FlowDPG: Deterministic Policy Gradient on Flow Matching Policies for Real-World Manipulation

Author Names Omitted for Anonymous Review.

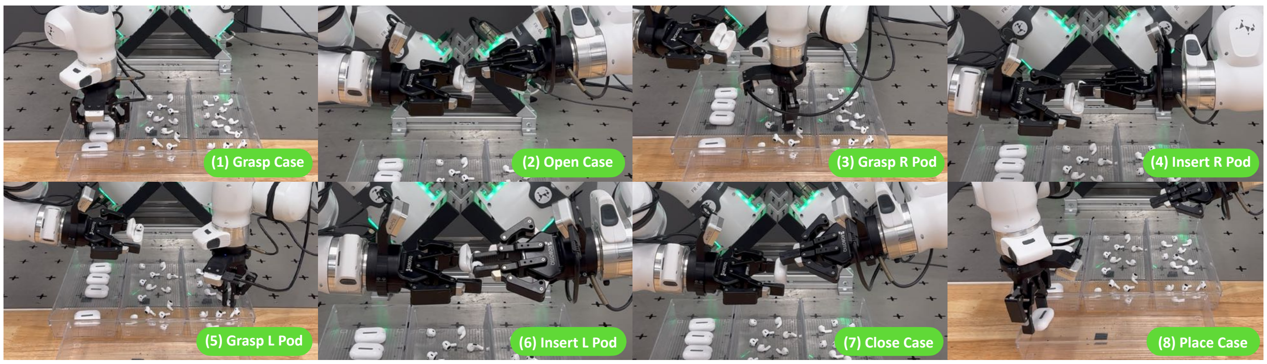

Fig. 1. The long-horizon, contact-rich, dual-arm AirPods assembly task used throughout this paper: 8 sequential sub-stages (grasp, open, insert, close, place) alternating between arms, with millimeter-level insertion tolerance. A single policy executes all stages end-to-end from raw observations, with case and pods at randomized poses and no stage-level resets or hand-engineered switching.

***Abstract*—Real-world reinforcement learning for robotic manipulation is especially hard for flow matching policies: applying policy gradient methods is fundamentally limited by the need to backpropagate through time (BPTT) along the multi-step ODE that maps noise to actions, which is computationally prohibitive and numerically fragile. We propose FlowDPG, a DDPG-style method for flow matching policies that distills the critic gradient into the velocity field at training time, bypassing BPTT entirely. Intuitively, FlowDPG combines two complementary vectors: the demonstration-driven velocity that keeps the action feasible, and the critic-driven correction that steers it toward higher value. Our contributions are threefold: (1) a BPTT-free distillation framework enabling stable DDPG-style policy improvement on flow matching policies, (2) a formal connection between the FlowDPG update and vanilla Deterministic Policy Gradient via three explicit approximations, and (3) real-world validation on a long-horizon, multi-stage, dual-arm AirPods assembly task, where FlowDPG attains a 92% end-to-end success rate, substantially outperforming recent RL methods spanning value-conditioning, auxiliary-module adaptation, and adjoint-based critic-gradient approaches. Videos and more results are on the project page https://flowdpg.github.io.**

I. Introduction

Imitation learning alone is insufficient for long-horizon, contact-rich manipulation. Collecting robot demonstrations at the scale and signal-to-noise ratio of language datasets is prohibitively expensive, and a supervised policy inherits its demonstration distribution and degrades sharply in unseen settings; on a long-horizon task, rare per-stage failures compound into catastrophic outcomes. This motivates letting the robot learn from its own actions through real-world reinforcement learning, which can push the policy beyond the original data manifold.

Recent state-of-the-art manipulation policies have converged on expressive generative action heads such as flow matching [15] and diffusion [6], which produce multi-modal action chunks through a multi-step denoising ODE. For these policies, the standard off-policy actor-critic update [14]—propagating the critic gradient $\nabla_a Q$ back through the actor—requires backpropagation through the entire denoising ODE. This is computationally prohibitive (cost grows linearly with solver steps) and numerically fragile (the chain of Jacobians compounds gradient explosion/vanishing, especially when the velocity field is not yet well-trained). As a result, expressive flow policies and value-driven RL have been hard to combine without either discarding the critic gradient or biasing the policy via coarse one-step approximations.

We propose **FlowDPG**, an off-policy actor-critic method for flow matching policies that obtains a stable critic-gradient update without backpropagating through the denoising ODE. The key insight is that a clean action x_1 can be estimated from any intermediate noisy action x_t by a single forward pass of the velocity predictor—a property unique to the flow-matching parameterization. We evaluate the critic gradient at this projected clean action, combine it with the demonstration-driven velocity into a value-improved target, and distill the result back into the velocity field via L2 regression. Intuitively, FlowDPG composes two complementary vectors: the demonstration-driven velocity that keeps the action feasible, and the critic-driven correction that steers it toward higher value. Inference is unchanged—a pure ODE solve with no deployment-time critic queries. Crucially, the FlowDPG update can be derived from the vanilla DPG gradient under three explicit, controllable approximations (Sec. III-B), placing it on a transparent footing relative to the classical DDPG literature, in contrast to recent critic-gradient approaches built on more abstract stochastic-control formulations [13].

We validate FlowDPG on a long-horizon, contact-rich, dual-arm AirPods assembly task with millimeter-level insertion tolerance, where a single policy executes eight sequential bimanual stages end-to-end from raw observations. FlowDPG reaches 92% end-to-end success—a 28% gain over the BC base and a 12% margin over the strongest prior RL baseline. In summary, our contributions are **(1)** a BPTT-free distillation framework enabling stable DDPG-style policy improvement on flow matching policies, **(2)** a formal connection between FlowDPG and vanilla DPG via three explicit approximations, and **(3)** real-world validation on long-horizon dual-arm assembly, with mechanism-level ablations supporting each design choice.

II. Related Work

a) Generative action heads for manipulation.: Large-scale demonstration datasets [18, 8, 10] have enabled generalist manipulation policies, evolving from transformer policies [4, 3] to vision-language-action models [11, 2, 17] that inherit representational strength from large VLMs [26, 7, 1]. At the action level, policies have moved from Gaussian heads to expressive generative heads: action-chunking transformers [32], diffusion policies [6, 24], and flow matching [15, 2, 16], which provides an ODE-based map from noise to actions. We focus on flow matching policies as the actor class.

b) RL fine-tuning of expressive policies.: One family discards the critic's action gradient and uses only scalar value/advantage signals: RA-BC [5] reweights the BC objective by progress, AWR [21] fits the policy to advantage-weighted demonstrations, and RECAP [22] conditions on advantage. A second family freezes the base policy and learns a separate adaptation module: DSRL [27] performs RL in the denoiser's latent noise space, PLD [30] learns a single-step residual, and RLT [31] attaches a lightweight actor-critic head. The closest family exploits the critic gradient directly: Q-score matching [23] relates the optimal policy's score to Q-gradients, while QAM [13, 28] converts $\nabla_a Q$ into step-wise supervision through an auxiliary adjoint flow. FlowDPG also uses first-order critic information, but distills a local critic-gradient correction directly into the velocity field via a simple DDPG-style objective, with an explicit derivation as an approximation of vanilla DPG. For value estimation we use IQL [12] with twin critics [9] and a stage-aware reward predictor [5].

III. FlowDPG

On top of a flow matching policy, our method has two coupled components (Fig. 2, 3): a twin-critic IQL value estimator

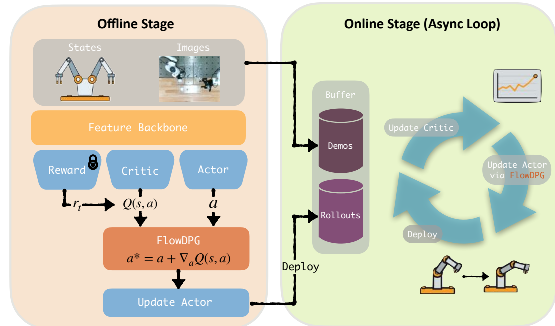

Fig. 2: Offline-to-online pipeline. **Offline (left):** a shared visual backbone feeds a frozen reward predictor, a critic, and a flow matching actor; FlowDPG combines the actor output a with the critic gradient $\nabla_a Q(s, a)$ into a value-improved target a^* distilled back into the actor. **Online (right):** the actor is rolled out on the real dual-arm setup, and critic and actor are asynchronously updated from a replay buffer mixing demonstrations with fresh rollouts.

(Sec. III-A), and **FlowDPG**, which distills the deterministic policy gradient (DPG) [25, 14] into the flow velocity field at training time (Sec. III-B).

A. Value Estimation with Twin-Critic IQL

We adopt Implicit Q-Learning (IQL) [12] with twin critics [9]. IQL replaces the $\max_a Q(s, a)$ bootstrap, which queries the critic on out-of-distribution (OOD) actions, with a value function $V_\psi(s)$ estimating a high-quantile in-distribution action-value, eliminating extrapolation error. Twin critics Q_{ϕ_1}, Q_{ϕ_2} mitigate overestimation; we take their minimum wherever a single Q is needed and pair each with an EMA target. V_ψ is trained by expectile regression of the twin-critic minimum on dataset actions,

$$\mathcal{L}_V(\psi) = \mathbb{E}_{\mathcal{D}}\big[\rho_\tau\big(\min_{i=1,2} Q_{\bar{\phi}_i}(s, \boldsymbol{a}) - V_\psi(s)\big)\big], \tag{1}$$

with $\rho_\tau(u) = |\tau - \mathbb{1}[u < 0]| \cdot u^2$ and expectile $\tau > 0.5$ pulling $V_\psi(s)$ toward the upper tail. The critics are bootstrapped from V_ψ via the H-step target $y_Q = r_t + \gamma^H V_\psi(s_{t+H})$:

$$\mathcal{L}_Q(\phi_i) = \mathbb{E}_{\mathcal{D}}\big[\big(Q_{\phi_i}(s_t, \boldsymbol{a}_t) - y_Q\big)^2\big], \tag{2}$$

again avoiding extrapolation to unseen actions. Chunk rewards r_t come from SARM [5], a frozen stage-aware predictor emitting a stage label $z(o_t)$ and within-episode progress $\phi(o_t) \in [0, 1]$, combined into

$$r_t = \underbrace{\phi(o_{t+H}) - \phi(o_t)}_{\text{progress}} + \underbrace{\beta_{\text{stage}} \cdot \mathbb{1}\big[z(o_{t+H}) > z(o_t)\big]}_{\text{stage}} + \underbrace{r_t^{\text{term}}}_{\text{terminal}} \tag{3}$$

i.e. dense per-step shaping, a fixed bonus at each predicted stage advance, and a whole-episode success term (Appendix B-C).

B. Policy Extraction via FlowDPG

A flow policy transports noise x_0 to a feasible action x_1 within the demonstration support, but x_1 is not necessarily optimal under the critic. FlowDPG combines *two vectors* (Fig. 3): the demonstration-driven velocity for feasibility, and a critic-driven correction along $\nabla_a Q$ for value improvement.

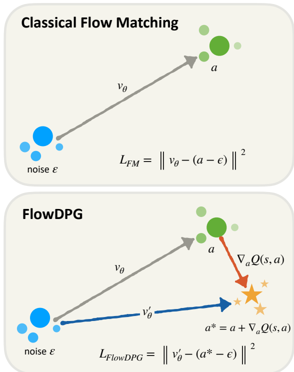

Fig. 3: **Top:** flow matching transports noise to a demonstration-feasible action a via v_θ. **Bottom:** FlowDPG adds a critic-driven correction along $\nabla_a Q(s,a)$ to reach a value-improved target a^*, then distills the resulting velocity v'_θ back into the flow field.

The combined target is distilled back into v_θ via one L2 regression, avoiding backpropagation of $\nabla_a Q$ through the full ODE. FlowDPG costs one extra forward pass and one critic-gradient evaluation per update, and leaves inference as a pure ODE solve.

a) Q-gradient.: The clean action x_1 is estimated from any intermediate x_t by a single forward pass of v_θ,

$$\hat{a} := x_t + (1-t) \cdot v_\theta(x_t, t, s), \tag{4}$$

which satisfies $\hat{a} \approx x_1$ when v_θ is accurate at (x_t, t), an approximation enforced by the consistency regularizer below. We compute $\hat{a}$ under `no_grad` and evaluate the critic gradient there,

$$g = \nabla_{\hat{a}} [\min_{i=1,2} Q_{\phi_i}(s, \hat{a})], \tag{5}$$

with stop-grad on the critic.

b) Adaptive shift.: As $\|g\|$ varies widely across states and steps, we normalize the step to the local velocity scale,

$$\Delta = \alpha \cdot \frac{\|u_t\|}{\|g\| + \varepsilon} \cdot g, \tag{6}$$

with base guidance scale $\alpha > 0$ and stabilizer ε.

c) Distilled target.: Summing gives the Q-improved clean action and velocity target,

$$a^* = \hat{a} + \Delta, \qquad u_t^* = a^* - x_0, \tag{7}$$

regressed alongside a behavior-cloning anchor,

$$\mathcal{L}_{\text{FlowDPG}}(\theta) = \lambda \underbrace{\|v_\theta - u_t^*\|^2}_{\mathcal{L}_{\text{distill}}} + (1-\lambda) \underbrace{\|v_\theta - u_t\|^2}_{\mathcal{L}_{\text{BC}}}, \tag{8}$$

($v_\theta = v_\theta(x_t, t, s)$), with λ warming up linearly from 0 over N_{warmup} steps so the unreliable early critic does not dominate.

d) Consistency regularizer.: A projection off the clean-action manifold leaves the critic queried OOD and g unreliable. Since the flow loss only supervises local velocity, we add

$$\mathcal{L}_{\text{cons}}(\theta) = \mathbb{E}\big[\|\hat{a} - x_1\|^2\big]. \tag{9}$$

Substituting $x_t = (1-t)x_0 + t x_1$ into Eq. 4 gives $\hat{a} - x_1 = (1-t)(v_\theta - u_t)$, so $\mathcal{L}_{\text{cons}}$ is a velocity-error penalty weighted by $(1-t)^2$, emphasizing small-t samples where the projection error is most amplified.

Algorithm 1 FlowDPG: Offline-to-Online Training

Require: Buffers $\mathcal{D}_{\text{off}}$, $\mathcal{D}_{\text{on}}$ (empty); scales $\alpha, \mu_{\text{cons}}, \lambda_{\max}$; warmup N_{warmup}; EMA rate ρ; expectile τ.
Pretrain SARM; init backbone, v_θ, $(V_\psi, Q_{\phi_1}, Q_{\phi_2})$; targets $Q_{\bar{\phi}_i} \leftarrow Q_{\phi_i}$

Offline phase

for $k = 1, \ldots, N_{\text{off}}$ **do**
  Sample batch from $\mathcal{D}_{\text{off}}$; get r_t; draw x_0, t; form x_t, u_t
  $v_t \leftarrow v_\theta(x_t, t, s)$; $\hat{a} \leftarrow x_t + (1-t)v_t$; $\mathcal{L}_{\text{cons}} \leftarrow \|\hat{a} - x_1\|^2$
  $g \leftarrow \nabla_{\hat{a}} \min_i Q_{\phi_i}(s, \hat{a})$ (stop-grad on critic)
  $\Delta \leftarrow \alpha \|u_t\| g / (\|g\| + \varepsilon)$; $u_t^* \leftarrow \hat{a} + \Delta - x_0$
  $\lambda \leftarrow \lambda_{\max} \min(k / N_{\text{warmup}}, 1)$
  Update θ by $\mathcal{L}_{\text{FlowDPG}} + \mu_{\text{cons}} \mathcal{L}_{\text{cons}}$ (Eqs. 8, 9)
  Update ψ, ϕ_1, ϕ_2 (Eqs. 1–2); EMA $\bar{\phi}_i \leftarrow \rho \bar{\phi}_i + (1-\rho)\phi_i$
end for

Online phase

Deploy π_θ for autonomous rollout collection
for $k = 1, \ldots, N_{\text{on}}$ **do**
  Append rollouts to $\mathcal{D}_{\text{on}}$; sample $\mathcal{D}_{\text{off}} \cup \mathcal{D}_{\text{on}}$ at $\sim$ 1:1
  Update $\theta, \psi, \phi_1, \phi_2$ as above (backbone, SARM frozen)
  if $k \bmod N_{\text{sync}} = 0$ **then**
    Synchronize π_θ to deployment
  end if
end for
return v_θ

e) Relationship to vanilla DPG.: Treating the ODE terminus x_1 as a deterministic actor, a vanilla DPG [25, 14] update yields, by the chain rule (Appendix A),

$$\nabla_\theta \mathcal{L}_{\text{DPG}} = -\int_0^1 g^\top \Phi(1,t) \cdot \nabla_\theta v_\theta(x_t, t, s)\, dt, \tag{10}$$

where $\Phi(1,t) := \partial x_1 / \partial x_t$ is the ODE sensitivity matrix—computing it requires backpropagating through all T ODE steps with $O(T)$ memory and the usual explosion/vanishing pathologies. Differentiating $\mathcal{L}_{\text{FlowDPG}}$ instead, with g detached, gives

$$\nabla_\theta \mathcal{L}_{\text{FlowDPG}} = -2\lambda\alpha \|u_t\| \cdot \hat{g}(\hat{a}) \cdot \nabla_\theta v_\theta(x_t, t, s), \tag{11}$$

$\hat{g}(\hat{a}) := g / \|g\|$. This recovers Eq. 10 under three approximations: *(i)* evaluating the critic gradient at $\hat{a}$ rather than x_1, collapsing $\Phi(1,t)$ to the identity (kept faithful by $\mathcal{L}_{\text{cons}}$); *(ii)* a single-$t$ Monte Carlo estimate of the trajectory integral, as in flow-matching training; and *(iii)* the adaptive scaling $2\lambda\alpha\|u_t\|$ in place of $\|g\|$, removing a known DDPG instability.

C. Offline-to-Online Pipeline

The two components slot into Fig. 2 (Algorithm 1). In the **offline phase**, backbone, action head, and value head are jointly trained on demonstrations $\mathcal{D}_{\text{off}}$ with the FlowDPG objective, producing a base policy. In the **online phase**, the policy collects real-world rollouts asynchronously appended to $\mathcal{D}_{\text{on}}$; the learner samples at $\sim$1:1 from $\mathcal{D}_{\text{off}} \cup \mathcal{D}_{\text{on}}$ to balance demonstration quality and fresh experience. Backbone and reward predictor are frozen to preserve representations and reduce compute; updated actor/critic checkpoints synchronize to deployment periodically.

IV. Experiments

Task and protocol. We evaluate on a long-horizon, dual-arm AirPods assembly task (Fig. 1): 8 sequential sub-stages,

Method	Grasp case	Open case	Grasp R pod	Insert R pod	Grasp L pod	Insert L pod	Close case	Place case	**Overall**	**Avg**
BC (base)	100%	78.32%	72.44%	68.54%	78.56%	66.23%	92%	96%	64%	81.51%
Value-conditioning										
RA-BC	100%	86.15%	80.52%	64.52%	82.49%	68.46%	96%	96%	76%	84.27%
AWR	100%	80.65%	78.43%	72.45%	78.77%	70.65%	100%	100%	76%	85.12%
RECAP	100%	89.29%	80.62%	69.69%	76.54%	64.52%	96%	100%	72%	84.58%
Auxiliary-module adaptation										
DSRL	100%	79.85%	75.47%	70.59%	78.13%	67.94%	92%	92%	68%	82.00%
PLD	100%	85.27%	80.13%	79.82%	80.43%	79.59%	96%	96%	76%	87.16%
RLT	100%	86.93%	82.41%	77.92%	84.14%	80.18%	96%	100%	80%	88.45%
Critic-gradient										
QAM	100%	88%	83.33%	70.58%	73.52%	80%	100%	100%	80%	86.93%
FlowDPG (offline)	100%	89.29%	90.63%	84.38%	96.43%	83.33%	100%	100%	88%	93.01%
FlowDPG (+online)	**100%**	**89.29%**	**92.31%**	**86.21%**	**96.67%**	**88.89%**	**100%**	**100%**	**92%**	**94.17%**

TABLE I: Per-stage, overall, and average success rates on the AirPods assembly task.

randomized case and pod poses, two Franka arms, and demonstrations collected via GELLO [29]. Each policy runs on 25 cases (50 pod manipulations); within-stage failures do not terminate the episode and retries are allowed. We report per-stage success, the unweighted stage mean (**Avg**), and end-to-end success (**Overall**), which requires all pods correctly inserted.

Baselines. We compare three families of RL methods for adapting a base flow policy (Sec. II): *value-conditioning* (RA-BC [5], AWR [21], RECAP [22]), *auxiliary-module* adapters (DSRL [27], PLD [30], RLT [31]), and the concurrent *critic-gradient* method QAM [13]. All baselines share the same backbone, action chunk horizon, reward, value head, and offline/online data split as FlowDPG.

A. Main Results

Three patterns in Table I connect each method's mechanism to its empirical ceiling. *Value-conditioning is bounded by the demonstration support*: reweighting or conditioning over existing data cannot synthesize OOD actions, so these methods improve on mode-selection stages (Open/Close case) but stay near the BC base on contact-rich stages (Insert R/L pod). *Auxiliary-module methods leave the base policy untouched*: DSRL optimizes the noise space (separated from the action by the entire ODE), PLD's single-step residual cannot reshape the multi-step denoising, and RLT's MLP actor—strongest of the three but discarding multi-modal expressivity—caps headroom at the adapter. *Critic-gradient methods directly update the base policy*: FlowDPG uses $\nabla_a Q$ to update the velocity field itself, removing both ceilings. The concurrent QAM [13] pursues the same direction via adjoint matching but needs an auxiliary adjoint flow and step-wise supervision at every timestep, making it sensitive to critic noise; FlowDPG distills a single critic-gradient evaluation via one L2 regression, derived as an approximation of vanilla DPG (Sec. III-B).

B. Online RL Improves Robustness to Unseen Disturbances

Online deployment lifts FlowDPG from 88% to 92% (Table I), with gains concentrated on contact-rich stages where deployment-time data exposes under-represented corner cases. To probe generalization, after the policy reaches the relevant stage we apply three disturbance families, each over 50 trials: *stage undo* (returning a grasped case/pod to the tray or inverting the case state), *in-place adjustment* (dislodging an inserted pod), and *scene perturbation* (reshuffling tray contents mid-reach). The offline policy handles undo disturbances reasonably (Table II), since the post-undo observation resembles a demonstration starting state; its weak point is scene perturbation, a mid-execution visual transition absent from demonstrations. Online RL improves all four cases—largest on scene perturbation—as the critic-guided update teaches the velocity field to steer perturbed and partial-success states back toward valid trajectories. Ablations isolating each design choice—the consistency regularizer, adaptive shift, and reward components, with mechanism-level diagnostics—appear in Appendices C and D.

Disturbance	**Offline**	**+Online**	Δ
Re-grasp (object replaced in tray)	78%	92%	+14%
Re-open/close (case state inverted)	82%	94%	+12%
Re-insert (pod dislodged)	86%	92%	+6%
Adapt grasp (tray reshuffled mid-execution)	70%	88%	+18%

TABLE II: Recovery rates under three disturbance families.

V. Conclusion and Limitations

We presented **FlowDPG**, a DDPG-style method for flow matching policies that distills the critic gradient into the velocity field via a single-step projection and L2 regression, avoiding backpropagation through the denoising ODE and recovering vanilla DPG under three explicit approximations. On a long-horizon dual-arm AirPods assembly task it attains 92% end-to-end success, with ablations supporting each design choice. **Limitations.** The reward needs stage-level annotations and we evaluate on a single platform; broader generalization is left for future work.

REFERENCES

[1] Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, Thomas Unterthiner, Daniel Keysers, Skanda Koppula, Fangyu Liu, Adam Grycner, Alexey Gritsenko, Neil Houlsby, Manoj Kumar, Keran Rong, Julian Eisenschlos, Rishabh Kabra, Matthias Bauer, Matko Bošnjak, Xi Chen, Matthias Minderer, Paul Voigtlaender, Ioana Bica, Ivana Balazevic, Joan Puigcerver, Pinelopi Papalampidi, Olivier Henaff, Xi Xiong, Radu Soricut, Jeremiah Harmsen, and Xiaohua Zhai. PaliGemma: A versatile 3B VLM for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

[2] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[3] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning (CoRL)*, 2023.

[4] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. RT-1: Robotics transformer for real-world control at scale. In *Robotics: Science and Systems (RSS)*, 2023.

[5] Qianzhong Chen, Justin Yu, Mac Schwager, Pieter Abbeel, Yide Shentu, and Philipp Wu. SARM: Stage-aware reward modeling for long horizon robot manipulation. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2509.25358.

[6] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems (RSS)*, 2023.

[7] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. PaLM-E: An Embodied Multimodal Language Model. *arXiv preprint arXiv:2303.03378*, 2023.

[8] Frederik Ebert, Yanlai Yang, Karl Schmeckpeper, Bernadette Bucher, Georgios Georgakis, Kostas Daniilidis, Chelsea Finn, and Sergey Levine. Bridge Data: Boosting Generalization of Robotic Skills with Cross-Domain Datasets. *arXiv preprint arXiv:2109.13396*, 2021.

[9] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning (ICML)*, pages 1587–1596. PMLR, 2018. URL https://arxiv.org/abs/1802.09477.

[10] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Youngwoon Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pannag R. Sanketi, Archit Sharma, Cody Simpson, Quan Vuong, Homer Rich Walke, Blake Wulfe, Ted Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, Christopher Agia, Rohan Baijal, Mateo Guaman Castro, Daphne Chen, Qiuyu Chen, Trinity Chung, Jaimyn Drake, Ethan Paul Foster, Jensen Gao, David Antonio Herrera, Minho Heo, Kyle Hsu, Jiaheng Hu, Donovon Jackson, Charlotte Le, Yunshuang Li, Kevin Lin, Roy Lin, Zehan Ma, Abhiram Maddukuri, Suvir Mirchandani, Daniel Morton, Tony Nguyen, Abigail O'Neill, Rosario Scalise, Derick Seale, Victor Son, Stephen Tian, Emi Tran, Andrew E. Wang, Yilin Wu, Annie Xie, Jingyun Yang, Patrick Yin, Yunchu Zhang, Osbert Bastani, Glen Berseth, Jeannette Bohg, Ken Goldberg, Abhinav Gupta, Abhishek Gupta, Dinesh Jayaraman, Joseph J. Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, and Chelsea Finn. DROID: A Large-Scale In-The-Wild Robot Manipulation Dataset, March 2024. URL http://arxiv.org/abs/2403.12945. arXiv:2403.12945 [cs].

[11] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. OpenVLA: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[12] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://arxiv.org/abs/2110.06169.

[13] Qiyang Li and Sergey Levine. Q-learning with adjoint matching. *arXiv preprint arXiv:2601.14234*, 2026.

[14] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2016.

[15] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

[16] NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An Open Foundation Model for Generalist Humanoid Robots. *arXiv preprint arXiv:2503.14734*, 2025.

[17] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[18] Open X-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Andrey Kolobov, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le, Chelsea Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Felipe Vieira Frujeri, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guangwen Yang, Guanzhi Wang, Hao Su, Hao-Shu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homanga Bharadhwaj, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine Hsu, Jay Vakil, Jeannette Bohg, Jeffrey Bingham, Jeffrey Wu, Jensen Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey Hejna, Jonathan Booher, Jonathan Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi "Jim" Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding, Minho Heo, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Muhammad Zubair Irshad, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Ning Liu, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Pannag R Sanketi, Patrick "Tree" Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundaresan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, and Roberto Martín-Martín. Open X-Embodiment: Robotic Learning Datasets and RT-X Models. *arXiv preprint arXiv:2310.08864*, 2023.

[19] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2024. URL https://arxiv.org/abs/2304.07193.

[20] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023.

[21] 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.

[22] Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, et al. $\pi^*_{0.6}$: A VLA that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025.

[23] Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via Q-score matching. In *International Conference on Machine Learning (ICML)*, 2024.

[24] Moritz Reuss, Maximilian Li, Xiaogang Jia, and Rudolf Lioutikov. Goal-conditioned imitation learning using score-based diffusion policies. In *Robotics: Science and Systems (RSS)*, 2023.

[25] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic

policy gradient algorithms. In *International Conference on Machine Learning (ICML)*, pages 387–395. PMLR, 2014.

[26] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv preprint arXiv:2307.09288*, 2023.

[27] Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2025.

[28] Yi Wang, Xinchen Li, Pengwei Xie, Pu Yang, Buqing Nie, Yunuo Cai, Qinglin Zhang, Chendi Qu, Jeffrey Wu, Jianheng Song, Xinlin Ren, Jingshun Huang, Mingjie Pan, Siyuan Feng, Zhi Chen, and Jianlan Luo. Learning while deploying: Fleet-scale reinforcement learning for generalist robot policies. *arXiv preprint arXiv:2605.00416*, 2026.

[29] Philipp Wu, Yide Shentu, Zhongke Yi, Xingyu Lin, and Pieter Abbeel. GELLO: A general, low-cost, and intuitive teleoperation framework for robot manipulators. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024.

[30] Wenli Xiao, Haotian Lin, Andy Peng, Haoru Xue, Tairan He, Yuqi Xie, Fengyuan Hu, Jimmy Wu, Zhengyi Luo, Linxi Fan, Guanya Shi, and Yuke Zhu. Self-improving vision-language-action models with data generation via residual RL. *arXiv preprint arXiv:2511.00091*, 2025.

[31] Charles Xu, Jost Tobias Springenberg, Michael Equi, Ali Amin, Adnan Esmail, Sergey Levine, and Liyiming Ke. RL token: Bootstrapping online RL with vision-language-action models. *arXiv preprint arXiv:2604.23073*, 2026.

[32] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. In *Robotics: Science and Systems*, 2023.

APPENDIX A
DERIVATION OF THE VANILLA DPG GRADIENT ON A FLOW MATCHING POLICY

We provide the full derivation of Eq. 10 for completeness. The deterministic actor is the ODE terminus $x_1 = x_0 + \int_0^1 v_\theta(x_t, t, s)\, dt$, with $x_0 \sim \mathcal{N}(0, I)$ independent of θ. The DDPG actor loss is $\mathcal{L}_{\text{DPG}} = -Q(s, x_1)$, so by the chain rule

$$\nabla_\theta \mathcal{L}_{\text{DPG}} = -\nabla_a Q(s, x_1) \cdot \nabla_\theta x_1 = -g^\top \cdot \nabla_\theta x_1, \tag{12}$$

where $g := \nabla_a Q(s, a)|_{a=x_1} \in \mathbb{R}^{|a|}$ is the critic gradient at the action terminus, treated throughout as a column vector. It remains to derive $\nabla_\theta x_1$ for the ODE actor.

a) Forward sensitivity equation.: Define the sensitivity matrix $M(t) := \partial x_t/\partial\theta \in \mathbb{R}^{|a|\times|\theta|}$, with initial condition $M(0) = 0$ since x_0 is parameter-free. Differentiating the ODE $\frac{dx_t}{dt} = v_\theta(x_t, t, s)$ with respect to θ and applying the chain rule on the right-hand side,

$$\frac{dM(t)}{dt} = \nabla_x v_\theta(x_t, t, s) \cdot M(t) + \nabla_\theta v_\theta(x_t, t, s). \tag{13}$$

This is a linear non-homogeneous ODE in $M(t)$. By variation of parameters, its solution at t=1 is

$$M(1) = \int_0^1 \Phi(1, t) \cdot \nabla_\theta v_\theta(x_t, t, s)\, dt, \tag{14}$$

where $\Phi(\sigma, t)$ is the state-transition matrix associated with the homogeneous part of Eq. 13, defined as the solution to

$$\frac{\partial\Phi(\sigma, t)}{\partial\sigma} = \nabla_x v_\theta(x_\sigma, \sigma, s) \cdot \Phi(\sigma, t), \qquad \Phi(t, t) = I. \tag{15}$$

$\Phi(1, t)$ describes how a perturbation of x_t propagates to a perturbation of x_1 — i.e., the Jacobian $\partial x_1/\partial x_t$.

b) Final form.: Substituting Eq. 14 into Eq. 12 gives the vanilla DPG gradient quoted in the main text:

$$\nabla_\theta \mathcal{L}_{\text{DPG}} = -\int_0^1 g^\top \Phi(1, t) \cdot \nabla_\theta v_\theta(x_t, t, s)\, dt. \tag{16}$$

c) Discrete-time form.: Under Euler discretization $x_{t_{i+1}} = x_{t_i} + \Delta t \cdot v_\theta(x_{t_i}, t_i, s)$ with T steps and $\Delta t = 1/T$, the chain-rule recursion gives

$$\begin{aligned} \nabla_\theta x_1 &= \Delta t \cdot \sum_{i=0}^{T-1} \left[\prod_{j=i+1}^{T-1} J_j \right] \cdot \nabla_\theta v_\theta(x_{t_i}, t_i, s), \\ J_j &:= I + \Delta t \cdot \nabla_x v_\theta(x_{t_j}, t_j, s), \end{aligned} \tag{17}$$

with the convention $\prod_{j=T}^{T-1}(\cdot) = I$, so that the term $i = T{-}1$ contributes only $\nabla_\theta v_\theta(x_{t_{T-1}}, \cdot)$. The matrix product $\prod_{j=i+1}^{T-1} J_j$ is the discrete analogue of $\Phi(1, t_i)$. In practice this recursion is what an autograd implementation would compute when backpropagating through the ODE solver. The product structure is precisely the source of the gradient explosion/vanishing pathology: small deviations in the spectral radius of J_j from unity compound multiplicatively across the T steps, making the update numerically fragile when v_θ is not yet well-trained.

d) Adjoint form.: Equivalently, define the adjoint state $\lambda(t) := \Phi(1, t)^\top g \in \mathbb{R}^{|a|}$, which propagates the terminal gradient g backward in time. It satisfies the linear backward ODE

$$\frac{d\lambda(t)}{dt} = -\nabla_x v_\theta(x_t, t, s)^\top \cdot \lambda(t), \qquad \lambda(1) = g, \tag{18}$$

integrated from $t = 1$ to $t = 0$. The DPG gradient can then be written as

$$\nabla_\theta \mathcal{L}_{\text{DPG}} = -\int_0^1 \lambda(t)^\top \cdot \nabla_\theta v_\theta(x_t, t, s)\, dt. \tag{19}$$

This is the form that adjoint methods solve directly, computing $\lambda(t)$ by backward integration and accumulating ∇_θ contributions at each t. FlowDPG bypasses both the forward sensitivity matrix Φ and the adjoint state λ entirely, instead distilling a single critic gradient evaluated at the in-distribution projected clean action $\hat{a}$.

APPENDIX B
IMPLEMENTATION DETAILS

A. Model Architecture

A DINOv2 [19] visual encoder fuses multi-view RGB images with proprioceptive states into a shared state embedding s. The flow matching velocity predictor $v_\theta(x_t, t, s)$ is a Diffusion Transformer [20] that produces an H-step action chunk in a single forward pass. On top of s, three MLPs of matched capacity form the value head: $V_\psi(s)$ and two Q-networks $Q_{\phi_{1,2}}(s, \boldsymbol{a})$ that additionally take the flattened action chunk, each paired with an EMA target $Q_{\bar{\phi}_i}$ used in the value loss (Sec. III-A). The reward head is the frozen SARM module sharing the same backbone.

B. Hyper-parameters

Table III lists the hyper-parameters used in our experiments. Unless otherwise noted, all baselines share the same backbone, action head, value head, reward, and offline/online data split, and only their policy-extraction objective differs.

C. SARM Signals and Value Estimates Along a Rollout

Figure 4 visualizes how the SARM signals and the learned value/Q-estimates evolve along a successful real-world execution. The progress curve $\phi(o_t)$ provides a meaningful per-step signal even within a single stage, supplying the dense shaping that FlowDPG distills via the critic, and stage-label jumps align with the keyframes in the top strip, illustrating when the stage-transition bonus in Eq. 3 fires. The value and Q-curves track each other closely throughout the rollout ($V_\psi \approx \min_i Q_{\phi_i}$), confirming that twin-critic IQL produces a well-calibrated value estimate suitable as a steering signal in FlowDPG. The value rises monotonically over the long horizon, with transient dips at contact-rich stages that reflect short-term recovery from minor failures before the next successful transition restores it.

APPENDIX C
ABLATION: CONSISTENCY REGULARIZER AND ADAPTIVE SHIFT

We ablate each design choice with end-to-end success and mechanism-level metrics (Fig. 5). Removing the consistency regularizer lets the projection error $\|\hat{a} - x_1\|$ plateau an order of magnitude higher (Fig. 5(a)), amplifying the Q-value discrepancy $|Q(s, \hat{a}) - Q(s, x_1)|$ (Fig. 5(b)): the critic gradient is then

TABLE III: Hyper-parameters for FlowDPG. Symbols match the notation in the main text where applicable.

Hyper-parameter	Symbol	Value
Flow matching policy		
Action chunk horizon	H	30
Number of ODE solver steps (inference)	T	12
Flow timestep sampling	t	$\text{Uniform}(0, 1)$
Noise distribution	x_0	$\mathcal{N}(0, I)$
Policy network	—	DiT, 14 layers, 16 heads, dim 1024
Value estimation (Sec. III-A)		
IQL expectile	τ	0.7
Discount factor	γ	0.99
EMA rate for target critic	ρ	0.005
Value network (V_ψ)	—	MLP, 3 layers, dim 512
Twin critic networks (Q_{ϕ_i})	—	MLP, 3 layers, dim 512
FlowDPG (Sec. III-B)		
Guidance scale	α	0.5
Numerical stabilizer	ε	$1\text{e}{-6}$
Distillation weight (max)	$\lambda_{\max}$	0.5
Warmup steps for λ	N_{warmup}	2,000
Consistency regularizer weight	μ_{cons}	1.0
Reward (Eq. 3)		
Stage-transition bonus	β_{stage}	1.0
Terminal success bonus	$r^{\text{term}}_{\text{success}}$	5.0
Terminal failure penalty	$r^{\text{term}}_{\text{fail}}$	-1.0
Optimizer and training pipeline		
Optimizer	—	AdamW
Learning rate (actor)	—	$1\text{e}{-4}$
Learning rate (critic)	—	$1\text{e}{-4}$
Weight decay	—	$1\text{e}{-4}$
Gradient clip norm	—	1.0
Mini-batch size	—	256
Offline training steps	N_{off}	10,000
Online training steps	N_{on}	6,000
Online/offline mini-batch ratio	—	1:1
Checkpoint sync interval (online)	N_{sync}	2,000
Perception backbone		
Visual encoder	—	DINOv2 ViT-B/14
Camera views	—	4 (front, top, left wrist, right wrist)
Image resolution	—	640×480
State embedding dimension	$\|s\|$	1024

evaluated OOD and points the wrong way, dropping Overall from 92% to 80%. Removing the adaptive shift reduces the update to a raw αg, whose unbounded magnitude makes the training loss oscillate (Fig. 5(c)) and produces the largest single-component drop (92% to 76%). The two components mitigate the same root cause—a misdirected critic-gradient signal—so removing both adds only a marginal further degradation (72%).

APPENDIX D
ABLATION: REWARD COMPONENTS

We compare the full composite reward (Eq. 3) against three variants; Figure 6 reports cumulative stage coverage with a distinct failure signature per variant. Removing the progress term causes $Q(s, a)$ to collapse toward a near-constant baseline within each stage, weakening $\nabla_a Q$ and producing a gradual erosion across the sequence (80%). Removing the stage-transition term keeps the dense progress shaping but eliminates the discrete bonus at sub-task completion, so $\nabla_a Q$ does not strongly favor closing out a sub-task over hovering near its end (76%). Removing both leaves only the terminal bonus: two visually similar states—the gripper holding a closed case before opening and again after re-closing—become indistinguishable under reward, so the policy frequently bypasses the entire pod-handling sequence to place the case directly, dropping sharply at Open case ($96\% \rightarrow 80\%$) and ending at 60%, below even the BC base. The progress and stage-transition terms are thus not redundant: when shaping is this sparse, the critic-gradient signal carries more noise than information and actively degrades the policy.

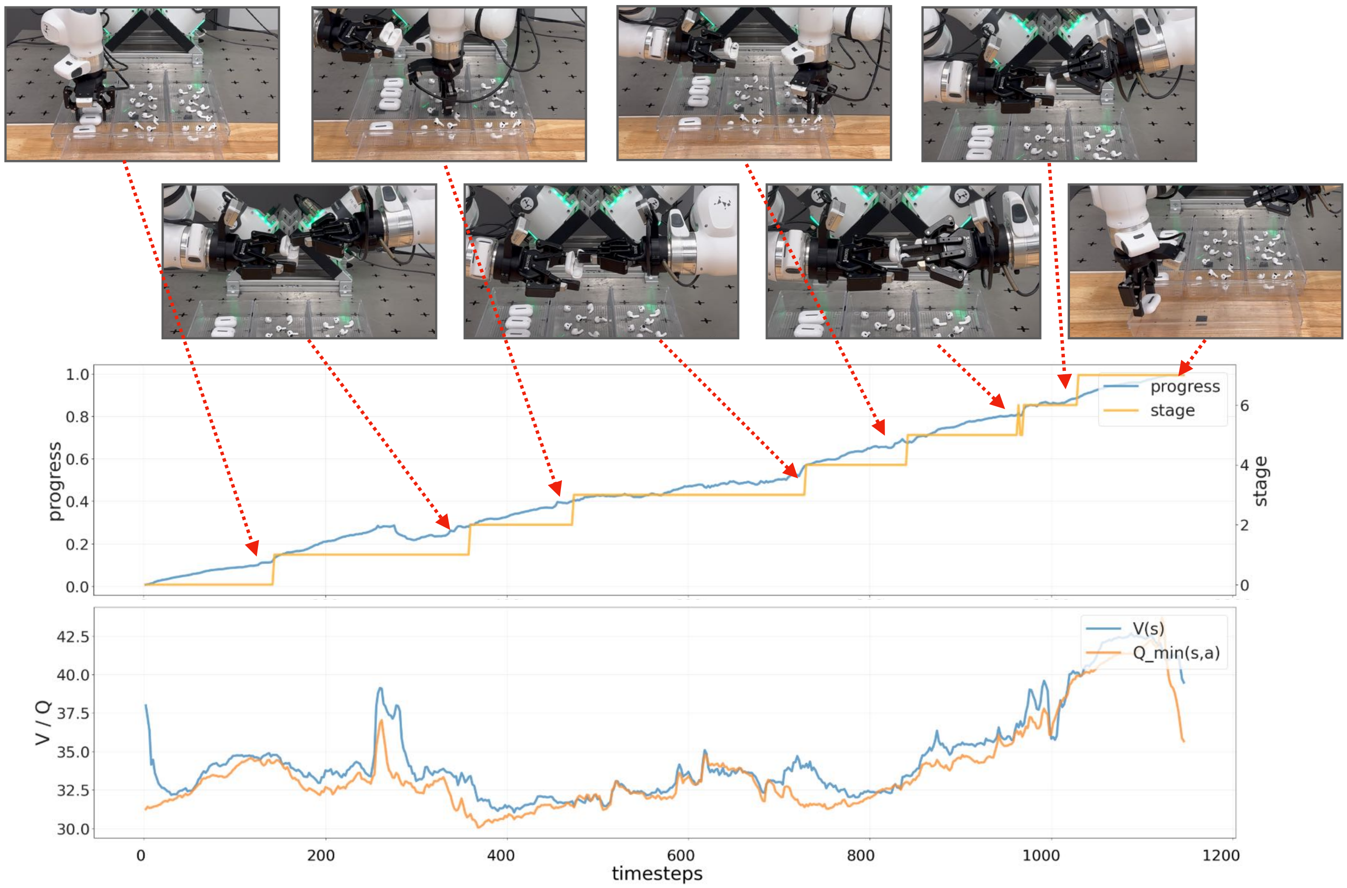

Fig. 4: SARM signals and value estimates along a successful AirPods assembly rollout. **Top:** keyframes at SARM stage transitions. **Middle:** predicted within-stage progress $\phi(o_t)$ (blue) and stage label $z(o_t)$ (yellow step). **Bottom:** learned value $V_\psi(s_t)$ (blue) and twin-critic minimum $\min_i Q_{\phi_i}(s_t, a_t)$ (orange) over the same rollout.

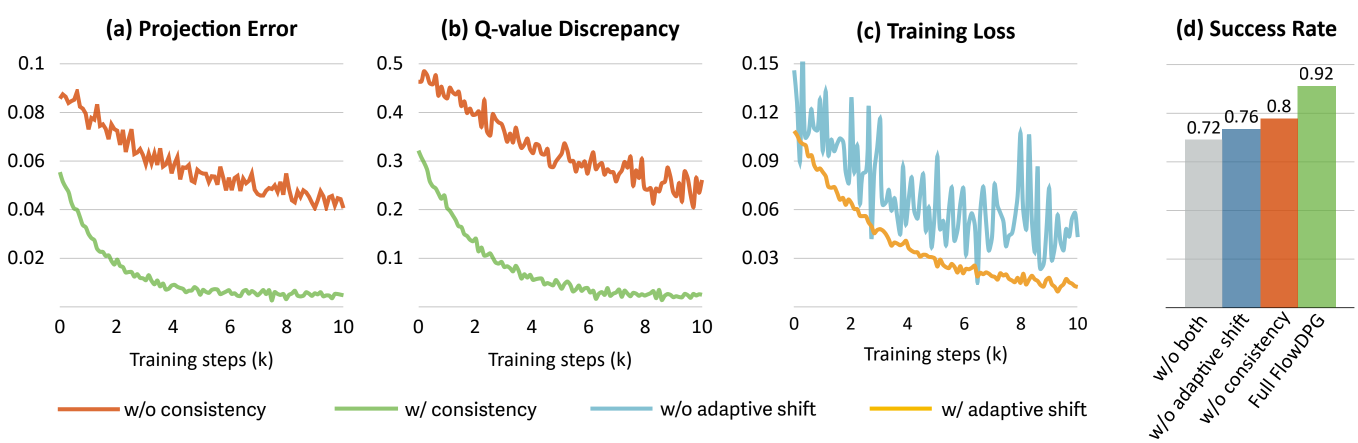

Fig. 5: Ablation on the consistency regularizer and adaptive shift.

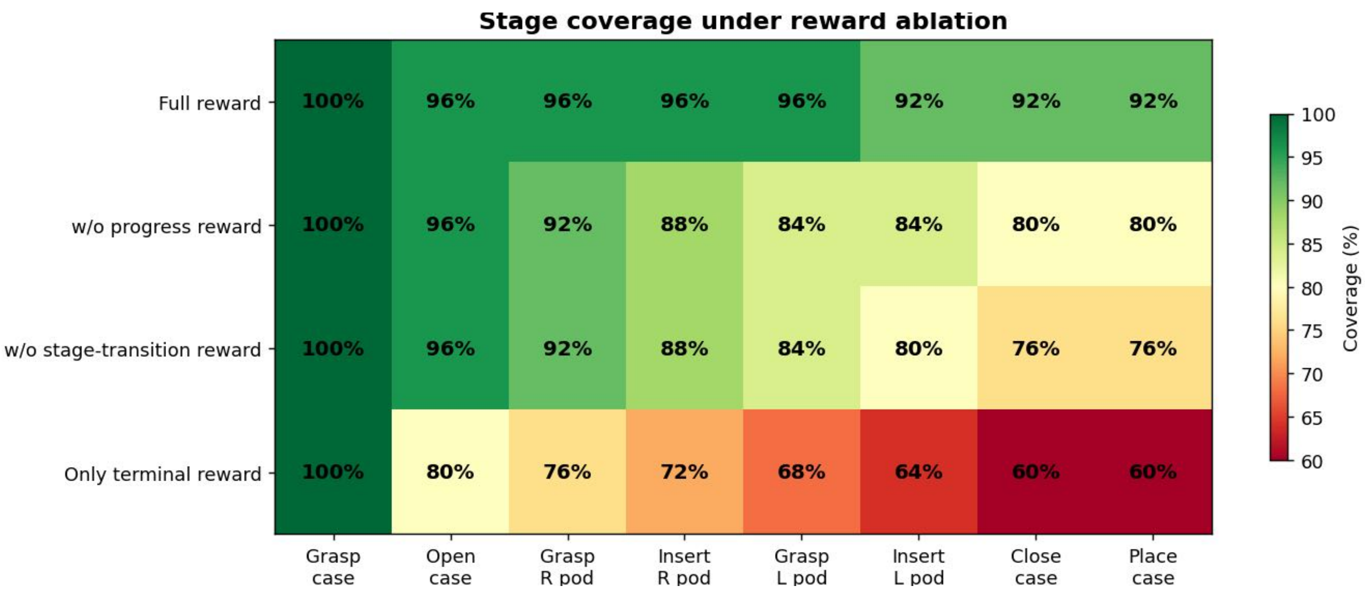

Fig. 6: Stage coverage under reward ablation.