A Policy-Decoupled Method for High-Quality Data Augmentation in Offline Reinforcement Learning

Shixi Lian [*1] **Yi Ma** [*1] **Jinyi Liu** [1] **Jianye Hao** [1] **Yan Zheng** [1] **Zhaopeng Meng** [1]

Abstract

Offline reinforcement learning (ORL) has gained attention as a means of training reinforcement learning models using pre-collected static data. To address the issue of limited data and improve downstream ORL performance, recent work has attempted to expand the dataset's coverage through data augmentation. However, most of these methods are tied to a specific policy (policy-dependent), where the generated data can only guarantee to support the current downstream ORL policy, limiting its usage scope on other downstream policies. Moreover, the quality of synthetic data is often not well-controlled, which limits the potential for further improving the downstream policy. To tackle these issues, we propose **HI**gh-quality **PO**licy-**DE**coupled (HIPODE), a novel data augmentation method for ORL. On the one hand, HIPODE generates high-quality synthetic data by selecting states near the dataset distribution with potentially high value among candidate states using the negative sampling technique. On the other hand, HIPODE is policy-decoupled, thus can be used as a common plug-in method for any downstream ORL process. We conduct experiments on the widely studied TD3BC and CQL algorithms, and the results show that HIPODE outperforms the state-of-the-art policy-decoupled data augmentation method and most prevalent model-based ORL methods on D4RL benchmarks.

[*]Equal contribution [1]College of Intelligence and Computing, Tianjin University, Tianjin, China. Correspondence to: Jianye Hao <jianye.hao@tju.edu.cn>.

Workshop on New Frontiers in Learning, Control, and Dynamical Systems at the International Conference on Machine Learning (ICML), Honolulu, Hawaii, USA, 2023.

1. Introduction

Offline Reinforcement Learning (ORL) (Lange et al., 2012) has garnered significant attention in recent years as it aims to learn from a dataset of previously collected experiences without further interaction with the environment. ORL is believed to be promising (Fu et al., 2020; Fujimoto et al., 2019), since online learning is not feasible due to the high cost of failures, and collecting new data is often expensive or even dangerous (Prudencio et al., 2023).

In the offline setting, prior off-policy RL methods are known to fail on fixed offline datasets (Haarnoja et al., 2018; Fujimoto et al., 2018), even on expert demonstrations (Fujimoto et al., 2019). The main reason of this could be the limited coverage of offline data. This can cause the policy visiting states that are out of the distribution (OOD) of the dataset, and suffer from the extrapolation error on these states (Fujimoto et al., 2019; Kumar et al., 2019). To alleviate extrapolation errors, most ORL researches attempt to avoid out-of-distribution states or actions, focusing on policy constraint (Fujimoto et al., 2019; Wu et al., 2019; Liu et al., 2020; Fujimoto & Gu, 2021), support constraint (Kostrikov et al., 2022; Kumar et al., 2019), value regularization (Kumar et al.; Ma et al., 2021b;a; Kumar et al., 2021; Kostrikov et al., 2021; An et al., 2021), and others. However, these approaches face the problem of the loss of generalization capability (Lyu et al., 2022).

Different from mitigating the extrapolation error, data augmentation has been applied in ORL recently to expand the coverage of the dataset. The simplest approach is to add noise to the original dataset to obtain augmented data (Sinha et al., 2022; Weissenbacher et al., 2022), which could result in inaccurate dynamics transition that may not match the real environment. In contrast, dynamics models used in model-based RL can augment the dataset by rolling out synthetic samples. Inspired by this, existing works use the forward or backward dynamics models (Yu et al., 2021; 2020; Kidambi et al., 2020; Lyu et al., 2022; Wang et al., 2021; 2022; Lu et al., 2022; Guo et al., 2022; Rigter et al., 2022; Fu et al.) to generate synthetic data and incorporate them into the policy training process. However, most of these methods are policy-dependent since they have to explicitly deal with unreliable data derived from inaccurate models to adapt to

the downstream policy, thus limiting their data's application to augment other ORL algorithms. Among them, (Wang et al., 2021; Lyu et al., 2022) achieve policy-decoupled data augmentation. However, these methods lack explicit constraints to ensure the quality of the generated data, making the underlying mechanism by which they work unclear and limiting the potential of further improvement to the downstream policy.

To overcome the above-mentioned issues, we investigate the data augmentation method that is not dependent on the downstream ORL policy, which also ensures the quality of generated data. We first empirically analyze that high-quality data is beneficial for enhancing ORL performance. Then, we propose the **HI**gh-quality **PO**licy-**DE**coupled (HIPODE) data augmentation approach.

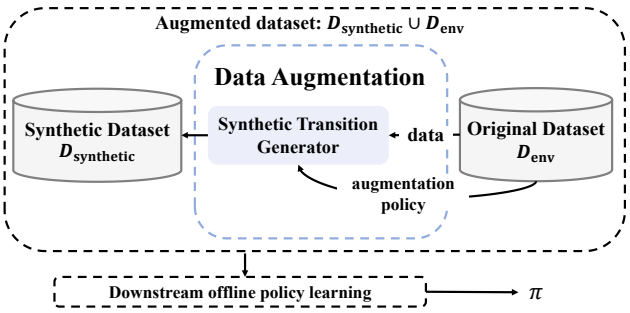

Figure 1. Outline of our data augmentation for ORL.

We present the outline of our policy-decoupled data augmentation process for ORL in Fig.1, which involves using specific augmentation policy to generate synthetic datasets based on the original dataset. These synthetic datasets are then used to expand the training data for any downstream ORL algorithm. Throughout this process, our key insight is to generate high-quality synthetic augmented data while ensuring authenticity (i.e. the proximity level between the synthetic data and the real data) as much as possible in such a policy-decoupled way. To this end, HIPODE uses a state transition model to generate next states and select the ones near the dataset distribution with potential highest value identified using a value network trained with negative sampling technique (Luo et al., 2019). Then the action and reward in the synthetic transition is replenished via inverse dynamics model and generative reward model, respectively. Note that data generation process of HIPODE does not need any information from the downstream policy, thus it's decoupled from the downstream offline policy learning process and can be used as a common plug-in method, similar to the role of data augmentation in Computer Vision (CV). Experimental results on D4RL benchmarks (Fu et al., 2020) demonstrate that HIPODE significantly improves different baselines' performance and outperforms existing policy-decoupled data augmentation methods for ORL. To show the benefit of policy-decoupled approach, we also conduct experiments to show that synthetic data generated from HIPODE can benefit several downstream offline policy learning processes while existing policy-dependent augmentation methods may fail.

To summarize, the contributions of this paper are:

- We investigate the impact of different types of augmented data on downstream ORL algorithms. Our findings indicate that high-quality data, as opposed to noisy data with high diversity, benefits downstream offline policy learning performance more.
- We propose a novel policy-decoupled data augmentation method HIPODE for ORL. HIPODE serves as a common plugin that can augment high-quality synthetic data for any ORL algorithm, and is decoupled with downstream offline policy learning process.
- We evaluate HIPODE on D4RL benchmarks and it significantly improves several widely used model-free ORL baselines. Furthermore, HIPODE outperforms state-of-the-art (SOTA) policy-decoupled data augmentation approaches for ORL.

2. Related Work

Data augmentation in ORL. To address the challenge of limited data in ORL, various methods have been proposed to generate more sufficient data. Most of these approaches are policy-dependent, meaning they generate data based on the current policy and use it to refine the training of the same policy. These policy-dependent data augmentation methods can be divided into two categories. The first category seeks to generate pessimistic synthetic data that would be pessimistic enough if it is OOD, thus expand the dataset's coverage while mitigating the extrapolation error caused by such OOD data. The literature (Yu et al., 2020; Kidambi et al., 2020) rely on the disagreement of dynamics ensembles or Q ensembles to construct a pessimistic MDP, and (Yu et al., 2021; Rigter et al., 2022; Guo et al., 2022) achieve the underestimation of synthetic data by unrolling the current policy in the model. The second category does not explicitly pursue the underestimation of synthetic data. Among them, (Fu et al.) generates and selected synthetic data with low model disagreement, and BooT (Wang et al., 2022) augments TT (Janner et al., 2021) with the synthetic data generated by itself. Besides, S4RL (Sinha et al., 2022) and KFC (Weissenbacher et al., 2022) add noise in a local area of states to smooth the critic.

An obvious drawback of these aforementioned approaches, which are all policy-dependent, is that the generated data is closely related on the policy itself, causing that applying the generated data directly to the learning process of other poli-

cies is not guaranteed to perform well. To overcome this limitation, recent studies have explored policy-decoupled data augmentation techniques, which is the focus of this paper. Bi-directional rolling proposed in (Lyu et al., 2022) induce the double-check mechanism into offline data augmentation ensure that the generated data is within the distribution and avoids inauthentic samples. In (Wang et al., 2021), a reverse dynamics model is proposed for ORL, and performs better than CQL (Kumar et al.) on maze environments. However, these two methods only consider the reliability of the synthetic data and neglect the quality of generated data, which may limit the performance. In contrast, HIPODE takes into account both the reliability and quality of the data.

Model-free ORL. Disregarding data augmentation, the model-free ORL algorithm investigates how to constrain the policy to approach the behavioral policy or support in static offline datasets. Existing methods implement this by policy constraint (Fujimoto et al., 2019; Wu et al., 2019; Liu et al., 2020; Fujimoto & Gu, 2021), support constraint (Kostrikov et al., 2022; Kumar et al., 2019), value regularization (Kumar et al.; Ma et al., 2021b;a; Kumar et al., 2021; Kostrikov et al., 2021; An et al., 2021), and others. Among them, we choose widely-used TD3BC (Fujimoto & Gu, 2021) and CQL (Kumar et al.) to be the downstream policy learning algorithm to evaluate different data augmentation methods.

3. What Kind of Augmented Data Can Help Improve ORL?

As mentioned before, data augmentation methods in ORL often neglect the data quality. However, high-quality data are regarded as beneficial for learning (Fu et al., 2020). Accordingly, we pose the question of whether the generation of high-quality data is also beneficial for ORL policies when compared to high-diversity data. We primarily investigate this question in this section.

Table 1. Results of downstream ORL algorithm using different augmented data. Original denotes using the original dataset; Original + Diversity σ and Original + Quality denote using high-diversity or high-quality augmented data; -r and -m-r denotes -random-v0 and -medium-replay-v0.

Task Name	Augmenting Type	TD3BC	CQL
halfcheetah-r	Original	12.8	17.0
	Original + Diversity 0.01	12.1	2.0
	Original + Diversity 0.1	12.0	16.1
	Original + Diversity 1.0	9.2	3.0
	Original + Quality	25.8	23.8
halfcheetah-m-r	Original	43.3	42.5
	Original + Diversity 0.01	44.6	38.4
	Original + Diversity 0.1	44.3	1.8
	Original + Diversity 1.0	41.8	25.6
	Original + Quality	46.8	52.6

To fairly investigate the effect of high-diversity data and high-quality data on the downstream ORL algorithm, we generate two types of data from real environments, instead of generating from other data augmentation techniques, to prevent potential bias due to inauthentic data impacting our findings. Moreover, to more closely match the offline setting, we limit the data generated in this section to only those that are not far away from the original dataset. Concretely, we choose the following two types of augmentation policies: (1) **Policy of high diversity**, where random noise with different scales is added to the behavioral policy. Formally, $\pi_{\text{noise}} := \mathcal{N}(a, \sigma \mathrm{I}), \text{s.t.}, a \sim \pi_\beta$, where π_β denotes the behavioural policy, $\mathcal{N}$ denotes the Gaussian distribution and I denotes a identity matrix. The dataset after augmentation by this policy exhibits higher **diversity** compared to the original dataset. We refer to this type of method as 'Diversity σ', where σ belongs to 0.01, 0.1, 1.0. (2) **Policy of high-quality**, a well-trained ORL policy, to ensure the action **quality**, i.e., return, derived from the ORL policy is similar to or higher than that of the actions in the dataset overall. Meanwhile, the generated data is also ensured to be close to the dataset. We refer this as 'Quality'.

Figure 2. Action distributions of the original dataset, noise-policy-augmented data and quality-policy-augmented data. Brighter color indicates higher reward in a single time-step.

The augmented data and the original data are together used to train the downstream ORL algorithms. Normalized score reported in Table 1 shows that using ORL policy to augment data can always benefit down stream offline policy learning performance while using random noise policy may not. We further visualize the distribution of the original dataset, the noise-policy-augmented data, and the quality-policy-augmented data through t-Distributed Stochastic Neighbor Embedding (t-SNE) (Hinton & Roweis, 2002) in Fig.2. As we can see from it, compared with the distribution of the original data, the distribution of the noise-policy-augmented data is similar to the original dataset's while the distribution of the quality-policy-augmented data is relatively concentrated in several clusters. In addition, the quality-policy-augmented data indeed has higher rewards in a single time-step, as the color of the most triangle points are brighter.

Based on these observations, we present the following takeaway:

> **Takeaway**: in the case that the augmented data is completely realistic, data with higher quality may be more beneficial than that of more diversity in improving downstream ORL algorithm.

4. Method

According to the takeaway above, we propose HIPODE to generate augmented data that maximizes its quality, i.e., return, while maintaining as much authenticity as possible in a policy-decoupled way. We illustrate HIPODE in Fig.3. Specifically, given any state s, we first generate several candidate next states $\tilde{S}'_{\text{cand}} = \{\tilde{s}'_1, ..., \tilde{s}'_n\}$ (Step 1 in Fig.3). Then we select the one with the highest value as $\tilde{s}'$ (Step 2). Finally, given s and $\tilde{s}'$, the action $\tilde{a}$ and the reward $\tilde{r}$ are produced using generative models (Step 3), thus generating a transition $\{s, \tilde{a}, \tilde{r}, \tilde{s}'\}$. In the following, we introduce Step 1 and 2 in Section 4.1 and Step 3 in Section 4.2.

4.1. Next State Generation with Negative Sampling

Given a state s, we generate the next state through a state transition model, and filter the high-quality data for our purpose. In the following, we describe these two steps in detail.

The forward state transition model. We first train a state transition model $\tilde{p}_\psi(s'|s)$ to generate candidate next states. To guarantee the authenticity of generated next state, we model the state transition within the dataset with a conditional variational auto-encoder (CVAE) following (Zhang et al., 2022), to ensure the generated next states are near the distribution of the dataset. Specifically, CVAE consists of an encoder and a decoder: the encoder takes the current state and the next state as input and manages to output an latent variable z under the Gaussian distribution; the decoder takes z and the current state as input and manages to map the latent variable z to the desired space. We denote the encoder as $E_\psi(s, s')$ and the decoder as $D_\psi(s, z)$. The state transition model is then trained by maximizing its variational lower bound, which is equivalent to minimizing the following loss:

$$\begin{aligned}\mathcal{L}(\psi) =& \mathbb{E}_{(s,s')\sim\mathcal{D}_{\text{env}}, z\sim E_\psi(s,s')}[(s' - D_\psi(s,z))^2 \\ &+ D_{\text{KL}}(E_\psi(s,s')\|\mathcal{N}(0,\mathbf{I}))].\end{aligned} \tag{1}$$

where I represents an identity matrix and $\mathcal{D}_{\text{env}}$ represents the original dataset. The first term of RHS of Eq.(1) represents the reconstruction loss where the approximated next state is decoded from z, given the current state. The second term of RHS represents the KL distance between the distribution of z and the Gaussian distribution so that a sampled z from a Gaussian distribution can be decoded to the desired state space when generating. Thus, given a state s, n candidate next states $\tilde{S}'_{\text{cand}} = \{\tilde{s}'_1, ..., \tilde{s}'_n\}$, s.t., $\tilde{s}'_i \sim D_\psi(\tilde{s}'_i|s, z)$ are sampled.

Value Approximation with Negative Sampling. To filter out the generated next states and form synthetic transitions, a value approximator is trained using SARSA-style updating to predict the value of different states. Since the generated next states may not be present in the dataset, the negative sampling technique (Luo et al., 2020) is employed to avoid overestimation of states outside the dataset. Specifically, for states within the dataset, standard TD-learning is performed as demonstrated in Eq.2:

$$\mathcal{L}^{\text{td}}_\theta(s) = \mathbb{E}_{(s,r,s')\sim\mathcal{D}_{\text{env}}}[r + \gamma V_{\bar{\theta}}(s') - V_\theta(s)]^2, \tag{2}$$

where $V_{\bar{\theta}}$ is the target value function and γ is the discount factor. Furthermore, to conservatively estimate the value of states outside the dataset distribution, we sample states around the dataset states by adding Gaussian noise and evaluate the L2 distances between the sampled noisy states and the original states. The greater the distance between the sampled state and the original state, the more severe the penalties imposed on the sampled state, as shown in Eq.(3):

$$\begin{aligned}\mathcal{L}^{\text{ns}}_\theta(s) =& \mathbb{E}_{s\sim\mathcal{N}(s_{\text{d}},\sigma\text{I}),(s_{\text{d}},r,s')\sim\mathcal{D}_{\text{env}}}[r + \gamma V_{\bar{\theta}}(s') \\ &- \alpha\|s - s_{\text{d}}\| - V_\theta(s)]^2,\end{aligned} \tag{3}$$

where s_{d} denotes the state sampled from the original dataset and α denotes the penalty weight. Thus, the optimization objective of the value approximator to minimize is represented by Eq.(4):

$$\mathcal{L}(\theta) = \mathcal{L}(\theta)^{\text{td}} + \mathcal{L}(\theta)^{\text{ns}}. \tag{4}$$

After training, all the candidate next states S'_{cand} are input into the value approximator to obtain their values. Then the candidate next state with the highest value estimation is selected, formally $\tilde{s}' = \text{argmax}_{\tilde{s}'_{\text{cand}}} V(\tilde{s}'_{\text{cand}}), \tilde{s}'_{\text{cand}} \in \tilde{S}'_{\text{cand}}$. Intuitively, a state can be selected in two cases:

- States within the dataset. This is because other candidate states that not in the dataset are severely underestimated. In this case, the selected state can be considered reliable.
- States with high true value near the dataset distribution. Since its estimated value is significantly penalized during training, there is a high probability that a selected state close to the distribution has a high true value.

Therefore, by filtering the candidate next states generated by the state transition model using the value approximator, we can obtain the augmented next state with similar or higher quality than that in the datasets while maintaining as much authenticity as possible.

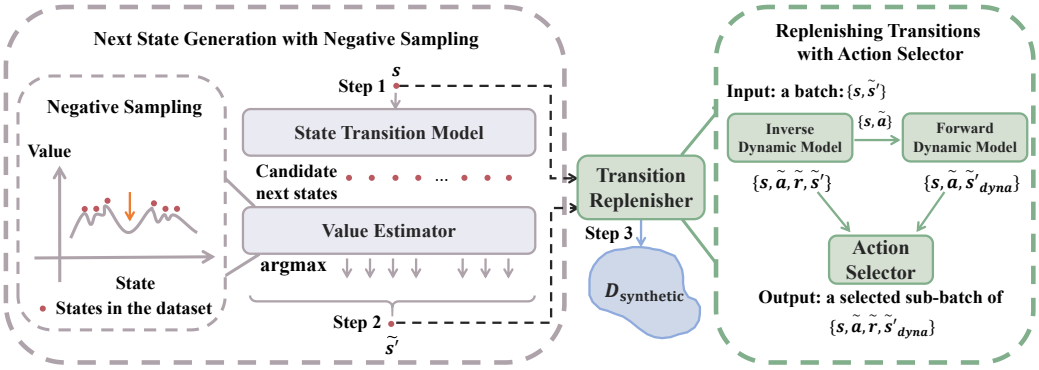

Figure 3. Illustration of HIPODE.

4.2. Replenishing Transitions with Action Selector

Based on the selected high-quality next state, in this section, we aim to generate an authentic action that can lead the current state to the generated next state. Specifically, an inverse model $M_{\text{inv}} = \tilde{p}_\epsilon(a|s, s')$ is trained to generate actions conditioned on s and the selected $\tilde{s}'$. Similar to the state transition model, we also use a CVAE for generating actions. We denote the encoder as $E_\epsilon(a, s, s')$ and the decoder as $D_\epsilon(s, s', z)$. The inverse model is then trained by maximizing its variational lower bound, which is equivalent to minimizing the following loss shown as Eq.(5):

$$\begin{aligned}\mathcal{L}(\epsilon) =& \mathbb{E}_{(a,s,s')\sim\mathcal{D}_{\text{env}}, z\sim E_\epsilon(a,s,s')}[(a - D_\epsilon(s, s', z))^2 \\ &+ D_{\text{KL}}(E_\epsilon(a, s, s')\|\mathcal{N}(0, \text{I}))].\end{aligned} \tag{5}$$

Besides, rewards are generated the same way as actions, using another model with encoder $E_\zeta(r, s, s')$ and decoder $D_\zeta(s, s', z)$.

Although the generated state have high quality and authenticity as described in Section 4.1, the action generated by the inverse dynamics model may be inauthentic, i.e. the generated action can not lead to the selected next state. Therefore, a filtering mechanism is imposed on actions for their reliability. We further draw on a forward dynamics model $M_{\text{for\_dyna}} = \tilde{p}_w(\tilde{s}'_{\text{dyna}}|s, a)$ representing the probability of the next state given the current state and action. The dynamics model is optimized by maximizing the log-likelihood of the static dataset, formally shown in Eq.(6):

$$\mathcal{L}(w) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}_{\text{env}}}[-\log \tilde{p}_w(s'|s, a)]. \tag{6}$$

Combining the forward dynamics model $M_{\text{for\_dyna}}$ and the inverse dynamics model M_{inv}, an action is assumed to be reliable to lead to the selected state $\tilde{s}'$ when the distance between the selected state $\tilde{s}'$ and the forward-predicted state $\tilde{s}'_{\text{dyna}} = M_{\text{for\_dyna}}(s, \tilde{a})$ is small enough. In practice, instead of setting a threshold for measuring the distance between $\tilde{s}'$ and $\tilde{s}'_{\text{dyna}}$, we pick up in a batch λ-portion of the generated data with the lowest $\|\tilde{s}'_{\text{dyna}} - \tilde{s}'\|$ values and consider them the most reliable subset of the batch. Then this subset of data is used as the final augmented data.

HIPODE is summarized in Algorithm 1, where $N(\mathcal{D}_{\text{env}})$ is the amount of data in the original dataset. Line 5-7 refers to selecting the next state with negative sampling, described in Section 4.1 for generating synthetic next states that are close to the dataset distribution and have the potential for the highest value. Line 8-10 and 12 refer to replenishing transitions with an action selector, as described in Section 4.2 for generating synthetic actions and rewards. Line 13-14 refers to merging the synthetic data with the original dataset for downstream offline policy training to obtain the offline policy π.

5. Experiments

In this section, we evaluate HIPODE based on two representative and widely-used offline policy learning algorithm: TD3BC (Fujimoto & Gu, 2021) and CQL (Kumar et al.).We aim to answer these questions:

Q1: Can our proposed algorithm HIPODE improve existing ORL algorithms and exhibit consistent superiority in comparison to other data augmentation technique?

Q2: Is augmenting synthetic data or high-quality synthetic data critical for ORL policy?

Q3: Does our policy-decoupled data augmentation algorithm HIPODE outperform the conventional policy-dependent data augmentation methods?

Q4: In HIPODE, what roles do the negative sampling and transition selector components play?

In the following, we answer *Q1* in Section 5.1, showing the effectiveness and superiority of HIPODE by combining it with offline RL algorithms on MuJoCo (Todorov et al., 2012) tasks. Then, we present an ablation study in details in Section 5.2 to answer *Q2*. We answer *Q3* in Section 5.3 by comparing HIPODE with policy-dependent data aug-

Algorithm 1 HIPODE

Input: Offline dataset $\mathcal{D}_{\text{env}} = \{(s, a, r, s')\}$, penalty wieght α, synthetic rate η, action selecting rate λ, number of candidate next states n
Output: Policy π
Train state transition model $D_\psi(s, z)$ by minimizing Eq.(1), value estimator $V_\theta(s)$ by minimizing Eq.(4), dynamics model $\hat{p}_w(s'|s, a)$ by minimizing Eq.(6) and inverse action model $D_\epsilon(s, s', z)$ as well as inverse reward model $D_\zeta(s, s', z)$ by minimizing Eq.(5) and a similar loss for reward generation respectively
repeat
  Sample a batch of s from $\mathcal{D}_{\text{env}}$
  Sample $\tilde{S}'_{\text{cand}}$ containing n $\tilde{s}'_i$ from $\tilde{s}'_i \sim D_\psi(s, z), z \sim \mathcal{N}(0, \mathrm{I})$
  $\tilde{s}' = \text{argmax}_{\tilde{s}'_{\text{cand}}} V(\tilde{s}'_{\text{cand}}), \tilde{s}'_{\text{cand}} \in \tilde{S}'_{\text{cand}}$
  Sample actions $\tilde{a} \sim D_\epsilon(s, s', z), z \sim \mathcal{N}(0, \mathrm{I})$ and sample rewards $\tilde{r} \sim D_\zeta(s, s', z), z \sim \mathcal{N}(0, \mathrm{I})$
  Sample $\tilde{s}'_{\text{dyna}}$ from $\tilde{s}'_{\text{dyna}} \sim \tilde{p}_w(\tilde{s}'_{\text{dyna}}|s, a)$
until reaching maximum generating amount, which is $\eta N(\mathcal{D}_{\text{env}})$
Select top λ-portion authentic actions to construct $D_{\text{synthetic}}$ with least $\|\tilde{s}'_{\text{dyna}} - \tilde{s}'\|$
Merge the synthetic dataset and the original dataset $D = D \cup D_{\text{synthetic}}$
Use any model-free offline policy learning algorithm to obtain π
return π

mentation methods. Finally, we answer *Q4* in Appendix B.

5.1. Performance on MuJoCo

Evaluation settings. We demonstrate the benefits of HIPODE on D4RL MuJoCo-v0 tasks (Fu et al., 2020), comparing with several baselines that augment data for policy training:

- **CABI** (Lyu et al., 2022), the SOTA policy-decoupled data augmentation algorithm for ORL. we reproduce CABI following their paper (Lyu et al., 2022) based on CORL (Tarasov et al., 2022).
- **COMBO** (Yu et al., 2021) and **MOPO** (Yu et al., 2020), two widely studied model-based ORL methods, which are policy-dependent. we re-run COMBO (Yu et al., 2021) using code of (Sun, 2023), and take the reported results of MOPO directly from the original paper (Yu et al., 2020). Additionally, We re-run MOPO on expert datasets using OfflineRL-Kit (Sun, 2023).

To ensure the fairness of the comparison, we implement HIPODE and the downstream ORL algorithms (TD3BC and CQL) based on CORL (Tarasov et al., 2022). We present detailed discussion about benchmark tasks and implementation in Appendix A.

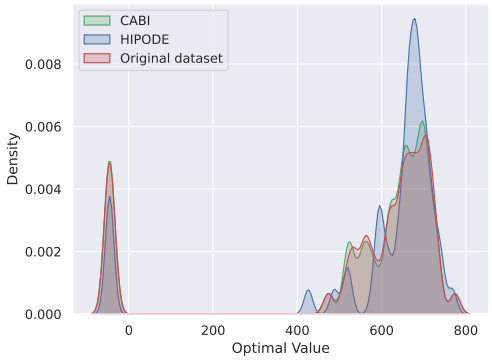

Figure 4. Density of different synthetic data and the original dataset.

Main results. Table 2 shows the results on 15 MuJoCo tasks comparing between the above-mentioned algorithms. HIPODE achieves remarkable improvements over baselines (TD3BC, CQL), and also significant gain outperforming SOTA data augmentation method (CABI), which confirm the effectiveness of HIPODE in handling these offline tasks. In this experiment, in order to make a fair comparison, we reproduce the CABI algorithm based on the same downstream policy learning process and the same hyper-parameters to demonstrate HIPODE's superiority. Under the premise of controlling downstream implementation and consistent hyper-parameters, the advantage of HIPODE performance all comes from the data augmentation process. On the other hand, compared to the reported results in (Lyu et al., 2022), our advantage remains consistent, as detailed in Appendix B. HIPODE is also effective in the Adroit tasks. Details are presented in Appendix B.

Also, HIPODE's predominance over the model-based policy-dependent baseline algorithms (COMBO, MOPO) demonstrates its strength. Noting that COMBO and MOPO need to access the true terminal function to ensure algorithm performance, whereas HIPODE achieves better performance without the need for such a function, by uniformly setting terminal flag of HIPODE's synthetic data to False.

To show HIPODE indeed generates high-quality transitions, we visualize the distribution of estimated discounted cumulative rewards of trajectories in the original dataset and synthetic data generated by HIPODE and CABI. Specifically, we train online SAC to converge to obtain the optimal value function V^* as authoritative value function. For each state-action pair (s, a), we use $r + V^*(s')$ to represent the discounted cumulative reward, where $r, s' \sim p(r, s'|s, a)$ are the true reward and next state in the environment respectively. Fig.4 illustrates the density of synthetic data generated by CABI and HIPODE on halfcheetah-medium-replay-v0, as well as the original dataset, where X axis represents

Table 2. Normalized average score and standard deviation over 3 random seeds of HIPODE based on CQL and TD3BC and of baseline performance. In the table, -m-e, -m-r, -m, -r, -e denote -medium-expert, -medium-replay, -medium, -random, -expert respectively.

Task Name	CQL + HIPODE	CQL + CABI	CQL	TD3BC + HIPODE	TD3BC + CABI	TD3BC	COMBO	MOPO
halfcheetah-m-e	99.5 ± 4.5	101.0	94.8	102.6 ± 2.2	94.6	98.0	38.7	63.3
hopper-m-e	112.1 ± 0.1	112.0	111.9	112.4 ± 0.3	102.8	112.0	75.1	23.7
walker2d-m-e	94.0 ± 4.0	92.3	70.3	105.3 ± 4.0	99.6	105.4	2.3	44.6
halfcheetah-m-r	46.0 ± 0.1	42.8	42.5	44.0 ± 0.7	43.4	43.3	46.9	53.1
hopper-m-r	34.7 ± 2.9	29.7	28.2	36.2 ± 0.8	32.7	32.7	19.7	67.5
walker2d-m-r	21.7 ± 6.5	12.7	5.1	36.4 ± 11.1	37.7	19.6	19.5	39.0
halfcheetah-m	39.3 ± 0.2	36.7	39.2	43.7 ± 0.5	43.0	43.7	27.4	42.3
hopper-m	30.4 ± 0.9	30.5	30.3	99.9 ± 0.3	99.7	99.9	71.6	28.0
walker2d-m	75.7 ± 5.1	46.8	66.9	80.1 ± 0.7	80.3	79.7	71.8	17.8
halfcheetah-r	23.1 ± 1.5	2.8	17.0	15.5 ± 0.9	12.5	12.8	5.5	35.4
hopper-r	10.4 ± 0.1	10.2	10.4	10.9 ± 0.2	10.9	10.9	7.5	11.7
walker2d-r	10.5 ± 0.6	-0.1	1.7	6.6 ± 1.0	3.0	0.37	1.6	13.6
Total	**592.9**	517.4	518.3	**693.6**	660.2	658.3	387.5	440.0
halfcheetah-e	109.1 ± 0.14	109.1	109.8	106.5 ± 0.5	106.3	107.0	44.2	102.1
hopper-e	112.2 ± 0.2	112.3	112.0	112.3 ± 0.5	110.8	107.3	112.3	0.7
walker2d-e	109.3 ± 2.0	98.7	104.7	106.6 ± 5.2	102.5	98.7	37.3	2.1
Total	**929.0**	837.5	844.9	**1019.0**	979.8	971.5	581.3	544.9

V^* and Y represents the density of synthetic transitions on V^*. Form the figure, the green shadow almost coincide with the red one, showing that CABI's data distribution almost coincide with the original dataset, while HIPODE indeed generates more high-quality data. In conjunction with the results in Table 2, the advantage of HIPIDE performance comes from more high-quality data in augmentation process, which sequentially demonstrates that high-quality data is more suitable rather than high diversity data as augmented data for ORL.

5.2. Ablation Study

In this section, we aim to further investigate how the generated data improves downstream offline policy. We conduct ablation experiments from three perspectives: not generating synthetic data (Repeat), generating vanilla synthetic data (NoV), and generating high-quality synthetic data (HIPODE). and the results are shown in Table 3.

Specifically, the difference of Repeat and HIPODE is that the synthetic data is replaced by 10% high-quality data from the original dataset in Repeat. The difference between NoV and HIPODE is that the value maximization mechanism is removed in NoV, i.e., the quality of generated data is not controlled. Generating high-quality synthetic data is exactly HIPODE.

As the results in Table 3, Repeat+TD3BC, i.e., repeating high-quality data in the dataset, brings little performance gain and even hurts performance on walker2d-medium-expert and walker2d-expert. Thus it's not effective for improving ORL performance. Besides, NoV+TD3BC achieves an improvement over TD3BC, indicating the importance of generating new synthetic data for data augmentation. However, the performance of NoV+TD3BC is worse than HIPODE, indicating the importance of generating high-quality data. To summarize, the result suggests that generating synthetic data is more effective than simply repeat data, but the pursuit of generating higher quality synthetic data can bring more significant performance improvements for downstream ORL performance.

Table 3. Normalized average score of generating different types of augmented data over 3 seeds on MuJoCo -v0 tasks.

Task Name	Repeat +TD3BC	NoV +TD3BC	HIPODE +TD3BC	TD3BC
halfcheetah-m-e	97.4	99.1	102.6	98.0
hopper-m-e	111.9	110.5	112.4	112
walker2d-m-e	38.0	102.5	105.3	105.4
halfcheetah-m-r	42.5	44.0	44.0	43.3
hopper-m-r	36.5	36.2	36.2	32.7
walker2d-m-r	18.0	30.0	36.4	19.6
halfcheetah-m	43.6	43.1	43.7	43.7
hopper-m	99.8	99.7	99.9	99.9
walker2d-m	79.5	79.7	80.1	79.7
halfcheetah-r	11.7	13.1	15.5	12.8
hopper-r	11.1	10.9	10.9	10.9
walker2d-r	1.9	2.1	6.6	0.4
halfcheetah-e	104.2	105.3	106.5	107.0
hopper-e	112.5	112.3	112.3	107.3
walker2d-e	78.1	104.9	106.6	98.7
Total	886.8	993.3	**1019.0**	971.4

Table 4. Normalized score comparison of policy-dependent methods for data augmentation v.s. HIPODE and the baseline on MuJoCo -v0 tasks. We report average normalized score over 3 random seeds each task. Full results consists of -random and -expert tasks are presented in Appendix B.

Task Name	MB+2.5 TD3BC	MB+0.001 TD3BC	MBPO	HIPODE+ TD3BC	TD3BC	BooT +CQL	CQL
halfcheetah-m-e	26.0	73.0	9.7	102.6	98.0	5.0	94.8
hopper-m-e	1.1	42.6	56	112.4	112.0	0.8	111.9
walker2d-m-e	42.9	8.5	7.6	105.3	105.4	26.4	70.3
halfcheetah-m-r	45.8	23.1	47.3	44	43.3	4.3	42.5
hopper-m-r	4.8	20.9	49.8	36.8	32.7	5.0	28.2
walker2d-m-r	0.0	7.5	22.2	36.4	19.6	5.8	5.1
halfcheetah-m	45.4	36.7	28.3	43.7	43.7	30.0	39.2
hopper-m	0.7	30.2	4.9	99.9	99.9	79.8	30.3
walker2d-m	4.3	16.9	12.7	80.1	79.7	6.4	66.9
Total	171.0	259.4	238.5	**661.2**	634.3	163.5	489.2

5.3. Comparison with Policy-Dependent Data Augmentation Methods

In policy-dependent data augmentation methods, the data generation process is tightly tied to the downstream ORL policy, which limits the applicability of the generated data. In this section we aim to illustrate the strength of our policy-decoupled data augmentation method, compared to policy-dependent methods on different downstream ORL policies. Specifically, on the downstream TD3BC algorithm, we evaluate the effect of data generated with some model-based policy dependent algorithms; on the downstream CQL algorithm, we analyze the effect of the more advanced policy dependent algorithm Boot (Wang et al., 2022).

We first evaluate the performance of dynamics-model-enhanced TD3BC based on the data generated by a previously trained dynamics model, by rolling-out current TD3BC policy on the dynamics model. The results are reported as MB+αTD3BC in Table 4, where α is a hyperparameter in TD3BC (Fujimoto & Gu, 2021). Results in Table 4 indicate that using dynamics-model-generated data as augmentation will damage the offline agent, and such damage can be mitigated when the policy of the offline agent is closed to the behavioural policy of the dataset. This suggests that the damage is caused by the difference between the policy the dynamics model trained on and the policy it generates data on, which these model-based policy-dependent methods fail to address.

We then directly take results report in (Wang et al., 2022) to form the BooT+CQL column in Table 4. BooT+CQL means directly using synthetic data generated by BooT on CQL. The results show that synthetic data generated by Boot has poor results as augmented data combined with CQL. This indicates that synthetic data generated by a policy-dependent data augmentation method can damage another offline agent. In contrast, HIPODE is policy-decoupled and our augmented data can benefit different offline agent without changing, as shown in Table 2.

In summary, synthetic data generated by policy-dependent data augmentation methods may have a detrimental effect on ORL processes, while the synthetic data generated by HIPODE can improve their performance, demonstrating the superiority of HIPODE.

6. Conclusions and Limitations

In this paper, we investigate the issues of data augmentation for ORL. We conduct extensive experiments to demonstrate that, in the context of ORL, high-quality data is a more suitable choice for augmented data than high-diversity data when the authority of the data is the same. Based on this observation, we propose a novel data augmentation method called HIPODE, which selects states with higher values as augmented data. This ensures that the synthetic data is both authentic and of high-quality and is generated in a policy-decoupled manner. Our experimental results on D4RL benchmarks demonstrate that HIPODE significantly improves the performance of several widely used model-free ORL baselines without changing the augmented data, thereby achieving policy-decoupled data augmentation and demonstrating superiority over policy-dependent methods. Furthermore, HIPODE outperforms SOTA policy-decoupled data augmentation methods for ORL, demonstrating the benefits by generating high-quality data.

The limitation of our work lies in the complexity of the method, as it requires several models to generate synthetic data. Additionally, HIPODE is outperformed by vanilla model-based ORL methods (e.g., MBPO) on -random datasets because the value penalty is excessively strict on those datasets. We believe that adjusting the penalty weight

to be state-dependent instead of initially setting it to a fixed value is a potential solution to this issue, which we leave for future work.

References

An, G., Moon, S., Kim, J.-H., and Song, H. O. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *Advances in neural information processing systems*, 34:7436–7447, 2021.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Fu, Y., Wu, D., and Boulet, B. A closer look at offline rl agents. In *Advances in Neural Information Processing Systems*.

Fujimoto, S. and Gu, S. S. A Minimalist Approach to Offline Reinforcement Learning. *arXiv:2106.06860 [cs, stat]*, December 2021. URL http://arxiv.org/abs/2106.06860. arXiv: 2106.06860.

Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pp. 2052–2062. PMLR, 2019.

Guo, K., Shao, Y., and Geng, Y. Model-based offline reinforcement learning with pessimism-modulated dynamics belief. *arXiv preprint arXiv:2210.06692*, 2022.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Hinton, G. E. and Roweis, S. Stochastic neighbor embedding. *Advances in neural information processing systems*, 15, 2002.

Janner, M., Li, Q., and Levine, S. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34: 1273–1286, 2021.

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33: 21810–21823, 2020.

Kostrikov, I., Fergus, R., Tompson, J., and Nachum, O. Offline reinforcement learning with fisher divergence critic regularization. In *International Conference on Machine Learning*, pp. 5774–5783. PMLR, 2021.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit q-learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative Q-Learning for Offline Reinforcement Learning. pp. 13.

Kumar, A., Fu, J., Soh, M., Tucker, G., and Levine, S. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in Neural Information Processing Systems*, 32, 2019.

Kumar, A., Agarwal, R., Ma, T., Courville, A., Tucker, G., and Levine, S. Dr3: Value-based deep reinforcement learning requires explicit regularization. *arXiv preprint arXiv:2112.04716*, 2021.

Lange, S., Gabel, T., and Riedmiller, M. Batch reinforcement learning. *Reinforcement learning: State-of-the-art*, pp. 45–73, 2012.

Liu, Y., Swaminathan, A., Agarwal, A., and Brunskill, E. Provably good batch off-policy reinforcement learning without great exploration. *Advances in neural information processing systems*, 33:1264–1274, 2020.

Lu, C., Ball, P., Parker-Holder, J., Osborne, M., and Roberts, S. J. Revisiting design choices in offline model based reinforcement learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=zz9hXVhf40.

Luo, Y., Xu, H., and Ma, T. Learning self-correctable policies and value functions from demonstrations with negative sampling. *arXiv preprint arXiv:1907.05634*, 2019.

Luo, Y., Xu, H., and Ma, T. Learning self-correctable policies and value functions from demonstrations with negative sampling. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rke-f6NKvS.

Lyu, J., Li, X., and Lu, Z. Double Check Your State Before Trusting It: Confidence-Aware Bidirectional Offline Model-Based Imagination, June 2022. URL http://arxiv.org/abs/2206.07989. Number: arXiv:2206.07989 arXiv:2206.07989 [cs].

Ma, X., Yang, Y., Hu, H., Liu, Q., Yang, J., Zhang, C., Zhao, Q., and Liang, B. Offline reinforcement learning with value-based episodic memory. *arXiv preprint arXiv:2110.09796*, 2021a.

Ma, Y., Jayaraman, D., and Bastani, O. Conservative offline distributional reinforcement learning. *Advances in Neural Information Processing Systems*, 34:19235–19247, 2021b.

Prudencio, R. F., Maximo, M. R., and Colombini, E. L. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

Rigter, M., Lacerda, B., and Hawes, N. Rambo-rl: Robust adversarial model-based offline reinforcement learning. *arXiv preprint arXiv:2204.12581*, 2022.

Seno, T. and Imai, M. d3rlpy: An offline deep reinforcement learning library. *Journal of Machine Learning Research*, 23(315):1–20, 2022. URL http://jmlr.org/papers/v23/22-0017.html.

Sinha, S., Mandlekar, A., and Garg, A. S4rl: Surprisingly simple self-supervision for offline reinforcement learning in robotics. In *Conference on Robot Learning*, pp. 907–917. PMLR, 2022.

Sohn, K., Lee, H., and Yan, X. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28, 2015.

Sun, Y. Offlinerl-kit: An elegant pytorch offline reinforcement learning library. https://github.com/yihaosun1124/OfflineRL-Kit, 2023.

Tarasov, D., Nikulin, A., Akimov, D., Kurenkov, V., and Kolesnikov, S. CORL: Research-oriented deep offline reinforcement learning library. In *3rd Offline RL Workshop: Offline RL as a "Launchpad"*, 2022. URL https://openreview.net/forum?id=SyAS49bBcv.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Wang, J., Li, W., Jiang, H., Zhu, G., Li, S., and Zhang, C. Offline reinforcement learning with reverse model-based imagination. *Advances in Neural Information Processing Systems*, 34:29420–29432, 2021.

Wang, K., Zhao, H., Luo, X., Ren, K., Zhang, W., and Li, D. Bootstrapped transformer for offline reinforcement learning. *arXiv preprint arXiv:2206.08569*, 2022.

Weissenbacher, M., Sinha, S., Garg, A., and Yoshinobu, K. Koopman q-learning: Offline reinforcement learning via symmetries of dynamics. In *International Conference on Machine Learning*, pp. 23645–23667. PMLR, 2022.

Wu, Y., Tucker, G., and Nachum, O. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J. Y., Levine, S., Finn, C., and Ma, T. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020.

Yu, T., Kumar, A., Rafailov, R., Rajeswaran, A., Levine, S., and Finn, C. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021.

Zhang, H., Shao, J., Jiang, Y., He, S., Zhang, G., and Ji, X. State deviation correction for offline reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 9022–9030, 2022.

A. Detailed Settings of Experiments

A.1. D4RL Tasks

In this section, we describe details about the MuJoCo and Adroit tasks in the D4RL (Fu et al., 2020) benchmark suite, on which we evaluate HIPODE.

MuJoCo contains a series of continuous locomotion tasks. Among them, Walker2d is a bipedal robot control task, where the goal is to maintain the balance of robot body and move as fast as possible. Hopper is a single-legged robot control task where the goal is to make the robot jump as fast as possible. Halfcheetah is to make a simulated robot perform a running motion that resembles a cheetah's movement, while trying to maximize the distance traveled within a fixed time period. In the MuJoCo-v0 tasks of D4RL, each environment has 5 different types of datasets: expert, medium-expert, medium-replay, medium and random. **Expert**: a large amount of data collected by a well-trained SAC agent. **Medium**: a large amount of data collected by a early-stopped SAC agent. **Medium-expert**: a large amount of mixed data of medium and expert at a 50-50 ratio. **Medium-replay**: replay buffer of a early-stopped SAC agent. **Random**: a large amount of data collected by a random policy.

Adroit contains a series of continuous and sparse-reward robotic environment to control a 24-DoF simulated Shadow Hand robot to twirl a pen, hammer a nail, open a door or grab a ball. These environments are even hard for online learning due to sparse rewards and exploration challenges (Fu et al., 2020). There are three types of datasets for each environment: expert, human and cloned. Among them, we evaluate HIPODE on human datasets, where a small number of demonstrations operated by a human (25 trajectories per task) is collected.

We report normalized score based on the protocol described in (Fu et al., 2020). A score of 0 represents the average return of random policies, and a score of 100 represents the return of a domain-specific expert. In our experiments, all score is the final performance of the downstream offline reinforcement learning (ORL) algorithms, which is the average cumulative reward of ten final policy rollouts.

Table 5. Hyper-parameters of HIPODE+CQL on 15 MuJoCo -v0 tasks.

HIPODE+CQL Task Name	**synthetic rate** η	**selecting rate** λ	**candidate number** n	**penalty weight** α	**penalty scope** σ	**CQL** α
halfcheetah-e	0.2	0.2	10	1.0	1.0	10.0
halfcheetah-m-e	0.1	0.2	10	1.0	1.0	10.0
halfcheetah-m-r	0.2	1.0	10	1.0	1.0	10.0
halfcheetah-m	0.2	1.0	10	1.0	1.0	10.0
halfcheetah-r	0.1	0.2	10	1.0	1.0	10.0
hopper-e	0.2	1.0	10	1.0	1.0	10.0
hopper-m-e	0.2	0.2	10	1.0	1.0	10.0
hopper-m-r	0.2	1.0	10	1.0	1.0	10.0
hopper-m	0.2	0.2	10	1.0	1.0	10.0
hopper-r	0.2	1.0	10	1.0	1.0	10.0
walker2d-e	0.2	0.2	10	1.0	1.0	10.0
walker2d-m-e	0.1	1.0	10	1.0	1.0	10.0
walker2d-m-r	0.2	1.0	10	1.0	1.0	10.0
walker2d-m	0.2	1.0	10	1.0	1.0	10.0
walker2d-r	0.2	1.0	10	1.0	1.0	10.0

A.2. Hyper-Parameters

In this section, we provide the key hyper-parameters used for HIPODE in the main experiments. They are listed in Table 5 and Table 6. Unless specified otherwise, all results in this paper are obtained using the hyper-parameters listed here. To reproduce CABI, we follow the hyper-parameters provided in (Lyu et al., 2022). For COMBO, we use the Offline-RL-Kit's code (Sun, 2023) and follow (Yu et al., 2021) to set the real ratio to 0.8 for both walker2d-medium-expert and walker2d-expert while 0.5 for other tasks. For all other hyper-parameters in COMBO and MOPO -expert, we use the default

Table 6. Hyper-parameters of HIPODE+TD3BC on 15 MuJoCo -v0 tasks and 4 Adroit-human-v0 tasks.

Task Name	**synthetic rate** η	**selecting rate** λ	**candidate number** n	**penalty weight** α	**penalty scope** σ	**TD3BC** α
halfcheetah-e	0.15	0.2	10	1.0	1.0	2.5
halfcheetah-m-e	0.15	0.2	10	1.0	1.0	2.5
halfcheetah-m-r	0.5	0.2	10	1.0	1.0	2.5
halfcheetah-m	0.2	0.2	10	1.0	1.0	2.5
halfcheetah-r	0.7	0.2	10	1.0	1.0	2.5
hopper-e	0.2	0.2	10	1.0	1.0	2.5
hopper-m-e	0.5	0.2	10	1.0	1.0	2.5
hopper-m-r	0.5	0.2	10	1.0	1.0	2.5
hopper-m	0.1	0.2	10	1.0	1.0	2.5
hopper-r	0.15	0.2	10	1.0	1.0	2.5
walker2d-e	0.2	0.2	10	1.0	1.0	2.5
walker2d-m-e	0.05	0.2	10	1.0	1.0	2.5
walker2d-m-r	0.5	0.2	10	1.0	1.0	2.5
walker2d-m	0.15	0.2	10	1.0	1.0	2.5
walker2d-r	0.4	0.2	10	1.0	1.0	2.5
door-human	0.4	0.2	10	1.0	1.0	0.001
hammer-human	0.2	0.2	10	1.0	1.0	0.001
pen-human	0.8	0.2	10	1.0	1.0	0.001
relocate-human	0.4	0.2	10	1.0	1.0	0.001

hyper-parameters provided in the Offline-RL-Kit's code [1].

A.3. Implementation Details

In this section, we describe details of the implementation to our experiments.

Data compression. For data augmentation, generating and storing large amount of synthetic data in a static dataset before the downstream ORL process can be resource-intensive. To address this issue, in practice, HIPODE is integrated into the downstream ORL process by generating synthetic data in real-time during the ORL process. This technique compress the size of synthetic data to the parameters of several generative models in HIPODE. In the data generating process, the synthetic rate η denotes the rate of synthetic data in every batch fed into downstream ORL algorithms. Therefor, a batch of size N contains ηN synthetic transitions and $(1-\eta)N$ real transitions.

Models in HIPODE. Here we describe the details about the models in HIPODE. There are 5 independent models in HIPODE:

- **Value network.** The value network is implemented using a Multi-Layer Perceptron (MLP) with one hidden layer of 256 units. We update the target value network every 2 gradient steps using the soft update method $\bar{\theta} = \tau\theta + (1-\tau)\bar{\theta}$, where $\tau = 0.005$ is the update rate.

- **Inverse action model.** The inverse action model is implemented using a CVAE (Sohn et al., 2015) to generate an action from a given state and next state. The encoder and decoder of the CVAE both have one hidden layer of 750 units, and the latent dimension is twice the state dimension of each task.

- **Inverse reward model.** The inverse reward model is implemented using a CVAE to generate a reward from a given state and next state. Its structure is similar to the inverse action model, but the two models are trained separately. Together, they are referred to as the 'inverse dynamics model'.

- **Forward dynamics model.** The forward dynamics model predicts the next state from a given current state and action. For

[1] CORL code URL: https://github.com/tinkoff-ai/CORL

a fair comparison, the implementation of the forward dynamics model is identical for both CABI and HIPODE and we refer to the implementation of the dynamics model in the D3RLPY (Seno & Imai, 2022) library [2] for the implementation of this part.

- **State transition model.** The state transition model is implemented using a CVAE to generate the next state from a given current state, following (Zhang et al., 2022). Its structure is the same as that of the inverse action model, except that the input dimension of the encoder and the output dimension of the decoder are different.

From the details of the HIPODE models, it can be seen that, although the data generating process is integrated into the downstream ORL process, the training process and data generating process of all the models in HIPODE is decoupled from the downstream ORL process, thus achieving policy-decoupled data augmentation.

B. Additional Results

B.1. HIPODE on Adroit Tasks

To further demonstrate the effectiveness of HIPODE, we also conduct experiments on Adroit-human tasks to evaluate its performance in a more challenging setting with limited human demonstrations and sparse rewards.

As shown in Table 7, HIPODE is effective, improving the baseline TD3BC on three out of four challenging Adroit tasks. However, the performance of HIPODE+TD3BC on Adroit-human datasets is less effective. We attribute this to the poor performance of the baseline offline agent on Adroit tasks due to the tasks' complexity (Fu et al., 2020).

Table 7. Average normalized score and standard deviation of HIPODE+TD3BC v.s. TD3BC over 3 random seeds on Adroit-human tasks.

Task Name	HIPODE+TD3BC	TD3BC
door-human	2.7 ± 2.6	1.3 ± 1.2
hammer-human	3.5 ± 4.0	3.9 ± 5.6
pen-human	85.0 ± 14.3	60.8 ± 14.0
relocate-human	0.2 ± 0.1	0.1 ± 0.1
Total	**91.4**	66.1

B.2. HIPODE V.S. Reported CABI Results

In Table 2, we report score of reproduced CABI for a fair comparison. However, our reproduced results are slightly worse than those reported in the original CABI paper. To provide a more comprehensive view of performance, we list the results reported in the original CABI paper in Table 8. As shown in Table 8, HIPODE's advantage compared to CABI is still significant when combined with CQL, while comparable when combined with TD3BC.

Additionally, to further demonstrate the strong potential of HIPODE, we also conduct experiments tuning the penalty weight α. We find that adjusting the penalty weight α on some tasks (e.g. walker2d-expert-v0 in Table 11) can improve the performance, leading to a higher total score than that of reported CABI+TD3BC. For the sake of fair comparison and ease of use, we only report the results of penalty-weight-non-tuned experiments in this paper.

B.3. Full Results for Policy-Dependent Methods V.S. HIPODE

In this section, we present the results of all 15 MuJoCo tasks to compare the performance of HIPODE with policy-dependent data augmentation methods, and the results are shown in Table 9. We begin by evaluating the performance of dynamics-model-enhanced TD3BC, using data generated by a previously trained dynamics model. Specifically, we roll out the current TD3BC policy on the dynamics model. The results are reported in Table 9 as MB+αTD3BC, where α is a hyper-parameter in TD3BC (Fujimoto & Gu, 2021). The MB+αTD3BC results, together with those from MBPO, suggest that using dynamics-model-generated data as augmentation can harm the offline agent. However, we find that the damage can be reduced when the policy of the offline agent is close to the behavioural policy of the dataset. This suggests that the damage is caused by the mismatch between the policy that the dynamics model was trained on and the policy it uses to generate data, which cannot be addressed by model-based policy-dependent methods.

We then conduct experiments on a more advanced policy-dependent data augmentation method, BooT (Wang et al., 2022). We directly take results report in (Wang et al., 2022) to form the BooT+CQL column in Table 4. BooT+CQL presents the use of synthetic data generated by BooT as augmentation data for CQL. The results indicate that the synthetic data generated

[2] D3RLPY code URL: https://github.com/takuseno/d3rlpy

Table 8. Average normalized score of HIPODE over 3 random seeds v.s. reported CABI on 15 MuJoCo -v0 tasks.

Task Name	HIPODE +CQL	CABI +CQL	HIPODE +TD3BC	CABI +TD3BC
halfcheetah-m-e	99.5	35.3	102.6	105.0
hopper-m-e	112.1	112.0	112.4	112.7
walker2d-m-e	94.0	107.5	105.3	108.4
halfcheetah-m-r	46.0	44.6	44.0	44.4
hopper-m-r	34.7	34.8	36.2	31.3
walker2d-m-r	21.7	21.4	36.4	29.4
halfcheetah-m	39.3	42.4	43.7	45.1
hopper-m	30.4	57.3	99.9	100.4
walker2d-m	75.7	62.7	80.1	82.0
halfcheetah-r	23.1	30.2	15.5	15.1
hopper-r	10.4	10.7	10.9	11.9
walker2d-r	10.5	7.3	6.6	6.4
halfcheetah-e	109.1	99.2	106.5	107.6
hopper-e	112.2	112.0	112.3	112.4
walker2d-e	109.3	110.2	106.6	108.6
Total	**928.0**	887.6	1019.0	1020.7

by BooT performs poorly when used as augmentation data combined with CQL.This suggests that synthetic data generated by a policy-dependent data augmentation method can have a detrimental effect on ORL algorithms. In contrast, HIPODE's synthetic data can benefit them without causing harm, as demonstrated in the HIPODE+TD3BC and HIPODE+CQL column in Table 9.

In summary, the synthetic data generated by policy-dependent data augmentation methods can have a negative impact on ORL processes, while HIPODE' synthetic data can improve them, demonstrating the superiority of HIPODE.

B.4. More Ablation Study

In this section, we investigate how HIPODE enhances downstream offline policy learning performance by examining two key components: the negative sampling mechanism and the state transition model. We analyze the impact of these components as independent variables of downstream offline policy learning performance.

Is the state transition model critical? To investigate the importance of state transition model, we remove the state transition model and randomly generate candidate next states inside the hypercube formed by the states in the original dataset with the other mechanisms and hyper-parameters stay the same. Results in Table 10 shows that removing state transition model severely drops compare to the baseline. In terms of the results in Table 10, although the negative sampling mechanism penalizes the OOD states in the no-state-transition-model condition, randomly choosing candidate next states can damage the downstream offline policy learning process, demonstrating the necessity of a state transition model. We believe the following reasons are responsible for this: (1) The value function is not authentic on randomly sampled next states, so the value is not very effective; (2) few in-support next states are generate so the winning next state may still be an OOD state resulting in a inauthentic synthetic transition; (3) The synthetic transition can lead the policy to a randomly state during evaluation. Hence, state transition model is significantly critical to ensure an authentic augmentation. This resembles the necessity of a behavioral policy in CABI and consistent with their conclusions (Lyu et al., 2022).

Table 10. Normalized results over 3 random seeds of HIPODE with randomly sampling next state v.s. HIPODE with CVAE next state.

Name	HIPODE(no stm) +TD3BC	HIPODE+ TD3BC	TD3BC
halfcheetah-m-r	0.6±0.1	44	43.3
halfcheetah-m	27.3±2.9	43.7	43.7
halfcheetah-m-e	58.2±3.0	102.6	98.0

Is negative sampling critical? In negative sampling, the penalty weight controls the severity of the penalty added to the value of OOD states. A larger penalty weight makes it less likely for

Table 9. Full results of normalized score comparison of policy-dependent methods for data augmentation v.s. HIPODE and the baseline. We report average score over 3 random seeds each task.

Task Name	**mb+2.5 TD3BC**	**mb+0.001 TD3BC**	**MBPO**	**HIPODE+ TD3BC**	**TD3BC**	**BooT +CQL**	**CQL**	**HIPODE+ CQL**
halfcheetah-m-e	26.0	73.0	9.7	102.6	98.0	5.0	94.8	99.5
hopper-m-e	1.1	42.6	56	112.4	112.0	0.8	111.9	112.1
walker2d-m-e	42.9	8.5	7.6	105.3	105.4	26.4	70.3	94.0
halfcheetah-m-r	45.75	23.1	47.3	44	43.3	4.3	42.5	46.0
hopper-m-r	4.8	20.9	49.8	36.8	32.7	5.0	28.2	34.7
walker2d-m-r	0.0	7.5	22.2	36.4	19.6	5.8	5.1	21.7
halfcheetah-m	45.4	36.7	28.3	43.7	43.7	30.0	39.2	39.3
hopper-m	0.7	30.2	4.9	99.9	99.9	79.8	30.3	30.4
walker2d-m	4.3	16.9	12.7	80.1	79.7	6.4	66.9	75.7
Total	171.0	259.4	238.5	**661.2**	634.3	163.5	489.2	553.4
halfcheetah-r	27.2	2.3	30.7	15.5	12.8	-	17.0	23.1
hopper-r	4.7	9.6	4.5	10.9	10.9	-	10.4	10.4
walker2d-r	0.1	1.3	13.6	6.4	0.4	-	1.7	10.5
Total	203.0	272.6	287.3	**694.0**	658.4		518.3	597.4
halfcheetah-e	-2.1	106.8	-	106.5	107.0	-	109.8	109.1
hopper-e	1.3	111.9	-	112.3	107.3	-	112.0	112.2
walker2d-e	49.3	84.4	-	106.6	98.7	-	104.7	109.3
Total	251.0	575.7	-	**1019.4**	971.4	-	844.9	929.0

Table 11. Normalized score of TD3BC combined with different penalty weight for data augmentation. The numbers in the headline denotes the penalty weight. We report average score over 3 random seeds each task.

Task Name	**-1+TD3BC**	**0+ TD3BC**	**1+ TD3BC**	**2+ TD3BC**	**4+ TD3BC**	**8+ TD3BC**	**TD3BC**
walker2d-e	105.2±2.1	102.9±8.1	106.6±5.2	105.2±2.2	106.6±1.0	109.1±0.2	98.7
halfcheetah-m-r	43.0±0.3	40.6±2.2	44.0±0.4	44.0± 0.7	43.8±0.6	43.8±0.3	43.3

HIPODE to generate an OOD next state. We change the penalty weight in $\{-1, 0, 1, 2, 4, 8\}$ and run the downstream offline policy learning algorithm, without changing the other hyper-parameters on walker2d-expert-v0 and halfcheetah-medium-replay-v0. The results in Table 11 show that penalty weights greater than 0 outperforms the others, but overall, the score difference is marginal. This indicates that penalizing the value function on OOD states can indeed benefit downstream offline policy learning process. On the other hand, the state transition model generates candidate states near the dataset, which makes the effect of the penalty insignificant.