Guided Flow Policy: Workshop Version

Franki Nguimatsia Tiofack[1,*], Theotime Le Hellard[1,*], Fabian Schramm[1,*], Nicolas Perrin-Gilbert[2], Justin Carpentier[1]

***Abstract*—Offline reinforcement learning often relies on constraint optimization techniques that enforce policy improvement to remain close to the behavior policy. However, such approaches fail to distinguish between high-value and low-value actions in their penalty components. We introduce Guided Flow Policy (GFP), which couples a multi-step flow-matching policy with a distilled one-step actor. The actor directs the flow policy through weighted behavior cloning to focus on cloning high-value actions from the dataset rather than indiscriminately imitating all state-action pairs. In turn, the flow policy constrains the actor to remain aligned with the dataset's best transitions while maximizing the critic. This mutual guidance enables GFP to achieve state-of-the-art performance across 144 tasks from the OGBench, Minari, and D4RL benchmarks, with substantial gains on suboptimal datasets and challenging tasks.**

***Index Terms*—Reinforcement Learning, Behavior Cloning, Flow Matching, Penalty Method**

I. Introduction

Offline Reinforcement Learning (RL) aims to learn policies from static datasets without further interaction with the environment [1], [2]. This paradigm is suited to robotics settings where online exploration can be unsafe or costly. However, standard online off-policy algorithms such as DDPG [3] and SAC [4], tend to underperform in offline settings since the RL agent cannot interact with the environment. The main challenge is extrapolation error, corresponding to the inability to properly evaluate out-of-distribution (OOD) actions [5]–[8].

Two main lines of work have been proposed to address this challenge. The first one focuses on learning a critic without querying the values of actions outside the dataset [9], [10]. The second one, known as the Behavior-Regularized Actor–Critic (BRAC) family, mitigates these errors by forcing the learned policy to stay "close" to the unknown behavior policy that generated the dataset [5], [11]–[14]. The key idea is that OOD state–action pairs are especially vulnerable to Q-value overestimation, so staying near the empirical distribution reduces extrapolation errors. This is achieved either using constrained optimization or by adding a behavior cloning penalty to regularize the policy toward some dataset actions. This approach raises a trade-off: regularizing too strictly to a potentially suboptimal dataset action may restrict the policy from exploiting higher-reward actions contained in the dataset.

Until recently, most offline RL algorithms were based on Gaussian policies. Recent development of flow and diffusion-based expressive models [15]–[17], led to new RL algorithms to capture complex and multimodal action distributions [18]–[21]. However, they come at the risk of high computational overhead: iterative sampling slows inference, and directly optimizing the values of output actions would result in unstable backpropagation through time (BPTT). To address these challenges, [22] proposed a BRAC method with a flow-matching BC model distilled into a one-step policy that also optimizes the critic, enabling expressive policy learning while avoiding BPTT and iterative sampling at inference. Still, a central limitation remains: the flow-based BC component, similar to standard BC, does not incorporate reward information.

We propose **Guided Flow Policy (GFP)**, a dual-policy BRAC framework with a bidirectional guidance mechanism between a multi-step flow-matching policy, termed Value-aware Behavior Cloning (VaBC), and a distilled one-step actor. VaBC acts as a distributional regularizer for the actor, encouraging it to remain within the support of the behavior policy. VaBC is trained via a weighted-BC mechanism, close to [21], [23], but leveraging the actor and the critic to prioritize cloning high-value actions from the dataset. Unlike previous BRAC approaches, in which the BC regularization indiscriminately treats all state-action pairs [11], [22], VaBC is a value-aware regularizer. In turn, the actor optimizes the critic while being distilled toward VaBC, allowing it to align with the dataset's high-value actions in a given state while maximizing expected returns. Fig. 1 illustrates the GFP framework, Tab. I shows how it differs in the regularization part compared to other BRAC methods, and Fig. 2 is a broader overview of related works.

We extensively evaluate GFP on **144 tasks** from offline RL benchmarks, showing strong performances with substantial gains on suboptimal datasets and challenging tasks. This workshop paper is based on a larger work accepted at ICLR 2026. For more details, please refer to the full paper [24].

TABLE I: **Overview of regularization mechanisms within the BRAC framework.**

	Regularization target	Value-aware regularization	Expressive variant
TD3+BC [11]			
ReBRAC [12]	↪ dataset actions	✗	Diffusion-QL [20]
FQL [22]	↪ learned BC policy	✗	✓
GFP (ours)	↪ learned value-aware BC policy	✓	✓

II. Background

Actor-critic framework. RL problems are formalized as a Markov Decision Process $(\mathcal{S}, \mathcal{A}, p, r, \rho, \gamma)$ [25], [26]. Here, $\mathcal{S}$ denotes the state space, $\mathcal{A}$ the action space, p the transition dynamics, r the reward function, ρ the initial state distribution,

*Equal contribution

[1]Inria and Département d'Informatique de l'École Normale Supérieure, PSL Research University, Paris, France

[2]Sorbonne Université, CNRS, Institut des Systèmes Intelligents et de Robotique, ISIR

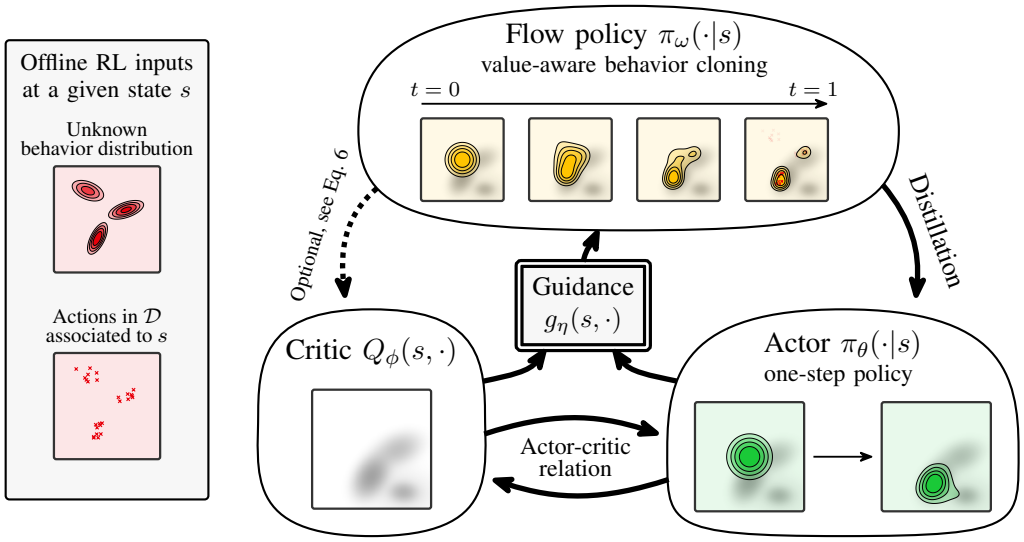


Fig. 1: **Overview of Guided Flow Policy.** GFP consists of three components: (i) in yellow, VaBC, a multi-step flow policy π_ω trained via weighted BC using the guidance term g_η, (ii) in green, a one-step actor π_θ distilled from the flow policy, and (iii) in gray, a critic Q_ϕ guiding action evaluation. π_ω regularizes the actor toward high-value actions from the dataset $\mathcal{D}$; in turn, the actor shapes the flow and optimizes the critic following the actor–critic approach. Each drawing represents the probability distribution of actions $a \in \mathcal{A}$ of a policy, in a current state s, except for the gray ones, where it is the value of actions $a \in \mathcal{A}$ in state s, according to the critic.

and $\gamma \in [0,1)$ the discount factor. The agent is governed by a policy π, mapping states to probability distributions over actions. The objective is to maximize the expected discounted return $\mathbb{E}_{a_t\sim\pi(\cdot|s_t)}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\right]$. In actor-critic approaches, the policy π, referred to as the actor, is trained jointly with a critic Q, which approximates the state–action value function: $Q^\pi(s,a) = \mathbb{E}_{a\sim\pi(\cdot|s)}\left[\sum_{t=0}^{\infty}\gamma^t r\left(s_t,a_t\right) \mid s_0=s, a_0=a\right]$, estimating the expected return after taking action a in state s and subsequently following π. Both actor and critic are parametrized as neural networks, with parameters θ and ϕ, and optimized by alternating gradient descent steps on:

$$\mathcal{L}^{\mathcal{A}}(\theta) = \mathbb{E}_{\substack{s\sim\mathcal{D},\\ a_\theta\sim\pi_\theta(\cdot|s)}}\big[-Q_\phi(s,a_\theta)\big], \tag{1}$$

$$\mathcal{L}^{\mathcal{C}}(\phi) = \mathbb{E}_{\substack{(s,a,r,s')\sim\mathcal{D}\\ a'\sim\pi_\theta(\cdot|s')}}\Big[\big(Q_\phi(s,a) - (r+\gamma Q_{\bar{\phi}}(s',a'))\big)^2\Big] \tag{2}$$

$\mathcal{L}^{\mathcal{A}}$ and $\mathcal{L}^{\mathcal{C}}$ are the actor and critic losses, and $\mathcal{D}$ is a set of transitions (s,a,r,s'). $Q_{\bar{\phi}}$ is a target Q-function parameterized by a slow Polyak averaging update of weights $\bar{\phi}$.

Constrained optimization in offline RL. In offline RL, the agent learns exclusively from a static dataset $\mathcal{D}$, consisting of transitions (s,a,r,s') generated by an unknown behavior policy π_β. At a state s, the distribution of actions of such π_β is illustrated on the left of Fig. 1. This introduces a key challenge compared to the online setting: the distributional shift [1], [7], [26]. The value estimates of OOD actions can be inaccurate, and a default actor may select actions outside the dataset's support. The BRAC approach constrains the policy π_θ to remain close to the behavior policy [7], [13], e.g.,

$$\max_{\pi}\ \mathbb{E}_{\substack{s\sim\mathcal{D}\\ a_\theta\sim\pi_\theta(\cdot|s)}}[Q_\phi(s,a_\theta)] \quad \text{s.t. } d\big(\pi(\cdot|s),\pi_\beta(\cdot|s)\big)\le\varepsilon \tag{3}$$

As emphasized in [11], a simple and effective relaxation of Eq. 3 consists of adding a BC term directly into the actor objective:

$$\mathcal{L}^{\mathcal{A}}(\theta) = \mathbb{E}_{\substack{(s,a)\sim\mathcal{D},\\ a_\theta\sim\pi_\theta(\cdot|s)}}\Big[-Q_\phi(s,a_\theta) + \alpha\overbrace{\|a_\theta - a\|^2}^{\text{BC term}}\Big] \tag{4}$$

where α is a hyperparameter that balances between exploiting high Q-values and staying close to the behavior policy. This objective encourages actions that both achieve high-expected returns and remain within the support of the dataset.

Behavior cloning with flow matching. Flow Matching [15] is a generative modeling framework that learns a transformation, or flow, mapping a simple base distribution (e.g. a standard Gaussian) to a target data distribution. This transformation is defined through intermediate time-dependent distributions governed by an ordinary differential equation (ODE).

In the context of BC, the flow models the behavior policy π_β underlying the dataset $\mathcal{D}$. This is achieved by learning a state and time dependent velocity field $v_\omega : [0,1]\times\mathcal{S}\times\mathbb{R}^d \to \mathbb{R}^d$ that governs the dynamics of a flow, where d is the action dimension. We adopt the simplest optimal transport variant of conditional flow matching [27], using linear interpolation with uniformly sampled times [15]. For $(s,a)\sim\mathcal{D}$, $\epsilon\sim\mathcal{N}(0,I_d)$, and $t\sim\mathcal{U}([0,1])$, the interpolated point is $a_t = (1-t)\epsilon + ta$, and target velocity is $a-\epsilon$. The velocity field v_ω is trained by least-squares regression toward this reference, yielding the conditional flow-matching BC loss:

$$\mathcal{L}^{\text{FM-BC}}(\omega) = \mathbb{E}_{\substack{(s,a)\sim\mathcal{D},\\ \epsilon\sim\mathcal{N}(0,I_d),\\ t\sim\mathcal{U}([0,1])}}\Big[\|v_\omega(t,s,a_t) - (a-\epsilon)\|_2^2\Big]. \tag{5}$$

Once trained, the flow $\psi_\omega : [0,1]\times\mathcal{S}\times\mathbb{R}^d \to \mathcal{A}$ that follows v_θ through the ODE, represents a behavior cloning policy π_ω approximating the behavior policy π_β. At inference, from state s, an action is obtained by sampling randomly $z\sim\mathcal{N}(0,I_d)$ and integrating the flow from 0 to 1 using an explicit Euler method. We denote by $\mu_\omega(s,z) := \psi_\omega(1,s,z)$ the value of the integrated flow at time 1.

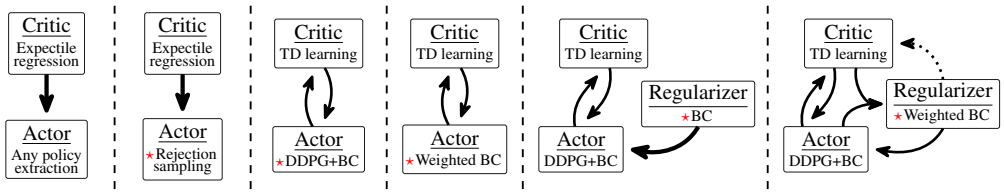


Fig. 2: **Overview of some offline-RL frameworks.** The symbol ⋆ indicates the use of a diffusion or a flow model, and specifically in which component. Each box, provides the name of the component (e.g. critic) and the working principle that is used to train it (e.g. TD learning). The arrows indicate how the components depend on each other, while the dashed arrow is optional (see Eq. 6)

Flow policy for offline RL. Following the idea of Diffusion Q-Learning by [20], for offline RL, one can directly replace the BC term in the actor loss (Eq. 4) with the flow-matching BC loss (Eq. 5). However, the iterative sampling procedure makes training expensive, due to recursive backpropagation through time (BPTT) in the actor loss, and also results in a slow inference at test time. To mitigate these limitations, [22] suggests distilling the iterative flow-matching BC policy into a one-step policy that also maximizes the critic, see Fig. 2e.

III. Guided Flow Policy

We now detail the GFP algorithm that builds on top of [11], [22]. GFP integrates a Value-aware Behavior Cloning (VaBC) flow policy with a distilled one-step actor through bidirectional guidance. VaBC leverages the actor and the critic to selectively clone high-value dataset actions, providing more targeted regularization than in standard BRAC approaches. The distilled actor, in turn, maximizes the critic while avoiding BPTT and iterative sampling. GFP is composed of three main components: the critic Q_ϕ, the actor π_θ, and the VaBC policy π_ω. The approach is illustrated in Fig. 1, and we refer to the full paper [24] for the detailed algorithm.

Step 1 – Learning the critic Q_ϕ. The critic is trained using the Bellman mean-squared loss Eq. 2. $y(s,r,s') := r + \gamma Q_{\bar{\phi}}(s',a')$ corresponds to the standard Bellman target in actor-critic methods, which we use by default in this work. Yet, since VaBC is designed to prioritize cloning the most promising dataset actions for a given state, we have also considered a more conservative variant of the Bellman target:

$$y^{\text{VaBC}}(s,r,s') = r + \frac{\gamma}{2}\Big(Q_{\bar{\phi}}(s',\mu_\theta(s',z)) + Q_{\bar{\phi}}(s',\mu_\omega(s',z))\Big) \tag{6}$$

where $\mu_\theta(s',z)$ denotes the action sampled from the actor and $\mu_\omega(s',z)$ the action from the VaBC policy, from noise $z \sim \mathcal{N}(0,I_d)$. The Bellman target $y^{\text{VaBC}}(s,r,s')$ averages two estimates of the Q-value: $Q_{\bar{\phi}}(s',\mu_\theta(s',z))$ which can overestimate the real Q-value; and $Q_{\bar{\phi}}(s',\mu_\omega(s',z))$ which can underestimate it. This choice can lead to substantial performance improvements in certain situations, as further studied in the full paper [24].

Step 2 – Learning the actor π_θ. The actor π_θ is trained by behavior regularized policy gradients, to maximize the Q-function while distilling the distribution of valuable dataset actions learned by the flow policy π_ω:

$$\mathcal{L}^{\mathcal{A}}(\theta) = \mathbb{E}_{\substack{s\sim\mathcal{D}\\ z\sim\mathcal{N}(0,I_d)}}\Big[-Q_\phi\left(s,\mu_\theta(s,z)\right) + \frac{\alpha}{\lambda}\|\mu_\theta(s,z)-\mu_\omega(s,z)\|_2^2\Big] \tag{7}$$

The normalization term $\lambda = \frac{1}{\frac{1}{N}\sum|Q_\phi(s,a)|}$ is estimated over mini-batches. The actor learns to select actions that maximize return while avoiding OOD, the distillation term constrains it to remain near the support of high-value dataset behaviors.

Step 3 – Learning the flow policy π_ω. The VaBC policy π_ω is optimized via a weighted flow-matching behavior cloning:

$$\mathcal{L}^{\text{VaBC}}(\omega) = \mathbb{E}_{\substack{(s,a)\sim\mathcal{D},\\ \epsilon\sim\mathcal{N}(0,I_d),\\ t\sim\mathcal{U}([0,1])}}\Big[g_\eta(s,a)\,\|v_\omega(t,s,a_t) - (a-\epsilon)\|_2^2\Big] \tag{8}$$

where

$$g_\eta(s,a) = \frac{\exp(\frac{\lambda}{\eta}Q_\phi(s,a))}{\exp(\frac{\lambda}{\eta}Q_\phi(s,a)) + \exp(\frac{\lambda}{\eta}Q_\phi(s,\mu_\theta(s,z)))} \tag{9}$$

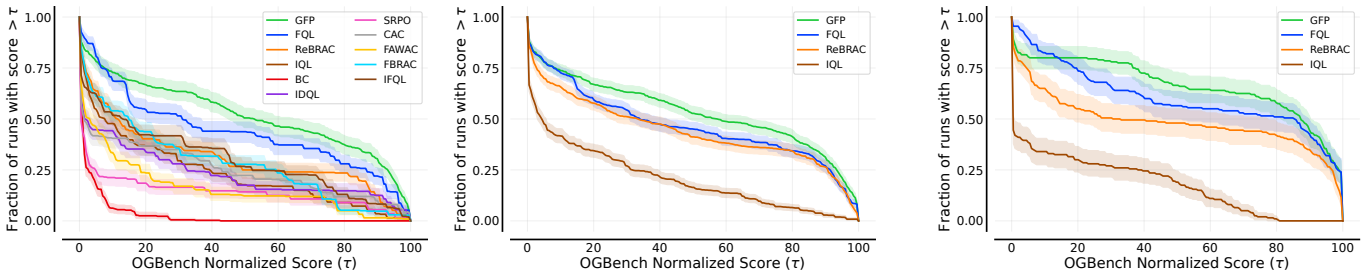
Intuitively, for a state-action pair (s,a) sampled from $\mathcal{D}$, $g_\eta(s,a)$ compares the quality between the dataset action a and a proposal of the actor $\mu_\theta(s,z)$ in a soft-max approach. If the dataset action has a higher Q-value, then $g_\eta(s,a) > 0.5$, placing greater emphasis on cloning it. Conversely, if the dataset action is worse, $g_\eta(s,a) < 0.5$ and it reduces its influence. This ensures that VaBC selectively clones high-value dataset behaviors. This makes sense because the actor itself is constrained to remain close to the dataset's action distribution. Importantly, VaBC is learned jointly with the actor and the critic, not beforehand.

λ is the same Q-normalization factor used in the actor loss, ensuring consistent scaling across components. $\eta > 0$ is a temperature hyperparameter that controls the sharpness of the filtering. Small η makes $g_\eta(s,a)$ more selective, shifting the policy from broadly imitating the dataset to emphasizing higher-value actions. η is tuned to achieve the best trade-off, prioritizing promising actions while preserving diversity.

The key contribution of GFP is to add value-awareness in the behavior regularization component of a BRAC framework, effectively combining the two predominant policy extraction methods studied by [28]: weighted-BC and behavior regularized policy gradients, illustrated Fig 2d and Fig 2c respectively. Note that different weighting functions g_η could be tested as alternatives to the soft-max Eq 9. An ablation study is provided in the full paper [24] using an advantage weighted term, similar to AWR [23] but using the actor to compute a baseline.

IV. Experiments

We conducted extensive experiments over a suite of robot locomotion and manipulation tasks, spanning three bench-



(a) (b) (c)

Fig. 3: **OGBench analysis.** (a) Performance profiles for 50 tasks comparing GFP against a wide range of prior works, showing the fraction of tasks where each algorithm achieves a score above threshold τ, using the evaluation reported by [22]. (b) Performance profiles on 105 tasks, including more challenging ones, and carefully reevaluated prior methods. (c) Performance profiles restricted to 30 noisy and explore tasks.

TABLE II: **Offline RL results.** Due to space constraints, we highlight only a few aggregated results. See [24] Tabs 2, 8, 9, 10, 11, and 12, for full results.

Task Category	**Offline RL algorithms**			
	IQL	ReBRAC	FQL	**GFP** actor π_θ
OGBench antmaze-giant-navigate-singletask (5 tasks)	*4* ± 1	**33.2** ± 5.7	16.3 ± 8.2	27.9 ± 8.5
OGBench humanoidmaze-medium-navigate-singletask (5 tasks)	*33* ± 2	59.2 ± 12.1	*58* ± 5	**72.0** ± 2.8
OGBench humanoidmaze-large-navigate-singletask (5 tasks)	*2* ± 1	12.9 ± 4.2	6.5 ± 2.7	**17.8** ± 9.6
OGBench cube-triple-play-singletask (5 tasks)	0.1 ± 0.1	2.9 ± 1.2	3.9 ± 1.5	**15.9** ± 2.0
OGBench cube-triple-noisy-singletask (5 tasks)	4.8 ± 1.2	5.2 ± 2.9	3.5 ± 1.6	**24.5** ± 2.8
OGBench puzzle-4×4-play-singletask (5 tasks)	*7* ± 1	17.1 ± 1.3	*17* ± 2	**26.1** ± 2.1
Average OGBench (105 tasks)	20.4	43.9	46.7	**53.2**
Average D4RL (18 tasks)	54.0	**64.8**	**62.1**	**63.0**
Average Minari (21 tasks)	–	–	65.9	**74.1**

marks: D4RL [29] (AntMaze and Adroit; 18 tasks), its successor Minari [30] (Gym-Mujoco Hopper, HalfCheetah, and Walker, and Adroit; 21 tasks) and the more challenging OGBench [31] reward-based tasks [22] (9 locomotion and 11 manipulation environments, each with 5 tasks; in total 100 state-based tasks, and 5 pixel-based tasks). Overall, we benchmark GFP on 144 tasks, our Jax implementation is available at github.com/Simple-Robotics/guided-flow-policy. Comparisons are synthesized in performance profiles presented Fig. 3, following [32], and some aggregated results are reported Tab. II.

Comparison against many previous works. Fig. 3a we first compare GFP against 10 prior methods on the 50 OGBench tasks reported by [22]. GFP clearly stands out compared to all these prior works. Results against 6 methods on D4RL and first evaluations on Minari are available in the full paper.

Extensive study on 144 tasks. GFP is further evaluated against 3 baselines: (i) FQL [22], as it comes second only to GFP on the first 50 tasks; (ii) ReBRAC [12], as we were able to improve its performance compared to previously reported results substantially by adjusting implementation details, and (iii) IQL [9] to represent in-sampling approaches. Tab. II summarizes our results, together with performance profile plots Figs. 3b and 3c, it confirms that GFP achieves state-of-the-art performance, with particularly substantial gains on noisy and challenging environments. For instance, on the cube-double-noisy and cube-triple-noisy datasets, GFP achieves an average score of 63 and 24, respectively, compared to 38 and 4 for FQL, and 20 and 5 for ReBRAC. Similarly, GFP stands out in some very challenging locomotion tasks, such as humanoidmaze-large-navigate (18 vs. 7 for FQL and 13 for ReBRAC), and manipulation tasks, such as cube-triple-play (16 vs. 4 for FQL and 3 for ReBRAC).

Additional experiments. We refer interested readers to the full paper [24] for ablation studies (e.g. on the modified Bellman target and an AWR guidance term), additional analyses (e.g. on η and α) and details on how we improved the performance of ReBRAC. Besides, while π_θ is GFP primary policy, we also evaluated the VaBC policy π_ω as a byproduct.

V. Conclusion

Guided Flow Policy (GFP) couples a multi-step flow-matching policy trained with value-aware behavior cloning and a distilled one-step actor through a bidirectional guidance mechanism. GFP leverages the expressiveness of flow policies while adding value awareness directly in the flow part, without the drawbacks of backpropagation through time. Our extensive study on 144 offline RL tasks demonstrates GFP effectiveness. Nonetheless, GFP depends on the availability of a sufficiently accurate critic. Future research directions could explore ways to reduce reliance on the critic or extend GFP to settings with weaker or sparse reward signals.

ACKNOWLEDGMENTS

This work has received support from the French government, managed by the National Research Agency, under the France 2030 program with the references Organic Robotics Program (PEPR O2R) and "PR[AI]RIE-PSAI" (ANR-23-IACL-0008). This research was funded, in part, by l'Agence Nationale de la Recherche (ANR), projects NIMBLE project (ANR-22-CE33-0008), RODEO (ANR-24-CE23-5886), PEPR O2R - AS2 (ANR-22-EXOD-0006), and PEPR O2R – PI3 ASSISTMOV (ANR-22-EXOD-0004). The European Union also supported this work through the ARTIFACT project (GA no. 101165695) and the AGIMUS project (GA no. 101070165). The Paris Île-de-France Région also supported this work in the frame of the DIM AI4IDF. The authors gratefully acknowledge the support and resources provided by the CLEPS infrastructure at Inria Paris. This work was performed using HPC resources from the GENCI-IDRIS Jean-Zay cluster (Grant 2024-AD010616763). Views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the funding agencies.

REFERENCES

[1] M. R. S. Lange, T. Gabel, "Batch reinforcement learning. in: M. wiering, m. van otterlo (eds) reinforcement learning." *Adaptation, Learning, and Optimization*, vol. 12, no. 3, p. 729, 2012.

[2] D. Ernst, P. Geurts, and L. Wehenkel, "Tree-based batch mode reinforcement learning," *Journal of Machine Learning Research*, vol. 6, 2005.

[3] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

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

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

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

[7] A. Kumar, J. Fu, M. Soh, G. Tucker, and S. Levine, "Stabilizing off-policy q-learning via bootstrapping error reduction," *Advances in neural information processing systems*, vol. 32, 2019.

[8] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," *Advances in neural information processing systems*, vol. 33, pp. 1179–1191, 2020.

[9] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," *arXiv preprint arXiv:2110.06169*, 2021.

[10] A. Nair, A. Gupta, M. Dalal, and S. Levine, "Awac: Accelerating online reinforcement learning with offline datasets," *arXiv preprint arXiv:2006.09359*, 2020.

[11] S. Fujimoto and S. S. Gu, "A minimalist approach to offline reinforcement learning," *Advances in neural information processing systems*, vol. 34, pp. 20 132–20 145, 2021.

[12] D. Tarasov, V. Kurenkov, A. Nikulin, and S. Kolesnikov, "Revisiting the minimalist approach to offline reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 11 592–11 620, 2023.

[13] N. Jaques, A. Ghandeharioun, J. H. Shen, C. Ferguson, A. Lapedriza, N. Jones, S. Gu, and R. Picard, "Way off-policy batch deep reinforcement learning of implicit human preferences in dialog," *arXiv preprint arXiv:1907.00456*, 2019.

[14] R. Laroche, P. Trichelair, and R. T. Des Combes, "Safe policy improvement with baseline bootstrapping," in *International conference on machine learning*. PMLR, 2019, pp. 3652–3661.

[15] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," *arXiv preprint arXiv:2210.02747*, 2022.

[16] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[17] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," *arXiv preprint arXiv:2011.13456*, 2020.

[18] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, p. 02783649241273668, 2023.

[19] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," *arXiv preprint arXiv:2205.09991*, 2022.

[20] Z. Wang, J. J. Hunt, and M. Zhou, "Diffusion policies as an expressive policy class for offline reinforcement learning," *arXiv preprint arXiv:2208.06193*, 2022.

[21] S. Zhang, W. Zhang, and Q. Gu, "Energy-weighted flow matching for offline reinforcement learning," *arXiv preprint arXiv:2503.04975*, 2025.

[22] S. Park, Q. Li, and S. Levine, "Flow q-learning," *arXiv preprint arXiv:2502.02538*, 2025.

[23] X. B. Peng, A. Kumar, G. Zhang, and S. Levine, "Advantage-weighted regression: Simple and scalable off-policy reinforcement learning," *arXiv preprint arXiv:1910.00177*, 2019.

[24] F. Nguimatsia Tiofack, T. Le Hellard, F. Schramm, N. Perrin-Gilbert, and J. Carpentier, "Guided flow policy: Learning from high-value actions in offline reinforcement learning," in *The Fourteenth International Conference on Learning Representations*, 2026. [Online]. Available: https://openreview.net/forum?id=EBjy1rmpv0

[25] R. S. Sutton, A. G. Barto, *et al.*, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[26] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," *Advances in neural information processing systems*, vol. 12, 1999.

[27] P. Holderrieth and E. Erives, "An introduction to flow matching and diffusion models," *arXiv preprint arXiv:2506.02070*, 2025.

[28] S. Park, K. Frans, S. Levine, and A. Kumar, "Is value learning really the main bottleneck in offline rl?" *Advances in Neural Information Processing Systems*, vol. 37, pp. 79 029–79 056, 2024.

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

[30] O. G. Younis, R. Perez-Vicente, J. U. Balis, W. Dudley, A. Davey, and J. K. Terry, "Minari," Sept. 2024. [Online]. Available: https://doi.org/10.5281/zenodo.13767625

[31] S. Park, K. Frans, B. Eysenbach, and S. Levine, "Ogbench: Benchmarking offline goal-conditioned rl," *arXiv preprint arXiv:2410.20092*, 2024.

[32] R. Agarwal, M. Schwarzer, P. S. Castro, A. Courville, and M. G. Bellemare, "Deep reinforcement learning at the edge of the statistical precipice," *Advances in Neural Information Processing Systems*, 2021.