ARCFLOW: TRAINING FLOW POLICIES FOR MULTIPLE SOLVER BUDGETS

Anonymous authors
Paper under double-blind review

ABSTRACT

Flow-matching policies adjust inference cost through the number of solver steps, but reinforcement learning at a single fixed solver budget can hide degradation at cheaper deployment budgets and even erase useful one-step behavior present in a pretrained policy. We introduce **Action–Ratio–Clip Flow (ARCFLOW)**, a deployment-aware algorithm for training one flow policy across a specified range of solver budgets. ARCFLOW combines action-aligned supervision, budget exposure, and worst-budget checkpoint selection. For reconstruction objectives, its bridge aggregates temporal probes into one surrogate ratio and one PPO-style clip per action, with a boundary residual exactly equal to deployed one-step reconstruction error. Cycling exposes the shared field to the deployment range, and selection optimizes its performance floor. For native velocity objectives, multi-budget endpoint supervision applies the same principle without replacing the native update. Both realizations preserve the architecture and deployment-time cost. Across DM Control, manipulation, and locomotion, ARCFLOW attains the highest mean one-step return among five flow-policy baselines on 25 of 30 tasks under each method's reported training settings. Component studies identify complementary improvements from boundary supervision, the fixed-grid bridge, budget exposure, and deployment selection. These results establish deployment-budget-aware training as an effective route to strong low-step control with one policy serving multiple inference costs. [1]

1 INTRODUCTION

Flow matching represents expressive action distributions by learning a velocity field that transports noise to actions (Lipman et al., 2023; Liu et al., 2023). Its solver controls inference cost: more steps refine integration, while fewer steps reduce network evaluations, helping robotic policies meet latency budgets. Consistency and shortcut models enable one- and few-step generation (Song et al., 2023; Frans et al., 2025), creating an opportunity to retain expressive policies at low deployment cost.

Reinforcement learning (RL) improves policies through interaction. FPO and FPO++ derive policy-gradient surrogates from flow-matching losses (McAllister et al., 2025; Yi et al., 2026); Diffusion Policy Policy Optimization (DPPO) applies policy gradients to a stochastic diffusion denoising chain (Ren et al., 2025); ReinFlow adds learnable exploration noise to flow policies (Zhang et al., 2025b); and PolicyFlow approximates importance ratios from velocity-field changes along interpolation paths (Yang et al., 2026). At deployment, solver steps determine latency and may change the action of the same trained field. High return at the training budget gives no assurance of high one-step return; RL can even erode low-step behavior already present in the prior. This gap motivates us to train and select one field across its deployment range (Figure 1).

Two properties of the training objective and solver discretization help explain this behavior. First, reconstruction-based objectives may provide limited supervision at the deployment boundary. In FPO (McAllister et al., 2025), for example, the noise-reconstruction formulation weights velocity error by $(1-t)^2$, so the boundary queried by a one-step solver receives zero weight. Direct velocity supervision instead provides a boundary-aligned signal. This observation complements prior work on time-dependent loss weighting in generative modeling (Choi et al., 2022; Hang et al., 2023) and highlights the importance of aligning the weighting scheme with the deployed action map. Second,

[1] Code can be accessed at: `https://anonymous.4open.science/r/arcflow-868A`

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

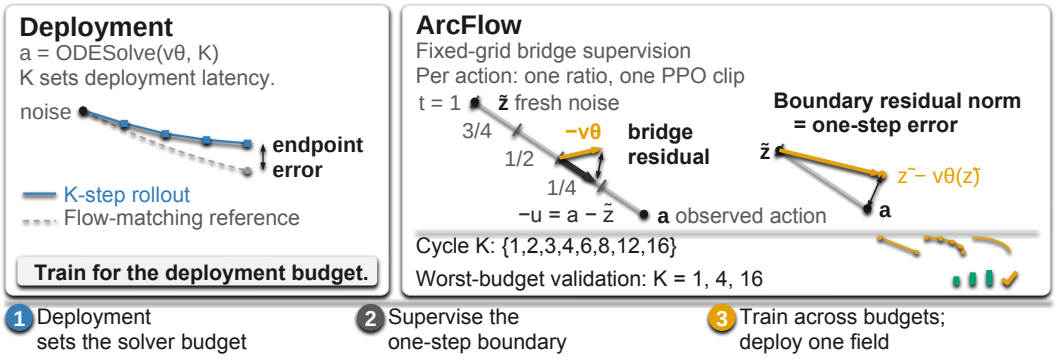


Figure 1: **ARCFLOW for multi-budget deployment.** Left: the solver budget K controls deployment cost. Right: the bridge-based control realization uses fresh noise and the observed action; at the boundary, the residual norm equals one-step reconstruction error. Probes are averaged before one surrogate ratio and PPO clip per action. Rollout budgets cycle, and checkpoint selection maximizes the worst validation return over $K \in \{1, 4, 16\}$.

because different solver budgets induce different action maps, updates based on rollouts from a single budget need not improve all budgets uniformly. We study this effect in the ReinFlow (Zhang et al., 2025b) fine-tuning pipeline, where native velocity supervision is already available.

We introduce **Action–Ratio–Clip Flow (ARCFLOW)**, an RL algorithm for one flow field across a specified solver-budget range. Its control realization uses a fixed-grid bridge with an exact one-step action-reconstruction residual. It aggregates temporal probes into one energy-based surrogate ratio and one PPO-style clip per action; the ratio is not the executed policy's likelihood ratio. For native velocity objectives, multi-budget endpoint losses and an anchored max–min update extend the same deployment principle without replacing the original objective. Both realizations retain one field and the original deployment architecture.

Our contributions are:

- **A diagnosis tied to deployment.** We expose one-step degradation hidden by training-budget evaluation and identify how the supervision target suppresses the relevant boundary signal.
- **An algorithm for multiple solver budgets.** We connect an exact action-reconstruction identity to action-aligned supervision, budget exposure, and worst-budget checkpoint selection.
- **Evidence across objectives and domains.** A controlled common-stack benchmark and a released manipulation pipeline test the algorithm, with component studies and deployment-cost accounting identifying the sources of improvement. Across DM Control, manipulation, and locomotion, ARCFLOW leads five flow-policy baselines in mean one-step return on 25 of 30 tasks including cross-regime manipulation comparisons.

2 PROBLEM FORMULATION

Let s denote an environment state, $z \sim p_0$ source noise, and $v_\theta(x, t \mid s)$ a velocity field with reverse-time convention $t : 1 \to 0$. A numerical solver S_K with K steps induces the policy $a = F_\theta^{S_K}(z, s)$. The environment observes the resulting action; the solver's internal queries do not constitute environment transitions. Write $J_K(\theta)$ for the expected episodic return of this policy. Our objective is to learn one field that remains useful across a specified set of deployment budgets, without supplying K as an input to the field.

Flow-policy algorithms use several tractable training signals, including flow-matching residuals, velocity-based surrogates, and likelihoods of stochastic internal transitions. These signals differ in their treatment of solver time. An improvement in a training surrogate, or in $J_{K_{\text{train}}}$, does not in itself imply an improvement in J_K for a different budget. We distinguish three outcomes. *Numerical failure* is a non-finite update or evaluation. *Cross-budget degradation* is a lower return at one deployment

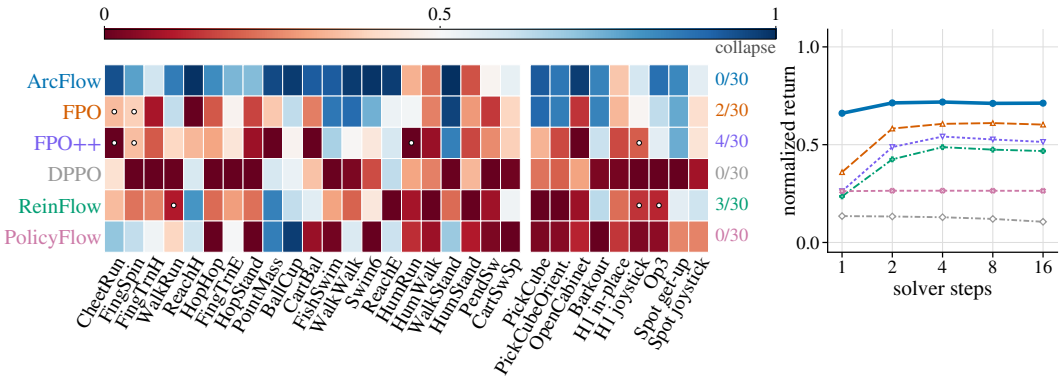


Figure 2: **ARCFLOW strengthens one-step deployment across the task census and solver budgets.** (a) Within-task normalized one-step return on all 30 retained tasks from the registered selection results; dots mark a one-to-sixteen-step mean-return ratio below 0.5. The right band shows nine Playground tasks on the display regimes of Table 1. (b) Mean normalized return versus solver steps over all 30 tasks for every method: 21 held-out DM Control tasks and 9 Playground tasks on their registered display regimes.

budget than another. *Temporal degradation* is a fall in return during training at a fixed budget. The census calls a task a *one-step collapse* when its specified one-to-sixteen-step return statistic falls below one half. This threshold is descriptive and depends on the seed pool and aggregation rule; it is not a synonym for temporal degradation. We report the graded return ratio alongside the threshold count.

3 SOLVER-BUDGET MISMATCH

Cross-budget performance. Figure 2 compares all six flow methods on the same 30-task census. In panel (a), ARCFLOW records no one-step collapses while maintaining strong one-step returns across the DM Control and Playground tasks. The marked collapse counts are $0/30$ for ARCFLOW, $2/30$ for FPO, $4/30$ for FPO++, $0/30$ for DPPO, $3/30$ for ReinFlow, and $0/30$ for PolicyFlow. Panel (b) shows that ARCFLOW has the highest mean normalized return at every evaluated solver budget. Its aggregate return increases from one to two steps and then remains nearly constant through sixteen steps, whereas the other methods plateau at lower levels; the separation is largest at one-step deployment.

Boundary suppression. For a straight bridge $x_t = (1-t)a + t\varepsilon$ and target velocity $u = \varepsilon - a$, the reference implementation predicts noise as $\hat{\varepsilon}_\theta = x_t + (1-t)v_\theta(x_t, t \mid s)$. Thus

$$\|\hat{\varepsilon}_\theta - \varepsilon\|_{2,d}^2 = (1-t)^2 \|v_\theta(x_t, t \mid s) - u\|_{2,d}^2, \tag{1}$$

where $\|r\|_{2,d}^2 = d^{-1}\sum_i r_i^2$. Both the loss and its actor gradient vanish at $t = 1$, the only node queried by a one-step reverse Euler solver. Nearby supervision is also quadratically down-weighted. Velocity supervision removes this factor. The ablations in Section 5.4 attributes most of the one-step recovery to this change. The released ReinFlow study complements this target-level diagnosis by testing deployment-aware training in a pipeline that already supervises velocity (Table 3).

Collapse versus absolute performance. The collapse count and the mean return capture different aspects of solver-budget behavior. DPPO and PolicyFlow have no marked collapses in Figure 2(a), yet their mean returns remain well below ARCFLOW at every displayed budget in panel (b). Conversely, a method can achieve a higher aggregate return while still containing tasks with severe one-to-sixteen-step degradation. Evaluating low-step return together with within-policy retention is therefore necessary: the absence of collapse alone does not imply an effective low-latency policy.

4 THE ARCFLOW ALGORITHM

Figure 1 previews ARCFLOW: its left panel contrasts the budget-dependent rollout action with a reference endpoint; its right panel connects a collected action to fresh noise on a fixed-grid bridge. The boundary residual measures one-step reconstruction error, while temporal probes yield one clipped surrogate weight per action. Rollout budgets cycle, and worst-budget validation selects the checkpoint. The deployment target is

$$\mathcal{J}(\theta) = \mathbb{E}_{K\sim p}[J_K(\theta)]. \tag{2}$$

Algorithm 1 specifies this control loop. The native-objective image-policy variant instead uses multi-budget endpoint supervision.

4.1 BUDGET-CYCLED ROLLOUTS

For $z \sim \mathcal{N}(0, I)$, a reverse Euler rollout produces

$$x_0 = z, \qquad x_{j+1} = x_j - h v_\theta(x_j, t_j \mid s), \qquad t_j = 1 - j/K, \quad h = 1/K, \quad a = x_K. \tag{3}$$

Training cycles through $\mathcal{S} = \{1, 2, 3, 4, 6, 8, 12, 16\}$ in increasing order, using one K for every rollout in an outer iteration. This gives equal exposure to each supported budget over a cycle. Parameters evolve across the cycle, and the field is not conditioned on K.

4.2 BOUNDARY-AWARE BRIDGE

For each collected tuple $(s, a, \widehat{A})$, draw fresh noise $\widetilde{z} \sim \mathcal{N}(0, I)$ and define

$$\widetilde{x}_t = (1-t)a + t\widetilde{z}, \qquad u = \widetilde{z} - a. \tag{4}$$

The bridge is an analytic reference path to the observed action, not the trajectory that generated it. On the fixed reference grid $G_4 = \{1, 3/4, 1/2, 1/4\}$, sample j uniformly and score

$$\ell_j(\theta; s, a, \widetilde{z}) = h\|v_\theta(\widetilde{x}_{t_j}, t_j \mid s) - u\|^2_{2,d}, \qquad h = 1/4. \tag{5}$$

The mean sampled energy estimates $L_\theta = \mathbb{E}_{\widetilde{z},j}[\ell_j]$. For fixed $\widetilde{z}$, this is $\frac{1}{4}\sum_{j=0}^{3}\ell_j$, not the unnormalized sum of grid residuals. The factor matters because it sets the scale of the exponential weight below. Every collecting solver uses the same grid and scale.

Proposition 1 (Exact one-step reconstruction)**.** *For any fixed $(s, a, \widetilde{z})$, the boundary residual satisfies*

$$h^{-1}\ell_0(\theta) = \|\widetilde{z} - v_\theta(\widetilde{z}, 1 \mid s) - a\|^2_{2,d} = \|F^{S_1}_\theta(\widetilde{z}, s) - a\|^2_{2,d}.$$

The identity follows by substituting $t_0 = 1$ into Eqs. 3–5. It links one quarter of the sampled-node distribution to the reconstruction error of a one-step action. The appendix extends this connection to endpoint error bounds under regularity assumptions (Propositions 3 and 5). The resulting design separates boundary supervision from the fixed-grid bridge used to score actions at every rollout budget.

Action-level surrogate. Let $\bar{\theta}$ be the rollout parameters. Cache M common noise–node probes for the old and new policies and form

$$\widehat{\Delta}_\theta = \frac{1}{M}\sum_{m=1}^{M}[\ell_{j_m}(\bar{\theta}) - \ell_{j_m}(\theta)], \qquad \widehat{w}_\theta = \exp\{\mathrm{clip}(\widehat{\Delta}_\theta, -c_E, c_E)\}. \tag{6}$$

We maximize a PPO-style clipped surrogate (Schulman et al., 2017),

$$\mathcal{J}_{\mathrm{Arc}}(\theta) = \mathbb{E}\left[\min\left(\widehat{w}_\theta\widehat{A}, \mathrm{clip}(\widehat{w}_\theta, 1-\epsilon, 1+\epsilon)\widehat{A}\right)\right]. \tag{7}$$

The probes are averaged before clipping, so each environment action receives one weight and one advantage. The weight is an energy-based surrogate, not the density ratio of the executed solver policy. Since common probes give $\widehat{\Delta}_{\bar{\theta}} = 0$, its local gradient is

$$\nabla_\theta \mathcal{J}_{\mathrm{Arc}}|_{\bar{\theta}} = -\,\mathbb{E}[\widehat{A}\nabla_\theta\widehat{L}_\theta(s,a)]\Big|_{\bar{\theta}}. \tag{8}$$

Positive advantages therefore decrease reconstruction energy locally. Appendix Proposition 4 relates this expression to an auxiliary normalized exponential-tilt policy under its stated centering assumptions; that auxiliary policy need not equal the deployed flow policy.

Algorithm 1 ARCFLOW: bridge-based control realization

Require: parameters θ, support $\mathcal{S}$, reference grid G_4, validation bank $\mathcal{B}_{\mathrm{val}}$
Evaluate θ at $\mathcal{K} = \{1, 4, 16\}$ on $\mathcal{B}_{\mathrm{val}}$.
$\theta^\star \leftarrow \theta$; $s^\star \leftarrow \min_{K\in\mathcal{K}} \widehat{J}_K(\theta; \mathcal{B}_{\mathrm{val}})$
for each training cycle **do**
  for $K \in \mathcal{S} = \{1, 2, 3, 4, 6, 8, 12, 16\}$ in order **do**
    Collect K-step rollouts and estimate advantages.
    Cache M noise–node probes per action and old bridge energies.
    Apply Eqs. 6–7 with those probes; update the critic with the backbone rule.
  Evaluate the current policy at $\mathcal{K}$ on $\mathcal{B}_{\mathrm{val}}$.
  $s \leftarrow \min_{K\in\mathcal{K}} \widehat{J}_K(\theta; \mathcal{B}_{\mathrm{val}})$
  if $s > s^\star$ **then** $(\theta^\star, s^\star) \leftarrow (\theta, s)$
return $\theta^\star$.

Algorithm 1 instantiates the right-hand sequence of Figure 1: cycle K for rollout collection, score actions on fresh fixed-grid probes, and apply one clipped update per action. After each cycle it retains the checkpoint with the highest minimum validation return over $\mathcal{K} = \{1, 4, 16\}$. The rollout path and reference bridge have distinct roles: the former chooses the action that receives an environment advantage, whereas the latter scores that action against fresh noise. Changing K therefore changes which actions enter the update without changing the grid or scale of their bridge energies. Locally, positive advantages lower the energy of collected actions and negative advantages raise it; the cached probes make the old and current energies comparable during each actor update. At every rollout budget, uniform sampling on G_4 assigns probability $1/4$ to $t = 1$. Thus the exact one-step residual in Proposition 1 contributes to the expected actor update even when rollouts take many steps. Averaging the M probes before exponentiation gives each collected action one weight, aligned with its one advantage, rather than separate weights for individual bridge nodes.

4.3 TRAINING AND SELECTION

The reference grid G_4 and its $1/4$ residual scale remain fixed as the rollout budget changes. Actor updates reuse cached probes, and the critic follows the backbone rule. Algorithm 1 evaluates the initial field and each cycle-end checkpoint at $\mathcal{K} = \{1, 4, 16\}$, then selects

$$\theta^\star \in \arg\max_{\theta\in\mathcal{C}} \min_{K\in\mathcal{K}} \widehat{J}_K^{\mathrm{val}}(\theta), \tag{9}$$

where $\mathcal{C}$ is the set of evaluated checkpoints. Taking the minimum prevents a high return at one validation budget from compensating for a low return at another. The selected $\theta^\star$ is one shared field: deployment changes the solver budget, not the checkpoint or the policy parameters. Training and selection have different roles: Eq. 7 updates the field through the advantage-weighted bridge surrogate, whereas Eq. 9 ranks the resulting checkpoints by environment return. No gradient is taken through the validation returns.

Native-objective realization. The bridge surrogate is unnecessary when the native learner already provides velocity supervision. In the released ReinFlow pipeline, ARCFLOW retains the native actor objective as an anchor, collects at $K = 4$, and supervises deterministic endpoint views at $\mathcal{V} = \{1, 2, 4, 8, 16\}$ against stored rollout actions, weighted by normalized advantage midranks within each minibatch. Let g_0 be the native ascent direction and g_K the negative gradient of the endpoint loss at budget K; write u_0, u_K for their unit directions. The update preserves the native gradient norm while combining its direction equally with a max–min direction across views:

$$p^\star \in \arg\max_{\|p\|\le 1,\ \langle p,u_0\rangle\ge 0} \min_{K\in\mathcal{V}} \langle p, u_K\rangle, \qquad d = \|g_0\| \frac{u_0 + p^\star}{\|u_0 + p^\star\|}. \tag{10}$$

The native AdamW optimizer proposes a step along d. Backtracking accepts a finite step only if endpoint losses stay within a small tolerance of their pre-step values and common-noise endpoints remain close across budgets. This check reuses training samples and makes no extra reward queries.

Appendix B.1 gives its thresholds, clock views, fallback, and optimizer-state handling. The separate cycling baseline changes only the collecting solver; both realizations deploy one field.

Table 1: **Cross-domain evaluation at three deployment budgets.** Deployed episodic return for ARCFLOW and five flow-policy baselines on 21 DM Control, three manipulation, and six locomotion tasks. Bold marks the highest displayed mean for each task and budget.

Task	FPO			FPO++			DPPO			ReinFlow			PolicyFlow			ARCFLOW		
K	1	4	16	1	4	16	1	4	16	1	4	16	1	4	16	1	4	16
DM Control																		
BallInCup	738	856	856	656	852	841	683	647	607	735	850	850	935	937	935	**937**	**949**	**949**
CartpoleBalance	611	695	732	487	793	816	668	659	631	772	**999**	891	522	496	492	**956**	959	**953**
CartpoleSwingupSparse	207	**522**	**495**	201	421	423	12.1	12.3	11.2	274	286	263	4.53	4.94	4.67	**284**	300	299
CheetahRun	409	**894**	**895**	156	827	813	469	457	437	410	728	706	672	677	677	**849**	868	859
FingerSpin	299	**670**	**666**	276	588	571	54.2	55.0	51.7	200	407	317	457	477	477	**554**	612	585
FingerTurnEasy	710	**894**	**901**	689	758	755	498	503	508	626	743	738	722	728	725	**820**	870	860
FingerTurnHard	430	755	**760**	487	656	647	383	401	398	517	629	617	663	671	667	**705**	**758**	751
FishSwim	575	**638**	**635**	485	588	580	187	189	185	320	500	497	193	195	196	**600**	620	616
HopperHop	14.4	32.2	30.6	22.8	36.6	33.0	0.30	0.30	0.30	16.4	44.0	44.0	0.44	0.41	0.42	**60.4**	**74.1**	**73.2**
HopperStand	85.2	108	110	39.4	75.1	70.9	4.92	4.89	4.74	115	243	234	4.28	3.89	3.90	**361**	**503**	**496**
HumanoidRun	**6.98**	**7.58**	**7.29**	1.51	5.77	4.98	4.73	4.74	4.59	2.42	3.22	2.73	2.90	2.92	2.90	4.94	5.66	5.41
HumanoidStand	23.8	26.0	24.3	17.7	**30.8**	**24.8**	**26.5**	25.7	23.8	9.06	11.4	9.66	14.2	14.2	14.2	18.5	23.0	21.5
HumanoidWalk	**12.2**	**12.9**	**13.0**	7.19	12.5	9.91	12.1	12.0	11.6	5.35	6.06	5.98	7.31	8.07	8.13	11.3	12.3	11.7
PendulumSwingup	35.9	56.7	56.2	63.9	66.8	63.9	0.98	1.09	1.02	19.5	30.6	28.3	8.34	7.65	7.77	**116**	**142**	**141**
PointMass	715	840	836	610	830	842	777	771	756	849	869	869	851	853	852	**881**	**885**	**883**
ReacherEasy	921	968	968	927	966	963	931	930	923	866	964	964	929	935	934	**968**	**970**	**969**
ReacherHard	675	**958**	**956**	770	956	954	840	834	808	899	945	942	846	849	849	**956**	955	954
Swimmer6	508	545	544	409	444	434	319	321	314	409	478	482	261	261	261	**591**	**595**	**591**
WalkerRun	416	**620**	**614**	292	484	475	80.6	78.8	77.1	130	312	308	290	289	289	**537**	567	563
WalkerStand	949	965	964	871	939	938	344	343	340	477	638	638	773	779	781	**970**	**972**	**971**
WalkerWalk	877	**955**	**956**	606	839	825	286	274	265	408	734	721	660	662	663	**942**	950	949
Manipulation																		
PandaPickCube	700	750	748	332	392	402	267	265	258	119	130	132	138	151	158	**719**	**769**	**754**
PandaPickCubeOrient	803	863	860	250	295	303	273	271	264	109	115	115	145	165	179	**817**	**931**	**919**
PandaOpenCabinet	131	164	167	50.7	58.0	57.8	225	224	220	92.2	94.9	93.8	118	131	133	**618**	**632**	**617**
Locomotion																		
BarkourJoystick	**6.06**	6.59	6.50	4.58	**7.07**	6.83	0.72	0.70	0.61	4.72	7.02	6.96	0.53	0.54	0.56	6.04	6.22	**7.15**
H1InplaceGait	13.5	15.5	15.5	11.1	18.0	17.9	8.84	8.77	8.52	11.8	**23.2**	**23.4**	10.6	10.8	10.9	**13.9**	14.6	19.6
H1JoystickGait	2.30	3.28	3.30	0.96	2.11	2.10	0.06	0.06	0.06	0.69	**4.61**	**4.54**	0.21	0.22	0.22	**2.73**	3.48	4.22
Op3Joystick	7.54	10.8	11.0	6.68	11.1	10.6	0.20	0.20	0.17	1.58	11.8	11.8	0.79	0.85	0.89	**10.5**	**11.9**	**11.9**
SpotGetup	7.23	7.55	7.53	7.16	**8.83**	**8.81**	2.14	2.13	2.12	5.86	8.25	8.22	3.86	4.58	4.66	**7.66**	8.78	8.65
SpotJoystickGait	22.9	28.0	28.0	23.2	29.7	29.3	16.5	16.4	16.1	**26.5**	**34.0**	**33.9**	19.8	15.8	15.0	25.5	29.5	29.2

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Common-stack benchmark. We instantiate all methods in a control stack derived from the released FPO Playground configuration. It covers 21 DM Control tasks (Tassa et al., 2018), 64 outer iterations, and 62.9 million interactions; fixed-budget arms collect at 4 steps. We compare FPO (McAllister et al., 2025), FPO++ (Yi et al., 2026), DPPO (Ren et al., 2025), ReinFlow (Zhang et al., 2025b), and PolicyFlow (Yang et al., 2026). All arms share the normalizer, architecture, initialization, critic, and training-interaction budget. The DPPO row applies its denoising-MDP policy-gradient objective to the shared flow actor, not the original diffusion actor; Appendix D specifies the adaptation. Raw returns use each configuration's registered seeds and display regime; the Panda comparisons are cross-regime. Figure 2 min–max normalizes each task over six methods, five budgets, and a Gaussian reference. "Collapse" means $R_1/R_{16} < 0.5$. The separate `can` study uses five matched training seeds.

5.2 MULTI-BUDGET PERFORMANCE

Broad low-budget gains. Table 1 reports the deployed return of all six methods at three deployment budgets on 30 tasks. At $K = 1$, ARCFLOW has the highest displayed mean on 25 of 30 tasks under the registered display regimes: 18 of 21 DM Control tasks, all 3 manipulation tasks, and 4 of six locomotion tasks. Across the 25 wins, the strongest baseline is FPO on 11 tasks, PolicyFlow on six, ReinFlow on four, and FPO++ and DPPO on two each. The gains are also often substantial. Using the displayed means, 12 of the 25 wins improve on the strongest baseline by at least 10%,

and nine improve by at least 20%. The largest multiplicative gains occur on HopperStand, where ARCFLOW reaches 361 versus 115 for the strongest baseline; HopperHop, with 60.4 versus 22.8; and PandaOpenCabinet, with 618 versus 225. The advantage is not limited to these outliers: at one step, ARCFLOW also obtains 956 versus 772 on CartpoleBalance, 849 versus 672 on CheetahRun, 554 versus 457 on FingerSpin, and 10.5 versus 7.54 on Op3Joystick. Thus the lowest-cost setting yields both broad task coverage and pronounced gains on several of the more difficult control and manipulation problems.

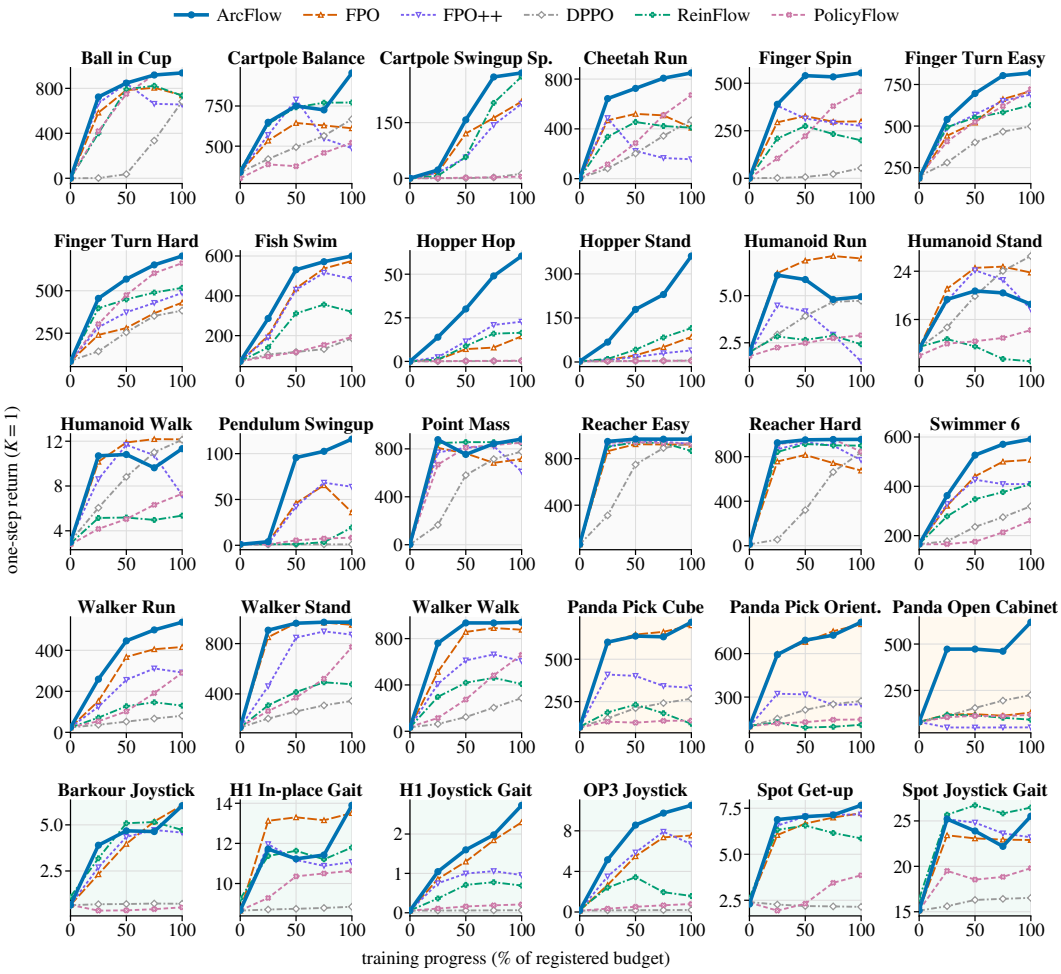


Figure 3: **One-step learning dynamics across 30 tasks.** Mean raw episodic return at $K = 1$, ordered as in Table 1. Training progress is the percentage of each method's registered interaction budget. Curves use the registered seeds. Axis ranges are task-specific; compare within panels.

Domain and budget coverage. The manipulation block provides the clearest uniform result: ARCFLOW has the highest displayed mean at all three budgets on PandaPickCube, PandaPickCube-Orient, and PandaOpenCabinet. In DM Control, it is best at all three budgets on eight tasks, including BallInCup, HopperHop, HopperStand, PendulumSwingup, PointMass, ReacherEasy, Swimmer6, and WalkerStand; it also leads all three budgets on Op3Joystick in locomotion. Across the full table, ARCFLOW is best on 13 tasks at $K = 4$ and 14 at $K = 16$, and on 12 tasks at every reported budget. Task-level leaders are therefore more mixed at higher solver counts than at one step, but the table shows that ARCFLOW combines a broad low-budget advantage with strong performance as additional computation becomes available. These task-level counts identify where each method leads; Figure 2(b) instead shows how each trained field performs as the deployment budget changes, averaged over the 30 tasks. ARCFLOW has the highest mean normalized return at every evaluated solver count, with the largest separation at one step. Its mean rises from one to two steps and then

remains nearly constant through sixteen steps. Figure 2(a) likewise records no one-step collapse for ARCFLOW across the 30-task census. Although the strongest high-budget method varies on the remaining tasks, the aggregate curve stays above all baselines. Thus the one-step gains do not come at the cost of losing the policy's higher-budget capability.

Table 2: **Boundary supervision, bridge training, and worst-K selection improve deployment robustness.** Point-estimate means on a 25-task component pool. $K = 1$ and Worst K evaluate normalized deployment return, with Worst K summarizing the weakest budget over $\mathcal{K}$. AUC is the mean four-step training-curve area, min–max normalized within each task over these rows. Bold: column maximum.

Configuration	$K = 1$	Worst K	AUC
FPO	0.350	0.334	0.587
+ u-supervision	0.534	0.508	0.333
bridge, fixed $K = 4$	0.612	0.593	**0.700**
bridge + cycle $\{1, 4, 16\}$	0.646	0.635	0.612
bridge + cycle $\mathcal{S}$	0.668	0.623	0.661
ARCFLOW (+ worst-K)	**0.711**	**0.685**	0.667

Table 3: **ARCFLOW restores full-range manipulation.** Mean `can` success (%) over five matched training seeds. BC is the behavior-cloned prior; "1-step" is the specialist. Worst and mean K summarize the five budget means. Bold: row maximum.

Budget	BC	ReinFlow	1-step	Cycling	ARCFLOW
$K = 1$	56.2	1.5	**98.0**	85.3	91.1
$K = 2$	56.2	71.7	83.3	**96.7**	95.3
$K = 4$	59.7	92.5	78.3	**98.2**	93.5
$K = 8$	65.0	92.4	74.6	78.3	**96.3**
$K = 16$	70.6	93.8	83.9	88.9	**97.1**
Worst K	56.2	1.5	74.6	78.3	**91.1**
Mean K	61.5	70.4	83.6	89.5	**94.7**

5.3 TRAINING DYNAMICS

Early emergence of low-step gains. Figure 3 shows that the deployed advantage often emerges before training ends. On CheetahRun, FingerSpin, HopperHop, HopperStand, the Walker tasks, and the manipulation tasks, ARCFLOW separates from most baselines by roughly mid-training. Its Hopper returns rise steadily while competing curves remain near the floor or plateau lower, so Table 1's differences are not created solely by the deployed endpoints.

Stable and late-stage improvement. On BallInCup, PointMass, ReacherEasy, and ReacherHard, ARCFLOW rises early and remains stable; PendulumSwingup, Swimmer6, H1JoystickGait, and Op3Joystick keep improving after mid-training. On CartpoleBalance, PandaOpenCabinet, and H1InplaceGait, the selected rightmost marker rises above the preceding chronological trajectory. The deployed result thus combines early acquisition, continued gains, and checkpoint selection.

5.4 CONTROL COMPONENT ANALYSIS

Table 2 isolates the contribution of each component on the separate 25-task component-study pool. Replacing noise reconstruction with direct u-supervision raises normalized one-step return from 0.350 to 0.534 and worst-budget return from 0.334 to 0.508, confirming that restoring the boundary signal accounts for a substantial part of the recovery. This direct repair alone reduces training AUC to 0.333. Introducing the fixed-scale bridge at $K = 4$ raises one-step and worst-budget return further to 0.612 and 0.593, while producing the highest training AUC, 0.700. This row shows the combined benefit of the fixed-grid bridge formulation and its fixed scale.

Budget exposure provides the next improvement. Cycling over $\{1, 4, 16\}$ raises the worst-budget return to 0.635, and the broader support $\mathcal{S}$ reaches 0.668 at one step with training AUC 0.661. Worst-budget checkpoint selection completes the algorithm: the full ARCFLOW row reaches 0.711 at one step and 0.685 at its weakest evaluated budget, the strongest deployment values in the table. The ladder shows that the components play distinct roles: boundary supervision repairs low-step execution, the fixed-grid bridge improves the training curve, cycling supplies budget coverage, and selection converts that coverage into a stronger deployment floor.

5.5 NATIVE-PIPELINE VALIDATION

Table 3 tests whether the same deployment-budget principle transfers beyond the reconstruction-based control realization. The released ReinFlow ShortCut pipeline starts from a behavior-cloned image policy, uses native velocity supervision, and fine-tunes a four-step actor. It tests the principle beyond

FPO's reconstruction target and compares one multi-budget policy with a one-step specialist and rollout-budget cycling. The released behavior-cloned prior retains moderate performance across solver budgets, but official ReinFlow fine-tuning raises four-step success to 92.5% while reducing one-step success to 1.5%. The specialist reverses this tradeoff: it reaches 98.0% at one step but has a worst-budget score of only 74.6%. Cycling performs best at two and four steps, yet falls to 78.3% at its weakest budget. In contrast, ARCFLOW remains between 91.1% and 97.1% across all five budgets, giving the highest worst-budget score (91.1%) and mean score (94.7%). Thus, its advantage is the high performance floor obtained with one field across the deployment range.

6 RELATED WORK

Generative transport and control. Diffusion models learn reverse transport from noise to data (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021b), while deterministic samplers and neural ODEs expose an accuracy–computation tradeoff (Song et al., 2021a; Chen et al., 2018). Flow matching and rectified flow learn velocity fields directly (Lipman et al., 2023; Liu et al., 2023); transport couplings and multisample matching straighten paths, and rectified flows scale to high-resolution generation (Tong et al., 2024; Pooladian et al., 2023; Esser et al., 2024). In control, generative models support planning and return conditioning (Janner et al., 2022; Ajay et al., 2023), plus visuomotor imitation across 2D, goal-conditioned, and 3D inputs (Chi et al., 2023; Reuss et al., 2023; Ze et al., 2024). AdaFlow adapts integration to action variance and FlowPolicy learns one-step consistency (Hu et al., 2024; Zhang et al., 2025a). This work instead trains one flow field with RL for a declared set of deployment budgets.

Reinforcement learning of generative policies. Trust-region and clipped policy objectives motivate many modern policy updates (Schulman et al., 2015; 2017). DDPO casts denoising as an MDP, while DRaFT differentiates rewards through the sampler (Black et al., 2024; Clark et al., 2024). For control, DPPO applies policy gradients to diffusion-policy denoising transitions and QVPO uses a value-weighted update (Ren et al., 2025; Ding et al., 2024). ReinFlow makes flow transitions stochastic; FPO and FPO++ form PPO-style ratios from flow losses; PolicyFlow and Flow-GRPO derive velocity- or transition-based objectives (Zhang et al., 2025b; McAllister et al., 2025; Yi et al., 2026; Yang et al., 2026; Liu et al., 2025). Offline and critic-based methods include Diffusion-QL, EDP, DTQL, BDPO, Q-score matching, Flow Q-Learning, and SAC Flow (Wang et al., 2023; Kang et al., 2023; Chen et al., 2024; Gao et al., 2025; Psenka et al., 2024; Park et al., 2025; Zhang et al., 2026). ARCFLOW focuses on the return floor of one policy across solver budgets.

Few-step and one-step policies. DDIM and DPM-Solver accelerate sampling numerically (Song et al., 2021a; Lu et al., 2022); progressive distillation, consistency models, and InstaFlow learn fast generators (Salimans & Ho, 2022; Song et al., 2023; Liu et al., 2024). Shortcut models and MeanFlow train directly over variable or one-step generation (Frans et al., 2025; Geng et al., 2025), and Consistency Policy distills a visuomotor policy (Prasad et al., 2024). One-step flow policy mirror descent, FLAME, and MFPO bring this objective to RL (Chen et al., 2025; Li et al., 2026; Dong et al., 2026). These methods optimize a designated short sampler or fast policy; ARCFLOW preserves one RL-trained field and measures it at every declared $K \in \mathcal{K}$.

Training-time weighting and deployment budgets. P2, Min-SNR, and EDM reweight diffusion times (Choi et al., 2022; Hang et al., 2023; Karras et al., 2022). Our problematic weight instead follows algebraically from noise reconstruction and is zero at the one-step boundary. Boundary supervision restores that signal, solver cycling exposes the intended budgets, and worst-budget selection makes their return floor an explicit selection criterion.

7 CONCLUSION

ARCFLOW aligns flow-policy learning with its deployment range through boundary supervision, solver-budget exposure, and worst-budget selection. Across three control domains it improves one-step performance while retaining competitive higher-budget returns; the native ReinFlow study extends the principle to a released image-policy pipeline and five budgets. Deployment retains one architecture and pays only for the selected solver budget. Component studies further separate the gains from action-aligned supervision, budget exposure, and selection. These results support training and evaluating generative policies over their full deployment range.

AI USE STATEMENT

Generative AI tools were used to help develop and critique the conceptual framework and hypotheses; formulate mathematical claims and proofs; design synthetic environments and experimental protocols; implement, test, and edit software; interpret results; search literature; prepare figures and tables; and draft and edit the manuscript. AI-generated outputs were not treated as empirical evidence: reported values come from versioned artifacts, citations are checked against primary sources, and assumptions and proofs are stated in the paper. Generative AI was not used to generate or clean experimental datasets or to translate source material; qualitative or thematic data analysis was not applicable. We reviewed the AI-assisted work and take responsibility for the final text, claims, code, and artifacts.

ETHICS STATEMENT

This work studies reinforcement learning for simulated control and does not involve human participants, personal data, or physical-robot experiments. Like other reinforcement-learning methods, ARCFLOW may amplify errors or unintended incentives in a specified reward function; improved optimization does not by itself provide safety guarantees. Applying the method to physical or safety-critical systems would therefore require independently validated objectives, operational constraints, monitoring, and fail-safe mechanisms. We report task-level outcomes and the limits of the present evaluation to support accurate interpretation of the results.

REPRODUCIBILITY STATEMENT

Code and artifacts are available in the anonymous repository at `https://anonymous.4open.science/r/arcflow-868A`. The repository contains the ARCFLOW implementation, experiment configurations. The main text and appendices specify the algorithms, solver budgets, checkpoint-selection rule, evaluation conventions, and matched five-seed manipulation protocol; the repository provides the experiment configurations.

REFERENCES

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B. Tenenbaum, Tommi S. Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=sP1fo2K9DFG.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/14f75513f0f1ca01de1e826b52e6b840-Abstract-Conference.html.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K. Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/69386f6bb1dfed68692a24c8686939b9-Abstract.html.

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

Tianyu Chen, Zhendong Wang, and Mingyuan Zhou. Diffusion policies creating a trust region for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/59a48c111f97f2174709ea9ed8e920d1-Abstract-Conference.html.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.026. URL https://roboticsproceedings.org/rss19/p026.html.

Jooyoung Choi, Jungbeom Lee, Chaehun Shin, Sungwon Kim, Hyunwoo Kim, and Sungroh Yoon. Perception prioritized training of diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11472–11481, 2022. URL https://openaccess.thecvf.com/content/CVPR2022/html/Choi_Perception_Prioritized_Training_of_Diffusion_Models_CVPR_2022_paper.html.

Kevin Clark, Paul Vicol, Kevin Swersky, and David J. Fleet. Directly fine-tuning diffusion models on differentiable rewards. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/13e8be77982beb73d7ed0bbf122f9f3c-Abstract-Conference.html.

Shutong Ding, Ke Hu, Zhenhao Zhang, Kan Ren, Weinan Zhang, Jingyi Yu, Jingya Wang, and Ye Shi. Diffusion-based reinforcement learning via Q-weighted variational policy optimization. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://openreview.net/forum?id=UWUUVKtKeu.

Xiaoyi Dong, Xi Sheryl Zhang, and Jian Cheng. Mean flow policy optimization. *arXiv preprint arXiv:2604.14698*, 2026. URL https://arxiv.org/abs/2604.14698.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 12606–12633. PMLR, 2024. URL https://proceedings.mlr.press/v235/esser24a.html.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.12557.

Chen-Xiao Gao, Chenyang Wu, Mingjun Cao, Chenjun Xiao, Yang Yu, and Zongzhang Zhang. Behavior-regularized diffusion policy optimization for offline reinforcement learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*. PMLR, 2025. URL https://openreview.net/forum?id=pUCYJ9JJuZ.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J. Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*, 2025. URL https://arxiv.org/abs/2505.13447.

Tiankai Hang, Shuyang Gu, Chen Li, Jianmin Bao, Dong Chen, Han Hu, Xin Geng, and Baining Guo. Efficient diffusion training via min-SNR weighting strategy. In *IEEE/CVF International Conference on Computer Vision*, pp. 7441–7451, 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Hang_Efficient_Diffusion_Training_via_Min-SNR_Weighting_Strategy_ICCV_2023_paper.html.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html.

Xixi Hu, Bo Liu, Xingchao Liu, and Qiang Liu. AdaFlow: Imitation learning with variance-adaptive flow-based policies. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/fa9cfdb49f7de3dee54007f84c0745b1-Abstract-Conference.html.

Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9902–9915. PMLR, 2022. URL https://proceedings.mlr.press/v162/janner22a.html.

Bingyi Kang, Xiao Ma, Chao Du, Tianyu Pang, and Shuicheng Yan. Efficient diffusion policies for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/d45e0bfb5a39477d56b55c0824200008-Abstract.html.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/a98846e9d9cc01cfb87eb694d946ce6b-Abstract-Conference.html.

Zeqiao Li, Yijing Wang, Haoyu Wang, Zheng Li, and Zhiqiang Zuo. Boosting maximum entropy reinforcement learning via one-step flow matching. *arXiv preprint arXiv:2602.01606*, 2026. URL https://arxiv.org/abs/2602.01606.

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*, 2023. URL https://arxiv.org/abs/2210.02747.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-GRPO: Training flow matching models via online RL. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/3a10c46572628d58cb44fb705f25cbbf-Abstract-Conference.html.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2209.03003.

Xingchao Liu, Xiwen Zhang, Jianzhu Ma, Jian Peng, and Qiang Liu. InstaFlow: One step is enough for high-quality diffusion-based text-to-image generation. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/4dc37a7bc61057252ce043fa3b83aac2-Abstract-Conference.html.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-Solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/260a14acce2a89dad36adc8eefe7c59e-Abstract-Conference.html.

David McAllister, Songwei Ge, Brent Yi, Chung Min Kim, Ethan Weber, Hongsuk Choi, Haiwen Feng, and Angjoo Kanazawa. Flow matching policy gradients. *arXiv preprint arXiv:2507.21053*, 2025. URL https://arxiv.org/abs/2507.21053.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48104–48127. PMLR, 2025. URL https://proceedings.mlr.press/v267/park25f.html.

Aram-Alexandre Pooladian, Heli Ben-Hamu, Carles Domingo-Enrich, Brandon Amos, Yaron Lipman, and Ricky T. Q. Chen. Multisample flow matching: Straightening flows with minibatch couplings. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 28100–28127. PMLR, 2023. URL https://proceedings.mlr.press/v202/pooladian23a.html.

Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, and Jeannette Bohg. Consistency policy: Accelerated visuomotor policies via consistency distillation. In *Proceedings of Robotics: Science and Systems*, 2024. doi: 10.15607/RSS.2024.XX.071. URL https://roboticsproceedings.org/rss20/p071.html.

Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via Q-score matching. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 41163–41182. PMLR, 2024. URL https://proceedings.mlr.press/v235/psenka24a.html.

Allen Z. Ren, Justin Lidard, Lars Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=mEpqHvbD2h.

Moritz Reuss, Maximilian Li, Xiaogang Jia, and Rudolf Lioutikov. Goal-conditioned imitation learning using score-based diffusion policies. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.028. URL https://roboticsproceedings.org/rss19/p028.html.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2202.00512.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1889–1897. PMLR, 2015. URL https://proceedings.mlr.press/v37/schulman15.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2256–2265. PMLR, 2015. URL https://proceedings.mlr.press/v37/sohl-dickstein15.html.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=St1giarCHLP.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=PxTIG12RRHS.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 32211–32252. PMLR, 2023. URL https://proceedings.mlr.press/v202/song23a.html.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. DeepMind control suite. *arXiv preprint arXiv:1801.00690*, 2018. URL https://arxiv.org/abs/1801.00690.

Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, pp. 1–34, 2024. URL https://openreview.net/forum?id=CD9Snc73AW.

Zhendong Wang, Jonathan J. Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=AHvFDPi-FA.

Shunpeng Yang, Ben Liu, and Hua Chen. PolicyFlow: Policy optimization with continuous normalizing flow in reinforcement learning. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=YETCQLcKtn.

Brent Yi, Hongsuk Choi, Himanshu Gaurav Singh, Xiaoyu Huang, Takara E. Truong, Carmelo Sferrazza, Yi Ma, Rocky Duan, Pieter Abbeel, Guanya Shi, Karen Liu, and Angjoo Kanazawa. Flow policy gradients for robot control. *arXiv preprint arXiv:2602.02481*, 2026. URL https://arxiv.org/abs/2602.02481.

Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3D diffusion policy: Generalizable visuomotor policy learning via simple 3D representations. In *Proceedings of Robotics: Science and Systems*, 2024. doi: 10.15607/RSS.2024.XX.067. URL https://roboticsproceedings.org/rss20/p067.html.

Qinglun Zhang, Zhen Liu, Haoqiang Fan, Guanghui Liu, Bing Zeng, and Shuaicheng Liu. FlowPolicy: Enabling fast and robust 3d flow-based policy via consistency flow matching for robot manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 14754–14762, 2025a. doi: 10.1609/aaai.v39i14.33617. URL https://ojs.aaai.org/index.php/AAAI/article/view/33617.

Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. ReinFlow: Fine-tuning flow matching policy with online reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, 2025b. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/98d6f928497d9eaad9f320fb2db3040d-Abstract-Conference.html.

Yixian Zhang, Shu'ang Yu, Tonghe Zhang, Mo Guang, Haojia Hui, Kaiwen Long, Yu Wang, Chao Yu, and Wenbo Ding. SAC Flow: Sample-efficient reinforcement learning of flow-based policies via velocity-reparameterized sequential modeling. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=zZvWj4JrYj.

A LIMITATIONS AND SCOPE

Empirical scope. Our evidence is limited to simulated control. The common-stack study covers DM Control, simulated Panda manipulation, and locomotion tasks, while the native image-policy study examines one manipulation task. These experiments do not establish robustness to real-robot sensing and actuation, safety constraints, distribution shift, or substantially larger policy and action spaces. The native study also implements the deployment-budget principle with endpoint supervision rather than the bridge objective, so it supports transfer of the principle but is not an independent test of the same update rule.

Comparison and statistical scope. Most baselines in the common-stack benchmark are controlled adaptations to a shared flow backbone, critic, and interaction budget; they are not reproductions of each method's native architecture, pretraining, or published benchmark.

Training and selection cost. ARCFLOW leaves deployment architecture and per-step solver cost unchanged, but training is not free: the bridge realization evaluates cached probes, cycles through several rollout budgets, and repeatedly measures validation return at $K \in \{1, 4, 16\}$. In the reported control protocol, validation adds 2.75% to the environment-interaction count, in addition to neural-network compute from the probes and multi-budget evaluations. Moreover, selection protects only the budgets it evaluates and gives no guarantee for unseen solver counts.

Objective and theory. The bridge weight is an energy-based surrogate, not the likelihood ratio of the executed policy, and PPO-style clipping therefore has no exact trust-region interpretation. The appendix bounds finite-solver reconstruction error for a deterministic Euler field under Lipschitz assumptions; it does not prove monotone improvement in environment return, and it does not directly cover the clipped ShortCut operator used in the native pipeline. The fixed four-node bridge, the chosen budget sets, and the associated thresholds are empirical design choices. Whether the same choices remain effective with adaptive or higher-order solvers, continuous budget ranges, longer horizons, or other generative-policy parameterizations remains open.

B IMPLEMENTATION DETAILS

Algorithm 1 and the complete bridge-based control realization appear in Section 4.3. This supplement gives the mathematical details, baseline scope, energy-scale diagnostic, and exact native-anchored manipulation update.

B.1 NATIVE-OBJECTIVE MANIPULATION UPDATE

The configuration collects with the released ReinFlow ShortCut actor at four steps. Its extra supervision evaluates the same actor at $\mathcal{V} = \{1, 2, 4, 8, 16\}$ using the stored rollout noise and actions. ShortCut retains its native time and duration inputs; the reverse-time field in the bridge realization has only time. The following specifies the actual manipulation update, including its distinction from the bridge loss of Eq. 7.

Endpoint views and advantage weights. For each captured minibatch B of $(s_i, z_i, a_i, \widehat{A}_i)$, z_i and a_i are the initial and final stored chain states. Let $E_{K,\alpha}(\theta; z, s)$ be the released ShortCut field integrated on $t_j = (j/K)^\alpha$, using its native forward-time convention, duration $t_{j+1} - t_j$, and action clipping after every step. The trained views use $\alpha = 1$. Within each minibatch, define normalized empirical midrank weights

$$r_i = \sum_{j \in B} \mathbf{1}\{\widehat{A}_j < \widehat{A}_i\} + \tfrac{1}{2}\left(1 + \sum_{j \in B} \mathbf{1}\{\widehat{A}_j = \widehat{A}_i\}\right), \qquad w_i = r_i \Big/ \sum_{j \in B} r_j. \tag{11}$$

The endpoint objective is

$$L_K^{\mathrm{end}}(\theta) = \sum_B \sum_{i \in B} w_i \| E_{K,1}(\theta; z_i, s_i) - \mathrm{stopgrad}(a_i) \|_{2,d}^2. \tag{12}$$

The loss sums the normalized minibatch losses. All 15 accumulation minibatches on `can` contribute to every view gradient. This matches the native update's accumulation window. Rank weights favor high-advantage actions and handle ties with their average rank; they are not signed policy-gradient coefficients.

Native anchor and direction. Let g_0 be the negative of the accumulated native actor gradient (policy loss plus the release's entropy and behavior-cloning terms), and $g_K = -\nabla L_K^{\mathrm{end}}$. Normalize these to unit vectors u_0, u_K. The small view-space solve is

$$p^\star \in \arg \max_{\|p\| \le 1,\ \langle p, u_0 \rangle \ge 0} \min_{K \in \mathcal{V}} \langle p, u_K \rangle. \tag{13}$$

It is implemented through the simplex-plus-ray dual $\min_{\beta \in \Delta, \lambda \ge 0} \| \sum_K \beta_K u_K + \lambda u_0 \|$, using a float64 Gram matrix and active-set enumeration. The applied direction uses a fixed equal blend,

$$d = \|g_0\| \frac{\frac{1}{2} u_0 + \frac{1}{2} p^\star}{\| \frac{1}{2} u_0 + \frac{1}{2} p^\star \|}. \tag{14}$$

A degenerate max–min solution falls back to $d = g_0$; zero or non-finite input gradients are rejected by the implementation. The norm floor is 10^{-12}, feasibility tolerance 10^{-8}, and cone-apex tolerance the square root of the input dtype's machine epsilon. The equal blend is an explicit native anchor, applied after the max–min solve.

Step acceptance. Replace the actor's gradient by $-d$ and call the native AdamW step to obtain a candidate parameter state θ_c and optimizer state. Try $\theta_f = \theta_0 + f(\theta_c - \theta_0)$ in descending order $f \in \{1, 1/2, 1/4, 1/8, 0\}$. Accept the first finite candidate for which every L_K^{end} increases by at most 10^{-7} from its pre-step value on the captured minibatches and the maximum old-to-candidate common-noise endpoint RMSE is at most $0.25 + 10^{-7}$. The displacement check uses the first 32 stored noise–state pairs and all 15 views $\mathcal{V} \times \{\alpha = 1, 2, 4\}$. The view losses use only the native-clock views; the clock-warped views constrain displacement.

Actor optimizer moments interpolate between the old and candidate moments with the accepted fraction f. A positive fraction retains the candidate step counter; $f = 0$ restores the old actor state completely. The critic's native update is unchanged. Acceptance reuses training advantages and samples and makes no additional reward queries; it is a finite-step surrogate check, not an independent return certificate. The separate cycling control uses the native update and cycles $K \in \{1, 2, 4\}$ at iteration boundaries, with fixed-K validation.

C Mathematical basis

The statements below concern the deterministic reverse Euler field in Section 4. They identify the supervised deployment boundary and bound action reconstruction under explicit regularity assumptions. They do not assert monotone environment return or transfer directly to the clipped ShortCut operator in Appendix B.1.

C.1 Boundary non-coercivity

Proposition 2 (Boundary non-coercivity)**.** *Fix a scalar velocity error c and let $b_\delta(t) = c \max\{0, (t - (1-\delta))/\delta\}$. This perturbation is continuous, has one-step boundary error $b_\delta(1)^2 = c^2$, and has continuous noise-reconstruction risk $\int_0^1 (1-t)^2 b_\delta(t)^2 \, dt = c^2 \delta^3 / 30$. Thus arbitrarily small continuous surrogate risk does not control the deployed one-step endpoint error, even over continuous fields.*

Proof. Substitute $q = (t - (1-\delta))/\delta$ on $[1-\delta, 1]$: $\int_0^1 (1-t)^2 b_\delta(t)^2 \, dt = c^2 \delta^3 \int_0^1 q^2 (1-q)^2 \, dq = c^2 \delta^3 / 30$, while $b_\delta(1)^2 = c^2$. □

C.2 Finite-solver endpoint bound

Proposition 3 (Finite-solver endpoint bound)**.** *With $x_j^\star$ the bridge nodes, $\widehat{x}_j$ the Euler trajectory from the same source, $r_j = v_\theta(x_j^\star, t_j \mid s) - u$, and v_θ L-Lipschitz in its state argument uniformly over the*

grid times,

$$\|\widehat{x}_K - a\| \le h \sum_{j=0}^{K-1} (1+hL)^{K-1-j} \|r_j\|, \quad \|\widehat{x}_K - a\|^2 \le h^2 \left[\sum_j (1+hL)^{2(K-1-j)}\right] \left[\sum_j \|r_j\|^2\right]. \tag{15}$$

Let $e_j = \widehat{x}_j - x_j^\star$, $h = 1/K$, and $r_j = v_\theta(x_j^\star, t_j \mid s) - u$. Lipschitz continuity gives

$$\|e_{j+1}\| \le (1+hL)\|e_j\| + h\|r_j\|. \tag{16}$$

With $e_0 = 0$, induction yields

$$\|e_K\| \le h \sum_{j=0}^{K-1} (1+hL)^{K-1-j} \|r_j\|. \tag{17}$$

Cauchy–Schwarz gives the squared form. □

C.3 One-step identity and local gradient

On the fixed reference grid G_4, $h = 1/4$, $t_0 = 1$, and the boundary query is $\widetilde{x}_{t_0} = \widetilde{z}$. Substituting $u = \widetilde{z} - a$ gives

$$h^{-1}\ell_0(\theta) = \|v_\theta(\widetilde{z}, 1 \mid s) - (\widetilde{z} - a)\|_{2,d}^2 \tag{18}$$

$$= \|\widetilde{z} - v_\theta(\widetilde{z}, 1 \mid s) - a\|_{2,d}^2. \tag{19}$$

The deployed one-step reverse-Euler endpoint from the same source is $F_\theta^{\mathrm{Euler}_1}(\widetilde{z}, s) = \widetilde{z} - v_\theta(\widetilde{z}, 1 \mid s)$, proving Proposition 1.

Write $\widehat{L}_\theta = M^{-1}\sum_m \ell_m(\theta)$, $\widehat{\Delta}_\theta = \widehat{L}_{\bar\theta} - \widehat{L}_\theta$, and $\widehat{w}_\theta = \exp\{\mathrm{clip}(\widehat{\Delta}_\theta, -c_E, c_E)\}$. At $\theta = \bar\theta$, $\widehat{\Delta} = 0$, $\widehat{w} = 1$, and, because the inner cap is locally inactive, $\nabla\widehat{w} = -\nabla\widehat{L}$. Away from the outer clip kink, the actor objective therefore has derivative $-\widehat{A}\nabla\widehat{L}$. Taking expectation establishes Eq. 8. This derivation requires common probes: independent old and new probes would make the sampled energy difference nonzero even at identical parameters.

Proposition 4 (Normalized local-policy interpretation)**.** *Let $L_\theta(s,a) = \mathbb{E}_{\widetilde{z},j}[\ell_j(\theta)]$ and define the normalized local policy*

$$q_\theta(a \mid s) = \frac{\pi_{\bar\theta}(a \mid s)\exp\{L_{\bar\theta}(s,a) - L_\theta(s,a)\}}{Z_\theta(s)}, \quad Z_\theta(s) = \mathbb{E}_{a\sim\pi_{\bar\theta}} \exp\{L_{\bar\theta} - L_\theta\}. \tag{20}$$

Then $q_{\bar\theta} = \pi_{\bar\theta}$ and, for any integrable, θ-independent $A(s,a)$,

$$\nabla_\theta \mathbb{E}_{a\sim q_\theta}[A]|_{\bar\theta} = -\mathbb{E}_{a\sim\pi_{\bar\theta}}\left[(A - \mathbb{E}[A \mid s])\nabla_\theta L_\theta\right]_{\bar\theta}, \tag{21}$$

so with a conditionally centered advantage, Eq. 8 is the exact first-order policy gradient of a normalized exponential tilt of the behavior policy. The finite clipped weight in Eq. 6 inherits no exact likelihood-ratio or trust-region interpretation.

For Proposition 4, differentiate

$$\log q_\theta(a \mid s) = \log \pi_{\bar\theta}(a \mid s) + L_{\bar\theta}(s,a) - L_\theta(s,a) - \log Z_\theta(s). \tag{22}$$

At $\bar\theta$, $q = \pi_{\bar\theta}$ and $\nabla \log Z = -\mathbb{E}_{\pi_{\bar\theta}}[\nabla L]$. Hence

$$\nabla \mathbb{E}_{q_\theta}[A]|_{\bar\theta} = \mathbb{E}_{\pi_{\bar\theta}}\left[A\{-\nabla L + \mathbb{E}_{\pi_{\bar\theta}}[\nabla L \mid s]\}\right] \tag{23}$$

$$= -\mathbb{E}_{\pi_{\bar\theta}}\left[(A - \mathbb{E}[A \mid s])\nabla L\right], \tag{24}$$

which proves Eq. 21.

Energy scale. For $\ell = wD$, write $\widehat{\Delta}_D$ for the probed old-minus-new residual difference. The local update is proportional to $-w\,\mathbb{E}[\widehat{A}\nabla D]$, while the uncapped likelihood-like weight is $\exp(w\widehat{\Delta}_D)$. Thus changing w changes both the local gradient scale and when the numerical ratio reaches the PPO clipping thresholds. A fixed $w = 1/4$ keeps that scale independent of the collecting budget. The matched-scale controls isolate this choice in Appendix E.1.

C.4 CROSS-BUDGET BOUND

For $G_K = \{1 - j/K : j = 0, \dots, K-1\}$ and $r(t) = v_\theta(x_t^\star, t \mid s) - u$, define

$$R_K = \left(\frac{1}{K} \sum_{t \in G_K} \|r(t)\|^2 \right)^{1/2}, \qquad \delta_{K,K'} = \max_{t' \in G_{K'}} \min_{t \in G_K} |t' - t|. \tag{25}$$

Proposition 5 (Cross-budget endpoint control)**.** *Assume the hypotheses of Proposition 3 for a K'-step deployment and additionally that $r(t)$ is H-Lipschitz along the analytic bridge. Then, with $h' = 1/K'$,*

$$\|\widehat{x}_{K'} - a\| \le C_{L,K'} \big(\sqrt{K} R_K + H \delta_{K,K'} \big), \quad C_{L,K'} = h' \sum_{j=0}^{K'-1} (1 + h' L)^{K'-1-j}. \tag{26}$$

Here $\delta_{4,1} = 0$ and $\delta_{4,16} = 3/16$.

For each $t' \in G_{K'}$, choose a nearest $t \in G_K$. The definition of R_K implies $\|r(t)\| \le \sqrt{K} R_K$, while time regularity gives $\|r(t') - r(t)\| \le H \delta_{K,K'}$. Thus every deployment-grid residual is at most $\sqrt{K} R_K + H \delta_{K,K'}$. Substitution into Eq. 15, with $h' = 1/K'$, proves Proposition 5. Large H or a shifted reference bridge makes the bound loose, and it is not a reward theorem. Throughout we keep Euler fixed: the endpoint results assert no invariance to changes of flow geometry, and state which residuals control one specified finite map.

D EXPERIMENTAL PROTOCOL AND BASELINES

Control training and checkpoint protocol. The control run uses 64 outer iterations, or eight passes through $\mathcal{S}$, with eight cached bridge probes per action. It evaluates the initialization and each cycle-end checkpoint at $\mathcal{K} = \{1, 4, 16\}$. At each checkpoint and budget, 64 evaluation episodes split into 32 for selection and 32 for disjoint reporting. The selection half applies Eq. 9; exact ties choose the earliest checkpoint. The report half cannot affect selection and supplies the selected checkpoint's reported return. On tasks with 1,000-step episodes, the nine evaluations add 1,728,000 environment steps, or 2.75% of the 62,914,560-interaction training budget. This is an empirical deployment rule, not a formal robustness certificate.

Common-stack comparison. FPO is the control backbone. FPO++, DPPO, ReinFlow, and PolicyFlow are adaptations to the same architecture, initialization, critic, and interaction budget. Their official releases use other tasks and training regimes. DPPO originally trains a diffusion policy: it treats stochastic denoising transitions as an inner MDP and applies policy gradients to their likelihood ratios (Ren et al., 2025). ReinFlow compares against that diffusion-policy implementation using separately pretrained actors (Zhang et al., 2025b). Our row labeled DPPO instead retains the shared flow velocity field and applies the denoising-MDP objective to Gaussian-noised Euler transitions, $z_{j+1} \mid z_j, o \sim \mathcal{N}(z_j + \Delta t_j v_\theta(z_j, t_j, o), \sigma_j^2 I)$; the PPO ratios use the corresponding transition log densities. Thus it tests the DPPO update on a flow backbone, rather than reproducing the original diffusion actor or its published results. The DM Control comparison starts every arm from a shared random initialization, whereas the original DPPO and ReinFlow studies fine-tune behavior priors. The native image-based ReinFlow study is therefore reported separately. The following distinctions define the baseline rows, rather than implying a reproduction of each paper's native benchmark results. Table 4 summarizes these adaptations.

Table 4: Implementation scope of the shared-backbone baselines.

Lineage	Adaptation and audited controls
FPO	Noise-reconstruction surrogate; four-step collection.
FPO++	Per-probe exponential ratios; the shared PPO objective replaces the release's ASPO objective.
DPPO (flow adaptation)	Gaussian-noised Euler transitions and denoising-MDP likelihood ratios; corrected coordinate reduction and denoising discount. Trained from scratch on the shared flow backbone.
ReinFlow	Stochastic-chain exploration model with the release-style noise-ceiling schedule restored; shared PPO clipping and backbone.
PolicyFlow	Velocity-based surrogate with audited Brownian regularization, entropy, and optimizer settings.

E ADDITIONAL CONTROLS

Table 2 reports the current component ladder. This appendix adds the collecting-grid isolation used to distinguish energy scale from node placement.

E.1 ENERGY SCALE AND NODE PLACEMENT

The collecting-grid controls further separate the time nodes from their weight. Using weight $1/K$ makes the energy scale depend on the collecting solver; holding it at $1/4$ removes that dependence.

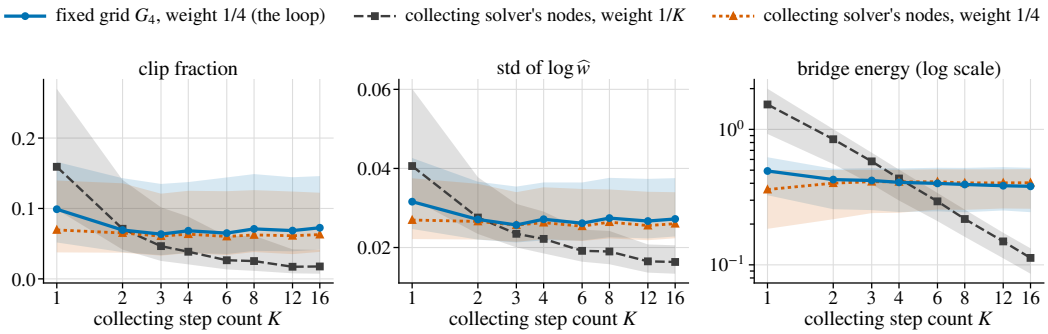


Figure 4: **A fixed energy scale stabilizes the numerical weight across collecting budgets.** Clip fraction, log-weight spread, and bridge energy versus collecting K; medians and quartiles over the selection-pool iterations. The collecting-grid $1/K$ weight changes with budget; both fixed-$1/4$ controls keep the energy scale comparable.

Figure 4 shows that weight $1/K$ makes the bridge energy fall as the collecting budget increases, whereas the fixed-$1/4$ variants keep bridge energy and log-weight spread on comparable scales. This is a scale diagnostic; it does not by itself compare episodic return.