ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230562

SDV: Simple Double Validation Model-Based Offline Reinforcement Learning

Xun Wang[a], Haonan Chen[a], Junming Yang[b], Zhuzhong Qian[a;*] and Bolei Zhang[c;*]

[a]State Key Laboratory for Novel Software Technology, Nanjing University, China
[b]School of Modern Posts, Nanjing University of Posts and Telecommunications, China
[c]School of Computer, Nanjing University of Posts and Telecommunications, China
ORCiD ID: Zhuzhong Qian https://orcid.org/0000-0003-1625-7575,
Bolei Zhang https://orcid.org/0000-0002-9406-4499

Abstract. Offline reinforcement learning (RL) aims to learn effective policies from recorded data without further interactions in the environments that are often costly or risky. Model-based algorithms, which begin by constructing an environmental model and then learn the policy under the model, have become a promising approach. However, most existing works have been over-conservative to avoid the out-of-distribution error induced by the model generated samples, leading to poor performance instead. In this work, we propose a novel model-based offline RL method, named Simple Double Validation (SDV). The main idea of SDV is to introduce an additional guidance model to assist the agent in determining the rationality of the states, combined with an advantage weighting factor to avoid effects that could potentially mislead the models due to suboptimal samples. In this way, the agent can be guided to more favourable states with reliable decisions. We evaluated SDV on the widely studied offline RL benchmarks and demonstrated its state-of-the-art performance. At the same time, our work introduces the idea of double validation and model advantage weighting into the field of model-based offline RL, providing new insights for future research.

1 Introduction

Reinforcement learning (RL) [24] has achieved state-of-the-art performance in many sequential decision problems [1, 21, 23]. However, the exploration in RL often carries high costs or significant risks, and has hindered its application in many real-world fields, such as robotics [4, 9], healthcare [25, 29], and autonomous driving [11, 32]. Alternatively, historical data records are much easier to collect, and can reveal system feedbacks under predefined policies. Recent offline RL works have utilized the collected data to learn the policies directly, so as to avoid expensive online interactions [16, 18], emerging as promising approaches for boosting the practical application of RL in aforementioned real-world fields.

In offline settings, as further interactions with the environment are not allowed, there may be distributional shift between the state-action pairs from the dataset and the real environment. As a result, policies with out-of-distribution (OOD) state-action pairs cannot be accurately evaluated in the training stage, and may further mislead the

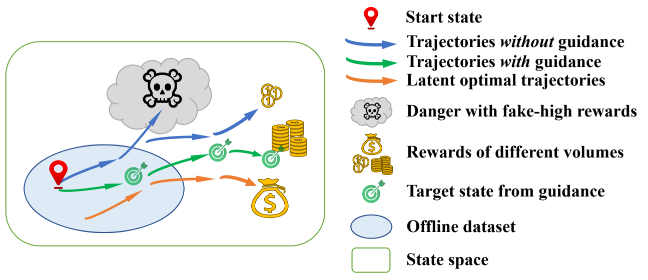

Start state
Trajectories *without* guidance
Trajectories *with* guidance
Latent optimal trajectories
Danger with fake-high rewards
Rewards of different volumes
Target state from guidance
Offline dataset
State space

Figure 1: The illustration of the effect of guidance. The orange arrows represent the latent optimal trajectories that achieve theoretically maximum reward. The blue arrows represent the trajectories without guidance which are prone to fall into danger or fail to achieve high rewards. The green arrows represent the guided trajectories which are close to optimal states within the dataset and avoid danger and achieve higher rewards outside of the dataset.

policy optimization, leading to poor performance during online deployment. Therefore, it is critical to handle the trade-offs between conservatism and optimism in offline settings. On one hand, conservatism is necessary so that the OOD state-action pairs will be visited less frequently; On the other hand, over-conservatism may also limit the performance of the policies severely.

To deal with the above problem, previous works have mainly adopted model-free and model-based methods. Model-free offline RL [3, 12, 13, 14, 15, 27, 28] directly trains policy with existing datasets and incorporates conservatism to constrain the policies. Although they have achieved significant performance gains compared to the online off-policy RL algorithm, the sample efficiencies are limited with the size of the dataset. In comparison, model-based offline RL [10, 20, 22, 26, 30, 31] begins by building an environment model and trains policy by querying the model to increase sample efficiency. The model-based approaches for overcoming OOD problem can be roughly divided into two categories: data augmentation [20, 26] and policy optimization [10, 22, 30, 31]. The former focuses on expanding the original dataset by generating more similar data using the model while the latter directly uses classical RL algorithms for policy optimization on the trained model. Although the latter can utilize the model more sufficiently, the agent-model interactions may accumulate significant extrapolation error and decrease the policy

* Correspondence to: Zhuzhong Qian (qzz@nju.edu.cn), Bolei Zhang (bolei.zhang@njupt.edu.cn).

performance. To address this issue, a common practice is to impose penalties on the uncertainty of the environment model. However, the uncertainty estimation of neural network models in practical implementations may be unreliable [19, 30], which tends to result in over-conservatism, decreasing the performance of the policy instead.

Rather than estimating uncertainty directly, one natural and feasible solution to the aforementioned problem is to provide guidance, navigating the agent to learn a satisfactory policy. Specifically, as illustrated in Figure 1, the orange arrows represent the latent optimal trajectories that avoid danger and achieve theoretically maximum reward. The goal of offline RL is to learn a policy as close as possible to these optimal trajectories, using only the available offline dataset, and maximize the cumulative reward. If without guidance, as shown by the trajectories consisting of blue arrows, the agent is prone to fall into danger due to extrapolation errors or fails to achieve high rewards for over-conservatism. On the contrary, guidance can inform the agent of the future state, just as the trajectories represented by the green arrows. With such guidance, the agent can approach optimal states within the dataset and be navigated to keep away from danger and achieve higher rewards outside of the dataset. This motivates us to provide guidance for the agent to improve the performance.

In this work, we propose a novel double-validation-based method: **S**imple **D**ouble **V**alidation Model-based Offline Reinforcement Learning (**SDV**) with a simple yet effective idea. In brief, SDV trains a guidance model to perform a secondary validation on the next state predicted by the transition model, which assists the agent in determining the rationality of that state and ultimately leads to a more effective policy. Since SDV does not directly estimate uncertainty, it effectively avoids the issue of excessive conservatism that exists in prior works. Considering the potential misleading that a large amount of suboptimal data may have on the models, SDV also introduces model advantage weighting factors to assist the models in extracting transitions with greater advantages, thereby improving training performance. To the best of our knowledge, the existing work on applying weights to model training mainly considers distributional differences, with few considering advantages.

In summary, the main contributions of this work are:

- SDV, a novel model-based offline RL algorithm that achieves excellent performance through combination of double validation and model advantage weighting.
- The idea of double validation and model advantage weighting may provide new insights for future research.

We tested[1] SDV on 9 MuJoCo datasets in the widely studied D4RL [2] benchmark, and demonstrated the state-of-the-art performance. In addition, the ablation results we provide indicate that double validation and model advantage weighting are crucial for the strong performance of SDV, especially double validation.

2 Related Work

Extensive efforts have been made to handle the trade-offs between conservatism and optimism in offline settings. Existing related works can be mainly divided into two categories: model-free offline RL and model-based offline RL. The former optimizes policies directly without training a model while the latter learns an environment model from offline data and trains policies with the help of the model.

Model-free offline RL Model-free offline RL methods [3, 12, 13, 14, 15, 27, 28] train policies directly on the offline dataset without learning an environment model. Previous works have studied serveral ways to guarantee conservatism, including constraining the target policy near the behavior policy through behavior regularization [3, 14, 27], or avoiding overvaluing OOD state-action pairs by optimizing policies with conservative Q-value functions [12, 13, 15]. Although these methods are compute-efficient, their inherent limitation of using only offline datasets constrains the policy from better generalization on OOD data. Xu et al. [28] recently have proposed to decouple the target state from the target action, enhancing the generalization performance of the policy on OOD actions. But since the policy is still trained with states inside the static dataset, their method fails to process the OOD states.

Model-based offline RL Model-based offline RL methods [10, 17, 20, 22, 26, 30, 31] begins by constructing an environment model from the offline dataset and then optimize the policy with the model. Compared to model-free methods, the model improves sample efficiency and enhances generalization ability of the agent to OOD data. The works related to this direction can be roughly divided into two subcategories based on the purpose of the model. **1) Data Augmentation** [20, 26]. Analogous to imitative learning, these works usually use models to generate more trajectories similar to offline data, and then use model-free methods to train policies. This kind of methods introduce more OOD information via augmented data than pure model-free methods, but the augmented dataset is still static and constrains the policy from further generalization. **2) Policy Optimization** [10, 22, 30, 31]. These methods, also called as pure model-based, directly optimize policy with classical RL algorithms on the pre-trained model. Therefore, as long as the model is appropriate enough, for example, can prevent the agent from dangerous OOD states, the policy is highly likely to achieve better performance. In order to learn such an appropriate model, previous approaches [10, 31] mainly focus on quantifying the uncertainty of the model and imposing penalties on state-action pairs with high uncertainty, pushing the agent to be conservative enough when exploring the OOD region. However, accurately quantifying the uncertainty of neural networks is difficult, so many subsequent works avoids directly quantifying uncertainty. For example, Yu et al. [30] used conservative policy evaluation to suppress the value function, but have to refer to a certain distribution. Rigter et al. [22] improved the conservatism through adversarially training pessimistic models, but have to couple model training with policy optimization. There are also some works [7, 17] have attempted to improve model accuracy by incorporating penalties for distributional bias into the learning objectives. However, these approaches seldom consider advantages when training the models.

In contrast to existing works, our method not only inherits the advantages of pure model-based offline RL methods, but also avoids directly quantifying model uncertainty. Instead, it adopts double validation to ensure conservatism without the requirement of reference distributions or coupled adversarial training. Additionally, it introduces model advantage weighting factors to further prevent the impact of suboptimal data.

3 Preliminaries

MDPs and Offline RL An MDP is defined using the six tuple $M = (S, A, T, R, \mu_0, \gamma)$. S and A represent the state and action space, respectively. $R(s, a)$ is the reward function, $T(s' \mid s, a)$ is the transition function, μ_0 is the initial state distribution, and γ is the discount factor. In this work, we consider Markovian policy, $\pi(a|s)$, which maps each state to the distribution of actions.

The state value function $V^{\pi}_{M}(s)$ (also known as the V-value function) represents the discounted return obtained by executing π from

[1] Code and appendix are available at https://github.com/Misakau/SDV.

state s in M: $V_M^\pi(s) = E_{\pi,M}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t, a_t)\right]$. We write V_M^π to indicate the value function under the initial state distribution, i.e. $V_M^\pi = E_{s\sim\mu_0}\left[V_M^\pi(s)\right]$. The goal of a standard MDP is to find the policy that can maximize V_M^π. In order to optimize the policy, it is also necessary to introduce the state-action value function $Q_M^\pi(s,a)$ (also known as the Q-value function), which is the expected discounted return obtained by executing action a in state s and then executing policy π thereafter.

In offline RL we only have an offline dataset $\{(s_i, a_i, r_i, s_i')\}_{i=1}^{|D|}$ composed of transitions sampled from real MDP. The goal of offline RL is to find policy that performs as well as possible during online deployment without interacting with the real environment.

Model-Based Policy Optimization In line with many existing works [22, 30, 31], we use Model-Based Policy Optimization (MBPO) [8] to optimize the policy. MBPO uses the standard Actor-Critic RL algorithm for policy optimization, but its value function is trained using the enhanced dataset $D \cup D_{model}$, where D_{model} is the set of synthetic data from the learned model. In order to obtain D_{model}, MBPO rolls out a batch of k-steps trajectories beginning with states in D. As for mini-batches to train the policy, MBPO samples a proportion of f of data points coming from real data D with the remaining $1-f$ proportion coming from D_{model} in each mini-batch.

Model-based Offline RL algorithm Model-based offline RL algorithm uses an estimated MDP model to train the policy. Specifically, it first trains a transition model $\widehat{T}$ from the dataset, typically through maximum likelihood estimation (MLE): $\min_{\widehat{T}} E_{(s,a,s')\sim D}\left[-\log \widehat{T}(s' \mid s,a)\right]$. If the reward model $\widehat{R}(s,a)$ is also unknown, it will also be trained, or generally for convenience, incorporated into $\widehat{T}(s', r \mid s, a)$ as part of the state. Apart from $\widehat{R}$ and $\widehat{T}$, the state and action space, initial state distribution and discount factor are consistent with the real MDP. As long as the estimated MDP model $\widehat{M} = (S, A, \widehat{T}, \widehat{R}, \mu_0, \gamma)$ is constructed, any planning or RL algorithm can be used to optimize the policy $\hat{\pi}$ within it.

However, directly applying model-based methods, such as MBPO, to offline policy optimization does not perform well due to aforementioned distributional shift [10]. To address this issue, we propose a novel method that improves the reliability of policy optimization through double validation. Meanwhile, it also prevents the models from being misled by suboptimal data with a weighting factor based on advantages.

4 Simple Double Validation Model-based Offline Reinforcement Learning (SDV)

To the best of our knowledge, almost all existing model-based methods train policies without guidance, although it may help the agent make better decisions. Therefore, inspired by the fact that humans think twice before making decisions, we propose a novel model-based algorithm, named **S**imple **D**ouble **V**alidation Model-based Offline Reinforcement Learning (**SDV**). Apart from the transition model, SDV trains a guidance model to provide guidance for the agent. Then, in order to prevent the models from being misled by suboptimal data, it also introduces model advantage weighting factors to common MLE losses. Finally, the policy is optimized by SAC algorithm [5] with an MBPO-like framework.

The final algorithm is shown in Algorithm 1. And to demonstrate the core idea and effectiveness of SDV better, we built two toy environments, which are detailed in Appendix B.

4.1 Double validation

Before making decisions, humans usually refer to past experiences and others' guidance to evaluate the future from different perspectives. Inspired by this, SDV not only follows the convention of training a transition model, but also trains a guidance model based on the offline dataset. The guidance model is only related to the states, excluding specific actions. Then the predicted rewards is reshaped according to the difference between the reference state suggested by the guidance model and the next state given by the transition model. Consequently, the agent is able to validate the rationality of the next state from two different perspectives and make better decisions.

Training guidance model From an intuitive standpoint, if the empirical data is not extremely poor, for an initial state, the more frequently a target state appears in past experiences (i.e. the offline dataset D), the more likely it is to be safe or even good, no matter what action the agent takes. Therefore, the guidance model $g_\omega(s' \mid s)$ should satisfy:

$$\omega = \arg\max_{\omega} E_{(s,s')\sim D}\left[\log g_\omega\left(s' \mid s\right)\right] \tag{1}$$

where ω is the parameters of the guidance model.

As a result, we trained the guidance model g_ω solely based on the transitions (s, s') between the current states and the next states in the offline dataset. In this way, the guidance can be encouraged to focus on macroscopic state transitions without being disturbed by specific actions. The corresponding loss function is as follows:

$$L_g(\omega) = -E_{(s,s')\sim D}\left[\log g_\omega\left(s' \mid s\right)\right] \tag{2}$$

Training transition model Then, as a model-based approach, SDV conventionally trains the transition model $T_\phi(s', r \mid s, a)$ through an offline dataset, incorporating the immediate reward r as part of the state into the transition model. The corresponding loss function is shown below:

$$L_T(\phi) = -E_{(s,a,r,s')\sim D}\left[\log T_\phi\left(s', r \mid s, a\right)\right] \tag{3}$$

where ϕ is the parameters of the transition model.

It is worth noting that, unlike the guidance model, the transition model also takes the specific actions into account, allowing for a more granular micro-perspective. The combination of micro and macro perspective is also what distinguishes SDV's double validation from directly training multiple ensemble models.

Reward shaping With the assistance of guidance model, SDV assesses the validity of the next state predicted by the transition model. Then it injects the judgment into the MDP model through reward shaping, thus guiding the agent to choose more reasonable states during policy optimization.

Specifically, SDV first obtains the reference state s'' suggested by the guidance model. Then it calculates the distance between s'' and the next state s' predicted by the transition model. Finally, SDV penalizes the predicted immediate reward based on the distance:

$$\hat{r} = r - \lambda\|s' - s''\|, \quad s' \sim T_\phi(s,a),\ s'' \sim g_\omega(s) \tag{4}$$

where λ is a non-negative constant controlling the severity of punishment.

It is obvious that the smaller the distance between the next state and the reference state is, the smaller the penalty on the immediate reward imposed. And the smaller penalty means that SDV considers the next state to be more reliable. It is also consistent with common sense that similar conclusions from different perspectives are generally more reliable than those from a single perspective. In addition, a reasonable assumption is that although both the two models may occasionally predict states deviating greatly from the expected output, the probability that the events occur simultaneously with the two output states being very close is very low. Indeed, it is this punishment mechanism that makes SDV inclined to choose the states it deems to be more reliable during policy optimization. Thus, SDV ensures the appropriate conservatism of the estimated MDP model and guides the agent to make more rational decisions.

4.2 *Model advantages weighting*

Although double validation with the guidance is more reliable compared to using only the transition model, the training of the models is entirely based on the offline dataset. Therefore, if there are many suboptimal data points, it may mislead the models to select states with higher frequency but are not optimal enough. Taking the potential misleading into consideration, SDV also employs model advantage weighting in addition to double validation. Specifically, the weights of data points with higher advantages are increased to help the models overcome the impact of suboptimal data.

Based on different determining factors, the advantage terms of the transition model and the guidance model are defined as follows, respectively:

$$A_T(s,a,s') = r + \gamma V_\psi\left(s'\right) - Q_\theta(s,a) \tag{5}$$

$$A_g(s,s') = r + \gamma V_\psi\left(s'\right) - V_\psi(s) \tag{6}$$

where Q_θ and V_ψ are value functions with parameters denoted by θ and ψ respectively. Moreover, r is the immediate reward and γ is the discount factor.

To obtain more reliable V-value and Q-value functions in Eq.5 and Eq.6, SDV uses expectile regression based on the offline dataset, rather than the current policy. This approach is similar to the one used by IQL[13] and is known to produce more accurate results. The loss functions are shown below:

$$L_V(\psi) = E_{(s,a)\sim D}\left[L_2^\tau\left(Q_{\hat{\theta}}(s,a) - V_\psi(s)\right)\right] \tag{7}$$

$$L_Q(\theta) = E_{(s,a,s')\sim D}\left[\left(r(s,a) + \gamma V_\psi\left(s'\right) - Q_\theta(s,a)\right)^2\right] \tag{8}$$

in which $L_2^\tau(\cdot)$ is the expectile regression loss defined as:

$$L_2^\tau(x) = |\tau - \mathbf{1}(x<0)|x^2 \tag{9}$$

where τ is a hyperparameter and $\mathbf{1}(\cdot)$ is an indicator function. Since the value functions are estimated only based on the offline dataset, SDV can maintain decoupling between model training and policy optimization while fine-tuning the models with advantage weighting.

After obtaining the value functions, SDV introduces an advantage weighting factor α to weight the models' loss:

$$L_T^\alpha(\phi) = -E_{(s,a,r,s')\sim D}\left[e^{\alpha\left(r+\gamma V_\phi\left(s'\right)-Q_\theta(s,a)\right)}\log T_\phi\right] \tag{10}$$

$$L_g^\alpha(\omega) = -E_{(s,r,s')\sim D}\left[e^{\alpha\left(r+\gamma V_\phi\left(s'\right)-V_\phi(s)\right)}\log g_\omega\right] \tag{11}$$

By advantage weighting, the models can extract transitions with greater advantages from the dataset. Subsequent ablation experiments also show that combined with double validation, the introduction of α can indeed improve the performance of the policy. It is also worth mentioning that α is not necessarily non-negative. When it is positive, it means choosing optimistic models, and when it is negative, it represents choosing models more pessimistically. If α exactly chosen as zero, it degenerates into common MLE models. In this way, SDV may be provided with more flexibility in its application. Previous work, such as POR[28], also used similar techniques. But as a model-free method, POR did not have the transition model, let alone weighting it, which discounted its generalization on OOD data.

Algorithm 1: Simple **D**ouble **V**alidation Model-based Offline Reinforcement Learning (**SDV**)

```
Input: offline data set $\mathcal{D}$
Output: policy $\pi(a \mid s)$
1  Initialize parameters $\psi, \theta, \hat{\theta}, \phi, \omega$.        ▷ Model training
2  for each gradient step do
3      $\psi \leftarrow \psi - \lambda_V \nabla_\psi L_V(\psi)$
4      $\theta \leftarrow \theta - \lambda_Q \nabla_\theta L_Q(\theta)$
5      $\hat{\theta} \leftarrow (1-\beta)\hat{\theta} + \beta\theta$
6      $\phi \leftarrow \phi - \lambda_T \nabla_\phi L_T^\alpha(\phi)$
7      $\omega \leftarrow \omega - \lambda_g \nabla_\omega L_g^\alpha(\omega)$
8  end
9  Initialize replay buffer $\mathcal{D}_{model} = \emptyset$.    ▷ Policy optimization
10 for epoch 1,2,... do
11     Sample $s_1$ with batch-size $b$ from $\mathcal{D}$ for the initialization
         of the rollout.
12     for $j = 1, 2, \ldots, k$ do
13         Sample action $a_j \sim \pi(s_j)$.
14         Sample $s_{j+1}, r_j \sim T_\phi(s_j, a_j)$.
15         Compute $\tilde{r_j} = r_j - \lambda\|s_{j+1} - g_\omega(s_j)\|$.
16         Add sample $(s_j, a_j, \tilde{r_j}, s_{j+1})$ to $\mathcal{D}_{model}$
17     end
18     Drawing samples from $\mathcal{D} \cup \mathcal{D}_{model}$, use SAC to update $\pi$.
19 end
```

5 Experiments

In this section, we conducted comprehensive experiments on 9 MuJoCo tasks from the widely used D4RL [2] dataset to evaluate the performance of SDV and tried to answer the following three questions:

- Does SDV have equally competitive or even superior performance compared to other baseline algorithms?
- Do double validation and model advantage weighting play a crucial role in improving the performance of SDV?
- Why is it counterintuitive that training with the random dataset collected by a stochastic policy is difficult, and has SDV made any efforts to improve its performance with such a dataset?

5.1 *Experiment setting*

Data collection We conducted experiments on three types of MuJoCo tasks from D4RL, namely: random, medium-replay, and medium. Their specific generation method is as follows and for all datasets we use the v2 version.

- Random: Roll out a randomly initialized policy for 1M steps.
- Medium: 1M samples from a policy trained to approximately 1/3 the performance of the expert by SAC.
- Medium-replay: Replay buffer of a policy trained up to the performance of the medium agent.

Table 1: The results of the D4RL benchmark using the normalization program proposed in [2]. We reported the normalization performance in the last 11 iterations of training, averaging over 3 seeds. ± represents the standard deviation of the seed. The bold numbers indicate that the results are within 2% of the most efficient algorithm.

	Ours	Model-based Baseline			Model-free Baseline				
	SDV	**COMBO**	**MOReL**	**MOPO**	**CQL**	**IQL**	**POR**	**BRAC-v**	**BC**
halfcheetah-r	**39.725** $\pm$ 2.114	38.8	25.6	35.4	35.4	11.2	29	31.2	2.1
hopper-r	23.415 $\pm$ 14.337	17.9	**53.6**	11.7	10.8	7.9	12	12.2	9.8
walker2d-r	7.074 $\pm$ 5.206	7	**37.3**	13.6	7	5.9	6.3	1.9	1.6
halfcheetah-mr	**55.264** $\pm$ 0.855	**55.1**	40.2	53.1	46.2	44.2	43.5	47.7	38.4
hopper-mr	**100.223** $\pm$ 2.902	89.5	93.6	67.5	48.6	94.7	**98.9**	0.6	11.8
walker2d-mr	**97.279** $\pm$ 2.902	56	49.8	39	32.6	73.8	76.6	0.9	11.3
halfcheetah-m	**62.648** $\pm$ 0.112	54.2	42.1	42.3	44.4	47.4	48.8	46.3	36.1
hopper-m	**96.966** $\pm$ 2.488	**97.2**	**95.4**	28	86.6	66.2	98.2	31.1	29
walker2d-m	**89.081** $\pm$ 3.342	81.9	77.8	17.8	74.5	78.3	81.1	81.1	6.6
Average Score	**63.519** $\pm$ 2.497	55.289	57.267	34.267	42.9	47.733	54.933	28.089	16.3

-r means -random, **-mr** means -medium-replay, **-m** means -medium

Experimental implementation The hyperparameters we mainly adjusted were the rollout step length k, the real data ratio f and the reward penalty coefficient λ. The advantage weighting factor α was set to 10 for all tasks.

We ran SDV on 3 random seeds for each task. In each run, we first conducted 500k steps of model training to obtain the guidance model and transition model. Next, considering fairness and consistency, we conducted 1M steps of policy training according to convention. We evaluated the policy every 1k steps.

Note that in order to highlight the simplicity and feasibility of SDV, we did not use preheating techniques such as behavior cloning, nor did we use the ensemble models commonly used by most existing model-based algorithms [10, 17, 22, 31]. Instead, we directly trained models conducted by single network, and optimized the policy with stochastic initialization. However, the results of experiments showed that even with such a simple setting, SDV still achieved good performance.

For more details about the specific architecture of networks, the hyperparameters settings, the evaluation procedure and the software and hardware environment, please refer to the Appendix A.

5.2 *Evaluation results on the D4RL dataset*

We compared SDV with pure model-based and model-free baseline algorithms. Pure model-based algorithms include: MOPO [31], COMBO [30] and MOReL [10]. Model-free algorithms include BC, BRAC-v, CQL [15], IQL [13], and POR [28]. Except for the results of IQL in the random dataset taken from [28], all the results were taken from the original papers or D4RL baseline[2]. The comparison results are shown in Table 1.

As the results show, although SDV only used single network models and did not perform auxiliary warm-up startup, it achieved SOTA on 6 tasks with a significant advantage over the others and was comparable to the previous best algorithms on most tasks, demonstrating its superiority.

Compared to RAMBO Due to the policy optimization of 2M steps and the coupling of model training and policy optimization, it is unfair to compare RAMBO [22] with the above algorithms. But we compared SDV separately with RAMBO in Table 2 to test whether the decoupled fixed models and shorter policy optimization would significantly reduce its performance. The results of RAMBO were taken from the original paper.

Table 2: The normalized results of SDV and RAMBO. The bold numbers indicate that the results are within 2% of the most efficient algorithm.

	SDV	**RAMBO**
halfcheetah-r	**39.725** $\pm$ 2.114	**40**
hopper-r	**23.415** $\pm$ 14.337	21.6
walker2d-r	7.074 $\pm$ 5.206	**11.5**
halfcheetah-mr	55.264 $\pm$ 0.855	**68.9**
hopper-mr	**100.223** $\pm$ 2.902	96.6
walker2d-mr	**97.279** $\pm$ 2.902	85
halfcheetah-m	62.648 $\pm$ 0.112	**77.6**
hopper-m	**96.966** $\pm$ 2.488	92.8
walker2d-m	**89.081** $\pm$ 3.342	86.9
Average Score	**63.519** $\pm$ 2.497	**64.444**

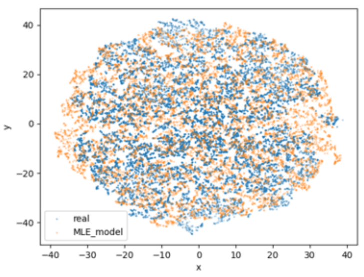
To our surprise, even with fixed models and only half the policy optimization steps of RAMBO, SDV outperformed RAMBO on 5 tasks and was competitive with it on most tasks. In particular, our algorithm performed much better than RAMBO on the hopper-m/mr and walker2d-m/mr tasks. The reason for the performance gap on the halfcheetah task may be that the policy has not fully converged due to the limited computational budget, which also resulted in SDV having an average score that is competitive with RAMBO but slightly lower.

5.3 *Ablation study*

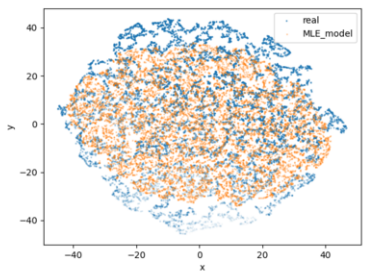
In order to investigate whether the double validation and model advantage weighting are indeed the key factors that make SDV perform better, we have conducted an ablation study on the medium and medium-replay datasets with the following three variants: **1) SDV-b:** double validation without model advantage weighting, **2) SDV-p:** model advantage weighting without double validation, and **3) SDV-n:** neither double validation nor model advantage weighting, i.e. offline MBPO.

The hyperparameters set for each ablation task were the same as those for evaluation experiments, except that the reward penalty coefficient λ was set to 0 for SDV-p and SDV-n while the advantage weighting factor α was set to 0 for SDV-b and SDV-n.

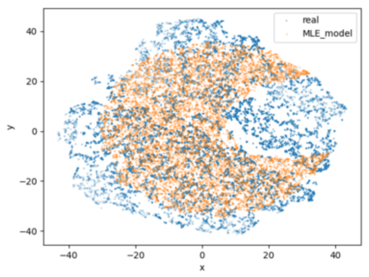
As the results shown in Table 3, the full implementation of SDV achieved better results on all tasks except for hopper-mr, which may



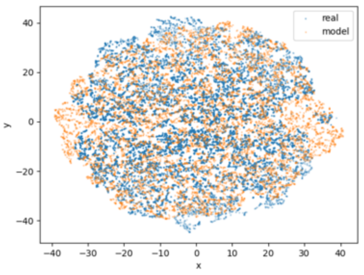
(a) Halfcheetah-random-MLE



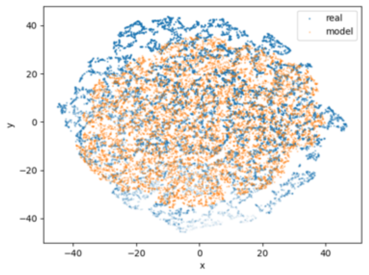
(b) Hopper-random-MLE



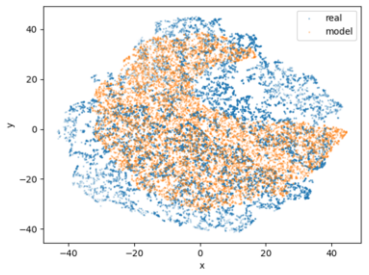
(c) Walker2d-random-MLE



(d) Halfcheetah-random-SDV



(e) Hopper-random-SDV



(f) Walker2d-random-SDV

Figure 2: T-SNE results, the orange points are the states rolled out from the model and the blue points are states sampled from real environment with darker blue indicating higher reward values.

have been overly optimistic due to the fixed setting of $\alpha = 10$, resulting in a slight decrease in its performance compared to SDV-b variant. Comparing the impact of the two techniques on the performance of SDV, on one hand, the double validation has made a significant contribution to the policy training performance. On the other hand, although the model advantage weighting technique alone may lead to worse results than the bare SDV-n due to being overly optimistic, it can provide some help in improving the policy performance when combined with the conservatism brought by the double validation. This indicates that it is necessary and effective to combine these two techniques to improve SDV's performance.

Table 3: The normalized results of ablation experiment. The bold numbers indicate that the results are the most efficient algorithm.

	SDV	**SDV-b**	**SDV-p**	**SDV-n**
halfcheetah-mr	**55.264**	51.872	52.519	49.877
hopper-mr	100.223	**102.778**	49.032	30.055
walker2d-mr	**97.279**	94.183	12.607	31.048
halfcheetah-m	**62.648**	62.014	38.778	44.151
hopper-m	**96.966**	76.68	0.686	1.9
walker2d-m	**89.081**	88.678	4.539	4.894
Total Score	**501.461**	489.290	158.162	161.925

5.4 *Exploration of random datasets*

Intuitively, when the dataset size is fixed, using random policy sampling should cover more transition distribution information, leading to a more accurate learned model. However, contrary to intuition, most methods, including SDV, perform worse on the hopper-r and walker2d-r datasets compared to halfcheetah-r while more conservative methods such as MOReL have an advantage. It indicates that the difference between the distribution supported by the two datasets and the true distribution may be much greater than that of halfcheetah-r, so more conservative approaches seem to be more appropriate, even though their performance on datasets with better data collection policies may not be that outstanding.

One possible explanation for the aforementioned difference is that for fine-tuned tasks such as robot control, a randomly initialized policy is more likely to get stuck in a suboptimal state. In the hopper and walker2d environments, if the robot's state violates certain constraints, the episode immediately will terminate (corresponding to high exploration costs and risks), which limits the space covered by trajectories and makes it difficult to explore the distribution. For example, the average length of trajectories in the hopper-r dataset is only 22 steps, and the highest normalized score is only 9.612. In contrast, the halfcheetah environment allows the robot to freely explore in 1k-step episodes, greatly increasing the possibility of trajectories covering more distribution information.

To validate the above explanation, we conducted the following experiments separately on halfcheetah-r, hopper-r and walker2d-r. Firstly, we trained an MLE model and a SDV model on the offline dataset and then used a random policy to run the simulation until reaching state s_{50} in the real MuJoCo environment. Next, we randomly sampled 10k actions from the action space. Finally, we used these actions to obtain the next states s'_{50} from both the real environment and the models, and mapped these two sets of s'_{50} to 2D using t-SNE [6] for comparison. If the distributions of the two sets of states are similar, their shapes after t-SNE processing should be similar; otherwise, the shapes will be different.

Figure 2 confirms that the models trained on the halfcheetah-r dataset is more accurate, while on the other two datasets, the deviation is significant, which verifies the above explanation. Furthermore, despite the low accuracy for the latter two tasks, the states predicted by the SDV model exhibit a stronger inclination towards areas with high rewards than those from the MLE model. This indicates that, even with a suboptimal dataset, the SDV approach can still enhance the likelihood of the agent reaching more favorable states.

6 Conclusion and Future Directions

In this work, we propose a simple yet effective model-based offline RL algorithm SDV inspired by human decision-making. It introduces an additional guidance model combined with model advantage weighting factors to provide more appropriate conservatism compared to previous works. In this way, the agent can be guided to more favourable states with reliable decisions. SDV demonstrated SOTA performance on the widely studied benchmark, and the ablation results indicated the crucial role the two key techniques play in its strong performance.

Meanwhile, our work introduces double validation and model advantage weighting into model-based offline RL, potentially providing new inspiration for future research, such as exploring the theoretical underpinnings of the effectiveness of double validation, investigating methods for automatically tuning hyperparameters, and integrating the concept of double validation into other tasks.

Acknowledgements

This work is partly supported by National College Student Innovation Training Program 2023, National Natural Science Foundation of China, under grants No. 61832005 and No. 62202238, China University Industry Research Innovation Foundation, under grant No. 2021FNA04005, and Natural Science Foundation of Jiangsu Province, under grant No. BK20200752.

References

[1] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al., 'Solving rubik's cube with a robot hand', *arXiv preprint arXiv:1910.07113*, (2019).

[2] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine, 'D4rl: Datasets for deep data-driven reinforcement learning', *arXiv preprint arXiv:2004.07219*, (2020).

[3] Scott Fujimoto, David Meger, and Doina Precup, 'Off-policy deep reinforcement learning without exploration', in *International conference on machine learning*, pp. 2052–2062. PMLR, (2019).

[4] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine, 'Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates', in *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 3389–3396. IEEE, (2017).

[5] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine, '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).

[6] Geoffrey E Hinton and Sam Roweis, 'Stochastic neighbor embedding', *Advances in neural information processing systems*, **15**, (2002).

[7] Toru Hishinuma and Kei Senda, 'Weighted model estimation for offline model-based reinforcement learning', *Advances in neural information processing systems*, **34**, 17789–17800, (2021).

[8] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine, 'When to trust your model: Model-based policy optimization', *Advances in neural information processing systems*, **32**, (2019).

[9] Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman, 'Mt-opt: Continuous multi-task robotic reinforcement learning at scale', *arXiv preprint arXiv:2104.08212*, (2021).

[10] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims, 'Morel: Model-based offline reinforcement learning', *Advances in neural information processing systems*, **33**, 21810–21823, (2020).

[11] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez, 'Deep reinforcement learning for autonomous driving: A survey', *IEEE Transactions on Intelligent Transportation Systems*, **23**(6), 4909–4926, (2021).

[12] Ilya Kostrikov, Rob Fergus, Jonathan Tompson, and Ofir Nachum, 'Offline reinforcement learning with fisher divergence critic regularization', in *International Conference on Machine Learning*, pp. 5774–5783. PMLR, (2021).

[13] Ilya Kostrikov, Ashvin Nair, and Sergey Levine, 'Offline reinforcement learning with implicit q-learning', *arXiv preprint arXiv:2110.06169*, (2021).

[14] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine, 'Stabilizing off-policy q-learning via bootstrapping error reduction', *Advances in Neural Information Processing Systems*, **32**, (2019).

[15] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine, 'Conservative q-learning for offline reinforcement learning', *Advances in Neural Information Processing Systems*, **33**, 1179–1191, (2020).

[16] Sascha Lange, Thomas Gabel, and Martin Riedmiller, 'Batch reinforcement learning', *Reinforcement learning: State-of-the-art*, 45–73, (2012).

[17] Byung-Jun Lee, Jongmin Lee, and Kee-Eung Kim, 'Representation balancing offline model-based reinforcement learning', in *International Conference on Learning Representations*, (2021).

[18] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu, 'Offline reinforcement learning: Tutorial, review, and perspectives on open problems', *arXiv preprint arXiv:2005.01643*, (2020).

[19] Cong Lu, Philip J Ball, Jack Parker-Holder, Michael A Osborne, and Stephen J Roberts, 'Revisiting design choices in offline model-based reinforcement learning', *arXiv preprint arXiv:2110.04135*, (2021).

[20] Jiafei Lyu, Xiu Li, and Zongqing Lu, 'Double check your state before trusting it: Confidence-aware bidirectional offline model-based imagination', *arXiv preprint arXiv:2206.07989*, (2022).

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al., 'Human-level control through deep reinforcement learning', *nature*, **518**(7540), 529–533, (2015).

[22] Marc Rigter, Bruno Lacerda, and Nick Hawes, 'Rambo-rl: Robust adversarial model-based offline reinforcement learning', *arXiv preprint arXiv:2204.12581*, (2022).

[23] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al., 'Mastering the game of go without human knowledge', *nature*, **550**(7676), 354–359, (2017).

[24] Richard S Sutton and Andrew G Barto, *Reinforcement learning: An introduction*, MIT press, 2018.

[25] Shengpu Tang and Jenna Wiens, 'Model selection for offline reinforcement learning: Practical considerations for healthcare settings', in *Machine Learning for Healthcare Conference*, pp. 2–35. PMLR, (2021).

[26] Jianhao Wang, Wenzhe Li, Haozhe Jiang, Guangxiang Zhu, Siyuan Li, and Chongjie Zhang, 'Offline reinforcement learning with reverse model-based imagination', *Advances in Neural Information Processing Systems*, **34**, 29420–29432, (2021).

[27] Yifan Wu, George Tucker, and Ofir Nachum, 'Behavior regularized offline reinforcement learning', *arXiv preprint arXiv:1911.11361*, (2019).

[28] Haoran Xu, Li Jiang, Li Jianxiong, and Xianyuan Zhan, 'A policy-guided imitation approach for offline reinforcement learning', *Advances in Neural Information Processing Systems*, **35**, 4085–4098, (2022).

[29] Chao Yu, Jiming Liu, Shamim Nemati, and Guosheng Yin, 'Reinforcement learning in healthcare: A survey', *ACM Computing Surveys (CSUR)*, **55**(1), 1–36, (2021).

[30] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn, 'Combo: Conservative offline model-based policy optimization', *Advances in neural information processing systems*, **34**, 28954–28967, (2021).

[31] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma, 'Mopo: Model-based offline policy optimization', *Advances in Neural Information Processing Systems*, **33**, 14129–14142, (2020).

[32] Zeyu Zhu and Huijing Zhao, 'A survey of deep rl and il for autonomous driving policy learning', *IEEE Transactions on Intelligent Transportation Systems*, **23**(9), 14043–14065, (2021).