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

Author Names Omitted for Anonymous Review.

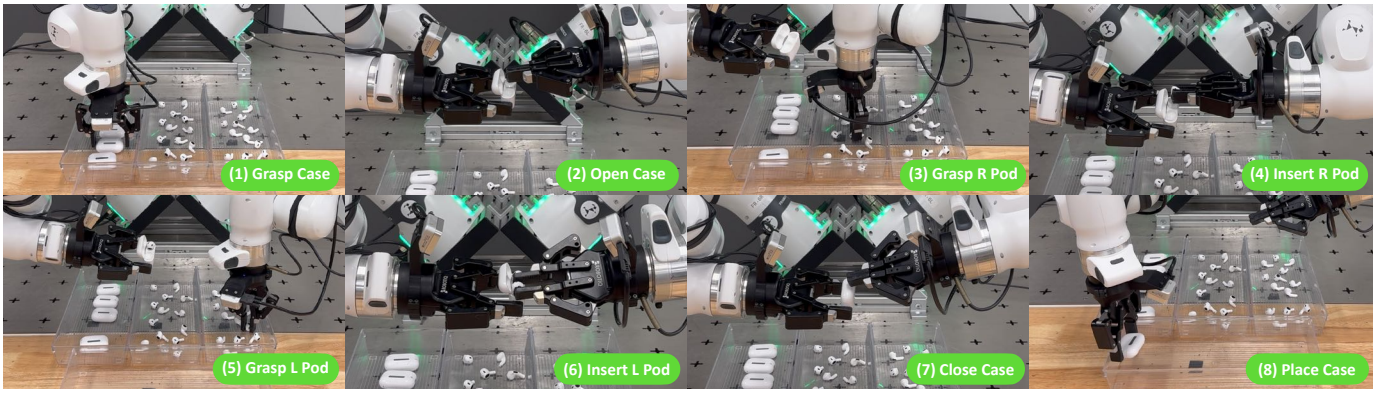

Fig. 1. The long-horizon, contact-rich, dual-arm AirPods assembly task used throughout this paper. The task decomposes into 8 sequential sub-stages, alternating between the two arms to grasp, open, insert, close, and place, and demands millimeter-level precision and bimanual dexterity at the insertion stage. The case and pods are initialized at randomized poses within the workspace, and a single policy must execute all 8 stages end-to-end from raw observations, without stage-level resets or hand-engineered switching.

***Abstract*—Real-world reinforcement learning for robotic manipulation remains challenging, and this difficulty is amplified for flow matching policies: applying policy gradient methods to these policies 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 specifically designed 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 that enables stable DDPG-style policy improvement on flow matching policies, (2) a formal connection between the FlowDPG update direction 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 provided on the project page https://flowdpg.github.io.**

I. Introduction

Despite the successes of large-scale supervised models in robotics, there is a clear shortcoming: imitation learning alone is insufficient for long-horizon, contact-rich tasks. Obtaining or collecting robotic datasets at the same scale and signal-to-noise ratio as datasets used to train large language models is prohibitively expensive. Even if one were able to get such a scale of high-quality teleoperated demonstrations, a supervised policy inherits its demonstration distribution and degrades sharply once it is deployed in unseen settings that are not covered by the data. On a long-horizon task, even rare failures at each stage can compound into a catastrophic performance. This naturally motivates the need for the robot to learn from its own actions, in a rather unsupervised fashion. A potential candidate is to employ real-world reinforcement learning, which has led to massive success in post-training LLMs, as it can push the policy beyond the original data manifold.

Recent state-of-the-art manipulation policies have converged on complex generative approaches such as flow matching [21] and diffusion [9] action heads, which produce expressive multi-modal action chunks through a multi-step denoising ODE. For these policies, the standard off-policy actor-critic update [20] propagating the critic gradient $\nabla_a Q$ back through the actor requires backpropagation through the entire denoising ODE. This is computationally prohibitive, since the cost grows linearly with the number of solver steps. It is also numerically fragile because the chain of Jacobians compounds gradient explosion and vanishing pathologies, particularly severe 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 designed for flow matching policies that obtains a stable critic-gradient update without backpropagating through the denoising ODE. The key insight is that clean action x_1 can be estimated from any intermediate noisy action x_t by a single forward pass of the velocity predictor, a unique property of flow-matching parameterization. We evaluate the critic gradient at this projected clean action, combine it with the demonstration-driven velocity to form 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 from standard flow matching, a pure ODE solve with no deployment-time critic queries. A core property of FlowDPG is that its update direction can be derived from the vanilla DPG gradient on a flow policy under three explicit, controllable approximations, on which we expand on later in Section III-B. This places the method on a transparent theoretical footing relative to the classical DDPG literature, in contrast to recent critic-gradient approaches that rely on more abstract stochastic-control formulations [19].

We validate FlowDPG on a long-horizon, contact-rich, dual-arm AirPods assembly task with millimeter-level insertion tolerance, where a single policy must execute eight sequential bimanual stages end-to-end from raw observations. FlowDPG reaches 92% end-to-end success, a 28% improvement over the BC base and a 12% margin over the strongest prior RL baseline. Mechanism-level ablations isolate the role of each design choice through diagnostic curves on projection error, Q-value discrepancy, and training-loss stability, and the online phase produces large gains on out-of-distribution disturbances that the offline policy alone cannot recover from. In summary, our contributions are **(1)** a BPTT-free distillation framework that enables 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) Generalist manipulation policies and generative action heads.: Large-scale robot demonstration datasets [24, 13, 11, 16] have enabled a wave of generalist manipulation policies that map perception, language, and proprioception directly to robot actions. Early systems used transformer policies trained on multi-task demonstrations [5, 4, 10, 6]. More recent vision-language-action models [17, 3, 29, 23, 22] inherit representational strength from large vision-language models [33, 8, 12, 15, 2] and demonstrate broad task coverage. At the action level, many recent robot policies have moved from deterministic or Gaussian action heads to expressive generative action heads. Action-chunking transformers [39] predict temporally extended action sequences, diffusion policies [9, 31, 1] generate action chunks through iterative denoising, and flow matching [21, 3, 22] provides an ODE-based parameterization for transporting noise to actions. Our work focuses on flow matching policies as the underlying actor class.

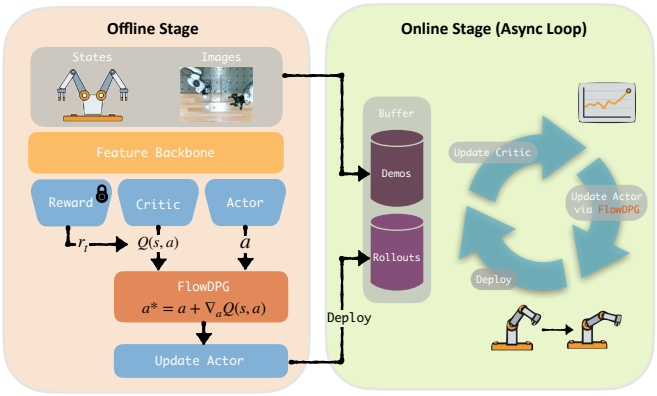

Fig. 2: Offline-to-online pipeline of FlowDPG. **Offline stage (left):** a shared visual backbone feeds a frozen reward predictor, a critic, and a flow matching actor; the FlowDPG block combines the actor output a with the critic gradient $\nabla_a Q(s,a)$ into a value-improved target a^* that is distilled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

b) RL fine-tuning of expressive manipulation policies.: Recent work has explored several ways to improve imitation-trained manipulation policies with reward signals. One family discards the critic's action gradient and uses only scalar value or advantage information: RA-BC [7] reweights the behavior-cloning objective by progress estimates, AWR [27] fits the policy to advantage-weighted demonstrations, and RECAP [28] conditions the policy on advantage as an additional input. A second family keeps the base policy frozen and learns a separate adaptation module: DSRL [34] performs RL in the latent noise space of the denoiser, PLD [37] learns a single-step residual and distills it back through supervised fine-tuning, and RLT [38] attaches a lightweight actor-critic head to a compact token extracted from the frozen base policy. The closest family exploits the critic's action gradient directly: Q-score matching [30] relates the score of an optimal policy to gradients of the Q-function, while QAM [19, 35] 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 flow velocity field using a simple DDPG-style objective, with an explicit derivation as an approximation of vanilla DPG. For value estimation, we use IQL [18] with twin critics [14] and a stage-aware reward predictor [7] as the dense reward source.

III. FlowDPG

Our method has two coupled components on top of a flow matching policy (Figures 2, 3): a twin-critic IQL value estimator (Sec. III-A), and **FlowDPG**, which distills the deterministic policy gradient (DPG) [32, 20] into the flow matching velocity

field at training time (Sec. III-B).

A. Value Estimation with Twin-Critic IQL

We adopt Implicit Q-Learning (IQL) [18] with twin critics in the style of TD3 [14]. IQL replaces the $\max_a Q(s,a)$ bootstrap target, which queries the critic on potentially out-of-distribution actions, with a value function $V_\psi(s)$ that estimates a high-quantile in-distribution action-value at s, eliminating extrapolation error during value learning. Twin critics Q_{ϕ_1}, Q_{ϕ_2} further mitigate Q-value overestimation from OOD actions and TD-bootstrap noise; we take their minimum wherever a single Q-estimate is needed, and pair each critic with an EMA target network used in the value loss. V_ψ is trained by expectile regression of the twin-critic minimum on dataset actions,

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

where $\rho_\tau(u) = |\tau - \mathbb{1}[u<0]|\cdot u^2$ is the expectile loss and the expectile $\tau > 0.5$ pulls $V_\psi(s)$ toward the upper tail of the action-value distribution. The critics are then bootstrapped from V_ψ using 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-level rewards r_t come from SARM [7], a frozen stage-aware reward predictor that outputs a stage label $z(o_t)$ and a within-episode progress $\phi(o_t) \in [0,1]$, combined into a composite chunk reward of three terms:

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

The progress term provides dense per-step shaping, the indicator term issues a fixed bonus β_{stage} at each predicted stage advance, and the terminal term encodes whole-episode success; we visualize an example reward and value curve in Appendix D.

B. Policy Extraction via FlowDPG

A flow matching policy transports noise x_0 to a feasible action x_1 within the demonstration support, but the resulting action is not necessarily optimal under the critic. FlowDPG combines *two vectors* (Figure 3): the demonstration-driven velocity for feasibility, and a critic-driven correction along $\nabla_a Q$ in the spirit of DDPG [20, 32] for value improvement. The combined target is distilled back into v_θ via a single L2 regression, avoiding the obvious alternative of backpropagating $\nabla_a Q$ through the full ODE, which is computationally prohibitive and numerically fragile. FlowDPG costs only one extra forward pass and one critic-gradient evaluation per update, and leaves inference as a pure flow ODE solve.

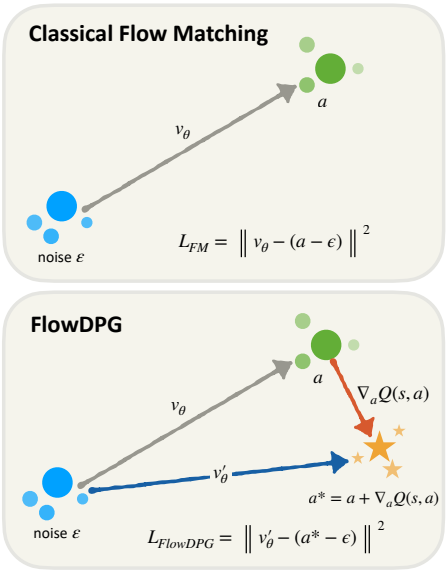

Fig. 3: **Top:** classical flow matching transports noise to a demonstration-feasible action a via v_θ. **Bottom:** FlowDPG adds a critic-driven correction along $\nabla_a Q(s,a)$ to reach a value-improved target a^*, then distills the resulting velocity v'_θ back into the flow field.

a) Q-Gradient.: The clean action x_1 can be estimated from any intermediate x_t by a single forward pass of v_θ:

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

This satisfies $\hat{a} \approx x_1$ when v_θ predicts the velocity field accurately 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:

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

with stop-grad on the critic parameters.

b) Adaptive shift.: The magnitude of g varies considerably across states and training steps, destabilizing updates if used as-is. We normalize the step to the local velocity scale:

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

with $\alpha > 0$ a base guidance scale and ε guarding against vanishing gradients.

c) Distilled flow target.: Summing $\hat{a}$ and Δ gives the Q-improved clean action, and subtracting noise gives the velocity target:

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

The policy is regressed against u_t^* alongside a behavior-cloning anchor:

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

with λ warming up linearly from 0 over N_{warmup} steps so that the unreliable early critic does not dominate.

d) Consistency regularizer.: The reliability of $\hat{a}$ is critical: a projection off the clean-action manifold leaves the critic queried OOD and g unreliable. Since the standard 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 $x_t = (1-t)x_0 + tx_1$ into Eq. 4 gives $\hat{a} - x_1 = (1-t)(v_\theta - u_t)$, so $\mathcal{L}_{\text{cons}}$ reduces to a velocity-error penalty weighted by $(1-t)^2$, emphasizing small-t samples where the projection error is most amplified.

e) Relationship to vanilla DPG.: A vanilla DPG [32, 20] update on a flow matching policy treats the ODE terminus x_1 as a deterministic actor and, by the chain rule (Appendix A), yields

$$\begin{aligned} \nabla_\theta \mathcal{L}_{\text{DPG}} &= -\nabla_a Q(s, x_1) \cdot \nabla_\theta x_1 \\ &= -\int_0^1 g^\top \Phi(1,t) \cdot \nabla_\theta v_\theta(x_t, t, s)\, dt, \end{aligned} \tag{10}$$

where $\Phi(1,t) := \partial x_1 / \partial x_t$ is the ODE sensitivity matrix. Computing Φ requires backpropagating through all T ODE steps with $O(T)$ memory and well-known gradient explosion/vanishing pathologies, particularly severe when v_θ is not yet well-trained. The FlowDPG gradient, by contrast, comes from differentiating $\mathcal{L}_{\text{FlowDPG}}$: after warmup, the only g-dependent term is the distillation loss, so treating g as a detached constant gives

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

with $\hat{g}(\hat{a}) := g/\|g\|$. This recovers Eq. 10 under three explicit approximations: *(i)* the critic gradient is evaluated at $\hat{a}$ rather than x_1, collapsing $\Phi(1,t)$ to the identity and eliminating ODE backpropagation, with $\mathcal{L}_{\text{cons}}$ keeping this faithful by directly minimizing $\|\hat{a} - x_1\|$; *(ii)* the trajectory-wide integral is replaced by a Monte Carlo estimate at a single $t \sim \text{Uniform}(0,1)$, the same estimator flow matching uses for training; and *(iii)* the un-normalized step size $\|g\|$ is replaced by the adaptive scaling $2\lambda\alpha\|u_t\|$ of Eq. 6, removing a known source of instability in DDPG-family methods [20, 14].

C. Offline-to-Online Pipeline

The two components above slot into the pipeline of Figure 2, detailed in Algorithm 1. In the **offline phase**, the backbone, action head, and value head are jointly trained on demonstrations $\mathcal{D}_{\text{off}}$ with the FlowDPG objective, producing a base policy that initializes online deployment. In the **online phase**, the policy is deployed to collect real-world rollouts that are asynchronously appended to $\mathcal{D}_{\text{on}}$; the learner samples mini-batches at $\sim$1:1 ratio from $\mathcal{D}_{\text{off}} \cup \mathcal{D}_{\text{on}}$ to balance demonstration quality and fresh experience. Backbone and reward predictor are frozen in this phase to preserve pretrained representations and reduce per-step compute; updated actor/critic checkpoints synchronize back to deployment periodically.

Algorithm 1 FlowDPG: Offline-to-Online Training

Require: Offline buffer $\mathcal{D}_{\text{off}}$; online buffer $\mathcal{D}_{\text{on}}$ (initially empty); guidance scale α; consistency weight μ_{cons}; max guidance weight $\lambda_{\max}$; warmup N_{warmup}; EMA rate ρ; expectile τ.

Pretrain SARM on the annotated subset
Initialize backbone, action head v_θ, value head $(V_\psi, Q_{\phi_1}, Q_{\phi_2})$; EMA targets $Q_{\bar{\phi}_i} \leftarrow Q_{\phi_i}$

Offline phase

for $k = 1, \ldots, N_{\text{off}}$ **do**
  Sample mini-batch from $\mathcal{D}_{\text{off}}$; compute r_t from the reward predictor
  Sample $x_0 \sim \mathcal{N}(0, I)$, $t \sim \text{Uniform}(0,1)$; form x_t, u_t
  Compute $v_t = v_\theta(x_t, t, s)$ and $\hat{a}$ via Eq. 4
  $\mathcal{L}_{\text{cons}} \leftarrow \|\hat{a} - x_1\|^2$
  $g \leftarrow \nabla_{\hat{a}} \min_i Q_{\phi_i}(s, \hat{a})$ with stop-grad on critic
  $\Delta \leftarrow \alpha \cdot \|u_t\| / (\|g\| + \varepsilon) \cdot g$
  $a^* \leftarrow \hat{a} + \Delta$; $u_t^* \leftarrow a^* - x_0$
  $\lambda \leftarrow \lambda_{\max} \cdot \min(k / N_{\text{warmup}}, 1)$
  Update θ by minimizing $\mathcal{L}_{\text{FlowDPG}} + \mu_{\text{cons}} \mathcal{L}_{\text{cons}}$ via Eqs. 8, 9
  Update ψ, ϕ_1, ϕ_2 via Eqs. 1–2
  EMA targets: $\bar{\phi}_i \leftarrow \rho\bar{\phi}_i + (1-\rho)\phi_i$
end for

Online phase

Deploy π_θ for autonomous rollout collection
for $k = 1, \ldots, N_{\text{on}}$ **do**
  Asynchronously append rollouts to $\mathcal{D}_{\text{on}}$ with rewards from the reward predictor
  Sample mini-batch from $\mathcal{D}_{\text{off}} \cup \mathcal{D}_{\text{on}}$ at $\sim$ 1:1 ratio
  Update $\theta, \psi, \phi_1, \phi_2$ as in the offline phase, with backbone and SARM frozen
  if $k \bmod N_{\text{sync}} = 0$ **then**
    Synchronize π_θ to deployment
  end if
end for
return v_θ

IV. Experiments

Task and protocol. We evaluate on a long-horizon, dual-arm AirPods assembly task (Figure 1) spanning 8 sequential sub-stages with randomized case and pod poses. The setup uses two Franka arms with demonstrations collected via GELLO [36] teleoperation. Each policy is evaluated on 25 cases (50 pod manipulations); failures within a stage do not terminate the episode and retries are allowed. We report per-stage success rate (with retries), the unweighted mean across stages (**Avg**), and end-to-end episode success (**Overall**), which requires all pods correctly inserted.

Baselines. We compare against three families of RL methods for adapting a base flow policy. *Value-conditioning* methods use only scalar value/advantage signals: RA-BC [7] reweights the BC objective by SARM progress, AWR [27] performs advantage-exponentially-weighted regression, and RECAP [28] conditions the policy on advantage. *Auxiliary-module* methods keep the base policy frozen and train a separate adapter: DSRL [34] performs RL in the noise space of the denoiser, PLD [37] learns a single-step residual, and RLT [38] attaches an actor-critic head to a compact RL token. The *critic-gradient* category contains the concurrent QAM [19], which converts $\nabla_a Q$ into step-wise supervision via adjoint matching. All baselines share the same backbone,

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 avg success rates on the AirPods assembly task.

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 algorithmic mechanism to its empirical ceiling.

a) Value-conditioning is bounded by the demonstration support: Reweighting or conditioning over existing demonstrations cannot synthesize an action outside their distribution. These methods improve where the bottleneck is selecting the better mode from existing data (Open/Close case) but remain near the BC base on contact-rich stages (Insert R/L pod) requiring action refinement beyond what demonstrations cover.

b) Auxiliary-module methods leave the base policy untouched: DSRL optimizes in noise space, an indirect channel separated from the final action by the entire ODE. PLD's single-step residual can nudge the trajectory endpoint but cannot reshape the multi-step denoising itself. RLT is the strongest here because its actor-critic head is easier to optimize, but the MLP actor discards the flow policy's multi-modal expressivity. In all three cases, improvement headroom is capped by the adapter rather than the base policy.

c) Critic-gradient methods directly update the base policy: FlowDPG uses $\nabla_a Q$ to update the velocity field itself, removing both ceilings: actions are no longer confined to the demonstration support, and the full capacity of the flow policy participates in the improvement. The concurrent QAM [19] pursues the same direction through adjoint matching, but requires an auxiliary adjoint flow at training time and step-wise supervision at every flow timestep, making it sensitive to critic noise across the trajectory. FlowDPG distills a single critic-gradient evaluation via one L2 regression, a simpler mechanism with an explicit derivation 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 the contact-rich stages (Grasp/Insert R/L pod) where deployment-time data exposes under-represented corner cases. To probe generalization beyond the demonstration distribution, we apply three families of disturbances after the policy reaches the relevant stage: (i) *stage undo*—returning a grasped case/pod to the tray or inverting the open/close state of the case; (ii) *in-place adjustment*—dislodging a successfully inserted pod; (iii) *scene perturbation*—reshuffling tray contents while the policy is reaching. Each disturbance is evaluated on 50 trials.

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.

The offline policy handles undo disturbances reasonably (Table II): the post-undo observation resembles the demonstration starting state of the corresponding stage, and visual conditioning re-triggers the appropriate behavior. The weak point is scene perturbation, a mid-execution visual transition absent from demonstrations. Online RL improves all four cases, with the largest gain on the scene perturbation and the smallest where demonstration coverage was already strong. Deployment-time failures expose the policy to dynamic perturbations and partial-success states, and the critic-guided update teaches the velocity field to steer such states back toward valid trajectories.

C. Ablation: Consistency Regularizer and Adaptive Shift

We ablate each design choice and report both end-to-end success and mechanism-level metrics (Figure 4). Removing the consistency regularizer lets the projection error $\|\hat{a} - x_1\|$ plateau an order of magnitude higher (Fig. 4(a)), amplifying the Q-value discrepancy $|Q(s, \hat{a}) - Q(s, x_1)|$ (Fig. 4(b)). The

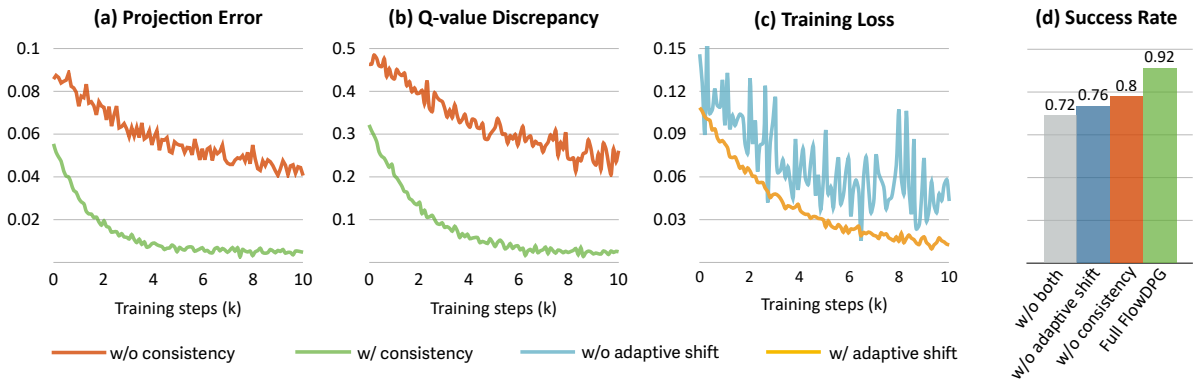

Fig. 4: Ablation on consistency regularizer and adaptive shift.

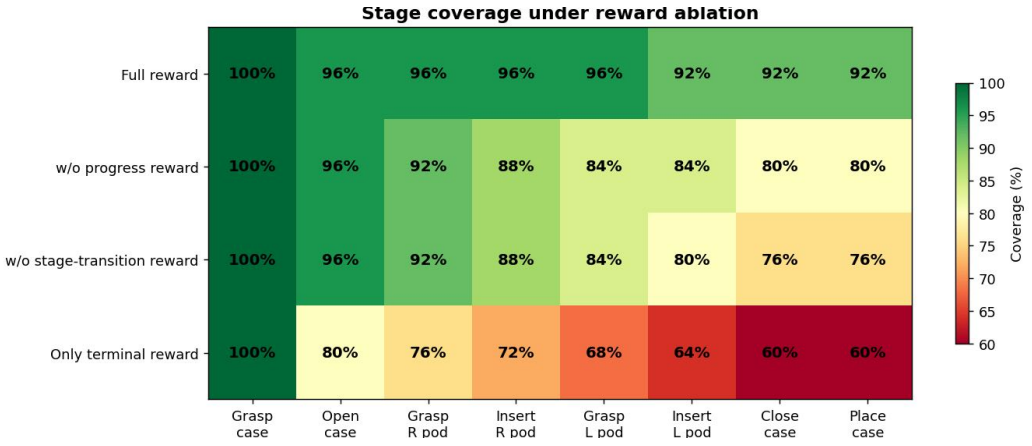

Fig. 5: Stage coverage under reward ablation.

critic gradient is then evaluated at an out-of-distribution action and points in the wrong direction, 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 rather than decrease smoothly (Fig. 4(c)) and produces the largest single-component drop (92% to 76%). The two components mitigate the same root cause, a misdirected critic-gradient signal, through complementary means, so removing both adds only a marginal further degradation (72%).

D. Ablation: Reward Components

We compare the full composite reward in Eq. 3 against three variants; Figure 5 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 entire 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 the case is re-closed, become indistinguishable under reward. 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 FlowDPG distills carries more noise than information and actively degrades the policy.

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. The update recovers vanilla DPG under three explicit approximations, and on a long-horizon dual-arm AirPods assembly task FlowDPG attains 92% end-to-end success with mechanism-level ablations supporting each design choice.

Limitations. The consistency regularizer minimizes projection error on average, leaving a few states with unreliable critic gradients that we do not detect or down-weight. The reward formulation requires stage-level annotations and does not apply to tasks lacking a clean sub-stage decomposition. Finally, we evaluate on a single bimanual platform and one task family; generalization to larger VLA policies and very different contact dynamics is left for future work.

References

[1] Jose Barreiros, Andrew Beaulieu, Aditya Bhat, Rick Cory, Eric Cousineau, Hongkai Dai, Ching-Hsin Fang, Kunimatsu Hashimoto, Muhammad Zubair Irshad, Masha Itkina, et al. A careful examination of large behavior models for multitask dexterous manipulation. *Science Robotics*, 11(113):eadp6201, 2026.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] Xi Chen, Josip Djolonga, Piotr Padlewski, Basil Mustafa, Soravit Changpinyo, Jialin Wu, Carlos Riquelme Ruiz, Sebastian Goodman, Xiao Wang, Yi Tay, et al. PaLI-X: On Scaling Up a Multilingual Vision and Language Model. *arXiv preprint arXiv:2305.18565*, 2023.

[9] 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.

[10] Sudeep Dasari and Abhinav Gupta. Transformers for one-shot visual imitation. In *Conference on Robot Learning*, pages 2071–2084. PMLR, 2021.

[11] Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. Robonet: Large-scale multi-robot learning. *arXiv preprint arXiv:1910.11215*, 2019.

[12] 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.

[13] 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.

[14] 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.

[15] Siddharth Karamcheti, Suraj Nair, Ashwin Balakrishna, Percy Liang, Thomas Kollar, and Dorsa Sadigh. Prismatic VLMs: Investigating the Design Space of Visually-Conditioned Language Models. In *International Conference on Machine Learning*, 2024.

[16] 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].

[17] 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.

[18] 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.

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

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

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

[27] 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.

[28] 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.

[29] Physical Intelligence, Kevin Black, Noah Brown, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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

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

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,

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

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

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

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

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

$\Phi(1, t)$ 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:

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

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

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

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

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

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

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

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 $\hat{a}$.

B. Model Architecture

A DINOv2 [25] 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 [26] 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.

C. 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.

D. SARM Signals and Value Estimates Along a Rollout

Figure 6 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.

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	$\mathrm{Uniform}(0, 1)$
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	$1\mathrm{e}{-6}$
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)	—	$1\mathrm{e}{-4}$
Learning rate (critic)	—	$1\mathrm{e}{-4}$
Weight decay	—	$1\mathrm{e}{-4}$
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	$\lvert s \rvert$	1024

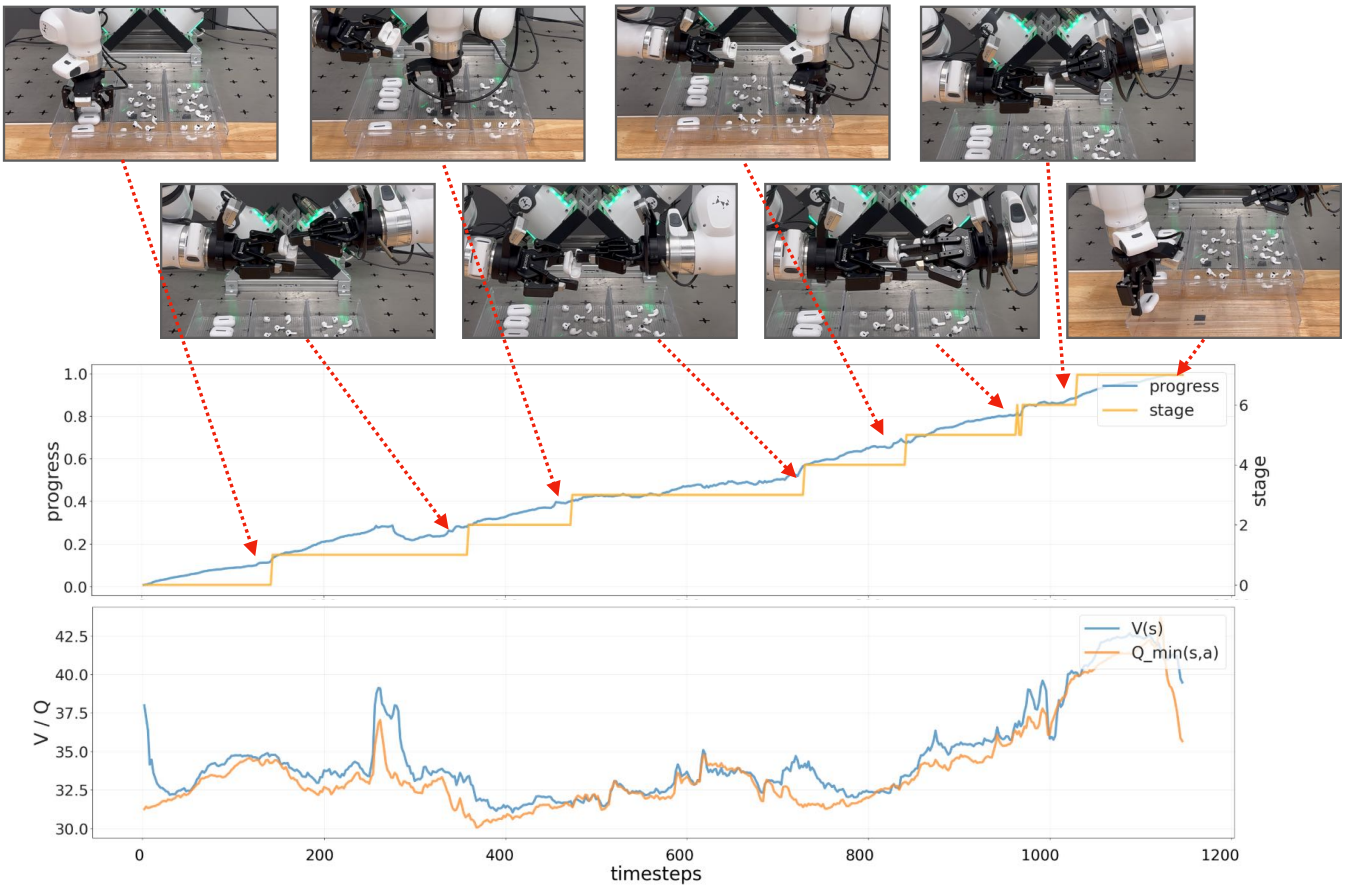

Fig. 6: SARM signals and value estimates along a successful AirPods assembly rollout. **Top:** keyframes at SARM stage transitions. **Middle:** predicted within-stage progress $\phi(o_t)$ (blue) and stage label $z(o_t)$ (yellow step). **Bottom:** learned value $V_\psi(s_t)$ (blue) and twin-critic minimum $\min_i Q_{\phi_i}(s_t, a_t)$ (orange) over the same rollout.