REINFORCEMENT LEARNING VIA ADVANTAGE GUIDANCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning (RL) holds the promise of learning a policy that can outperform what is shown in the data. However, in practice, such systems suffer from instability and hyperparameter sensitivity. In contrast, supervised learning provides a stable and scalable training objective, but is fundamentally limited by the quality of the demonstrated behavior. In this paper we ask whether we can perform policy optimization with something as simple and stable as supervised learning but as performant as RL. Unfortunately, naïvely training the policy with supervised learning and conditioning on future rewards or advantages typically underperforms dedicated RL methods. In this paper, we introduce a new policy extraction method, Advantage Guidance, that retains the simple supervised learning structure with advantage conditioning while outperforming prior RL methods through effective conditional training and a novel formulation of contrastive diffusion guidance during sampling. We show that Advantage Guidance can significantly improve a model's ability to respond to conditional advantage signals, and the contrastive guidance during sampling can use the learned advantage-conditioned policy to target the KL-regularized optimal policy. Empirically, we find that Advantage Guidance significantly outperforms prior state-of-the-art RL methods on 26 long-horizon, sparse reward tasks across datasets with varying degrees of optimality.

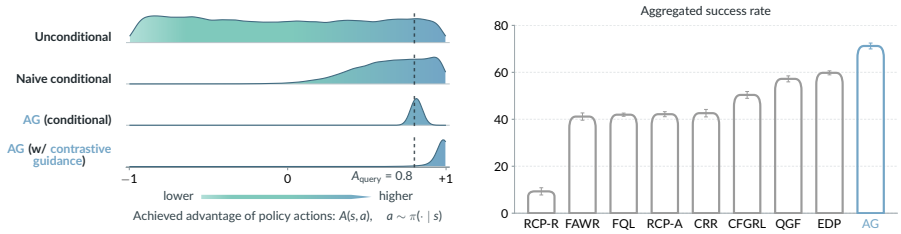

Figure 1: We introduce a new RL method, **Advantage Guidance (AG)**, that uses a simple supervised learning objective with advantage conditioning while outperforming prior RL methods through effective conditional training and a novel formulation of contrastive guidance during sampling. *Left:* Naïve advantage-conditioned policies often fail to match the requested advantage, producing actions whose advantages vary widely and frequently fall below the requested value. We call this the *advantage alignment problem*. Our approach (AG) significantly improves advantage alignment, and our novel contrastive guidance method further enables advantage maximization rather than matching a prespecified value, as desired for policy extraction. *Right:* Our method significantly outperforms baselines on 26 long-horizon sparse-reward tasks across datasets with varying degrees of optimality.

1 INTRODUCTION

Reinforcement learning (RL) holds the promise of improving policy performance beyond what is shown in the data, and can in theory outperform imitation learning under suitable conditions (Sutton & Barto, 2018; Kumar et al., 2022). However, RL algorithms suffer from lots of instability and hyperparameter sensitivity issues in practice (Tsitsiklis & Van Roy, 1996; Fujimoto et al., 2018; Henderson et al., 2018), especially over off-policy or offline data. Imitation learning, on the other hand, has been shown to be stable and scalable for policy training with modern generative modeling techniques like diffusion and flow matching. This raises the question: *can we perform policy training with something as simple and stable as imitation learning, but as performant as RL?*

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

To achieve the stability of imitation learning, one might be tempted to directly apply the supervised-learning loss function, training the policy to maximize the likelihood of actions seen in the data. Previous works have explored accomplishing this with conditional, filtered, or weighted supervised learning (Emmons et al., 2021). One set of methods, which we will refer to as *advantage-conditioned methods* (Kumar et al., 2019; Amin et al., 2025; Gao et al., 2024; Qing et al., 2024), resorts to training a policy conditioned on advantage values, which estimate how "good" actions are. During inference, the policy is queried with a high advantage to generate "good" actions.

Unfortunately, such naïve advantage-conditioned policies typically underperform dedicated RL methods (see Figure 1). This is in part because advantage-conditioning does not generally result in a policy that always favors actions with higher advantages as RL policies do: the resulting policy is restricted to the queried advantage level, and this restricts the degree of policy improvement. Moreover, we find it difficult to train a policy to respond to advantage conditioning in the first place: the policy often cannot match the requested advantage (see Figure 1), and can ignore the conditioning altogether and revert to unconditional imitation.

In this work, our goal is to train a policy with the simple and stable supervised learning objective conditional on continuous advantage estimates, while enabling the policy to generate actions that maximize advantage. To accomplish this, we first derive a novel guidance formulation we call *contrastive guidance* for sampling at test time, which can target actions that maximize the advantage under a KL behavior constraint. Then, to effectively train a flow-matching conditional policy that responds to the conditional values, we re-parameterize the conditional policy as a residual flow on top of a reference unconditional flow field. We also propose an *action imagination* scheme: we use "imagined actions" sampled from the unconditional policy, and their advantage estimates, to train the advantage-conditioned policy under a broader action distribution for each state.

The resulting algorithm, **Advantage Guidance**, significantly improves the policy's ability to respond to advantage inputs, and can target the KL-regularized optimal policy during sampling. Across 26 long-horizon, sparse reward tasks with both play-style and demonstration datasets in the offline RL setting, Advantage Guidance outperforms prior strong RL algorithms.

2 Related Work

Offline RL and policy extraction. Offline RL aims to learn policies that outperform the behavior represented in a fixed dataset. Prior works focus on learning value functions that mitigate erroneous extrapolation (An et al., 2021; Garg et al., 2023; Kostrikov et al., 2021; Kumar et al., 2020; Xu et al., 2023), as well as policy extraction from that value function (Chen et al., 2022; Fujimoto & Gu, 2021; Hansen-Estruch et al., 2023; Nair et al., 2020; Peng et al., 2019; Tarasov et al., 2024; Wu et al., 2019). However, many prior policy extraction methods can be difficult to apply in practice due to instability and hyperparameter sensitivity. Another line of work, which closely relates to ours, circumvents this instability by converting the policy training objective to an imitation learning loss via advantage, reward, or return conditioning (Kumar et al., 2019; Chen et al., 2021; Emmons et al., 2021; Srivastava et al., 2019), data filtering (Chen et al., 2021; Wang et al., 2020; Chen et al., 2020), and loss weighting (Peng et al., 2019; Peters & Schaal, 2007; Neumann & Peters, 2008). We specifically study advantage conditioning, which avoids instability of loss-weighting, uses the full dataset for generalization, and has lower variance than return conditioning. Existing advantage conditioning methods condition on continuous values (Kumar et al., 2019; Gao et al., 2024; Qing et al., 2024) or binarized labels (Amin et al., 2025; Team et al., 2026), and during inference query with a fixed high advantage value, which may fall outside the training distribution, or learn an extra model to choose a high in-support query (Kumar et al., 2019; Gao et al., 2024). In contrast, our method uses a novel form of contrastive diffusion guidance to maximize advantage without risking out-of-support advantage queries or needing additional models. In addition, we find that previous methods struggle to respond to advantage inputs, and we introduce residual parameterization and action imagination training schemes to improve the policy's responsiveness to advantages.

Flow and diffusion policies for RL. Diffusion and flow matching have been widely studied for parameterizing the policy in RL (Ren et al., 2024; Ding et al., 2024; Xue et al., 2025; Chen et al., 2024; Koirala & Fleming, 2025; Chen et al., 2025; Zhou et al., 2026). Using these iterative denoising methods is challenging because leveraging the action gradient requires backpropagating through a multi-step denoising process (Wang et al., 2022; He et al., 2023; Ding & Jin, 2023; Zhang et al., 2024; Wang et al., 2024; Lv et al., 2025), which can be expensive and unstable in practice (Zhou et al., 2026; Park et al., 2025). Previous works avoid this unstable backpropagation process by

approximating the multi-step policy with single-step models (Park et al., 2025; Chen et al., 2025; Kang et al., 2023), but suffer from approximation errors and limited expressivity. Another line of work avoids backpropagation entirely, instead directly guiding the iterative sampling process using critic gradient (Zhou et al., 2026) or classifier-free guidance (Frans et al., 2025; Foffano et al., 2026). Like the latter, our approach avoids critic gradients, which are difficult to estimate for high-dimensional or partially noised actions. Most similar to our work is CFGRL (Frans et al., 2025), which applies CFG between an unconditional policy trained on the entire dataset and another "filtered policy" trained on the positive-advantage subset. Our approach instead trains on the entire dataset conditioned on continuous advantage values, and uses a novel form of contrastive guidance during sampling. Empirically, we find that CFGRL usually underperforms in practice because its binary filtering criterion discards information on action quality.

3 Preliminaries and Background

KL-regularized RL. We characterize the environment with state space $\mathcal{S}$, action space $\mathcal{A}$, a transition function $p(s'|s,a) : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, and a reward function $r(s) : \mathcal{S} \to \mathbb{R}$. A policy $\pi(a \mid s) : \mathcal{S} \to \Delta(\mathcal{A})$ is a mapping from state to an action distribution, and value functions $V^\pi(s)$ and $Q^\pi(s,a)$ characterize the expected future return under a policy π. The advantage $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$ measures how much better action a is than following π at state s. The KL-regularized RL objective is $J(\pi) = \mathbb{E}_\tau\left[\sum_t \gamma^t r(s_t,a_t)\right] - \beta\,\mathbb{E}_{s\sim p^\pi(s)}\left[D_{\mathrm{KL}}(\pi(\cdot \mid s) \,\|\, \hat{\pi}(\cdot \mid s))\right]$, where β controls regularization toward the reference policy $\hat{\pi}$. The KL regularizer prevents the policy from exploiting Q-values, especially in offline RL. It is well-known (Dayan & Hinton, 1992; Abdolmaleki et al., 2018; Peng et al., 2019) that this objective under fixed-state has a closed form solution:

$$\pi^*_{\mathrm{KL}}(a \mid s) \propto \hat{\pi}(a \mid s) \cdot \exp(A(s,a)/\beta), \tag{1}$$

which "tilts" the reference distribution to exponentially favor high advantage actions. We set this as the policy-extraction target distribution we want to achieve.

Advantage conditioned policies. The main idea for advantage-conditioned methods is straightforward: first, compute the advantage estimate $A(s,a)$ for each (s,a) transition in the data via a learned value function or Monte Carlo sampling; then, train a policy $\pi(a \mid s, A(s,a))$ to predict actions given their advantage values; finally, query the policy with some specified target value A_{query}. Intuitively, this associates action quality with advantage values, allowing high-advantage queries to elicit good actions. This results in the policy $\pi(a \mid s, A_{\mathrm{query}}) \propto \hat{\pi}(a \mid s) \cdot \mathbf{1}\left[|A^{\hat{\pi}}(s,a) - A_{\mathrm{qeury}}| < \epsilon\right]$ where $0 < \epsilon \ll 1$. This can be undesirable because $\pi(a|s, A_{\mathrm{qeury}})$ only incentivizes actions with advantage A_{qeury}, but actually discourages actions with $A > A_{\mathrm{qeury}}$, unlike Equation (1). An excessively high A_{query} can place it outside the training distribution and result in bad actions.

Classifier free guidance. Diffusion and flow matching are generative models that transform noise into samples x through an iterative denoising process $p_t(x_t)$. We use flow-matching in this paper for simplicity. The denoising direction is closely related to the score function $\nabla_x \log p_t(x_t)$, which diffusion models learn directly and flow-matching models recover under appropriate transformations (Song et al., 2020; Lipman et al., 2022). Importantly, modifying the score during sampling allows us to guide the denoising process toward a desired distribution, such as one that favors higher-advantage actions. For a generative model $p(x)$ with conditional labels y, classifier free guidance (CFG) (Ho & Salimans, 2022) modifies the sampling at timestep t to instead use score $\nabla_x \log p_t(x_t) + w\left[\nabla_x \log p_t(x_t \mid y) - \nabla_x \log p_t(x_t)\right] = \nabla_x \log \left[p_t(x_t) p_t(y \mid x_t)^w\right]$, where $w > 0$ is a guidance weight. For generated clean samples, this tilts the generated distribution to $p(x)\exp(w \log p(y \mid x))$. Intuitively, this generates samples that maximize the probability of having a specific value of y. Frans et al. (2025) showed that CFG can be used to achieve policy improvement in RL when $p_t(x_t)$ is the policy and the conditional variable y is binary $y \in \{0,1\}$ and $p(y=1 \mid A(s,a))$ is a non-negative and monotonic increasing function of $A(s,a)$.

4 Analysis of CFG for Advantage Conditioned Policies

Since CFG can induce policy improvement (Frans et al., 2025), one might wonder whether we can apply it to policies conditioned on continuous advantages to sample from our target, the KL-constrained optimal policy π^*_{KL} in Equation (1). In this section, we derive the solution of CFG on an advantage conditioned policy and show that it does not achieve this target.

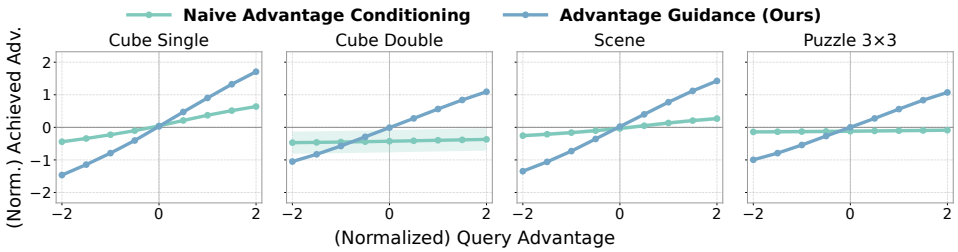

Figure 2: Naïve advantage conditioned policy suffers from the **advantage alignment problem**, generating actions whose advantages don't align with the requested value. 5 tasks per domain.

Diffusion (and under some transformations flow matching) learn the score function, and so we can analyze the score function space directly. The score of the target distribution π^*_{KL} in Equation (1) is

$$\nabla_a \log \pi^*_{\mathrm{KL}}(a \mid s) = \nabla_a \log \left[\hat{\pi}(a \mid s) \exp(A(s,a)/\beta)\right]. \tag{2}$$

Next, we will derive the score function when we perform CFG sampling on an advantage conditioned policy. Let us denote the conditional policy as $\pi(a \mid s, y)$, where $y \sim Y$ is the conditional random variable. For simplicity, assume it has Gaussian likelihood: $Y \mid s, a \sim \mathcal{N}(A(s,a), \sigma_y^2)$ and $\log p_Y(y \mid A(s,a)) = -\frac{(y - A(s,a))^2}{2\sigma_y^2} - \frac{1}{2}\log(2\pi\sigma_y^2)$. We will denote the score of the conditional and unconditional policy as $g_y = \nabla_a \log \pi(a \mid s, y)$ and $g = \nabla_a \log \hat{\pi}(a \mid s)$. Bayes rule gives: $g_y - g = \nabla_a \log \left[\frac{p_Y(y|A(s,a))\hat{\pi}(a|s)}{p(y|s)}\right] - \nabla_a \log \hat{\pi}(a \mid s) = \nabla_a \left[-\frac{(y - A(s,a))^2}{2\sigma_y^2}\right]$. Consequently, the score when sampling with CFG at guidance weight w is

$$g + w(g_y - g) = \nabla_a \log \left[\hat{\pi}(a \mid s) \exp\left(-\frac{w(y - A(s,a))^2}{2\sigma_y^2}\right)\right]. \tag{3}$$

This is generally not the same as the score of the desired tilted distribution in Equation (2). Intuitively, sampling with CFG induces a distribution that favors actions that match the target advantage value, and penalizes deviations above and below it; it does not always favor high-advantage actions as in Equation (2). Higher CFG weights more strongly favor matching the target advantage.

5 Advantage Alignment Problem of Naïve Conditioned Policies

While sampling with CFG does not necessarily produce actions that maximize the advantage, we might imagine that conditioning on a sufficiently high target advantage can still yield a good policy that is an improvement over $\hat{\pi}(a \mid s)$. By corollary, advantage conditioned policies should also enjoy this benefit because they equate to the special case of CFG sampling with guidance weight $w = 1$. For simplicity, we focus our analysis on this special case, analyzing advantage-conditioned policies to understand whether they can effectively adapt to conditioning and generate high-advantage actions.

To measure how well a policy responds to the advantage signal, we define ***advantage alignment*** as the extent to which the achieved advantage of a policy action matches the advantage used to query the conditional policy (*i.e.*, A_{query} vs. $A(s, \pi(\cdot \mid s, A_{\mathrm{query}}))$). Intuitively, if a conditional policy responds properly to advantage inputs, it should be advantage aligned: actions generated conditioned on high advantage should have high advantages, and vice versa. Figure 2 shows the advantage alignment on 20 different OGBench (Park et al., 2024b) tasks across 4 different domains, and the advantages shown are normalized using the mean and variance of the advantages of 32 actions sampled from the same state. A naïve advantage-conditioned policy $\pi(a \mid s, A)$, which we implement following RCP-A (Kumar et al., 2019), responds very weakly to advantage inputs, often generating actions that have the same advantages even when requested with different values, especially on hard long-horizon tasks. When queried with a high advantage, it often generates actions whose advantages fall below the requested value. We refer to this as the ***advantage alignment problem***. And as we will show in Section A.4, using a more expressive policy architecture such as a transformer does not solve this problem, and misalignment is also reflected in policy performance when using different A_{query} (see Section A.3). In comparison, our method Advantage Guidance (which we will introduce later in Section 6) is substantially more responsive to advantage conditioning, with the resulting actions exhibiting achieved advantage values that closely align with requested advantages.

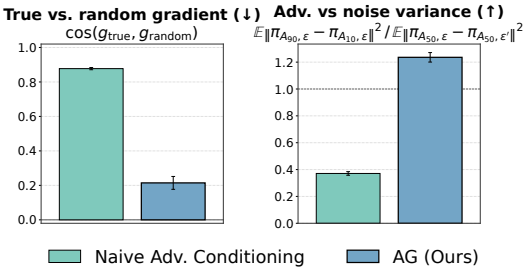

Figure 3: **Naïve advantage conditioned policies often ignore advantage inputs.** Averaged over 20 OGBench tasks.

To gain a mechanistic understanding of the optimization process, we investigate (1) how the gradient of the policy loss responds to advantage conditioning, and (2) how strongly the output actions respond to different conditional signals. First, we compare the gradient of the policy loss conditioned on the ground truth versus random advantage values. We plot their cosine similarity in Figure 3 (left), and find that naïve advantage-conditioned policy has gradient cosine similarity close to 1. This suggests that the loss is largely insensitive to advantage labels, leading the policy to ignore them. So does the policy respond to any conditioning at all, or does it just ignore the advantage? To investigate, we compare the sensitivity of generated actions to advantage inputs and initial flow noise. We measure $\frac{\mathbb{E}\|\pi(a|s,A_{90},\epsilon)-\pi(a|s,A_{10},\epsilon)\|^2}{\mathbb{E}\|\pi(a|s,A_{50},\epsilon)-\pi(a|,s,A_{50},\epsilon')\|^2}$, where A_n is the n-th percentile advantage for state s and ϵ, ϵ' are different initial noises. Figure 3 (right) shows that varying the advantage changes the naïve agent's actions by less than half the effect of sampling noise, whereas our method's response exceeds it.

The analysis above shows that the policy network largely ignores the advantage value , effectively regressing to an unconditional policy $\pi(a \mid s)$. In the next section, we will present solutions to this challenge and show that our method can significantly improve advantage alignment.

6 Advantage Guidance

In this section, we introduce our proposed method, Advantage Guidance, an advantage-conditioned policy extraction method that improves advantage alignment and can target the KL-constrained optimal policy through guidance. We first derive a guidance formulation we term *contrastive guidance* that uses a policy conditioned on continuous advantage values to target $\pi^*_{\text{KL}}(a \mid s) \propto \hat{\pi}(a \mid s) \exp(A(s,a)/\beta)$. Then, we introduce practical techniques during training to improve advantage alignment, including reparameterizing the conditional policy as a residual flow over an unconditional flow field and training on multiple imagined actions per state.

6.1 Contrastive Guidance for Advantage Maximization

First, we aim to derive a sampling technique based on diffusion guidance that can enable the policy to generate advantage-maximizing actions rather than matching a prespecified value. Our target is the KL-regularized optimal policy π^*_{KL}, which maximizes advantage under KL constraint to $\hat{\pi}(a \mid s)$.

Recall that for flow and diffusion policies, we can analyze the score function directly, and the score of our target distribution π^*_{KL} is in Equation (2). Following Section 4, we denote the conditional policy as $\pi(a \mid s, y)$ where $y \sim Y$ is the conditional random variable with likelihood $p_Y(y \mid A)$.

Remark 1 (Contrastive Guidance.). *Assume p_Y belongs to exponential family $p_Y(y \mid A) = h(y) \exp\left(T(y)A - F(A)\right)$ where $h(y) \geq 0$ and $F(A)$ is a log-partition function. The action score of the KL-regularized optimal policy $\pi^*_{\text{KL}}(a \mid s) \propto \hat{\pi}(a \mid s) \exp(A(s,a)/\beta)$ matches*

$$\nabla_a \log \hat{\pi}(a \mid s) + w \left[\nabla_a \log \pi(a \mid s, y_+) - \nabla_a \log \pi(a \mid s, y_-)\right] \tag{4}$$

if some fixed $w, y_+, y_- \in \mathbb{R}$ satisfy $1/\beta = w[T(y_+) - T(y_-)]$.

We provide a formalization and proof of this claim in Section A.1. In other words, if we sample according to Equation (4), which is easily derived with the score of the conditional and unconditional policy, we can directly match the score of the desired exponentially-tilted distribution. This is akin to performing diffusion guidance with guidance weight w and guidance term $\nabla_a \log \pi(a \mid s, y_+) - \nabla_a \log \pi(a \mid s, y_-)$. Since the guidance term contains the score difference of a contrastive pair (y_+, y_-), we call this *contrastive guidance*.

Remark 2 (Advantage Maximization.). *If $\beta > 0$ (which is ensured if $T(y_+) > T(y_-)$ and $w > 0$), the tilted policy $\pi^*_{\text{KL}}(a \mid s) \propto \hat{\pi}(a \mid s) \exp(A(s,a)/\beta)$ is the solution to the KL-constrained advantage maximization problem* $\arg\max_{\pi(\cdot|s)} \left\{\mathbb{E}_{a\sim\pi(\cdot|s)}[A(s,a)] - \beta D_{\text{KL}}(\pi(\cdot \mid s) \,\|\, \hat{\pi}(\cdot \mid s))\right\}$.

This follows directly from Remark 1 and prior work (Dayan & Hinton, 1992; Abdolmaleki et al., 2018; Peng et al., 2019). Thus, with a positive guidance weight $w > 0$ and $T(y_+) > T(y_-)$, the

Algorithm 1 Advantage Guidance (AG)

Training	**Sampling**
Input: dataset $\mathcal{D}$, advantage $A(s,a)$, K, advantage label std σ_y	**Input:** state s, trained velocities $v_{\hat{\pi}}, v_\theta$ conditional query pair $y_+ > y_-$, guidance weight w, steps T
while not converged **do**	
$(s,a) \sim \mathcal{D}, \epsilon \sim \mathcal{N}(0,I), t \sim \mathcal{U}(0,1)$	$x_0 \sim \mathcal{N}(0,I)$
Train ref. policy $\hat{\pi}$ with $\mathcal{L}_{\text{BC}} = \|v_{\hat{\pi}} - (a-\epsilon)\|_2^2$	**for** $n = 0, \dots, T-1$ **do**
$a_0 \leftarrow a$; $a_i \sim \hat{\pi}(\cdot \mid s)$, $i = 1, \dots, K$ ◀ *action imagination*	$t \leftarrow n/T$
$A_i \leftarrow A(s,a_i)$, $i = 0, \dots, K$	▼ *contrastive guidance*
$\tilde{A}_i \leftarrow (A_i - \text{mean}(A_{0:K}))/\sqrt{\text{var}(A_{0:K}) + \delta}$ ◀ *adv. norm.*	$v \leftarrow v_{\hat{\pi}} + w\big[v_\theta(y_+) - v_\theta(y_-)\big]$
$y_i \sim \mathcal{N}(\tilde{A}_i, \sigma_y^2)$, $i = 0, \dots, K$ ◀ *Gaussian label*	$x_{n+1} \leftarrow x_n + \frac{1}{T} v$
$\Delta\theta \leftarrow \nabla_\theta \sum_{i=0}^{K} \|v_{\hat{\pi}} + v_\theta(y_i) - (a_i - \epsilon)\|_2^2$ ◀ *residual param.*	
end while	**end for**
return $v_{\hat{\pi}}, v_\theta$	**return** $a = x_T$

Velocity-field arguments are abbreviated above: $v_{\hat{\pi}} = v_{\hat{\pi}}(x_t, t, s)$ and $v_\theta(y) = v_\theta(x_t, t, s, y)$

target of contrastive guidance induces advantage maximization, rather than matching a fixed value. The contrastive labels y_+, y_- and the guidance weight w control the maximization degree together: a higher $w[T(y_+) - T(y_-)]$ leads to weaker constraint and stronger advantage maximization.

Practical Implementation. In practice, we can choose a simple exponential family for the conditional variable Y, the Gaussian distribution. In other words, we choose $Y \mid s, a \sim \mathcal{N}(A(s,a), \sigma_y^2)$, which corresponds to $h(y) = \frac{1}{\sqrt{2\pi}\sigma_y} \exp(-\frac{y^2}{2\sigma_y^2}), T(y) = \frac{y}{\sigma_y^2}, F(A) = \frac{A^2}{2\sigma_y^2}$. For $y_+ > y_-$, we automatically have $T(y_+) > T(y_-)$ and we can choose guidance weight $w = \frac{\sigma_y^2}{\beta(y_+ - y_-)} > 0$ to give the desired tilt in Equation (1).

6.2 Residual Parameterization for Conditional Policy

While contrastive guidance can target the KL-regularized optimal policy in theory, it makes an implicit assumption that the conditional policy is advantage-aligned with y_+ and y_-. Therefore, we focus next on training techniques that can improve advantage alignment. Recall that we found in Section 5 that naïve advantage conditioned policies often ignore advantage inputs and largely optimize the policy loss based on state and initial noise alone. In other words, the policy is minimizing the loss for the advantage-conditional policy by learning to minimize the unconditional loss. This motivates an approach that separates out the components that are not conditioned on advantage, and leaves the conditional policy to focus on the advantage input. In Advantage Guidance, we propose to parameterize the conditional policy as a residual flow v_θ to the flow field of an unconditional reference policy $v_{\hat{\pi}}(a \mid s)$. Specifically, both policies are instantiated as flow-matching networks, where the model learns a velocity field conditioned on a partially noised action x_t, along with the noise scale t, state s, and conditioning variable y. The residual parameterization in velocity space is:

$$v_\pi(x_t, t, s, y) = v_{\hat{\pi}}(x_t, t, s) + v_\theta(x_t, t, s, y). \tag{5}$$

The training loss for the conditional policy is:

$$\mathcal{L}(\theta) = \mathbb{E}_{s,a \sim \mathcal{D}_{\text{aug}}}\left[||v_{\hat{\pi}}(s, x_t, t) + v_\theta(s, x_t, y, t) - (a - \epsilon)||^2\right] \quad \text{where} \quad x_t = (1-t)\epsilon + ta. \tag{6}$$

The noise scale $t \sim U[0,1]$ and $\epsilon \sim \mathcal{N}(0,1)$ is Gaussian noise. The unconditional reference flow policy $\hat{\pi}$ is trained to minimize regular flow-matching loss $\mathbb{E}_{s,a \sim D}\left[||v_{\hat{\pi}}(s, x_t, t) - (x - \epsilon)||^2\right]$. With this parameterization, the reference policy $\hat{\pi}(a \mid s)$ minimizes the advantage-independent loss, therefore encouraging the residual v_θ to focus on minimizing $\mathcal{L}(\theta)$ based on conditional input y.

6.3 Action Imagination for Counterfactual Supervision

Beyond optimization dynamics, learning an advantage-conditioned policy presents a more fundamental challenge: the dataset typically contains only one action for any given state, especially in continuous state spaces. As a result, the policy receives little direct supervision on how actions should change as requested advantages vary. We address this lack of counterfactual supervision with *action*

imagination: for each state-action pair (s, a_d) in the dataset, we augment the training distribution for the conditional policy with K additional actions on state s sampled from some proposal distribution:

$$\mathcal{D}_{\text{aug}} = \bigcup_{(s,a_d)\in\mathcal{D}} \left\{\left(s, a_i, A(s, a_i)\right)\right\}_{i=0}^{K}, \qquad a_0 = a_d, \quad a_i \sim \pi_p(\cdot \mid s) \text{ for } i = 1, \ldots, K.$$

One convenient solution is to set the proposal distribution as the reference unconditional policy: $\pi_p = \hat{\pi}$. As we will show in Section 7, training with residual parameterization and action imagination drastically improves advantage alignment and policy performance, especially on long horizon tasks.

6.4 Implementation Details

Advantage Guidance is an RL policy extraction algorithm based on advantage-conditioned training and contrastive guidance during sampling. It is agnostic to how the advantage estimates are derived. For simplicity, in this paper, we train value functions $Q(s, a)$ and $V(s)$ using IQL (Kostrikov et al., 2021), and calculate advantage as $A(s, a) = Q(s, a) - V(s)$. Algorithm 1 shows a summary of the training and sampling procedure of Advantage Guidance. During training, we standardize advantages separately for each state using $\tilde{A}(s, a_i) = (A(s, a_i) - \mu_s)/\sqrt{\sigma_s^2 + \delta}$, where $\mu_s = \frac{1}{K+1}\sum_{j=0}^{K} A(s, a_j)$ and $\sigma_s^2 = \frac{1}{K+1}\sum_{j=0}^{K}(A(s, a_j) - \mu_s)^2$, with dataset action $a_0 = a_d$, imagined actions $a_{i=1,\ldots,K}$, and $\delta > 0$ ensuring numerical stability. Then, we condition the policy on $\tilde{A}$ using AdaLN (Xu et al., 2019), which adaptively scales and shifts hidden representations at each layer.

7 Experiments

We conduct experiments to analyze the empirical effectiveness of Advantage Guidance on a range of long-horizon, sparse reward manipulation tasks. Concretely, we study the following research questions: **(1)** How well does Advantage Guidance perform compared to prior RL methods on datasets with varying degrees of optimality? **(2)** Do residual parameterization and action imagination improve advantage alignment and policy performance? **(3)** Does contrastive guidance improve policy performance compared to CFG or no guidance? Additionally, we conduct further ablation experiments in Section A.2. All detailed results are documented in Section A.10.

Experimental Setup. We evaluate all methods in the offline RL setting with 20 tasks from OGBench (Park et al., 2024b) and 6 tasks from OCBench (Anonymous, 2026)[1]. OGBench offers a suite of long-horizon challenging manipulation tasks, and OCBench is a recently proposed extension of OGBench that provides a more realistic action distribution meant to emulate the statistical properties of real-world human demonstrations. In particular, the two benchmarks differ substantially in their offline dataset composition. OGBench is a standard offline RL benchmark with *play-style* datasets generated by scripted behaviors pursuing random goals, often yielding highly suboptimal trajectories for the task objective. OCBench offers *demonstration-based* datasets, consisting of trajectories generated by near-expert policies with high success rates on the task objective. Compared to play-style datasets, these demonstrations cover narrower states and predominantly near-optimal actions, making it more challenging for offline RL because extrapolation errors are harder to avoid. However, such data more closely resembles real-world robotics datasets, and OCBench's design to emulate human demonstration statistics makes it particularly relevant. We use `cube-single-T{1-5}`, `cube-double-T{1-5}`, `scene-T{1-5}`, `puzzle-3x3-T{1-5}` from OGBench and lite tasks `block-single-T1`, `block-double-T{1-2}`, `chamber-easy-T{1-3}` from OCBench, all with sparse rewards. In addition, we follow the action chunking setting (Li et al., 2025), where the higher-dimensional actions make the action distribution more complex and therefore more challenging.

Baselines. We compare **Advantage Guidance** against a representative set of strong flow-matching RL baselines, grouped into three categories: **(1) actor critic methods** train a parameterized actor to maximize the critic $Q(s, a)$ subject to behavioral constraints. **FQL** (Park et al., 2025) distills a multi-step flow policy into a single-step policy to avoid backpropagation through time, while **EDP** (Kang et al., 2023) approximates the multi-step integration with a single large Euler step. **(2) Supervised-learning-based methods** formulate policy optimization as a conditional, filtered, or weighted supervised learning problem. **CRR** (Wang et al., 2020) trains a behavior cloning policy

[1] OCBench is a concurrent benchmark submission to ICLR, which was made available to us for our evaluation. The referenced submission includes the OCBench datasets and code. See Figure 16 for task illustrations.

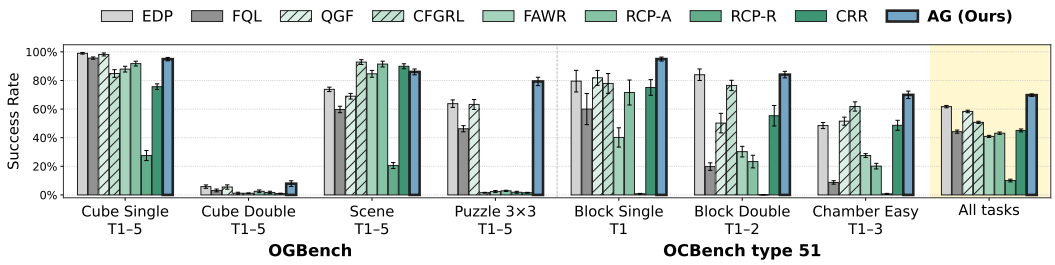

Figure 4: **Offline RL performance** on OGBench (20 tasks) and OCBench (6 tasks) with 10 seeds: Advantage Guidance (blue) outperforms all previous methods, including strong actor-critic methods (grey), supervised-learning-based methods (green), and diffusion guidance methods (shaded green).

on the subset of data with positive advantage; **FAWR** is an extension of AWR (Peng et al., 2019) to flow policies by weighting the flow-matching loss with exponentiated advantages. **RCP-R** and **RCP-A** (Kumar et al., 2019) train a policy conditioned on the return-to-go and advantage estimate, respectively. **(3) Diffusion guidance methods** steer the denoising process toward high-value actions at inference time. **QGF** (Zhou et al., 2026) guides a behavior cloning policy using an approximation of the critic gradient $\nabla_a Q(s, a)$ at partially noised actions. **CFGRL** (Frans et al., 2025) applies CFG between a policy trained on the full dataset and a policy trained on the positive-advantage subset. Section A.6 provides a more detailed description of each method. To ensure fair comparison among all the policy extraction methods, we use the same value function for all. For simplicity, use the IQL (Kostrikov et al., 2021) Q and V functions, and use $A(s, a) = Q(s, a) - V(s)$.

7.1 How Does Advantage Guidance Compare with Prior Methods?

While we expected the main benefit of Advantage Guidance to be its simplicity, in practice we found that it actually outperforms much more sophisticated RL methods. Figure 4 compares the performance of Advantage Guidance against baselines. Overall, Advantage Guidance significantly outperforms previous methods on both play-style datasets from OGBench and demonstration-based datasets from OCBench. In particular, we find that previous supervised-learning-based methods perform poorly compared to strong actor critic methods such as EDP. The performance gap is especially large on challenging long horizon tasks such as `puzzle-3x3` and `block-double`. In comparison, Advantage Guidance significantly outperforms these baselines, including strong actor critic methods such as EDP and FQL. Advantage Guidance also significantly outperforms previous diffusion guidance methods: compared to CFGRL, Advantage Guidance can utilize continuous advantage signals for denser supervision, and can target the KL-regularized optimal policy with contrastive guidance; compared to QGF, Advantage Guidance doesn't rely on explicit Q-gradients in high-dimensional action spaces, which can be unstable or inaccurate.

Best-of-N sampling. Prior work has shown that when additional test-time compute is available, best-of-N (BFN) sampling is an expensive but effective method to improve flow policies (Zhou et al., 2026; Hansen-Estruch et al., 2023). BFN samples multiple actions from a reference BC policy and chooses the action with the highest Q-value. However, even with a small number of samples such as $N = 4$, BFN is magnitudes more expensive due to the sampling process (See Fig 6 in Zhou et al. (2026)). Regardless, we compare against this strong baseline in Figure 5 with different values of N. On OGBench, Advantage Guidance significantly outperforms BFN at low values of N despite using less test-time compute, but eventually slightly underperforms BFN with large enough N, which also outperforms all prior methods in Figure 4. On OCBench, however, Advantage Guidance is significantly better even with large N. This suggests that while BFN can be effective on play-style dataset, it does not do well in more realistic demonstration-based datasets.

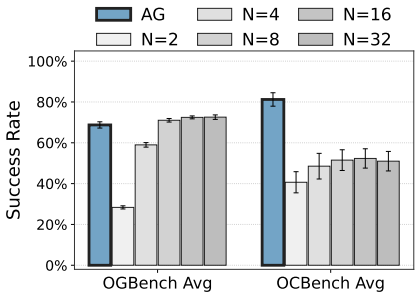

Figure 5: **Comparison with best-of-N sampling**. Advantage Guidance outperforms BFN at small N on OGBench and consistently on OCBench, despite using much less test-time compute.

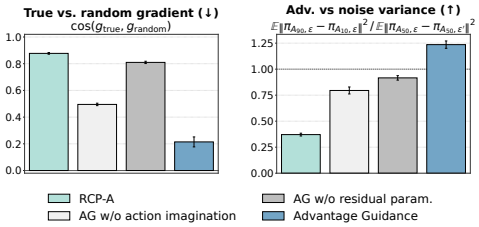

Figure 6: **Effect of residual parameterization and action imagination on advantage alignment**: both improve gradient sensitivity and output variance on 20 OGBench tasks.

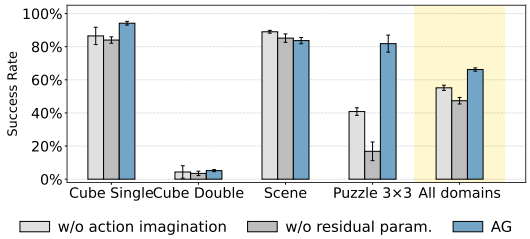

Figure 7: **Ablation of residual parameterization and action imagination training**: both improve policy performance across 20 OGBench tasks.

7.2 Does AG Training Improve Advantage Alignment and Policy Performance?

We analyze whether the residual parameterization and the action imagination training scheme introduced in Section 6 improve advantage alignment and policy performance. We first plot in Figure 6 the two metrics introduced in Section 5 that measure how much a conditional policy responds to advantage conditioning: the gradient similarity between true and random gradients and the advantage-noise variance ratio. Figure 6 shows that both the residual parameterization and action imagination improve advantage alignment, and combining both does even better. Figure 7 shows that such increased alignment translates to better policy performance, especially on hard long horizon tasks such as `puzzle-3x3`. We also ablate the effect of different numbers of imagined actions in Figure 9 in Section A.2, showing that increasing K can increase performance but will plateus after $K = 7$.

7.3 Does contrastive guidance improve performance over CFG or no guidance?

We showed in Section 6.1 that contrastive guidance can target the KL-regularized optimal policy and always favor high-advantage actions. Here, we empirically validate whether contrastive guidance improves policy performance. Specifically, we compare performance of the same AG-trained policy with (1) contrastive guidance, (2) CFG, and (3) just querying the conditional policy. For contrastive guidance, we use fixed (y_+, y_-) per benchmark, while we tune the best conditional values per-domain per benchmark for the latter two to make them as competitive as possible. Figure 8 shows that contrastive guidance performs similarly to CFG and pure conditional policy on OGBench, but significantly improves performance on OCBench. It is not entirely surprising that all three methods can perform well: while CFG and pure conditional policy can only match a target advantage value instead of maximizing advantage, conditioning an advantage-aligned policy on a high-enough advantage should have a similar maximizing effect. However, as observed on OCBench, conditional policies and standard CFG may perform poorly when high query advantages lie outside the training distribution, a common issue in demonstration-based datasets. In this regime, contrastive guidance is crucial for advantage maximization. Furthermore, all methods only perform well if the conditional policy is advantage aligned (such as AG here); using an unaligned policy such as RCP-A leads to substantially worse performance (see Figure 4).

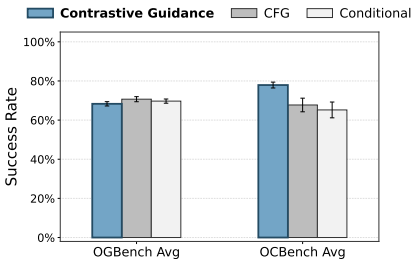

Figure 8: **Ablation of sampling method for Advantage Guidance.** Contrastive guidance significantly outperforms CFG and no guidance on 6 OCBench tasks.

8 Conclusion

In this paper, we introduced Advantage Guidance, a new RL method that uses a simple supervised learning objective with advantage conditioning and a novel formulation of contrastive diffusion guidance during inference. While we expected the main benefit of Advantage Guidance to be its simplicity, we find empirically that it is actually able to significantly outperform much more complex RL methods. We show that our proposed approach enables the policy to match requested advantages much better than prior approaches, and our proposed contrastive guidance further enables advantage maximization during inference, leading to much better policy improvement.

9 LIMITATIONS

Our method Advantage Guidance relies on using advantage estimates to condition and guide policy extraction. While we find this to be empirically effective, the extracted policy is dependent on the accuracy of advantage estimates. One direction for future research is to characterize how much Advantage Guidance performance depends on the advantage estimate accuracy, how to deal with noisy advantages, and how to improve the advantage estimation accuracy. In addition, we note that while diffusion guidance in general (our proposed contrastive guidance included) matches the clean-action score of a tilted distribution, it is extended to intermediate noise steps as a practical and performant approximation. Another direction for future work is to sidestep diffusion guidance during inference and directly learn an advantage-maximizing policy with advantage conditioning during training.

10 AI USE STATEMENT

In this work, we used generative AI tools for assist in the writing of proofs / formulate mathematical claims / design or provide feedback on research methodology or experiments / implement methods / support qualitative and thematic data analysis. We have not used generative AI tools for provide critical ingredients for proving mathematical claims / propose or refine hypotheses / interpret results, and help develop theoretical models or conceptual frameworks / assist with translation / generate synthetic data sets / clean and reformat dataset are not applicable to this work. We have reviewed all AI-assisted work. We checked LLM-generated code and visualization of results. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

11 ETHICS STATEMENT

We, the authors, acknowledge that we have read the ICLR Code of Ethics and fully adhere to it.

12 REPRODUCIBILITY STATEMENT

We describe all the implementation details in Section 6.4 and Section A.7, and hyperparameter choices in Section A.8, which should enable researchers to reproduce our results. We will share the code during the review process and also release the code publicly.

REFERENCES

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*, 2018.

Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, et al. $\pi^*_{0.6}$: a vla that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025.

Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *Advances in neural information processing systems*, 34:7436–7447, 2021.

Anonymous. OCBench: a controllable benchmark for studying behavioral cloning, 2026. URL https://openreview.net/forum?id=bTZVvwj6R6. Submitted to ICLR 2027.

Huayu Chen, Cheng Lu, Chengyang Ying, Hang Su, and Jun Zhu. Offline reinforcement learning via high-fidelity generative behavior modeling. *arXiv preprint arXiv:2209.14548*, 2022.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34, 2021.

Tianyi Chen, Haitong Ma, Na Li, Kai Wang, and Bo Dai. One-step flow policy mirror descent. *arXiv preprint arXiv:2507.23675*, 2025.

Tianyu Chen, Zhendong Wang, and Mingyuan Zhou. Diffusion policies creating a trust region for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 37: 50098–50125, 2024.

Xinyue Chen, Zijian Zhou, Zheng Wang, Che Wang, Yanqiu Wu, and Keith Ross. Bail: Best-action imitation learning for batch deep reinforcement learning. *Advances in Neural Information Processing Systems*, 33:18353–18363, 2020.

Peter Dayan and Geoffrey E Hinton. Feudal reinforcement learning. *Advances in neural information processing systems*, 5, 1992.

Shutong Ding, Ke Hu, Zhenhao Zhang, Kan Ren, Weinan Zhang, Jingyi Yu, Jingya Wang, and Ye Shi. Diffusion-based reinforcement learning via q-weighted variational policy optimization. *Advances in Neural Information Processing Systems*, 37:53945–53968, 2024.

Zihan Ding and Chi Jin. Consistency models as a rich and efficient policy class for reinforcement learning. *arXiv preprint arXiv:2309.16984*, 2023.

Scott Emmons, Benjamin Eysenbach, Ilya Kostrikov, and Sergey Levine. Rvs: What is essential for offline rl via supervised learning? *arXiv preprint arXiv:2112.10751*, 2021.

Daniele Foffano, Arvid Eriksson, David Broman, Karl H Johansson, and Alexandre Proutiere. Advantage-guided diffusion for model-based reinforcement learning. *arXiv preprint arXiv:2604.09035*, 2026.

Kevin Frans, Seohong Park, Pieter Abbeel, and Sergey Levine. Diffusion guidance is a controllable policy improvement operator. *arXiv preprint arXiv:2505.23458*, 2025.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

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

Chen-Xiao Gao, Chenyang Wu, Mingjun Cao, Rui Kong, Zongzhang Zhang, and Yang Yu. Act: Empowering decision transformer with dynamic programming via advantage conditioning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 12127–12135, 2024.

Divyansh Garg, Joey Hejna, Matthieu Geist, and Stefano Ermon. Extreme q-learning: Maxent rl without entropy. *arXiv preprint arXiv:2301.02328*, 2023.

Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. IDQL: Implicit Q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.

Longxiang He, Li Shen, Linrui Zhang, Junbo Tan, and Xueqian Wang. Diffcps: Diffusion model based constrained policy search for offline reinforcement learning. *arXiv preprint arXiv:2310.05333*, 2023.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Bingyi Kang, Xiao Ma, Chao Du, Tianyu Pang, and Shuicheng Yan. Efficient diffusion policies for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36: 67195–67212, 2023.

Diederik Kingma and Ruiqi Gao. Understanding diffusion objectives as the elbo with simple data augmentation. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 65484–65516, 2023.

Prajwal Koirala and Cody Fleming. Flow-based single-step completion for efficient and expressive policy learning. *arXiv preprint arXiv:2506.21427*, 2025.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit Q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Aviral Kumar, Xue Bin Peng, and Sergey Levine. Reward-conditioned policies. *arXiv preprint arXiv:1912.13465*, 2019.

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.

Aviral Kumar, Joey Hong, Anikait Singh, and Sergey Levine. When should we prefer offline reinforcement learning over behavioral cloning? *arXiv preprint arXiv:2204.05618*, 2022.

Qiyang Li, Zhiyuan Zhou, and Sergey Levine. Reinforcement learning with action chunking. *arXiv preprint arXiv:2507.07969*, 2025.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Lei Lv, Yunfei Li, Yu Luo, Fuchun Sun, Tao Kong, Jiafeng Xu, and Xiao Ma. Flow-based policy for online reinforcement learning. *arXiv preprint arXiv:2506.12811*, 2025.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Gerhard Neumann and Jan Peters. Fitted q-iteration by advantage weighted regression. *Advances in neural information processing systems*, 21, 2008.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. *ArXiv*, 2024a.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. *arXiv preprint arXiv:2410.20092*, 2024b.

Seohong Park, Qiyang Li, and Sergey Levine. Flow Q-learning. *arXiv preprint arXiv:2502.02538*, 2025.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pp. 745–750, 2007.

Yunpeng Qing, Shunyu Liu, Jingyuan Cong, Kaixuan Chen, Yihe Zhou, and Mingli Song. A2po: Towards effective offline reinforcement learning from an advantage-aware perspective. *Advances in Neural Information Processing Systems*, 37:29064–29090, 2024.

Allen Z Ren, Justin Lidard, Lars L Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. *arXiv preprint arXiv:2409.00588*, 2024.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Rupesh Kumar Srivastava, Pranav Shyam, Filipe Mutz, Wojciech Jaśkowski, and Jürgen Schmidhuber. Training agents using upside-down reinforcement learning. *arXiv preprint arXiv:1912.02877*, 2019.

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

Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

NeoteAI Team, Fudan TEAI Team, et al. n_0-vtla: Scaling vision-tactile-language-action model with latent tactile tokens. *arXiv preprint arXiv:2607.23782*, 2026.

John Tsitsiklis and Benjamin Van Roy. Analysis of temporal-diffference learning with function approximation. *Advances in neural information processing systems*, 9, 1996.

Yinuo Wang, Likun Wang, Yuxuan Jiang, Wenjun Zou, Tong Liu, Xujie Song, Wenxuan Wang, Liming Xiao, Jiang Wu, Jingliang Duan, et al. Diffusion actor-critic with entropy regulator. *Advances in Neural Information Processing Systems*, 37:54183–54204, 2024.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

Ziyu Wang, Alexander Novikov, Konrad Zolna, Josh S Merel, Jost Tobias Springenberg, Scott E Reed, Bobak Shahriari, Noah Siegel, Caglar Gulcehre, Nicolas Heess, et al. Critic regularized regression. *Advances in Neural Information Processing Systems*, 33:7768–7778, 2020.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

Haoran Xu, Li Jiang, Jianxiong Li, Zhuoran Yang, Zhaoran Wang, Victor Wai Kin Chan, and Xianyuan Zhan. Offline rl with no ood actions: In-sample learning via implicit value regularization. *arXiv preprint arXiv:2303.15810*, 2023.

Jingjing Xu, Xu Sun, Zhiyuan Zhang, Guangxiang Zhao, and Junyang Lin. Understanding and improving layer normalization. *Advances in neural information processing systems*, 32, 2019.

Shuchen Xue, Chongjian Ge, Shilong Zhang, Yichen Li, and Zhi-Ming Ma. Advantage weighted matching: Aligning rl with pretraining in diffusion models. *arXiv preprint arXiv:2509.25050*, 2025.

Ruoqi Zhang, Ziwei Luo, Jens Sjölund, Thomas B Schön, and Per Mattsson. Entropy-regularized diffusion policy with q-ensembles for offline reinforcement learning. *Advances in neural information processing systems*, 37:98871–98897, 2024.

Zhiyuan Zhou, Andy Peng, Charles Xu, Qiyang Li, Tobias Springenberg, Kevin Frans, and Sergey Levine. Test-time gradient guidance of flow policies in reinforcement learning. *arXiv preprint arXiv:2606.11087*, 2026.

A Appendix

A.1 Contrastive Guidance Theoretical Guarantee.

Theorem 1 (Contrastive guidance)**.** *Let $\hat{\pi}$ be a reference policy and $\pi(a \mid s, y)$ be the conditional policy, where the conditional random variable Y has likelihood*

$$p_Y(y \mid A) = h(y) \exp\left(T(y)A - F(A)\right), \qquad F(A) = \log \int h(y) e^{T(y)A}\, dy, \tag{7}$$

where $h(y) \geq 0$. For fixed w, y_+, y_- satisfying $0 < h(y_\pm) < \infty$, $|T(y_\pm)| < \infty$, the policy $\pi^*_{\mathrm{KL}}(a \mid s) \propto \hat{\pi}(a \mid s) e^{A(s,a)/\beta}$ satisfies

$$\nabla_a \log \pi^*_{\mathrm{KL}}(a \mid s) = \nabla_a \log \hat{\pi}(a \mid s) + w\left[\nabla_a \log \pi(a \mid s, y_+) - \nabla_a \log \pi(a \mid s, y_-)\right]. \tag{8}$$

Proof. By Bayes' rule, we have

$$\pi(a \mid s, y) = \frac{\hat{\pi}(a \mid s) p_Y(y \mid A(s,a))}{p_Y(y \mid s)}, \tag{9}$$

where $p_Y(y \mid s) = \int_{\mathcal{A}} \hat{\pi}(a' \mid s) p_Y(y \mid A(s,a'))\, da'$ is the marginal distribution of Y. Taking the ratio of the two conditional policies gives

$$\log \frac{\pi(a \mid s, y_+)}{\pi(a \mid s, y_-)} = \log \frac{p_Y(y_+ \mid A(s,a))}{p_Y(y_- \mid A(s,a))} + \log \frac{p_Y(y_- \mid s)}{p_Y(y_+ \mid s)} \tag{10}$$

$$= [T(y_+) - T(y_-)]\, A(s,a) + \log \frac{h(y_+)}{h(y_-)} + \log \frac{p_Y(y_- \mid s)}{p_Y(y_+ \mid s)}, \tag{11}$$

where the log-normalizer $F(A(s,a))$ cancels out. Since the last two terms are independent of a, differentiating yields

$$\nabla_a \log \pi(a \mid s, y_+) - \nabla_a \log \pi(a \mid s, y_-) = [T(y_+) - T(y_-)]\, \nabla_a A(s,a). \tag{12}$$

Using $\beta^{-1} = w[T(y_+) - T(y_-)]$, we therefore obtain

$$\nabla_a \log \hat{\pi}(a \mid s) + w\, [\nabla_a \log \pi(a \mid s, y_+) - \nabla_a \log \pi(a \mid s, y_-)] \tag{13}$$

$$= \nabla_a \log \hat{\pi}(a \mid s) + \frac{1}{\beta} \nabla_a A(s,a) \tag{14}$$

$$= \nabla_a \log \left(\frac{\hat{\pi}(a \mid s) e^{A(s,a)/\beta}}{Z_\beta(s)} \right) \tag{15}$$

$$= \nabla_a \log \pi^*_{\mathrm{KL}}(a \mid s), \tag{16}$$

□

We note that the result above characterizes the desired policy through its clean-action score. For sampling with contrastive guidance, we extend it to intermediate denoising steps as a practical approximation, following standard classifier free guidance (Ho & Salimans, 2022).

A.2 Additional Advantage Guidance Ablation Experiments

Action imagination. Figure 9 shows the ablation of using different numbers of imagined actions for Advantage Guidance. For each, the conditional policy is trained on $K+1$ actions per state, with K imagined actions and 1 dataset action. As described in Section 6.4, the advantages are also normalized with the mean and variance of these $K+1$ actions. Figure 9 shows that having a very small K such as $K=3$ hurts performance, and having more imagined actions helps, though performance plateaus after roughly $K=7$.

Advantage Normalization. Figure 10 shows the ablation of using different methods of advantage normalization. We consider three different types of normalization methods that normalize with the mean and variance with $\tilde{A}(s,a_i) = (A(s,a_i) - \mu)/\sqrt{\sigma^2 + \delta}$, and each method calculates the mean and variance statistics differently:

1. Per-state norm (Ours): as described in Section 6.4, we calculate mean and variance per-state using K imagined action together with 1 dataset action on the same state:

$$\mu_s = \frac{1}{K+1} \sum_{j=0}^{K} A(s,a_j), \qquad \sigma_s^2 = \frac{1}{K+1} \sum_{j=0}^{K} \left(A(s,a_j) - \mu_s\right)^2,$$

   where $a_0 = a_d$ is the dataset action and $a_1, \ldots, a_K$ are imagined actions.

2. Per-batch norm: we calculate the mean and variance across the current minibatch of advantages

$$\mu_{\mathcal{B}} = \frac{1}{B} \sum_{i=1}^{B} A(s_i,a_i), \qquad \sigma_{\mathcal{B}}^2 = \frac{1}{B} \sum_{i=1}^{B} \left(A(s_i,a_i) - \mu_{\mathcal{B}}\right)^2.$$

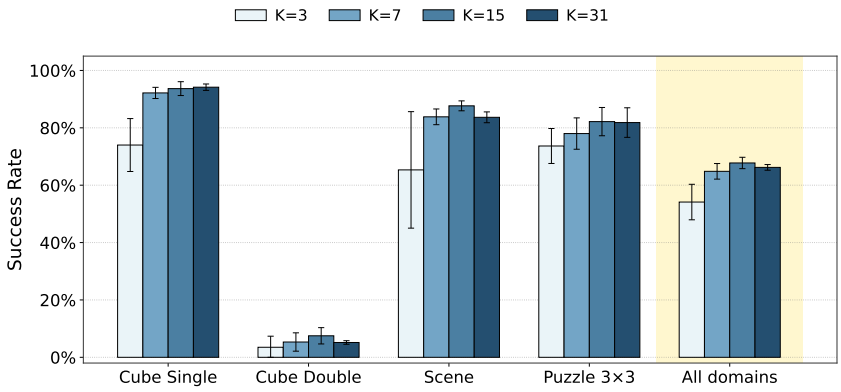

Figure 9: Ablating the number of imagined actions, K, used for training the conditional policy in Advantage Guidance. A small K underperforms larger values, but performance quickly plateaus. Averaged over 5 tasks per domain and 4 seeds.

3. Dataset-level norm: we calculate the mean and variance across the entire dataset:

$$\mu_{\mathcal{D}} = \frac{1}{N}\sum_{i=1}^{N} A(s_i, a_i), \qquad \sigma^2_{\mathcal{D}} = \frac{1}{N}\sum_{i=1}^{N} \left(A(s_i, a_i) - \mu_{\mathcal{D}}\right)^2.$$

To only ablate the effect of normalization, the different normalization methods all train the conditional policy with action augmentation at $K = 31$ and only differ in how the advantages are normalized. Figure 10 shows that our per-state normalization scheme offers a big advantage over other normalization schemes.

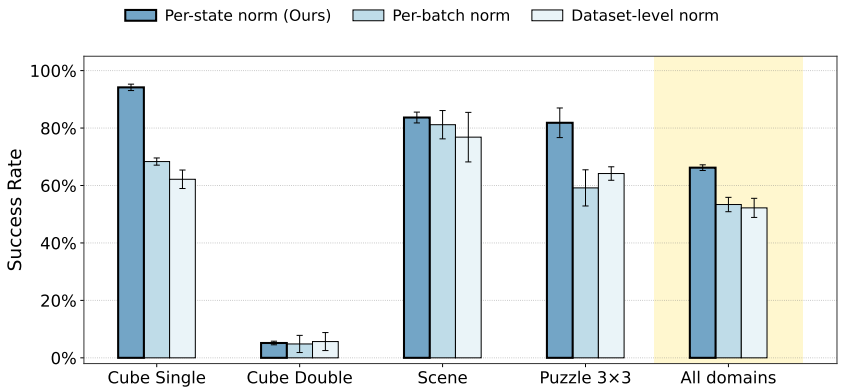

Figure 10: Ablating different advantage normalization schemes: our per-state advantage normalization outperforms batch-level and dataset-level normalization. Averaged over 5 tasks per domain and 4 seeds.

Contrastive guidance conditions and guidance weight. Additionally, we measure how the contrastive conditions (y_+, y_-) pair and the guidance weight w affect policy performance. Figure 11 shows the performance with 6 different contrastive pairs at each guidance weight and at the best guidance weight. With the best guidance weight w, different contrastive conditions perform similarly. This aligns with our derivation in Section 6.1, which shows that the optimality of the sampled action is determined by the product $w(y_+ - y_-)$, and so in practice it's sufficient to tune only one.

Advantage label standard deviation. Figure 12 ablates different values of standard deviation we use to sample the advantage label $Y \mid s, a \sim N(A(s,a), \sigma_y^2)$. To plot the performance, we use contrastive pair $(y_+, y_-) = (2, 0)$ for all variations, and pick the best guidance weight for each

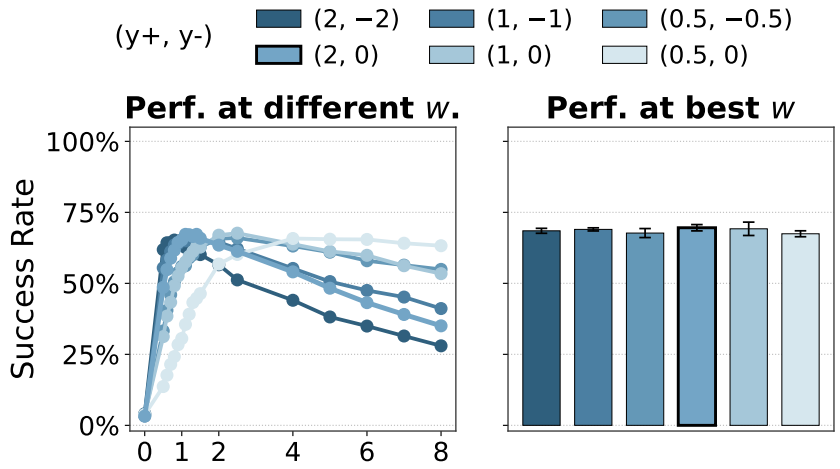

Figure 11: **Sensitivity to contrastive conditions and guidance weight:** each contrastive condition performs similarly given the best guidance weight for that condition.

variation on each domain according to our hyperparameter selection scheme in Section A.8. The results show that different values of σ_y^2 result in roughly similar levels of performance. This is perhaps not so surprising given that we showed in Section 6.1 the label standard deviation, guidance weight, and contrastive pair together determine the advantage tilt: $\beta = \frac{\sigma_y^2}{w(y_+ - y_-)}$. As a result, using different σ_y does not have a significant effect on performance as long as the guidance weight w is changed accordingly as well. This is similar to the finding in Figure 11. We also note that while our derivation in Section 6.1 require strictly $\sigma_y > 0$, $\sigma_y = 0$ works well in practice.

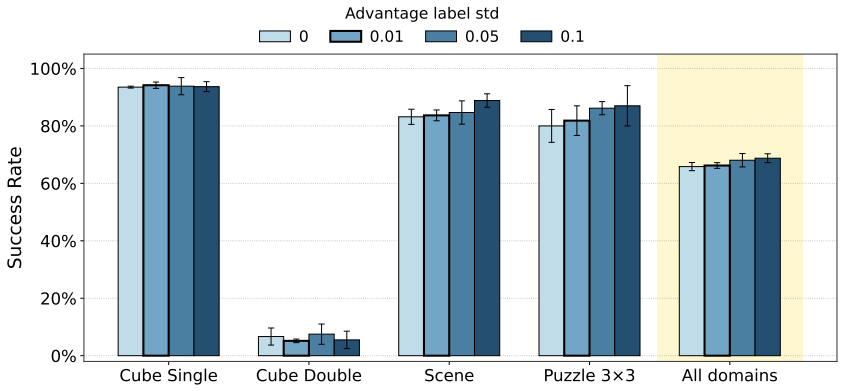

Figure 12: Sensitivity to different advantage label standard deviation: performance is not sensitive to the label standard deviations as long as the best guidance weight is picked. Averaged over 5 tasks per domain and 4 seeds.

A.3 Empirical Evaluation of Naïve Advantage Conditioned Policy RCP-A

Section 5 showed that a naïve advantage conditioned policy such as RCP-A suffers from the advantage alignment problem. We empirically evaluate that here, evaluating RCP-A on 20 OGBench tasks when queried with different advantage values. As described in Section A.6, RCP-A models the distribution of advantage value over the dataset by modeling it as a Gaussian distribution and learning its mean μ and variance σ^2. During inference, RCP-A queries with advantage $\mu + k\sigma$, which is k standard deviations above the mean dataset advantage. While the original paper (Kumar et al., 2019) always set $k = 1$, we experiment with different k values during inference and plot the result in Figure 13. The

results show that different query values has effectively no impact on performance, which corresponds with our empirical analysis in Section 5.

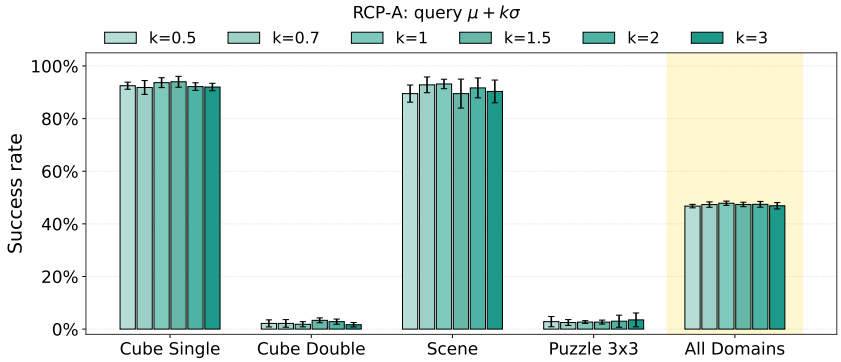

Figure 13: Naïve advantage conditioned policy RCP-A has the same performance across 20 OGBench tasks (5 tasks per domain) when queried with different advantage values. Averaged over 4 seeds.

A.4 Does A More Expressive Architecture Help with Advantage Alignment?

In this section, we investigate whether we can improve advantage alignment, and in turn policy performance, of naïve advantage conditioning by using a more expressive conditional policy architecture. Specifically, we experiment with using a transformer architecture to ingest the conditional advantage. We compare RCP-A, which uses a FiLM-like architecture that uses the advantage input to modulate the hidden states at every layer of the policy network, to a variant that instead uses a transformer to handle conditional inputs. We represent each action position in the $H = 5$ action chunk with a separate token. These action tokens, together with an advantage token, an observation token, and a flow-time token, are processed jointly through self-attention. Each contextualized action token is then projected directly to the velocity vector for its corresponding action position. We use four transformer blocks with model width 256, eight attention heads (32 dimensions per head), and feed-forward width 1024. The FiLM RCP-A actor has approximately 1.39M parameters, while the transformer actor has approximately 3.23M parameters. We tune the learning rate smaller ($1e-4$) for the transformer version to prevent overfitting.

We plot the performance comparison between RCP-A and its transformer variant, as well as Advantage Guidance, in Figure 14. Figure 14 (left) shows that using a transformer is not significantly better in terms of advantage alignment: while the transformer policy output varies more when advantage input is changed, the policy gradient is still invariant to advantage inputs. In comparison, Advantage Guidance is much more responsive to different advantage inputs. Figure 14 (right) plots the policy performance on OGBench. Overall, using a transformer does not affect policy performance much, and even slightly underperforms the original RCP-A agent. This shows that there is no free lunch: we cannot expect to improve advantage alignment by simply swapping in a transformer policy backBFNe, and the residual parameterization and action imagination training schemes are critical.

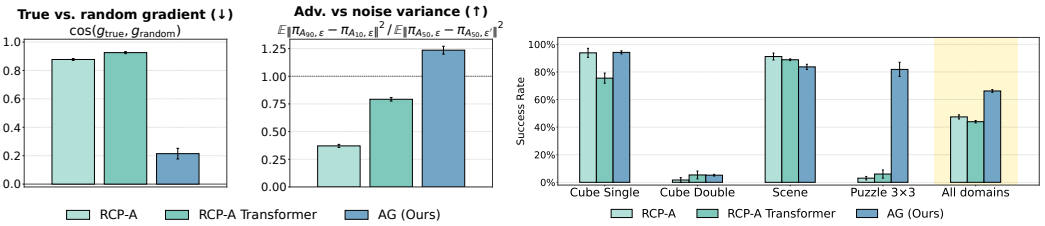

Figure 14: Using a transformer for the policy architecture does not significantly improve advantage alignment (left), and does not improve policy performance overall on OGBench (right).

A.5 Analysis of Pure Conditional Sampling with Advantage Guidance

In Section 7.3 we compared Advantage Guidance using different sampling schemes. In Figure 15, we plot the performance of Advantage Guidance with pure conditional sampling on different environments using different query advantages. On OGBench (top row), high query advantages leads to some performance decline, and experiences a sharp decline in `puzzle-3x3`, the longest horizon task. On OCBench, all tasks experience a sharp drop in performance at high advantage values. This shows that high advantage value is likely out of the training distribution on OCBench where the offline dataset is demonstration-based and therefore more narrow. `puzzle-3x3` could also suffer from the same issue because its state-action coverage is relatively smaller given its longer horizon for the same amount of offline transitions as other OGBench tasks.

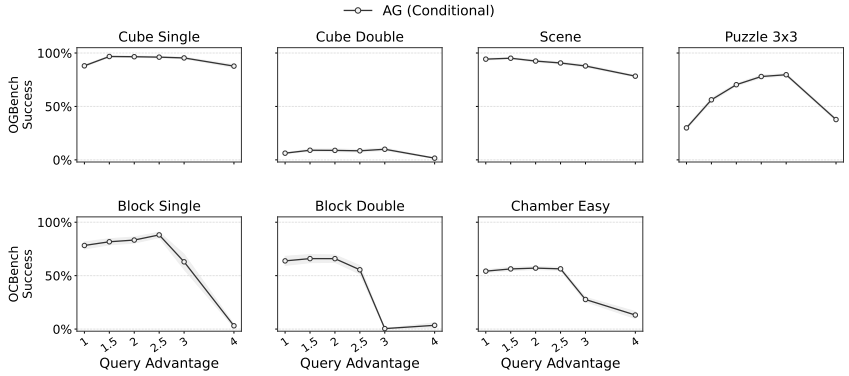

Figure 15: Pure conditional sampling of Advantage Guidance: querying with large advantage values leads to a sharp performance drop on OCBench and hard long horizon task `puzzle-3x3`. Averaged over different tasks in the same domain with 10 seeds.

A.6 Description of Baseline Methods

We describe in detail the different categories of baseline methods. While some methods are implemented as Gaussian policies or diffusion policies in the original paper, we implement the flow-matching variant for all methods for fair comparison.

For all the agents below, we will assume that noise $\epsilon \sim \mathcal{N}(0, I)$, time $t \sim \mathcal{U}(0, 1)$, and the expectations of training losses are taken over $(s, a) \sim \mathcal{D}$ unless otherwise specified.

(1) Actor-critic methods

FQL (Park et al., 2025) learns a distilled one-step policy π_ω to maximize the critic, avoiding the need to take action gradient through the multi-step denoising process with backpropagation through time. Specifically, policy extraction with FQL requires learning two actors, a multi-step BC actor π_θ, and a one-step policy π_ω. The BC actor is trained to minimize the standard flow matching loss $\mathbb{E}\left[||v_\theta(x_t, s, t) - (a - \epsilon)||^2\right]$, while the one-step actor is trained to minimize:

$$\mathbb{E}\left[-Q_\phi(s, a^\pi) + \alpha \left\|a^\pi - \mathrm{ODE}(v_\theta(s, \epsilon))\right\|_2^2\right] \quad \text{where} \quad a^\pi = v_\omega(s, \epsilon). \tag{17}$$

$\mathrm{ODE}(\cdot)$ is the Euler integration process, and α is a coefficient that controls the behavior regularization strength.

EDP (Kang et al., 2023) optimizes a multi-step flow policy π_θ to maximize the critic using a one-Euler-step approximation of the full denoising process, avoiding the full backpropagation through time. Specifically, given $x_t = (1 - t)\epsilon + ta$, we approximate the full denoising process with the one-Euler step integration $\hat{a}_1(x_t) = x_t + (1 - t)v_\theta(x_t, s, t)$, and train the policy to minimize:

$$\mathcal{L}_{\mathrm{EDP}}(\theta) = \mathbb{E}\left[-Q_\phi(s, \hat{a}_1(x_t)) + \alpha \left\|v_\theta(x_t, s, t) - (a - \epsilon)\right\|_2^2\right]. \tag{18}$$

The coefficient α controls the behavior regularization strength. During training, the action gradient passes through the one-step action approximation and is taken with respect to x_t.

(2) Supervised-learning-based methods

CRR (Wang et al., 2020) uses the advantage estimates to filter the actions used for policy training. We adopt its binary-weight variant, training the flow policy only on dataset actions with positive estimated advantages:

$$\mathcal{L}_{\text{CRR}}(\theta) = \mathbb{E}\left[\mathbf{1}\{A(s,a) > 0\} \left\| v_\theta(x_t, s, t) - (a - \epsilon) \right\|_2^2\right]. \tag{19}$$

This corresponds to behavior cloning on the positive-advantage subset.

FAWR is a baseline we construct by extending advantage weighted regression (AWR) (Peng et al., 2019) to flow-matching policies. AWR is introduced for Gaussian policies, where the log likelihood of each action is weighted by its exponentiated advantage. Unfortunately, this is not readily extendable to flow policies because the likelihood of diffusion and flow policies cannot be evaluated exactly. However, since diffusion (and under some constraints flow matching) optimize a lower bound on the log-likelihood (Kingma & Gao, 2023), we can optimize the lower bound of an AWR objective by directly using the flow matching loss to substitute the action log likelihood. Specifically, we weight each dataset action's flow-matching loss by its exponentiated advantage:

$$\mathcal{L}_{\text{FAWR}}(\theta) = \mathbb{E}\left[\exp\left(\frac{A(s,a)}{\beta}\right) \left\| v_\theta(x_t, s, t) - (a - \epsilon) \right\|_2^2\right]. \tag{20}$$

The temperature $\beta > 0$ controls how strongly training favors high-advantage actions. In practice, we clip $\exp(\frac{A(s,a)}{\beta})$ to have max value 100.

RCP-R and RCP-A (Kumar et al., 2019) train policies conditioned on an estimated return-to-go (RCP-R) and advantage (RCP-A). While the original implementation of both RCP-A and RCP-R use Gaussian policies, we instantiate both with flow policies. Specifically, for a conditional signal y (advantage or return-to-go), we optimize policy $\pi_\theta(a \mid s, y)$ with:

$$\mathcal{L}_{\text{RCP}}(\theta) = \mathbb{E}\left[\left\| v_\theta(x_t, s, t, y) - (a - \epsilon) \right\|_2^2\right]. \tag{21}$$

While the original RCP-R implementation estimates the return-to-go from (s, a) using Monte Carlo estimation, we directly use the lower variance estimate $Q(s, a)$ to maintain comparability with other methods. For RCP-A, we estimate the advantage as $A(s, a) = Q(s, a) - V(s)$. For both methods, the conditioning variable modulates the policy network's hidden features through FiLM (Perez et al., 2018), which applies multiplicative interactions of the hidden features and the conditional variable at every layer. Following Kumar et al. (2019), we estimate the mean μ_y and variance σ_y^2 of the conditioning variable using a running average, and, at inference time, query the policy with $y_{\text{query}} = \mu_y + k\sigma_y$. The original paper always use $k = 1$, whereas we sweep over different k values.

(3) Diffusion guidance methods

QGF (Zhou et al., 2026) is a test-time RL method that trains a critic and a BC flow policy with standard flow matching loss and optimizes actions solely at test time with critic gradients guidance during sampling. The main idea is to guide the action towards high-value directions at each denoising step using the critic's action gradient. To estimate the critic gradient at partially noised action x_t, QGF uses a single large Euler integration step to approximate the clean action and evaluate the critic gradient at this approximation $\hat{a}$. Specifically, QGF uses the following velocity:

$$\begin{aligned} \hat{a} &= x_t + (1 - t) v_{\hat{\pi}}(x_t, s, t), \\ v_{\text{QGF}}(x_t, s, t) &= v_{\hat{\pi}}(x_t, s, t) + w \, \nabla_a Q(s, a)|_{a=\hat{a}}. \end{aligned} \tag{22}$$

Here w controls the guidance strength. QGF omits the Jacobian of the one-step approximation, using the critic gradient directly at $\hat{a}$. This is akin to classifier guidance of the policy using the Q-function as a classifier.

CFGRL (Frans et al., 2025) applies classifier-free guidance between an unconditional policy trained on the full dataset and a policy conditioned on positive advantage. CFGRL trains two policies: a conditional policy $\pi(a \mid s, o)$, where $o \in \{0, 1\}$ is a binary random variable that indicates optimality (*i.e.*, $o = 1$ corresponds to optimal), and an unconditional policy $\pi(a \mid s)$. Both policies are trained with flow matching. Following the original implementation, we equate the unconditional policy $\pi(a \mid s)$ with the conditional $\pi(a \mid s, o = 0)$; optimality is defined as $A(s, a) = Q(s, a) - V(s) > 0$ and $\pi(a \mid s, o = 1)$ is trained only on data with positive advantage estimates. Conceptually, this is the same as training one unconditional policy on the whole dataset and another "filtered" policy on the positive-advantage subset. During inference, CFGRL integrates:

$$v_{\text{CFGRL}}(x_t, s, t) = v_{\hat{\pi}}(x_t, s, t) + w\left[v_+(x_t, s, t) - v_{\hat{\pi}}(x_t, s, t)\right]. \tag{23}$$

The CFG weight w controls the contribution of the positive-advantage policy.

With additional test-time compute:

Best-of-N (BFN) performs best-of-N sampling of a base behavior cloning policy $\hat{\pi}$ trained with the flow matching objective. Specifically, given a critic $Q(s, a)$, BFN selects action

$$a = \underset{a_1, \cdots, a_N \sim \hat{\pi}}{\arg\max}\ Q(s, a_i). \tag{24}$$

A.7 Critic Training Details

As described in Section 7, we follow the action chunking setting and learn both actors that produce action chunks and critics that are conditioned on action chunks.

Our method is agnostic to how the advantage estimate is derived. For simplicity, in this paper we train value functions with IQL (Kostrikov et al., 2021) and derive the advantage function $A(s, a) = Q(s, a) - V(s)$. The IQL objective is as follows:

$$\mathcal{L}_Q(\theta) = \mathbb{E}_{(s_t, a_{t:t+h}, r_{t:t+h}, s_{t+h}) \sim \mathcal{D}}\left[\left(Q_\theta(s_t, a_{t:t+h}) - \left(\sum_{t'=t}^{t+h-1} [\gamma^{t'-t} r'_t] + \gamma^h\, V_{\bar{\psi}}(s_{t+h})\right)\right)^2\right], \tag{25}$$

$$\mathcal{L}_V(\psi) = \mathbb{E}_{(s_t, a_{t:t+h}) \sim \mathcal{D}}\left[L_2^\tau\big(Q_{\bar{\theta}}(s_t, a_{t:t+h}) - V_\psi(s_t)\big)\right], \tag{26}$$

where $L_2^\tau(u) = |\tau - \mathbf{1}(u < 0)|u^2$ is the expectile loss with expectile τ, and $\bar{\theta}, \bar{\psi}$ are the target networks computed with EMA. We train all IQL state-action value functions with an ensemble of 2 $Q(s, a)$ and aggregate them by taking the minimum.

A.8 Hyperparameter and Tuning Details

The main results in Figure 4 were run with 10 random seeds, and all ablation studies were run with 4 random seeds unless otherwise specified. All error bars report the 95% confidence interval.

Hyperparameter tunning. For each method, including Advantage Guidance and all baseline methods, we tune the task-specific hyperparameter per domain in the ranges specified in Table 4. We tune hyperparameters with 4 seeds, and re-run the experiments with 10 new seeds using the chosen hyperparameter to report performance. Specifically, on OGBench, each domain contains 5 similar tasks of increasing hardness. We therefore tune hyperparameters per domain on tasks 2 and 4 only, and use the picked hyperparameter for all 5 tasks on that domain. Tunning on task 2 and 4 represents an average performance on that domain since task 1 to 5 gradually increases in difficulty. On OCBench, there are different numbers of tasks available per domain, and different tasks in the domain can be very different. Therefore, we tune the hyperparameter per-task per domain in OCBench. The resulting hyperparameters we use are documented in Table 3.

For all methods, we use shared hyperparemters in Table 1. For Advantage Guidance, we use the same value per benchmark for σ_y, and the same y_+, y_-, K universally (see Table 2). This is because solely tunning the guidance weight is sufficient: we can observe this from the theoretical derivation in Section 6.1, and we also empirically observe this in our ablation studies in Figure 12 and Figure 11. We use these hyperprameters for Advantage Guidance in all experiments unless otherwise specified. We use $\sigma_y = 0.1$ on OGBench instead of 0.01 because Figure 12 shows it to be slightly more performant, though the advantage is tiny. We did not tune σ_y on OCBench.

Hyperparameter	OGBench	OCBench
Batch size	256	1024
Discount factor γ	0.99	0.999
IQL expectile τ	0.9	0.5
Critic ensemble size	2	2
Critic aggregation method	min	min
Target-network update rate	0.005	0.005
Action chunk horizon H	5	3
Flow integration / denoising steps	10	10
Actor hidden dimensions	$[512, 512, 512, 512]$	$[1024, 1024, 1024, 1024]$
Critic/value hidden dimensions	$[128, 128]$	$[1024, 1024, 1024, 1024]$
Actor / Critic Learning rate	3×10^{-4}	1×10^{-4}
Optimizer	Adam	Adam
Offline training steps	5×10^5	1×10^6 (1×10^5 for AG on Block Single)
Evaluation episodes	30	500

Table 1: **Common hyperparameters** used for the OGBench and OCBench experiments in Figure 4.

Hyperparameter	OGBench	OCBench
Positive condition y_+	1	1
Negative condition y_-	-1	-1
Advantage label standard deviation σ_y	0.1	0.01
Number of Imagined Actions K	7	7

Table 2: **Common hyperparameters for Advantage Guidance**. As shown in Figure 12 and Figure 11, different values of y_+, y_-, σ_y hyperparameters lead to very similar results. We therefore fix these parameters across tasks on the same benchmark and only tune the guidance weight for Advantage Guidance.

A.9 Tasks and Environment Details

We use all the available task for each domain in two of the benchmarks. OCBench offers different numbers of tasks per domain, ranging from 1 to 3. OGBench tasks all have offline datasets with 1M transitions. OCBench block single and chamber easy has 10k offline transitions, while block double has 30k transitions.

Figure 16 shows illustrations of the OCBench tasks. Illustrations of OGBench tasks can be found in Park et al. (2024a).

A.10 Detailed Results

Suite	Domain	EDP (α_{EDP})	FQL (α_{FQL})	QGF (w_{QGF})	CFGRL (w_{CFGRL})	FAWR (β)	RCP-A (k)	RCP-R (k)	CRR	AG (w)
OGBench	Cube Single	100	1000	0.08	2	0.3	1	1	–	1
OGBench	Cube Double	10	100	0.16	2	0.3	1	1	–	1.5
OGBench	Scene	30	300	0.16	2	10	1	1	–	1
OGBench	Puzzle 3×3	30	300	0.16	3	0.3	1	1	–	2.5
OCBench	Block Single	1	300	0.02	10	3	1	1	–	1.5
OCBench	Block Double	$(1, 1)$	$(100, 100)$	$(0.32, 0.32)$	$(15, 15)$	$(3, 0.3)$	1	1	–	$(3, 4)$
OCBench	Chamber Easy	$(1, 1, 1)$	$(100, 100, 100)$	$(0.32, 0.32, 0.32)$	$(5, 15, 10)$	$(1, 1, 1)$	1	1	–	$(4, 3, 2)$

Table 3: **Domain-specific hyperparameters** used for OGBench and OCBench results in Figure 4. For OCBench, parenthesized values are ordered by task number. For example, $(3, 4)$ denotes the values used for Tasks 1 and 2 of Block Double. A dash indicates that the method has no tuned domain-specific hyperparameter.

Method	Hyperparameter	Tunning Range
EDP	BC coefficient α_{EDP}	OGBench: $\{10, 30, 100, 300\}$; OCBench: $\{1, 10\}$
FQL	BC coefficient α_{FQL}	$\{100, 300, 1000\}$
QGF	Guidance weight w	$\{0, 0.002, 0.004, 0.01, 0.02, 0.04, 0.08, 0.16, 0.32\}$
CFGRL	Guidance weight w	OGBench: $\{0, 0.5, 1, 2, 3, 5, 10, 15\}$; OCBench: $\{0, 0.5, 1, 1.5, 2, 3, 5, 10, 15\}$
FAWR	Inverse temperature β	OGBench: $\{0.3, 1, 3, 10\}$; OCBench: $\{0.3, 1, 3\}$
RCP-A	inference std multiplier k	OGBench: $\{0.5, 0.7, 1, 1.5, 2, 3\}$; OCBench: $\{1\}$
AG (Ours)	Guidance weight w	OGBench: $\{0, 0.75, 1, 1.25, 1.5, 2, 2.5, 4\}$; OCBench: $\{0, 0.5, 1, 1.5, 2, 3, 4\}$

Table 4: Hyperparameter ranges for OGBench and OCBench experiments.

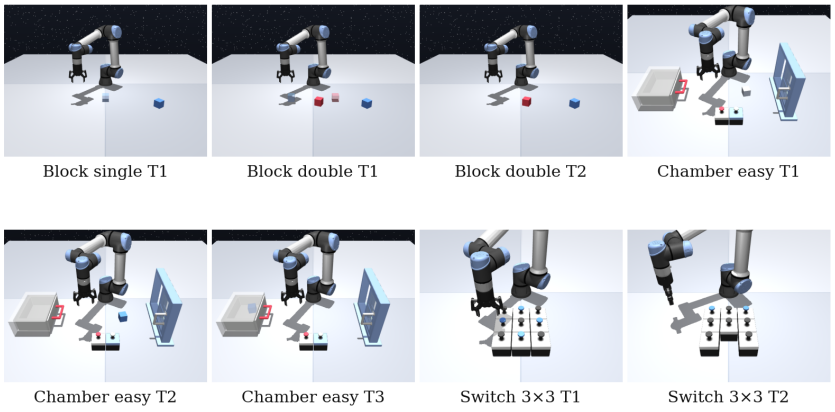

Figure 16: Illustration of OCBench tasks.

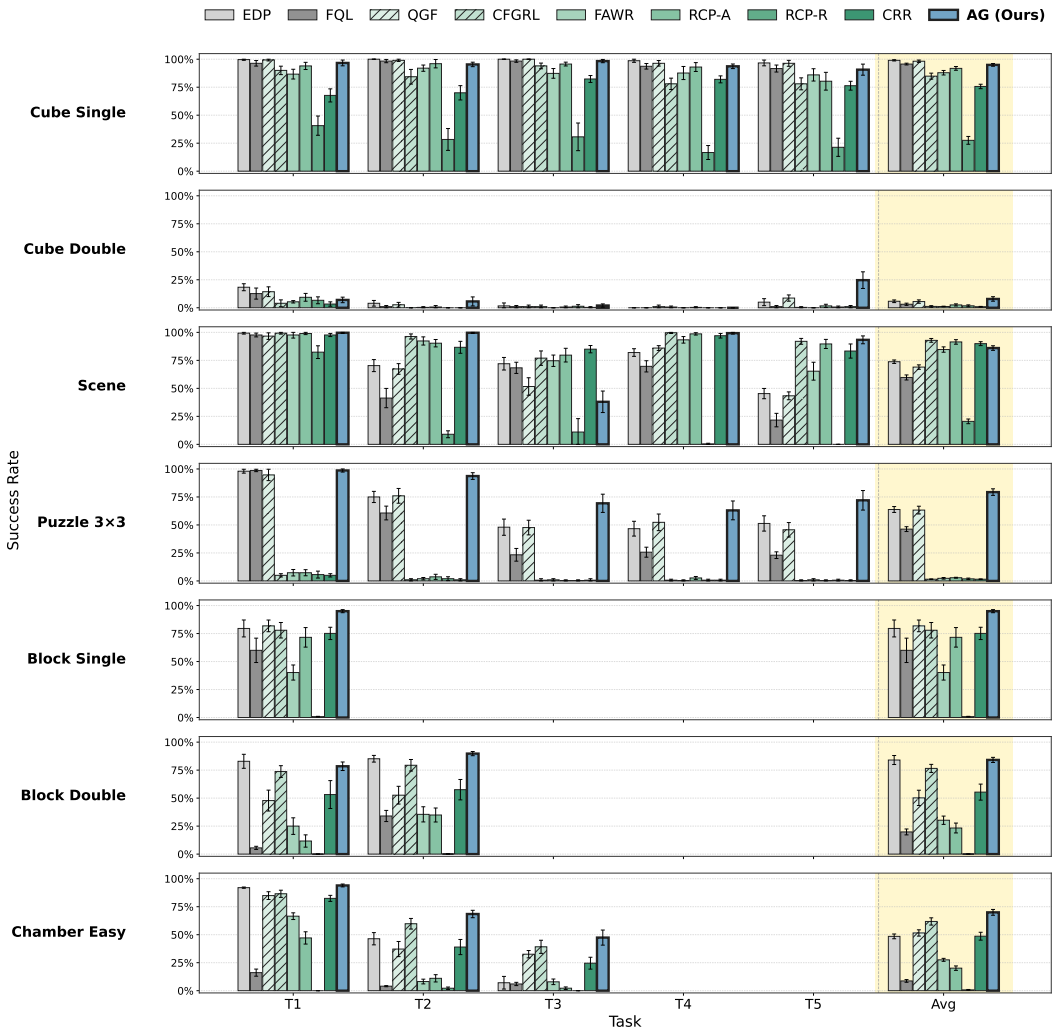

Figure 17: Task breakdown of Figure 4: Comparing Advantage Guidance against prior methods in offline RL setting on 20 OGBench tasks and 6 OCBench tasks. 10 random seeds.

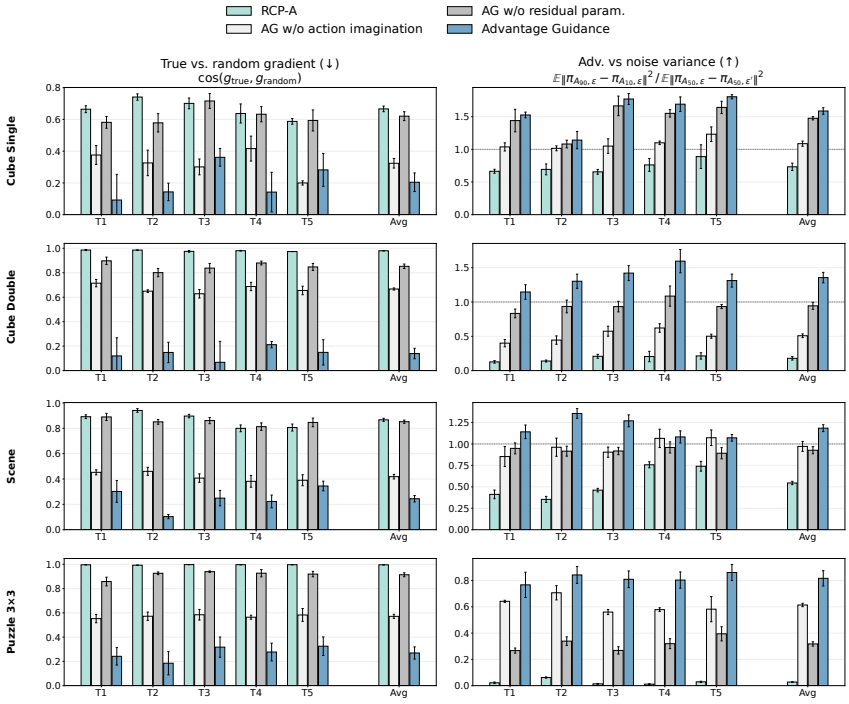

Figure 18: Task breakdown of Figure 6: Comparing (1) gradient similarity between gradient of policy loss conditioned on true and random advantage values and (2) the variance of policy output conditioned on different advantage values or initial noise.

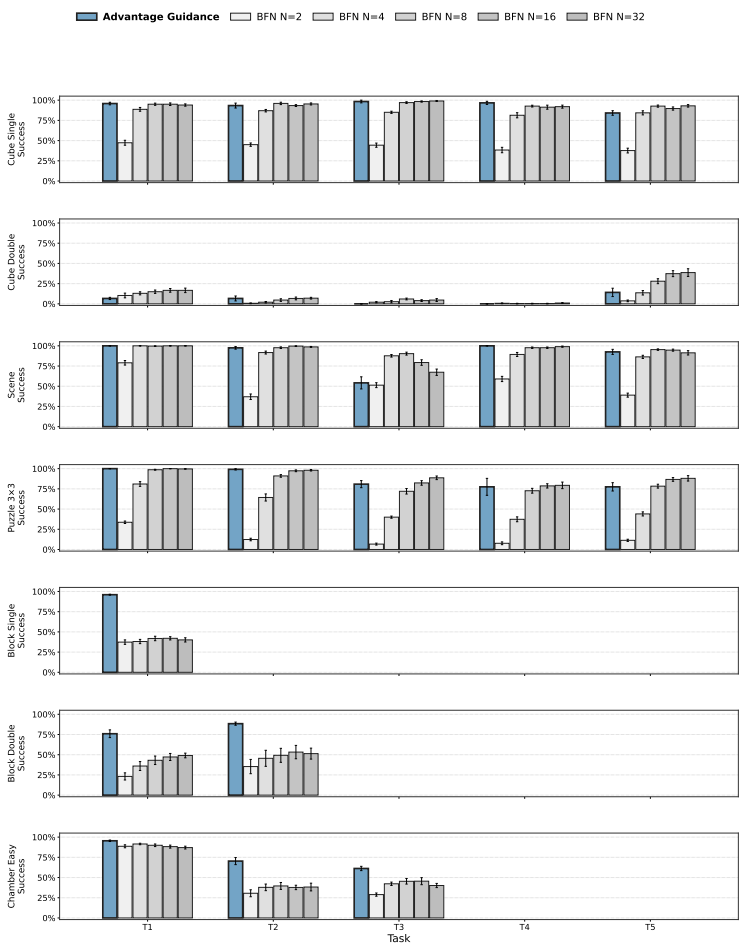

Figure 19: Task breakdown of Figure 5, comparing Advantage Guidance with best-of-N (BFN) sampling at different values of N on OGBench and OCBench.

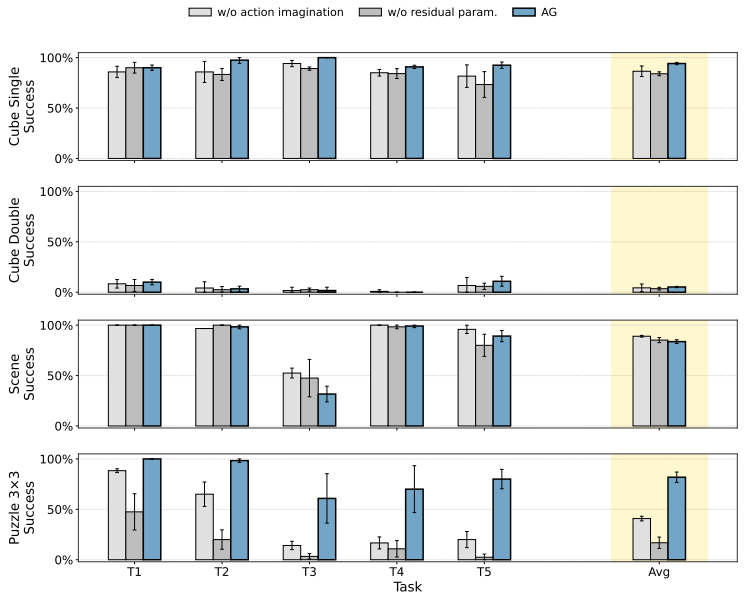

Figure 20: Task breakdown of Figure 7: ablation of residual parameterization and action imagination.

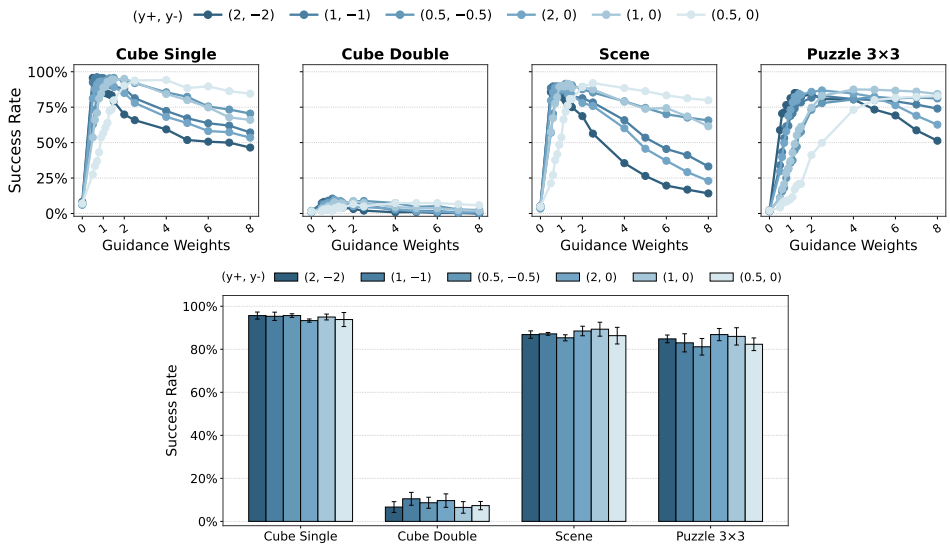

Figure 21: Task breakdown of Figure 11: sensitivity to contrastive condition pair (y_+, y_-) and guidance weight w. *Top:* Performance at different guidance weights. *Bottom:* Performance at best guidance weight.

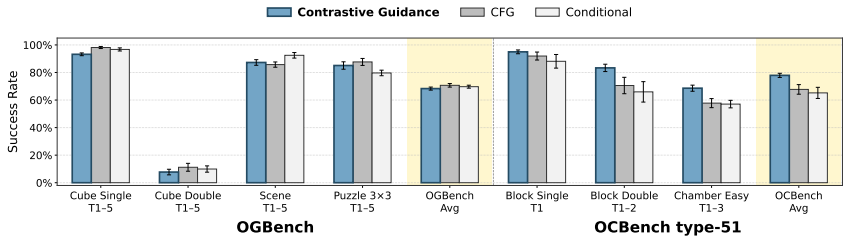

Figure 22: Domain breakdown of Figure 8: Ablation of sampling method for Advantage Guidance. 10 random seeds.