisherBRC [29], CQL [32], MOPO [75], etc. We additionally compare CABI+BCQ against BEAR here. The results of BC, TD3_BC, CQL, and FisherBRC are taken directly from [15], and the results of UWAC are acquired by running the official codebase over 5 different random seeds. The results of BEAR are taken directly from [73]. We do not report standard deviation for BEAR and BCQ.

Table 11: Normalized average score comparison of CABI+IQL against different baselines on the Adroit "-v0" tasks, where score 0 represents the performance of a random policy and 100 corresponds to an expert policy performance. The highest mean scores are in **bold**.

| Task Name | CABI+IQL | IQL | UWAC | BEAR | BC | AWR | CQL | MOPO | COMBO |
|---|---|---|---|---|---|---|---|---|---|
| pen-cloned | 42.2±6.1 | 35.2±7.3 | 33.1 | 26.5 | **56.9** | 28.0 | 39.2 | -2.1 | -2.4 |
| pen-human | **72.0**±9.1 | 68.7±8.6 | 21.7 | -1.0 | 34.4 | 12.3 | 37.5 | 9.7 | 27.7 |
| pen-expert | **129.1**±0.6 | 118.4±6.9 | 111.9 | 105.9 | 85.1 | 111.0 | 107.0 | -0.6 | 11.5 |
| door-cloned | **0.8**±0.4 | 0.7±0.5 | 0.0 | -0.1 | -0.1 | 0.0 | 0.4 | -0.1 | 0.0 |
| door-human | **11.5**±3.6 | 3.3±1.3 | 2.1 | -0.3 | 0.5 | 0.4 | 9.9 | -0.2 | -0.3 |
| door-expert | **105.7**±0.2 | 105.2±0.3 | 104.1 | 103.4 | 34.9 | 102.9 | 101.5 | -0.2 | 4.9 |
| relocate-cloned | **-0.1**±0.0 | -0.2±0.0 | -0.3 | -0.3 | **-0.1** | -0.2 | **-0.1** | -0.3 | **-0.1** |
| relocate-human | 0.4±0.3 | 0.0±0.0 | **0.5** | -0.3 | 0.0 | 0.0 | 0.2 | -0.3 | -0.3 |
| relocate-expert | **107.4**±0.2 | 105.6±0.5 | 105.6 | 98.6 | 101.3 | 91.5 | 95.0 | -0.2 | 17.2 |
| hammer-cloned | **2.4** ±0.2 | 1.6±1.0 | 0.4 | 0.3 | 0.8 | 0.4 | 2.1 | 0.2 | 0.4 |
| hammer-human | **4.8**±1.8 | 2.3±0.6 | 1.1 | 0.3 | 1.5 | 1.2 | 4.4 | 0.2 | 0.2 |
| hammer-expert | **127.9**±1.2 | 121.7±1.3 | 110.6 | 127.3 | 125.6 | 39.0 | 86.7 | 0.3 | 0.3 |
| Total Score | **604.1** | 562.5 | 490.8 | 460.3 | 440.8 | 386.5 | 483.8 | 6.4 | 59.1 |

Table 12: Normalized average score comparison of CABI+IQL vs. baseline methods on the D4RL MuJoCo "-v0" dataset, where score 0 corresponds to a random policy performance and 100 corresponds to an expert policy performance. The highest mean scores are in **bold**.

| Task Name | CABI+IQL | IQL | UWAC | MOPO | BCQ | BC | CQL | FisherBRC |
|---|---|---|---|---|---|---|---|---|
| halfcheetah-random | 18.4±0.9 | 16.2±0.2 | 2.3 | **35.4** | 2.2 | 2.0 | 21.7 | 32.2 |
| hopper-random | 11.4±0.1 | 9.3±1.8 | 9.8 | **11.7** | 10.6 | 9.5 | 10.7 | 11.4 |
| walker2d-random | 8.0±0.5 | 6.2±2.2 | 3.8 | **13.6** | 4.9 | 1.2 | 2.7 | 0.6 |
| halfcheetah-medium-replay | 42.2±0.2 | 40.5±0.4 | 38.9 | **53.1** | 38.2 | 34.7 | 41.9 | 43.3 |
| hopper-medium-replay | 36.8±0.4 | 33.4±1.1 | 18.0 | **67.5** | 33.1 | 19.7 | 28.6 | 35.6 |
| walker2d-medium-replay | 17.2±0.8 | 15.8±1.7 | 8.4 | 39.0 | 15.0 | 8.3 | 15.8 | **42.6** |
| halfcheetah-medium | 41.6±0.1 | 41.2±0.1 | 37.4 | **42.3** | 40.7 | 36.6 | 37.2 | 41.3 |
| hopper-medium | 40.0±12.9 | 30.7±0.0 | 30.3 | 28.0 | 54.5 | 30.0 | 44.2 | **99.4** |
| walker2d-medium | 55.1±2.3 | 50.8±7.7 | 17.4 | 17.8 | 53.1 | 11.4 | 57.5 | **79.5** |
| halfcheetah-medium-expert | **96.7**±1.3 | 89.0±0.7 | 40.6 | 63.3 | 64.7 | 67.6 | 27.1 | 96.1 |
| hopper-medium-expert | **112.8**±0.2 | 111.5±1.0 | 95.4 | 23.7 | 110.9 | 89.6 | 111.4 | 90.6 |
| walker2d-medium-expert | **104.8**±1.0 | 99.7±2.9 | 14.8 | 44.6 | 57.5 | 12.0 | 68.1 | 103.6 |
| halfcheetah-expert | 104.7±0.9 | 100.8±3.7 | 104.0 | - | 89.9 | 105.2 | 82.4 | **106.8** |
| hopper-expert | **112.8**±0.2 | 112.0±0.0 | 109.1 | - | 107.0 | 111.5 | 111.2 | 112.3 |
| walker2d-expert | **106.8**±3.7 | 103.6±2.0 | 88.4 | - | 102.3 | 56.0 | 103.8 | 79.9 |
| Total Score | 909.3 | 860.7 | 618.6 | - | 784.6 | 595.3 | 764.3 | **974.6** |

Table 13: Normalized average score comparison of CABI+BCQ against different baselines on the Adroit tasks, where score 0 represents the performance of a random policy and 100 corresponds to an expert policy performance. The highest mean scores are in **bold**.

| Task Name | CABI+BCQ | BCQ | UWAC | BEAR | BC | AWR | CQL | SAC-off | MOPO | COMBO |
|---|---|---|---|---|---|---|---|---|---|---|
| pen-cloned | 54.7±2.0 | 44.0 | 33.1±10.1 | 26.5 | **56.9** | 28.0 | 39.2 | 23.5 | -2.1±4.0 | -2.4±1.1 |
| pen-human | **75.1**±1.5 | 68.9 | 21.7±2.2 | -1.0 | 34.4 | 12.3 | 37.5 | 6.3 | 9.7±4.1 | 27.7±10.7 |
| pen-expert | **127.6**±2.0 | 114.9 | 111.9±1.2 | 105.9 | 85.1 | 111.0 | 107.0 | 6.1 | -0.6±1.9 | 11.5±2.3 |
| door-cloned | **0.5**±0.2 | 0.0 | 0.0±0.0 | -0.1 | -0.1 | 0.0 | 0.4 | 0.0 | -0.1±0.1 | 0.0±0.0 |
| door-human | 1.7±0.1 | -0.0 | 2.1±1.6 | -0.3 | 0.5 | 0.4 | **9.9** | 3.9 | -0.2±0.1 | -0.3±0.2 |
| door-expert | **105.3**±0.5 | 99.0 | 104.1±1.8 | 103.4 | 34.9 | 102.9 | 101.5 | 7.5 | -0.2±0.1 | 4.9±1.2 |
| relocate-cloned | -0.2±0.0 | -0.3 | -0.3±0.0 | -0.3 | **-0.1** | -0.2 | **-0.1** | -0.2 | -0.3±0.1 | **-0.1**±0.1 |
| relocate-human | 0.1±0.1 | **0.5** | **0.5**±0.6 | -0.3 | 0.0 | 0.0 | 0.2 | 0.0 | -0.3±0.1 | -0.3±0.1 |
| relocate-expert | **105.9**±1.0 | 41.6 | 105.6±1.6 | 98.6 | 101.3 | 91.5 | 95.0 | -0.3 | -0.2±0.0 | 17.2±3.1 |
| hammer-cloned | **4.3** ±1.6 | 0.4 | 0.4±0.0 | 0.3 | 0.8 | 0.4 | 2.1 | 0.2 | 0.2±0.0 | 0.4±0.2 |
| hammer-human | 3.1±2.2 | 0.5 | 1.1±0.6 | 0.3 | 1.5 | 1.2 | **4.4** | 0.5 | 0.2±0.0 | 0.2±0.0 |
| hammer-expert | **128.9**±0.9 | 107.2 | 110.6±20.7 | 127.3 | 125.6 | 39.0 | 86.7 | 25.2 | 0.3±0.0 | 0.3±0.1 |
| Total Score | **607.0** | 476.7 | 490.8 | 460.3 | 440.8 | 386.5 | 483.8 | 72.7 | 6.4 | 59.1 |

Table 14: Normalized average score of CABI+TD3_BC vs. prior state-of-the-art methods on the D4RL MuJoCo dataset, where score 0 corresponds to a random policy performance and 100 corresponds to an expert policy performance.

| Task Name | CABI+TD3_BC | TD3_BC | UWAC | MOPO | BEAR | BCQ | BC | CQL | FisherBRC |
|---|---|---|---|---|---|---|---|---|---|
| halfcheetah-random | 15.1±0.4 | 10.2±1.3 | 2.3±0.0 | **35.4**±2.5 | 25.1 | 2.2 | 2.0±0.1 | 21.7±0.9 | 32.2±2.2 |
| hopper-random | **11.9**±0.1 | 11.0±0.1 | 9.8±0.1 | 11.7±0.4 | 11.4 | 10.6 | 9.5±0.1 | 10.7±0.1 | 11.4±0.2 |
| walker2d-random | 6.4±1.5 | 1.4±0.6 | 3.8±1.3 | **13.6**±2.6 | 7.3 | 4.9 | 1.2±0.2 | 2.7±1.2 | 0.6±0.6 |
| halfcheetah-med-replay | 44.4±0.2 | 43.3±0.5 | 38.9±0.3 | **53.1**±2.0 | 38.6 | 38.2 | 34.7±1.8 | 41.9±1.1 | 43.3±0.9 |
| hopper-med-replay | 31.3±0.7 | 31.4±3.0 | 18.0±5.6 | **67.5**±24.7 | 33.7 | 33.1 | 19.7±5.9 | 28.6±0.9 | 35.6±2.5 |
| walker2d-med-replay | 29.4±1.3 | 25.2±5.1 | 8.4±1.2 | 39.0±9.6 | 19.2 | 15.0 | 8.3±1.5 | 15.8±2.6 | **42.6**±7.0 |
| halfcheetah-medium | **45.1**±0.1 | 42.8±0.3 | 37.4±0.2 | 42.3±1.6 | 41.7 | 40.7 | 36.6±0.6 | 37.2±0.3 | 41.3±0.5 |
| hopper-medium | **100.4**±0.3 | 99.5±1.0 | 30.3±0.3 | 28.0±12.4 | 52.1 | 54.5 | 30.0±0.5 | 44.2±10.8 | 99.4±0.4 |
| walker2d-medium | **82.0**±0.4 | 79.7±1.8 | 17.4±8.5 | 17.8±19.3 | 59.1 | 53.1 | 11.4±6.3 | 57.5±8.3 | 79.5±1.0 |
| halfcheetah-med-expert | **105.0**±0.2 | 97.9±4.4 | 40.6±6.6 | 63.3±38.0 | 53.4 | 64.7 | 67.6±13.2 | 27.1±3.9 | 96.1±9.5 |
| hopper-med-expert | **112.7**±0.0 | 112.2±0.2 | 95.4±23.5 | 23.7±6.0 | 96.3 | 110.9 | 89.6±27.6 | 111.4±1.2 | 90.6±43.3 |
| walker2d-med-expert | **108.4**±1.3 | 101.1±9.3 | 14.8±1.4 | 44.6±12.9 | 40.1 | 57.5 | 12.0±5.8 | 68.1±13.1 | 103.6±4.6 |
| halfcheetah-expert | **107.6**±0.9 | 105.7±1.9 | 104.0±2.0 | - | **108.2** | - | 105.2±1.7 | 82.4±7.4 | 106.8±3.0 |
| hopper-expert | **112.4**±0.1 | 112.2±0.2 | 109.1±3.9 | - | 110.3 | - | 111.5±1.3 | 111.2±2.1 | 112.3±0.2 |
| walker2d-expert | **108.6**±1.5 | 105.7±2.7 | 88.4±3.7 | - | 106.1 | - | 56.0±24.9 | 103.8±7.6 | 79.9±32.4 |
| Total Score | **1020.7** | 979.3 | 618.6 | - | 802.6 | - | 595.3 | 764.3 | 974.6 |