Authors' Response to Reviews of ICLR

We would like to thank all the reviewers for their detailed comments. We summarize all the reviewers' concerns into three categories: 1. On the experimental details and the reproducibility of the proposed algorithm; 2. On the runtime cost and computational complexity of the algorithm. 3. On the algorithm's connection with several related works.

We have made a number of changes to address reviewers' suggestions and concerns. A short summary of the modifications made:

- Provide the source code of SfBC in the supplementary material to ensure reproducibility.
- Expand the implementation details in the Appendix B to provide more information on the implementation of our method (especially on the runtime and the sampling method).
- Increase the test seeds for Antmaze tasks from 20 to 100. Increase the independent experiment seed numbers from 3 to 4.
- Adopt a recent advance in diffusion-specialized ODE solver [1] to replace the standard 4th order Runge–Kutta solver for algorithm evaluation. Reduce sampling time required by three times.
- Conduct an ablation study on the diffusion steps used.
- Conduct an ablation study on the value iteration K.
- Add citations on several related works [2,3], and add a new section (Appendix G) to discuss the connection and difference between our method and two related methods (VEM [4] and EMaQ[2]).
- Fix several typos in the paper.

[1] DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps. https://arxiv.org/abs/2206.00927

[2] EMaQ: Expected-Max Q-Learning Operator for Simple Yet Effective Offline and Online RL. https://arxiv.org/abs/2007.11091

[3] Diffusion Policies as an Expressive Policy Class for Offline Reinforcement Learning. https://arxiv.org/ab

[4] Offline Reinforcement Learning with Value-based Episodic Memory. https://arxiv.org/abs/2110.09796

Response to Reviewer 1

Comment R1.1 *I believe the norm is to include the source code in the submission rather than a promise of future code.*

Response to R1.1 We thank the reviewer for this suggestion. We have submitted the source code in the supplementary material to ensure reproducibility.

Comment R1.2 *Ideally the results would be over more than 3 seeds, but this is not vital.*

Response to R1.2 Three random seeds is a shared evaluation metric adopted by many of our referenced baselines (D4RL, Diffuser, DT, etc.). Still, we have increased the random seeds used from 3 to 4 for MuJoCo and Antmaze tasks in order to clear the reviewer's concern. We have also increased the test seed number for all Antmaze tasks from 20 to 100 to remove possible randomness. After the update, the averaged performance for MuJoCo tasks almost doesn't change. The averaged performace for Antmaze tasks drop by 4 percent but still remains the highest among all referenced baselines. Note that the performance numbers for Kitchen and Maze2d have not been

updated due to the very limited time for rebuttal revision. We plan to update those numbers before camera-ready version of the manuscript.

Comment R1.3 *I did not see in Appendix C the number of diffusion steps used (quite an important detail).*
Response to R1.3 In the previous version of the paper, we use a standard 4th order Runge–Kutta ODE solver which requires 35 to 50 diffusion steps given different environments. In the updated manuscript, we are able to decrease the diffusion steps required to 15 steps in all tasks. This is because we adopt a recent advance in specialized diffusion sampler, namely DPM-solver [1]. We additionally conduct a series of experiments to study the influence of diffusion steps to the overall performance (Table A, summarized below). Generally speaking, we find that 10-25 diffusion steps result in similar overall performance.

[1] DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps. https://arxiv.org/abs/2206.00927

Comment R1.4 *It would also be helpful to discuss the runtime costs of sampling from diffusion models.*
Response to R1.4 For a concrete example, it takes 3.2s to evaluate our algorithm for every episode (1000 steps) in "halfcheetah-medium-v2" the task, when using 15 diffusion steps and 20 parallel environments (Hardware being a 2080Ti GPU). This runtime cost roughly matches the runtime of non-diffusion algorithms (diffusion step is 1) that doesn't use parallel evaluation. Although sampling from diffusion models is known to be slow, we use two techniques to make our method more experimentally practical.

1. We leverage recent advances of diffusion sampling method, which allows us to sample from the diffusion model with only 15 steps.

2. We implement a parallel evaluation scheme similar to [5]. Instead of sampling actions sequentially for multiple test seeds, we sample all actions in a single network inference by using a larger batchsize. This allows us to fully ultilize the parral computing power of GPUs and greatly accelerates action sampling.

We have conducted an ablation study to discuss the runtime costs of diffusion sampling, the detailed results is provided in Appendix C.

[5] Efficient parallel methods for deep reinforcement learning. https://arxiv.org/abs/1705.04862

Comment R1.5 *Is a closely related work that uses diffusion models for offline RL that should be mentioned in related work (although the method is not identical).*
Response to R1.5 We have mentioned and cited this work (Diffusion-QL) in Section 4. Diffusion-QL [2] and SfBC (our method) are two concurrent and independent lines of work. We have noticed that their work is also under submission to ICLR 2023.

In principle, it unnecessary for us to compare to a concurrent work. Still, we list three aspects from which the two works are easily distinguishable to assist understanding and clarify our method: 1. SfBC is theoretically derived from weighted regression methods where the behavior policy serves as an explicit prior of the final policy (e.g. AWAC, CRR), while Diffusion-QL is based on another line of methods where the behavior policy serves as an implicit regularization of the final policy (e.g. TD3+BC, BCQ). 2. SfBC additionally proposes an computationally efficient in-sample planning Q-operator to further avoid selecting OOD actions, while Diffusion-QL uses one-step bootstrapping. 3. SfBC samples from the learned policy using a rejection sampling technique,

while diffusion-QL uses an approximated guided sampling method. Besides, the implementation details of two algorithms are significantly different.

Response to Reviewer 2

Comment R2.1 *This paper lacks novelty. Policy improvement via weighted regression has been well studied in many papers, such as IQL and AWAC paper.*
Response to R2.1 We think the fact that weighted regression has been widely used and studied in many papers (e.g., IQL, AWAC) actually strengthened the novelty and the importance of our proposed method. This is because our method exactly aims to address existing problems of weighted regression methods instead of being one of them. We believe there is some misunderstanding here that needs to be clarified. Sorry that we may have not expressed the motivation of our algorithm clearly. Although our proposed method shares the same theoretical foundation with weighted regression methods, it is not one of them. one of our main contributions is that we observe existing weighted regression methods requires calculating exact log likelihood which limits policy expressivity. Such limits introduce unnecessary extrapolation error during Q-learning and may have been the bottleneck of existing algorithms. To address this problem, we introduce our method: sampling from behavior candidates in order to adopt a much more expressive generative behavior modeling method (diffusion models). Our method improves over existing weighted regression method in terms of policy expressivity and diversity (As visualized in a diverse-optimal-solution task that we propose in Figure 1 and Figure 3).

Comment R2.2 *Where the novelty of our proposed in-sample planning-based operator lies compared with the Q-learning method proposed by VEM (Ma et al. 2022) ?*
Response to R2.2 The proposed in-sample planning-based Q operator in our paper is by no means a trivial application of VEM [4]. We briefly summarize the novelty of our proposed method, and compare three key differences between VEM and our method:

First, a simplified version of $\mathcal{T}_{vem}$ is defined as:

$$\mathcal{T}_{vem}V(s) := \max_{0\leq n\leq c}\{(\mathcal{T}^{\mu})^n\mathcal{T}_{\mu}^{\tau}V(s)\}, \tag{1}$$

which is is built on $\mathcal{T}_{\mu}^{\tau}$, an expectile-based V-learning operator proposed by VEM:

$$\mathcal{T}_{\mu}^{\tau}V(s) := E_{a\sim\mu(\cdot|s)}\left\{\begin{array}{lll} \tau[r(s,a)+\gamma V(s')]+(1-\tau)V(s) & if & r(s,a)+\gamma V(s')\geq V(s) \\ (1-\tau)[r(s,a)+\gamma V(s')]+\tau V(s) & if & r(s,a)+\gamma V(s')< V(s) \end{array}\right\} \tag{2}$$

$\tau\in(0,1)$ is the hyperparameter for expectile operation.

1. VEM is built on an expectile operator $\mathcal{T}_{\mu}^{\tau}$, which means that it's only applicable to algorithms that use expectile regression (e.g., IQL), while our method has no such limitations because it's built on the current policy $\mathcal{T}^{\pi}$. As a result, we believe our proposed method is in a more general form. The theoretical property for the two operators are also different (See our proofs in Appendix B).

2. VEM requires comparing $V(s)$ and $r(s,a)+\gamma V(s')$ for theoretical derivation of the expectile operator $\mathcal{T}_{\mu}^{\tau}$, which cannot be applied to stochastic environments both in theory and in implementation. This is because $V(s')$ ($s'\sim P(\cdot|s,a)$) is a random variable while $V(s)$ is an arbitrary

scalar function, which is not comparable. In other words, $P(s'|s,a)$ and $r(s,a)$ have to be Dirac for VEM, but could be any distribution for our proposed method since we require no expectile operation.

3. VEM is a V-operator which learns a value function, while our method introduces a Q-operator which learns a Q function.

We notice that this part was not discussed in detail in the previous manuscript, and may cause some misunderstanding. We have strengthened this in the updated paper by adding a new section (Appendix F) in the updated paper to discuss the connection and the difference between two methods in more details.

[4] Offline Reinforcement Learning with Value-based Episodic Memory. https://arxiv.org/abs/2110.09796

Comment R2.3 *Applying diffusion models in offline RL is new while using diffusion models for behavior cloning seems an easy plug-in.*

Response to R2.3 Technically, applying diffusion models for behavior cloning is straightforward given that we need a much more expressive generative modeling method, but we believe this doesn't affect the empirical contributions of the paper. Although diffusion models have been successfully applied to image generation in recent years, there have been very limited research that studies the application of diffusion models in low-dimensional data space. Especially, diffusion models have not been previously used to model the behavior policy in reinforcement learning so there is few prior work that we could reference. As a result, we have made much efforts trying to design an appropriate network architecture, to search for suitable hyperparameters and to compare different sampling methods with a large amount of experiments. We have also made much efforts trying to reduce the training and inference time required (will be discussed in detail later) to make the algorithm more practical.

Besides, we believe keeping the our method simple can directly reveal the fact that policy expressivity is an important factor that introduces extrapolation error. We hope such a simple application of diffusion models could point to a new direction for offline RL research and serve as a neat baseline for related future works. For instance, our method could readily be combined with recent advances in energy-guided diffusion sampling methods to transform the behavior model to an expressive policy model and remove the need for rejection sampling.

.

Comment R2.4 *Moreover, by looking at the Algorithm 1, performing implicit in-sample planning has high computational complexity. There are two for loops, one for K and the other for N, which is dataset size and usually huge.*

Response to R2.4 The "two for loops" is a part of the critic training time, which is roughly 155 minutes for the "halfcheetah-medium-v2" task (1M data points) if we set value iteration K=5, so a total of 500 epochs and 122k gradient steps are considered. We think this runtime cost is acceptable since CQL roughly requires 4 hours of training for the same task (1M gradients) with the same hardware setup (a 2080Ti GPU). More specifically, "two for loops" refers to the update-value-target runtime and is about 6.3 minutes for each value target update. We think the main understanding gap here are due to three reasons:

1. K in the outer loop is a very small number (at max 5 for all tasks, mostly between 2 and 4).

2. The inner loop for N is actually not executed sequentially, but parallelized with a batchsize of 4096 to fully utilize the computing power of GPUs.

3. We require much less gradient steps compared with CQL because the training target for the

critic model is fixed (our method) instead of moving (CQL). This is one of the claimed contributions of our proposed method (fast contraction rate).

We notice that the previous pseudocode may easily cause such understanding gap. We thus thank the reviewer for the comment have updated the pseudocode (See Appendix A).

Comment R2.5 *Computational complexity is a concern and not discussed.*
Response to R2.5 The computational complexity has been discussed above (roughly matches several existing algorithms). For inference (evaluation) time, we manage to reduce the evaluating runtime to 3.2s per episode (1k action steps) when using 20 parallel environments and a diffusion step of 15. This is about the same time required for a non-diffusion algorithm if not using parallel evaluation techniques, making our algorithm practical.

Specifically, the main problem of SfBC is that the computational cost is expensive for algorithm evaluation because we require solving an ODE in order to get the final action. Theoretically SfBC requires at least D times of network inference times compared with non-diffusion methods, D being the diffusion steps. In our work, we make the following efforts in order to reduce the averaged evaluation runtime and make our algorithm practical:

1. We adopt a recent advance in diffusion-specialized ODE solver [2] to replace the standard 4th order Runge–Kutta solver and reduce to D from typically 35-50 to 10-15 in our work.

2. We implement a parallel evaluation scheme [5] that could help evaluating the algorithm under multiple test seeds at the same time, allowing us to significantly reduce the evaluation runtime by utilizing the parallel computing power of GPUs.

We have expanded experiment details to provide more experiments of runtime required both for algorithm training and evaluation under different diffusion steps and parallel environment numbers (Appendix C). Hopefully this could address the reviewer's concern.

[1] DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps. https://arxiv.org/abs/2206.00927

[5] Efficient parallel methods for deep reinforcement learning. https://arxiv.org/abs/1705.04862

Comment R2.6 *What is the number of timesteps for the diffusion model used in the paper? I guess an ablation study on the number timesteps is needed to show the effect of it.*
Response to R2.6 As we have discussed above, we adopt recent advances in diffusion sampling technique and use a uniform of 15 diffusion steps for the final performance of the paper. Generally we find that 10-25 diffusion steps yield similar performance, while a diffusion step of 5 would result in drastic performance degrade. We have provided an ablation study to compare the effect of using different diffusion steps in Appendix C (summarized below).

Comment R2.7 *Better provide a training curve in the appendix to show the training stability.*
Response to R2.7 We have provided the training curves under four different diffusion steps for MuJoCo and Antmaze tasks in Appendix C.

Comment R2.8 *The policy network is much larger than the current policy networks, which makes the computational cost expensive and comparison not very fair.*
Response to R2.8 SfBC requires using a more expressive policy, this partly means a network of high capacity and more parameters, which we think is a trade-off between policy expressivity and computational cost. We believe the comparison with referenced baselines is fair for the following reasons:

1. We have included two baselines that also use high-capacity networks for comparison. Concretely, DT (NIPS 2021, [6]) uses transformers which has 0.6-1.5M parameters; Diffuser (ICML 2022 Oral, [7]) uses an architecture that combines diffusion models with the transformer architecture which nearly has 4M parameters. The parameters used in SfBC is approximately 1.6M for all tasks. However, the FLOPs required by our method is more than 10 times less than these sequential-based methods because we do not operate in the trajectory level.

2. Other referenced baselines which use a much smaller policy network are not likely to have any performance improvement if switching to a high-capacity model. In other words, their reported results have mostly been finetuned. We might be able to run a few experiments on one or two classic offline RL algorithms with our network architecture later to prove this point.

Comment R2.9 *Expect to an ablation study on the K, since it is a key parameter in Algorithm 1.*
Response to R2.9 We have provided an ablation study on $K \in \{1, 2, 3, 4, 5\}$ in Appendix C.

Comment R2.10 *For the reported values in Table 1, what are these values? Are them the statistics based on the last points after the training or the highest points during the training?*
Response to R2.10 The are statistics based on the last points.

Comment R2.11 *Minor: typo in Equation (12), I think it is* R_{n+1}^{k-1}
Response to R2.11 We sincerely thank the reviewer for the careful reading. The typo is fixed now, though we think the superscript should be k instead of $k-1$.

Response to Reviewer 3

We thank the reviewer for the very constructive comments by mentioning a very important related work EMaQ, which we have missed during the literature-search phase. Before we give a point-to-point response to the reviewer, we think it would be helpful to first present a brief review of EMaQ. A more detailed review and comparison is presented in Appendix C in the updated manuscript.

EMaQ is built upon BCQ [Fujimoto et al., 2019], which computes the training target in Q-Learning by:

$$\mathcal{T}^*_{BCQ}Q(s,a) := r(s,a) + \gamma \max_{a' \sim \mu_\theta(\cdot|s')} \{Q(s', a' + \xi_\phi(s', a'))\}, \tag{3}$$

where $\xi_\phi(s,a)$ is an explicitly constrained perturbation network that helps relax the constraint of behavior policy μ_θ.

The core motivation for EMaQ is to remove the perturbation model $\xi_\phi(s,a)$, by taking max over N Q-function evaluations:

$$\mathcal{T}_{EMaQ}Q(s,a) := r(s,a) + \gamma E_{\{a'_i\}^N \sim \mu_\theta(\cdot|s')}[\max_{a'_i \in \{a'_i\}^N} Q(s', a'_i)]. \tag{4}$$

For EMaQ, N serves as a hyperparameter to interpolate $\mathcal{T}^\mu$ and $\mathcal{T}^*$. When $N = 1$, $\mathcal{T}_{EMaQ}$ becomes $\mathcal{T}^\mu$. When $N \to \infty$, $\mathcal{T}_{EMaQ}$ approaches $\mathcal{T}^*$ because $\{a'_i\}^N$ nearly covers the whole action space.

In contrast, for SfBC, the hyperparameter N is the number of Monte Carlo samples used to estimate the training Q-targets:

$$\mathcal{T}^\pi_{SfBC}Q(s,a) \approx r(s,a) + \gamma \sum_N \left[\frac{\exp(\alpha Q(s',a'))}{\sum_N \exp(\alpha Q(s',a'))} Q(s',a')\right] \tag{5}$$

Although $\mathcal{T}^{\pi}$ may look like a softmax version of EMaQ, which is only briefly mentioned but not studied in detail in Appendix J in [1], it is not an interpolation of $\mathcal{T}^{\mu}$ and $\mathcal{T}^{*}$. Changing N only influence the variance of the Monte Carlo estimate for $\mathcal{T}^{\pi}Q(s,a)$, but does not affect the fixed point of $\mathcal{T}^{\pi}$.

Besides, SfBC uses a multi-step planning-based operator $\mathcal{T}_{\mu}^{\pi}Q(s,a) := \max_{n\geq 0}\{(\mathcal{T}^{\mu})^{n}\mathcal{T}^{\pi}Q(s,a)\}$, while EMaQ only considers one-step bootstrapping.

During inference, both methods select actions from candidates genererated by a behavior model, although EMaQ adopts an autoregressive masked autoencoder architecture for behavior modeling while SfBC uses diffusion models. However, the focus and the claimed contribution of EMaQ and SfBC are almost entirely different. EMaQ focuses on studying how the choices of hyperparameter N affect the performance of the algorithm, and only mentions that they empirically find an expressive policy model yield good performance when explaining why they choose an autoregressive masked autoencoder architecture (MADE) over VAE in the Experiment section (Section 3.5). In contrast, SfBC makes much more efforts discussing how offline RL may benefit from a much more expressive behavior policy model.

Comment R3.1 *The related work section misses some important prior work in particular, which also introduces a similar approach.*

Response to R3.1 We appreciate the reviewer for this very insightful comment. EMaQ is indeed a very important related work, which we have missed during the literature-search phase. We now have mentioned EMaQ in the Introduction Section, Method Section, and Related works Section in our updated manuscript. We have also added a new section in the Appendix to discuss in more detail about the connection and difference between the two works.

Comment R3.2 *The approach has a limited novelty. Expressive policies have been previously studied in EMAQ.*

Response to R3.2 EMaQ and our research both find that offline RL could benefit greatly from an expressive behavior policy. However, we believe it would be more appropriate to state that expressive policies have been discussed or mentioned (instead of studied) in EMaQ, so the focus and claimed contribution of EMaQ and our method are different. We have a few observations to support our claims:

1. EMaQ neither mentioned expressive policies (or the MADE architecture) in the Abstract Section nor in the Introduction Section in its manuscript. Policy expressivity is only briefly talked about in the Section 3.5 on the methodology. On the other hand, the main focus of our research is to discuss and explain how the policy expressivity may affect the performance of algorithms.

2. EMaQ actually concentrates on using models with good inductive bias instead of high expressivity, as we quote: "We do not believe that autoregressive models are intrinsically better than VAEs", and "Our results suggest that for a given domain, focusing efforts on building-in good inductive biases in the generative models and value functions might be sufficient to obtain strong offline RL performance in many domains".

3. The focus of EMaQ's experiments is to demonstrate how the hyperparameter N affects the performance of the algorithm instead of how an expressive policy affects the performance of the algorithm, though they have compared with a VAE-based baseline (BEAR) to support their architecture choices. In contrast, we aim to visualize the importance of expressive policies by designing new tasks and compare our method to other baselines as well as other generative models in details (Figure 1, 3, Table 2, Section 3.1, 6.2, 6.3).

In short, we think EMaQ is an important related work but doesn't affect the novelty of our proposed method.

Comment R3.3 *Moreover, the value computation technique is similar to a soft version of EMAQ.*
Response to R3.3 We think the value computation technique used in our method and EMaQ only bear limited similarity, and the similarity lies only in the empirical implementation part which was not claimed to be a contribution of our method:

1. The core contribution of our proposed planning-based in-sample Q-operator is that we combines multi-step Q learning with one-step bootstrapping which provably ensures policy improvement while still guarantees a fast-contraction rate. However, EMaQ only considers one-step bootstrapping.

2. As we have described in the background of EMaQ above, although EMaQ and our method both use N samples to calculate the Q-targets, the N used for EMaQ aims to interpolate between $\mathcal{T}^{\mu}$ and $\mathcal{T}^{*}$ while it's only a Monte Carlo sample number for our proposed method. For experiments, one of the main focuses of EMaQ is to study how N affects the property of the backup Q-operator and performance of the algorithm, while two hyperparameters N and α are fixed to constants for all tasks in our research. The soft version of EMaQ does not change the theoretical differences between EMaQ and our proposed method. Besides, it is only briefly discussed in Appendix J of EMaQ's manuscript with no further experiments or analysis.

Comment R3.4 *At the end of section 4.2, it's mentioned that the n-step operator proposed in this work works in the stochastic setting. However, this statement is not completely clear. For example, does it mean that it's unbiased?*
Response to R3.4 No, we make this statement because a prior method VEM[1] which inspires our method is derived under the assumption that the environment is deterministic ($r(s, a)$ and $P(\cdot|s, a)$ are both Dirac) and could not be applied to stochastic environments. This is because VEM requires a max operation "$\max(V(s), r(s, a) + \gamma V(s'))$", and $V(s')$ would be a random variable if $s' \sim P(\cdot|s, a)$ while $V(s)$ is a scalar function. Another reviewer also has similar questions. We therefore thank the reviewer for bringing up this question, making us realise that we should have provided enough information on the background. We have now added a new section to talk about the connection between VEM and our proposed planning-based n-step operator in details in Appendix C.

Comment R3.5 *It is not clear how the method is implemented, and the paper can be significantly improved by expanding the implementation details ... it might be difficult to reproduce the results in the paper.*
Response to R3.5 We have made the following adjustments to improve the clarity and ensure reproducibility of our paper:

1. We have submitted our source code in the supplementary material to ensure reproducibility.

2. We have increased independent seeds (3 to 4) for MuJoCo environments and test seeds (20 to 100) for the Antmaze environments.

3. We have expanded the provided implementation details in Appendix C (especially on computational cost, sampling method and evaluation metrics).

4. We have added a new hyperparameter table (Table C) to list all hyperparameters used in our experiments.

5. We have rewritten the pseudocode in Appendix A to include more implementation details.

6. We have provided ablation studies for two important hyperparameters (value iteration K and diffusion step D) and provided training curves of our algorithm.

We hope we can clear the reviewer's concern and clarify our algorithm's implementation. If

the reviewer has further questions or more specific suggestions, we would be very happy to address them.

References

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062. PMLR, 2019.