WHAT DOES IT TAKE FOR A GENERALIST ROBOT POLICY TO ADAPT FROM MINIMAL DATA?

Anonymous authors
Paper under double-blind review

ABSTRACT

Abstract. A central goal in robot learning is to move beyond task-specific human data collection toward robots that improve through autonomous interaction. Yet fully autonomous learning remains difficult with current policies: sparse rewards and weak zero-shot exploration often prevent online learning from discovering successful behavior. As a first step towards fully autonomous learning, we propose the **minimal-data adaptation** regime, in which a pretrained robot policy must learn a new task from as little as one demonstration followed by autonomous online interaction. We establish that, although pretrained policies fail zero-shot on many new tasks, a single demonstration suffices to direct their behavioral prior to coarse task-following behavior, albeit with unreliable performance. From this insight, we isolate the key design principles required to convert coarse task-following into reliable task success. These include (1) the surprising efficacy of pretrained VLA representations to support sample-efficient value learning from online interaction (2) the sufficiency of residual policy learning to learn corrective behavior and (3) the need for careful data-balancing to prevent exploration collapse. As a representative of these principles, we propose **MIDAS**, a simple offline-to-online RL recipe that combines few-demonstration behavior cloning with value-based residual learning over frozen VLA representations. Across LIBERO and RoboCasa tasks, MIDAS substantially improves performance from single-demonstration initializations, including policies that succeed only rarely, validating the principles highlighted above. On a bimanual YAM platform, starting from a brittle initial policy, MIDAS improves task success after approximately six hours of online interaction.

1 INTRODUCTION

A long-standing goal in robotics is to develop robots that improve through their own experience: a robot enters a new environment, attempts a task, and autonomously refines its behavior until it can perform the task reliably. Achieving this goal poses challenges ranging from physical considerations, such as providing resets and managing hardware instrumentation, to algorithmic questions about how learning should be initialized and carried out. In this work, we focus on a central algorithmic challenge: enabling rapid adaptation to a new task, with minimal task-specific supervision or design. Prior work on autonomous robot adaptation has largely relied on specialist, single-task policies (Luo et al., 2025; Haldar et al., 2023; Vosylius and Johns, 2025), often coupled with substantial task-specific engineering. The recent emergence of generalist robot policies pretrained on broad and diverse datasets (Team et al., 2024; Brohan et al., 2023b,a; Intelligence et al., 2025; Black et al., 2026; Ye et al., 2026; Kim et al., 2025) provides an opportunity to revisit this problem: can the broad prior encoded by a pretrained generalist policy enable efficient online adaptation with ***only minimal task-specific supervision***?

To answer this question, we first note that autonomous adaptation depends on three ingredients. The policy must represent high-dimensional observations in a form useful for policy and critic learning, produce behavior that reaches reward-bearing regions during deployment, and use this experience to improve efficiently. Given the strong downstream transfer enabled by large-scale pretraining in language, vision, and representation learning (Devlin et al., 2019; Radford et al., 2021; Brown et al., 2020), one would expect recent generalist robot policies (Team et al., 2024; Brohan et al., 2023b,a; Intelligence et al., 2025; Black et al., 2026; Kim et al., 2025) to make progress on the first two

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.

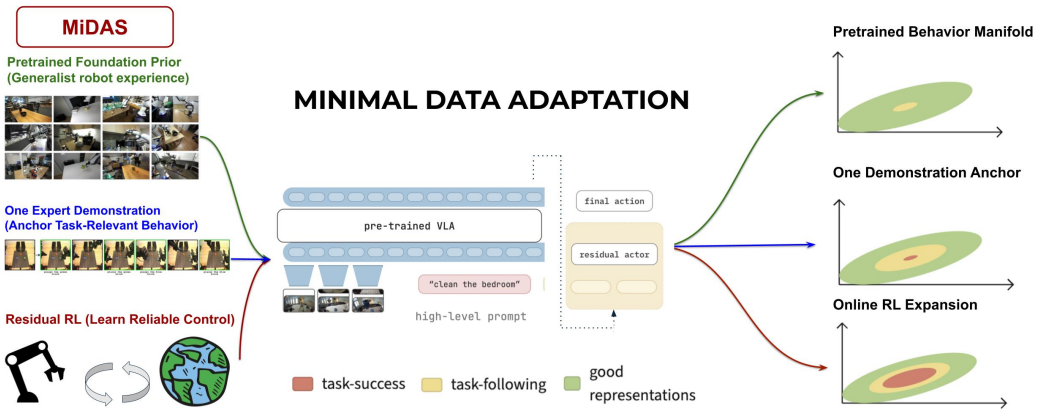


Figure 1: **Overview of minimal-data adaptation.** A single demonstration is sparse and underdetermined on its own, but pretrained VLA representations turn it into a useful anchor for adaptation. Few-demo behavior cloning narrows the search space to a task-relevant region around the demonstration, and online residual RL then expands local success by learning value-guided corrections on top of the warm-started base policy.

ingredients: task-relevant representations for control and behavioral (action) priors acquired from diverse manipulation data. However, these policies are still not reliable zero-shot learners in new environments. They may fail to follow the task instruction, miss sparse reward entirely, or visit only a narrow set of useful states, leaving online learning with too little signal to bootstrap improvement.

We study this gap by proposing the ***minimal-data adaptation (MDA)*** setting, in which a pretrained policy must learn to improve autonomously from only minimal task-specific supervision, such as a single successful demonstration. Prior work has developed effective methods for autonomous policy improvement, but these methods generally operate in more favorable regimes. They typically assume either that the initial policy already has substantial task competence (Wagenmaker et al., 2025; Ren et al., 2024; Hu et al., 2024; Patil et al., 2026) or that tens to hundreds of demonstrations are available (Kumar et al., 2023; Nakamoto et al., 2024; Hu et al., 2024; Ball et al., 2023). While MDA resembles prior autonomous adaptation settings, minimal task-specific supervision makes stable online learning substantially harder: under sparse rewards, a weak initial policy collects mostly failures, often leading to policy collapse. As a result, methods designed for stronger initial policies or larger demonstration sets do not transfer cleanly to this regime. At the same time, few-shot imitation learning methods suggest that learning from one or two demonstrations is possible in some settings (Duan et al., 2017; Finn et al., 2017; Palo and Johns, 2024), but most available VLA policies do not naturally exhibit consistent few-shot improvement. MDA targets the narrow regime between zero-shot deployment and standard task-specific finetuning: the pretrained policy is not reliable enough to deploy directly, but may be close enough that one demonstration can make online learning tractable.

Our main finding is that, in many settings, minimal-data adaptation is tractable for today's generalist robot policies. We show that, while behavior cloning of a single demonstration of a successful trajectory does not provide enough coverage for high rates of task success, a suitably designed online interaction phase can achieve reliable task performance. In particular, ***our work isolates precisely how broad pretraining, minimal per-task demonstration, and online interaction jointly enable adaptation.*** We establish that (1) pretraining features provide not just visual information, but task-relevant features that are essential for learning task-specific critics from limited data (2) that value-based actor-critic policy learning, but not filtered supervised imitation of a policy's own success, affords sufficient exploration to increase task success and (3) how offline and online data should be carefully balanced to prevent being overly constrained by the former, or collapsing by overfitting to the latter. Together, we consolidate these principles in a representative algorithm we call **MiDAS**: an offline-to-online recipe that first fine-tunes a pretrained VLA with imitation learning on K demonstrations to obtain a policy with coarse behaviours, then freezes the VLA and trains a lightweight residual actor on its representations using value-based RL (Mark et al., 2024). Moreover, we carefully test the capabilities and limitations of the generalization afforded by adaptation, showing that: (1) robustness to visual and language changes is largely preserved from pretraining;

(2) online learning makes the task-relevant behavior induced by the demonstration more reliable across nearby object configurations; (3) gradually expanding the reset distribution during online interaction broadens this generalization without additional demonstrations, but settings that require new manipulation strategies remain challenging.

To summarize, we make three contributions in this paper. **First**, we introduce *minimal-data adaptation*, a regime where a pretrained robot policy is not reliable zero-shot, but may be close enough that one or a few demonstrations make online learning tractable. **Second**, we show that ***minimal-data adaptation is possible from as few as 1 demonstration.*** We do so by studying a set of algorithmic choices that enable learning in this regime and identify MIDAS, a simple value-based RL recipe combining few-demo behavior cloning with frozen-backbone residual RL and stabilizing choices for sparse-reward adaptation. Across LIBERO-Long and RoboCasa, we show that MiDAS recovers substantial task success from policies initialized with a single demonstration, even when behavior cloning yields an initial success rate below 5%, significantly outperforming imitation and steering baselines. MIDAS transfers to a real bimanual YAM platform with roughly 6 hours of autonomous interaction. **Third**, we empirically characterize the contributions of demonstrations to task-level behavior, pretrained VLA representations to sample-efficient value learning, and online residual RL to corrective actions beyond those the few-demo policy reliably produces. We systematically study the capabilities and limitations of generalization achieved by successful adaptation in the MDA regime.

2 Problem Setting: Minimal Data Adaptation

We now define the **minimal-data adaptation** (MDA) setting and motivate why it is a useful regime to study. Given a pretrained vision-language-action policy π_{base}, K successful demonstrations $\mathcal{D}^K_{\text{demo}}$ collected in a target environment (denoted by $\mathcal{M}_{\text{target}}$), and access to autonomous interaction with $\mathcal{M}_{\text{target}}$, minimal-data adaptation asks whether we can learn an adapted policy π^K_θ that achieves high return in $\mathcal{M}_{\text{target}}$ using only this limited supervision. The goal is not to solve the task from demonstrations alone. Instead, the demonstrations serve as a minimal scaffold that makes subsequent autonomous learning tractable. The case $K{=}1$ is the sharpest version of this setting: learning on a single demonstration is often insufficient to learn previously unseen tasks, but may still expose enough task-relevant behavior for interaction-driven improvement to begin.

MDA targets the regime where autonomous learning first becomes tractable. With no task-specific data, current pretrained policies often fail to reach task-relevant behavior often enough for sparse-reward interaction to improve them. A small number of demonstrations can make the problem just structured enough: they expose the intended behavior and bring the policy into the vicinity of success, while still leaving robustness and distribution coverage to be learned through autonomous interaction. This setting therefore tests whether pretrained robot policies have reached a regime where a few demonstrations can specify the task while online interaction supplies the missing robustness. It also separates the roles of the available sources of supervision: the structure inherited from pretraining, the task information provided by demonstrations, and the corrections learned through autonomous interaction.

3 Preliminaries

As defined in Section 2, the MDA regime seeks to learn a policy through autonomous interaction with a target environment $\mathcal{M}_{\text{target}}$. The objective of this autonomous learning is to solve the target task consistently, and in as few environment interactions as possible. We formalize this objective in the standard reinforcement learning (RL) setting. Concretely, the goal is to learn a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ that maximizes the discounted return $J(\pi; \mathcal{M}) = \mathbb{E}^{\mathcal{M},\pi}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$, where $s_0 \sim \rho$, $a_t \sim \pi(\cdot \mid s_t)$, and $s_{t+1} \sim P(\cdot \mid s_t, a_t)$. We focus on sparse binary rewards, where $r_t = 0$ if the task is completed within the time limit and -1 otherwise. Because the policy incurs a penalty of -1 at every step until the task is solved, maximizing this cumulative return implicitly optimizes for solving the task consistently and in the fewest environment interactions, thereby recovering our original formulation of autonomous learning. A central object utilized by our approach is the action-value function $Q^\pi(\mathbf{s}, \mathbf{a})$, which estimates the expected discounted return from taking action $\mathbf{a}$ in state $\mathbf{s}$ and following π thereafter. In our robotic setting, states include high-dimensional visual and proprioceptive observations, and actions are chunks (Zhao et al., 2023): $\mathbf{a}_t$ denotes a sequence of low-level controls executed open-loop. Details are in Appendix A.

Value-Based RL. Value-based methods estimate Q^π from experience: they maintain a replay buffer $\mathcal{B}$ of previously collected transitions and train a critic using temporal-difference (TD) learning. Given a transition $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1}) \sim \mathcal{B}$, the TD target and loss are

$$y_t = r_t + \gamma Q_{\bar{\psi}}(\mathbf{s}_{t+1}, \mathbf{a}'),$$
$$\mathcal{L}_{\mathrm{TD}}(\psi) = \mathbb{E}_{\mathcal{B}}\left[\left(Q_\psi(\mathbf{s}_t, \mathbf{a}_t) - y_t\right)^2\right], \tag{3.1}$$

where $Q_{\bar{\psi}}$ is a target critic and $\mathbf{a}'$ is the next action. The policy π_θ is then updated toward actions that receive high value under the critic Fujimoto et al. (2018); Haarnoja et al. (2018).

Residual RL. Residual RL adapts a frozen base policy by learning a lightweight module that modifies the base policy's proposed action (Dong et al., 2026; Xu et al., 2026; Ankile et al., 2024, 2025). Given a policy π_θ, the residual policy $\pi_\theta^{\mathrm{res}}$ conditions on both the state and an action proposed by the base policy, and outputs the action executed in the environment, i.e. $\mathbf{a}_{\mathrm{exec}} \sim \pi_\theta^{\mathrm{res}}(\cdot \mid \mathbf{s}, \mathbf{a}_{\mathrm{base}})$ with $\mathbf{a}_{\mathrm{base}} \sim \pi_\theta(\cdot \mid \mathbf{s})$. This residual update can be parameterized in several ways, such as predicting an additive offset in action space or directly predicting the modified action. In this work, we adopt the latter parameterization as discussed in Section 4.

Policy-Agnostic RL. Policy-agnostic RL (PA-RL) (Mark et al., 2024) is an online improvement method that applies across policy classes. It first improves sampled actions against the learned Q-function, then distills the optimized actions back into the parametric policy using the appropriate supervised objective, such as flow matching for flow policies or likelihood for discrete-action policies. We use PA-RL for this policy-agnostic structure, extending it to action-chunked residual policies (details in Appendix F.1). For each replay state $\mathbf{s}_t$, we sample candidate action chunks from the residual policy and refine them by taking gradient steps against Q_ψ,

$$\mathbf{a}_t \sim \pi_\theta^{\mathrm{res}}(\cdot \mid \mathbf{s}_t),$$
$$\mathbf{a}_t^\star \leftarrow \mathbf{a}_t + \eta \nabla_{\mathbf{a}} Q_\psi(\mathbf{s}_t, \mathbf{a}_t), \tag{3.2}$$

where η is the step size, and then distill the optimized chunk $\mathbf{a}_t^\star$ back into the residual policy.

Generalist Robot Policy. We take the base policy π_{base} to be a generalist robot policy, such as a pretrained vision-language-action model like $\pi_{0.5}$ (Intelligence et al., 2025). Concretely, π_{base} consists of a pretrained vision-language-model (VLM) backbone and a flow head that parameterizes a flow policy conditioned on the output of the VLM backbone.

4 MIDAS: A SIMPLE RECIPE FOR MINIMAL DATA ADAPTATION

In this section, we propose **Mi**nimal-**D**ata **A**daptation **S**trategy (MIDAS), a simple representative recipe for adapting a pretrained policy from minimal demonstrations and autonomous interaction. We use MIDAS to provide a common experimental framework for examining the roles of demonstrations, pretrained representations, and online corrective learning through controlled comparisons and targeted ablations. MIDAS builds prior work on residual and offline-to-online RL methods (Zhou et al., 2025b; Ball et al., 2023), but introduces subtle-yet-important modifications to how past methods utilize interaction data so as to account for *minimal* pretrained success. These serve to expose the differences between MDA and the traditional, large-BC-success offline-to-online regime. We note that concurrent work (Sun and Song, 2026) introduces alternative recipes that are similar in spirit and motivation.

The main motivation for MIDAS is that fine-tuning on a small number of K demonstrations seeds coarse task-following behavior but does not ensure reliable completion (shown in Section 6). This "coarse" behavior following provides a stronger prior than using pretraining solely for representation learning (Shah and Kumar, 2021; Parisi et al., 2022; Kim et al., 2024; Zhou et al., 2025a), but is a weaker prior in that pretraining does not attain non-trivially large task completion success, as is the common regime in robot learning and language modeling (Hu et al., 2024; Guo et al., 2025).

In our intermediate regime, we find that learning a lightweight residual policy through value-based RL often suffices to close the remaining gap. These findings directly lead to MIDAS: a simple two-stage recipe combining few-demo behavior cloning with online residual RL. We denote the resulting adapted policy by π_{RL}^K and provide the full algorithm in Appendix D. We note that the two-stage structure follows established approaches to initializing policies from offline data and improving them

through online interaction (Zhou et al., 2025b; Ball et al., 2023). Our focus here is on how to use *minimal offline* data to complete a given task.

Stage I: Behavior cloning on minimal demonstrations. Stage I adapts the pretrained policy π_{base} to the downstream task using the K demonstrations in $\mathcal{D}^K_{\text{demo}}$. We first fine-tune the policy on the available demonstrations to obtain a task-directed initialization for subsequent online RL. Implementation details are deferred to Appendix F.2. As described in Section 6, the resulting π^K_{base} coarsely attempts the task, but does not complete it reliably.

Stage II: Online improvement via value-based residual RL. Stage II improves π^K_{base} through autonomous interaction using value-based RL. We instantiate this adaptation module as a residual policy conditioned on the action proposed by π^K_{base}, following prior work (Dong et al., 2026; Xu et al., 2026; Ankile et al., 2024, 2025) with π^K_{base} kept frozen. At each state $\mathbf{s}_t$, the base policy proposes an action chunk $\mathbf{a}^{\text{base}}_t$, and the residual policy π^{res}_θ proposes an edit conditioned on the state and the chunk itself. We parameterize the residual policy as follows [1] (further details on design choices in Appendix C)

$$\pi^{\text{res}}_\theta(\cdot \mid \mathbf{s}_t, \mathbf{a}^{\text{base}}) = \tanh\big(\mathcal{N}\big(\mu_\theta(\mathbf{s}_t, \mathbf{a}^{\text{base}}),\ \sigma^2_\theta(\mathbf{s}_t, \mathbf{a}^{\text{base}})\big)\big)\,. \tag{4.1}$$

Overcoming the challenges introduced by minimal demonstrations. The two-stage structure alone does not ensure reliable learning in MDA. With only a few demonstrations, the residual actor and critic have little data for initialization, and autonomous rollouts from π^K_{base} may succeed only 2–5% of the time. Moreover, the initial policy may not reliably produce the corrective actions needed for task completion (Section 6). Our residual PA-RL implementation addresses these challenges through the initialization, replay sampling, and policy-update choices described below.

1) Offline warmup: residual actor initialization and critic calibration from few demonstrations. A randomly initialized residual actor would produce actions unrelated to the base policy, making early exploration unstable. We therefore construct a "warmup" buffer $\mathcal{B}_{\text{warm}}$ from the K demonstrations in $\mathcal{D}^K_{\text{demo}}$ together with a small number of autonomous rollouts from π^K_{base} that may or may not be successful. For each state in this buffer, we query the frozen base policy to obtain $\mathbf{a}^{\text{base}} \sim \pi^K_{\text{base}}(\cdot \mid \mathbf{s})$, and train the residual actor to reproduce this base proposal:

$$\mathcal{L}_{\text{warm}}(\theta) = \mathbb{E}_{(\mathbf{s},\mathbf{a}^{\text{base}})\sim\mathcal{B}_{\text{warm}},\ \mathbf{a}\sim\pi^{\text{res}}_\theta(\cdot|\mathbf{s},\mathbf{a}^{\text{base}})}\left[\left\|\mathbf{a} - \mathbf{a}^{\text{base}}\right\|_2^2\right]. \tag{4.2}$$

This initialization makes the residual actor approximately preserve the behavior of π^K_{base}: Online RL also requires a critic that can guide improvement from the start. While prior work can pretrain such critics from offline data (Kumar et al., 2023; Nakamoto et al., 2024; Zhou et al., 2025b; Kostrikov et al., 2021), our setting has no additional offline data beyond the few demonstrations and autonomous rollouts from π^K_{base}. We therefore draw inspiration from Zhou et al. (2025b) and train the critic on the warmup buffer $\mathcal{B}_{\text{warm}}$ using TD learning (Eq. (3.1)). Importantly, this critic is trained on the state-action distribution induced by π^K_{base}, making it calibrated to the regions where online improvement begins and limiting forgetting.

2) Success Balancing: overcoming limited initial policy performance. Since the base policy may only succeed 2-5% in some cases after Stage I, we employ *success balancing* to amplify the sparse reward signal during both the warmup and online improvement phases. To do so, we maintain a replay buffer $\mathcal{B}$ containing all collected trajectories and a success buffer $\mathcal{B}_{\text{succ}}$ containing only successful trajectories, both initialized with the given K demonstrations. Critic minibatches are sampled from a mixture of these buffers, oversampling $\mathcal{B}_{\text{succ}}$ to prevent the value function from being dominated by failed rollouts and to guide policy improvement toward reward-bearing regions. Details are in Appendix F.3.

3) Residual policy training with PA-RL (Mark et al., 2024). Lastly, Section 6 reveals the need for actions that go beyond the support of the base policy. While many value-based RL algorithms can provide this, we opt for "policy-agnostic" PA-RL approach, which we apply to the residual actor. Having a policy-agnostic parameterization allows us to easily extend our method to more sophisticated residual policies like a minimal-iterative-policy (Pan et al., 2026), while also being competitive with other common policy extraction approaches Haarnoja et al. (2018); Fujimoto et al. (2018). We implement this by sampling *a single* proposal action $\mathbf{a}^{\text{base}}_t$ from the base policy, and then applying PA-RL style Best-of-N distillation objective with an action gradient to a batch of N

[1] Unlike typical residual formulations, (4.1) directly predicts the edited action $\mathbf{a}^{\text{exec}}_t \sim \pi^{\text{res}}_\theta(\cdot|\mathbf{s}_t, \mathbf{a}^{\text{base}}_t)$.

LIBERO-Long

Task	Zero-Shot	1 Demo				
		BC	DSRL	Filt. BC	DiceRL	MIDAS
Alph. Soup + Cr. Cheese → Basket	0.0	12.7±1.2	12.0±2.0	14.0±2.0	98.0±2.0	**99.3**±1.2
Alph. Soup + Tom. Sauce → Basket	0.0	4.0±3.5	3.3±2.3	2.0±0.0	**92.0**±4.0	82.7±13.3
Black Bowl → Bottom Drawer	0.0	22.7±6.4	28.7±4.2	28.7±10.3	90.7±3.1	**98.7**±1.2
Book → Caddy	0.0	37.3±7.0	42.7±6.1	28.0±2.0	98.0±3.5	**98.7**±2.3
Both Moka Pots → Stove	0.0	13.3±3.1	14.7±4.6	19.3±7.0	74.7±9.5	**76.0**±14.0
Cr. Cheese + Butter → Basket	0.0	34.0±9.2	28.0±2.0	68.0±2.0	88.7±2.3	**96.0**±0.0
Moka Pot → Stove	0.0	84.0±0.0	92.0±0.0	85.3±8.3	**98.7**±1.2	95.3±3.1
White Mug + Choc. Pudding	0.0	33.3±11.7	45.3±6.4	42.0±4.0	**97.3**±4.6	96.7±5.8
White Mug + Plates	0.0	30.0±2.0	17.3±1.2	34.7±6.4	61.3±11.5	**79.3**±11.0
Yellow-White Mug → Microwave	0.0	64.0±2.0	50.7±4.6	68.0±7.2	**93.3**±3.1	89.3±8.1
Average	0.0	33.5±1.8	33.5±0.8	39.0±2.4	89.3±1.8	**91.2**±4.2

RoboCasa

Task	Zero-Shot	1 Demo			
		BC	DSRL	Filt. BC	MIDAS
Banana: Fridge Drawer → Shelf	0.0	13.3±3.1	12.7±2.3	19.3±3.1	**93.3**±7.0
Hot Dog: Counter → Cabinet	0.0	29.3±1.2	20.0±10.4	36.0±29.5	**100.0**±0.0
Mug → Coffee Machine + Start	0.0	24.0±8.7	24.7±4.2	46.7±7.6	**74.7**±20.0
Cup + Bowl → Dishwasher + Close†	0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
Average	0.0	22.2±3.4	19.1±3.2	34.0±10.6	**89.3**±8.7

Table 1: **One demonstration adaptation results measured over 3 seeds.** Success rates (%) are mean ± std. †Excluded from the RoboCasa average.

actions $\mathbf{a}_t^{(i)} \sim \pi_\theta^{\text{res}}(\cdot \mid \mathbf{s}_t, \mathbf{a}_t^{\text{base}})$ sampled from the residual (See Appendix F.1 for more details). Importantly, sampling requires *only* a single expensive forward pass through π_{base}^K while sampling the remaining actions from the lightweight residual policy for diversity.

5 MDA IS POSSIBLE FROM AS FEW AS ONE DEMONSTRATION

In this section, we evaluate representative adaptation strategies in the MDA regime and show that successful adaptation is possible from as little as one demonstration.

Experimental setup. We use $\pi_{0.5}$ (Intelligence et al., 2025) as the pretrained base policy π_{base} and run MIDAS with $K{=}1$ demonstration only on two simulation benchmarks: **a)** LIBERO-Long (Liu et al., 2023), which studies long-horizon language-conditioned manipulation, and **b)** RoboCasa-365 (Nasiriany et al., 2026), which studies long-horizon household manipulation in diverse kitchen scenes (Details in Appendix E).

Experimental Results. Table 1 compares representative strategies corresponding to the three ingredients of adaptation in MDA: pretraining alone (*zero-shot*), adaptation with a single demonstration (*BC*), and subsequent online improvement. For online improvement, we evaluate learning from successful rollouts (*Filtered BC*), value-guided noise-space steering (*DSRL* (Wagenmaker et al., 2025)), and value-guided action-space adaptation with MIDAS.

On LIBERO, we additionally compare against *DICE-RL* (Sun and Song, 2026), which is a concurrent method that adapts frozen generative policies with value-based residual corrections and selective behaviour regularization. DICE-RL shares key design principles underlying MIDAS (see Appendix L). This allows us to test whether successful adaptation extends to alternative implementations of our shared principles. We see that MIDAS and DICE-RL recover strong performance from one-demonstration initializations on LIBERO. Filtered BC and DSRL provide smaller and less consistent improvements, particularly when initial success is below 20%. On RoboCasa, MIDAS also substantially improves performance on three of four evaluated tasks; the dishwasher task remains unsolved by all evaluated methods. Implementation details in Appendix E.

Design choices affect online improvement. We examine the effects of replay sampling and continued behavior-cloning regularization in Appendix G. We find that removing success balancing lowers performance. Early in training, most autonomous rollouts fail, so uniform replay sampling provides relatively few examples of successful execution. Success balancing keeps these rare suc-

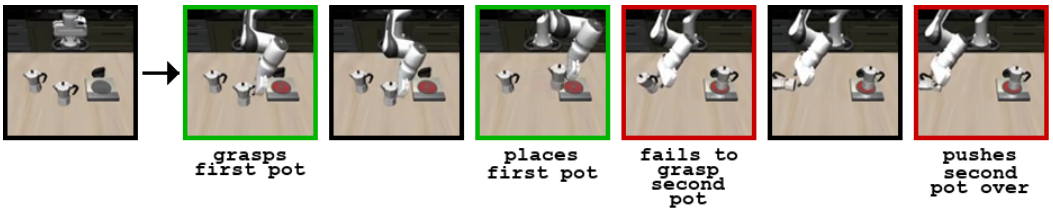


Figure 2: **Few-demo BC (π^K_{base}) recovers task-level behavior, but not fine-grained control.** The policy reaches the right objects but fails to grasp the second pot under orientation changes of the reset distribution.

cessful transitions present in actor and critic updates. However, explicitly encouraging the actor to imitate successful trajectories throughout online learning also reduces performance, suggesting that continued imitation can restrict useful departures from previously observed actions. These ablations highlight the importance of different design choices in successful adaptation with MIDAS.

Real world experiments. We finally evaluate whether minimal-data adaptation is tractable in the real world using a bimanual YAM platform. We consider two pick-and-place tasks: **(T1)** placing a green block in the right container and a blue block in the left container, and **(T2)** placing a knife and a donut on a plate (Figure 13). In both tasks, we use $\pi_{0.5}$ (Intelligence et al., 2025) as π_{base} and apply the same MIDAS protocol with a single demonstration ($K{=}1$); implementation details are deferred to Appendix M. On **T1**, the one demonstration policy π^1_{base} already reaches 40% success, which we hypothesize reflects overlap with the pretraining distribution of $\pi_{0.5}$. Online RL further improves success to 67% and produces qualitatively new corrective behaviors not observed in base-policy rollouts (Appendix M). On the harder **T2** task, where the warm-start policy is substantially weaker, online adaptation improves success from 27% to 80% after 5-6 hours of autonomous interaction.

6 What Enables Minimal-Data Adaptation?

Previously, we motivated minimal-data adaptation with the premise that a few demonstrations can recover enough task-directed behavior to make online improvement tractable. In this section, we present detailed empirical evidence examining how pretraining, demonstrations, and online interaction enable successful adaptation in this regime. First, the few-demonstration BC finetuning of the pretrained VLA provides actions which follow task instructions coarsely, reducing the burden of exploration. Second, the pretrained VLA provides visual representations, enabling efficient learning without direct state access. Finally, online reinforcement fills in the gaps in precise-control, which necessarily requires value-guided online updates to finetuning actions beyond what π^K_{base} can reliably produce on its own.



Figure 3: **Performance of different state representations for BC.**

1) Pretraining and demonstrations enable learning of coarse task-specific behaviors: Before online adaptation, we ask whether $K{=}1$ can recover coarse task behavior. Figure 3 compares π^1_{base} to three approaches: **a)** a flow-matching policy trained on pretrained ResNet features, which tests whether generic visual pretraining is sufficient; **b)** a flow-matching policy trained on privileged state inputs, which tests whether near-optimal state information alone enables recovery of task relevant behavior; and **c)** the zero-shot pretrained policy π_{base}, which tests whether pretraining alone identifies the task. All three perform poorly: the ResNet flow-policy achieves non-zero success on only 3/10 LIBERO-Long tasks, while the privileged-state policy and π_{base} obtain 0% success. Together, these failures suggest that MDA requires both task grounding from the demonstration and behavior-aware representations from pretraining; privileged state inputs alone do not recover this behavior, suggesting that the benefit of pretraining goes beyond improved state representation.

Qualitative rollouts in Fig. 2 show what π^1_{base} recovers despite its low success. The policy often approaches the correct object, attempts the correct subtask sequence, and follows a trajectory resembling the demonstration. Its failures are primarily in execution: contact, alignment, grasping, and recovery remain brittle, necessitating the need for online adaptation to learn reliable control.

2) Pretrained representations simplify online adaptation and substantially improve sample efficiency: The preceding results show that large-scale pre–training, together with minimal demonstrations, anchors the search for task-relevant behavior. We now find that the representations pro-

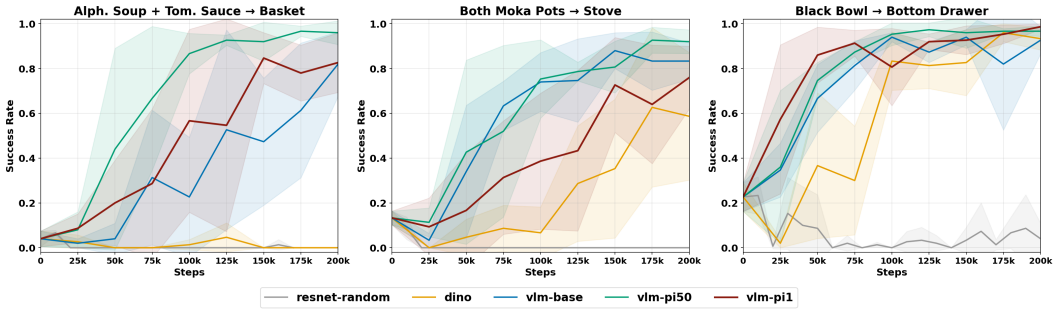
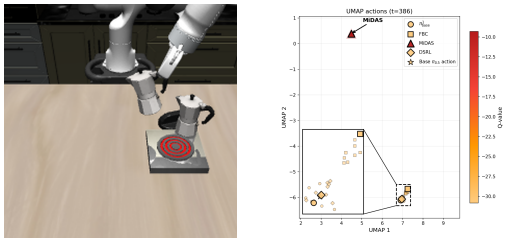


Figure 4: **Performance of different representations for online adaptation.**

vided to the residual policy also determine how efficiently online RL can exploit this initialization. Figure 4 isolates this effect by varying the representation used by the residual learner. We compare **(i)** a **ResNet** trained from scratch during online RL, representing no visual pretraining; **(ii)** frozen **DINO** features (Caron et al., 2021), representing generic large-scale self-supervised visual pretraining; and **(iii)** representations from the pretrained VLA backbone. Within the VLA representations, we further compare π_{base}, π^1_{base}, and π^{50}_{base} to assess the effect of increasing amounts of demo data.

Without visual pretraining, MIDAS fails to converge at $K = 1$, indicating that sparse interaction is insufficient to simultaneously learn task-relevant visual features and residual action corrections. Generic visual pretraining with frozen DINO substantially improves learning, but still requires considerably more interaction than VLA features, suggesting that generic visual invariances alone are insufficient for sample-efficient adaptation. In contrast, representations from the pretrained $\pi_{0.5}$ backbone enable rapid improvement throughout training. The similar performance of π_{base}, π^1_{base} and π^{50}_{base} on two of the three tasks indicates that task-specific fine-tuning provides little additional improvement to the representations used for online adaptation. Although we study a single VLA, these results suggest that representations learned through action-aligned pretraining can substantially improve the sample efficiency of downstream online adaptation.

3) Value-based RL learns fine-grained control actions beyond the base policy's prior: The previous analysis showed that π^K_{base} often produces task-relevant behavior, but solves the task inconsistently. We ask: do the right actions already lie within its support but are not reliably selected, or do they fall outside its effective support? We test this by comparing MIDAS to the baselines introduced in Section 5.



(a) Trajectory critical state (b) UMAP of actions

Figure 5: **MIDAS reaches actions outside the effective support of π^K_{base} with higher Q^* and succeeds..**

We examine a critical state in a LIBERO-Long rollout in Fig. 5 to make the distinction between MIDAS and the baselines. At this critical state, actions from Filtered BC and DSRL remain concentrated near samples from π^1_{base} and fail, while MIDAS selects an action in a disjoint region of the action space and succeeds. We further validate this finding in the robot's action space (further details in Appendix K).

7 Generalization and Robustness in Minimal Data Adaptation

The preceding analysis examined how pretraining, demonstrations, and online interaction enable minimal-data adaptation. We now study how these capabilities extend beyond the conditions encountered during adaptation: what robustness is inherited from pretraining, what generalization online learning enables, and which shifts remain difficult for the adapted policy.

We consider two categories of shifts: **(a) observation shifts**, which alter object appearance (color, texture) or language phrasing while preserving the task and its physical requirements, and **(b) state shifts**, which change dynamically relevant quantities such as starting positions, object shapes, or object instances. We ask: **(1)** does adaptation preserve robustness to visual and language changes? **(2)** Under which state shifts does online improvement transfer? and **(3)** can broader online experience improve robustness to larger state shifts without additional demonstrations?

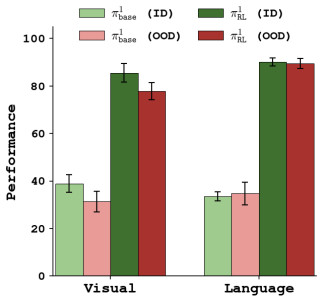


Figure 6: **Performance of BC and MIDAS under observation shifts**

We evaluate on 10 long-horizon tasks in LIBERO-Long using the perturbation definitions from LIBERO-PRO (Zhou et al., 2026) (Further details in Appendices E.2 and E.3).

(Q1) Observation-level robustness is inherited from the frozen vision-language-model (VLM) backbone. We first test whether minimal-data adaptation preserves robustness to semantic input changes, rather than merely fitting the visual and language conditions seen in the demonstrations. Under visual shifts such as color and texture changes (Fig. 6), π^{K}_{base} (K=1) retains most of its in-distribution performance, and the adapted policy π^{1}_{RL} shows a similar retention pattern despite using only a lightweight residual actor. The same holds under language paraphrases (Fig. 6). Since both policies share the same frozen VLM backbone, and the RL module adds little perceptual or linguistic capacity, this robustness is best explained by invariances already present in the pretrained representation. Thus, online RL improves control while retaining observation-level robustness.

(Q2) Generalization to state shifts depends on whether new behavior is required. We next evaluate shifts that require the policy to change its actions. Figure 7, shows that π^{1}_{RL} improves over π^{1}_{base} under some state shifts, especially shape changes where the required grasp is similar. This result can simply be explained by the fact that the policy just needs to execute the same behavior in a different part of the state space. However, performance under both drops sharply under object swaps and object changes: changes in object affordance or placement can require distinct manipulation strategies, which are difficult to infer from one demonstration and online interaction around it. Tables 5, 6, and 7 in Appendix E.3 summarize the results for object swap, shape change, and object change perturbations, respectively. Further implementation details are provided there.

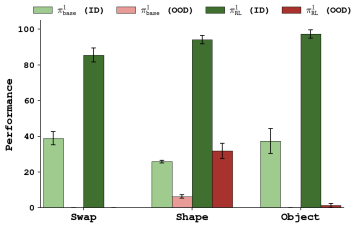


Figure 7: **Performance of BC and MIDAS under state shifts**

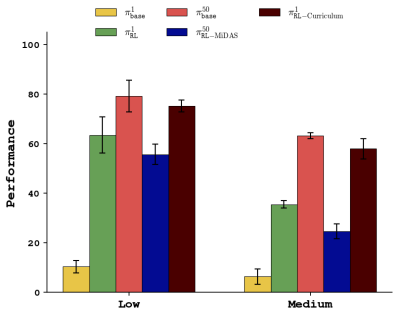


Figure 8: **Progressively widening the reset distribution during online RL recovers performance** .

(Q3) Broader coverage, either from demonstrations or curriculum, expands the generalization boundary. Lastly, we ask: can autonomous adaptation ensure robustness to large perturbation shifts that are outside the support of the reset distribution, provided that they do not require fundamentally new behavior (in view of Q2)? In Appendix I, we show that naively increasing the breadth of demonstration can improve robustness. In Appendix H, we describe a curriculum that gradually increases the width of the distribution, and achieves similar robustness to a policy trained on 50 demonstrations. Fig. 8 summarizes our results and shows that by gradually widening the reset distribution with $\pi^{1}_{\text{RL-Curriculum}}$, we are able to recover most of the performance of π^{50}_{base} under large perturbation shifts.

8 Limitations

The results in this paper should be interpreted within two limitations. First, the present instantiation remains limited on longer-horizon tasks, where sparse-reward value learning must assign credit through long sequences of actions proposed by a frozen prior. Scaling MDA may therefore require stronger value priors, such as pretrained critics or value decompositions that expose intermediate progress signals. Second, Stage II remains reset-dependent and interaction-intensive. This is manageable in simulation, but becomes a central constraint on hardware, where failures, resets, and safety limit autonomous practice without an operator in the loop. Reducing this dependence will require safer exploration, better failure detection, learned resets, or lightweight intervention mechanisms that preserve the low-demonstration setting. We discuss directions for future work in Appendix N.

AI USAGE DISCLOSURE

We used large language models to assist with writing and polishing the manuscript and to support literature retrieval and discovery, including identifying relevant related work. We also used LLMs as coding assistants. The authors take full responsibility for the final content of the paper.

REFERENCES

Lars Ankile, Anthony Simeonov, Idan Shenfeld, Marcel Torne, and Pulkit Agrawal. From imitation to refinement – residual rl for precise assembly, 2024. URL https://arxiv.org/abs/2407.16677.

Lars Ankile, Zhenyu Jiang, Rocky Duan, Guanya Shi, Pieter Abbeel, and Anusha Nagabandi. Residual off-policy rl for finetuning behavior cloning policies, 2025. URL https://arxiv.org/abs/2509.19301.

Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data, 2023. URL https://arxiv.org/abs/2302.02948.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2026. URL https://arxiv.org/abs/2410.24164.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023a. URL https://arxiv.org/abs/2307.15818.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-1: Robotics transformer for real-world control at scale, 2023b. URL https://arxiv.org/abs/2212.06817.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers, 2021. URL https://arxiv.org/abs/2104.14294.

Pim De Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423/.

Perry Dong, Qiyang Li, Dorsa Sadigh, and Chelsea Finn. Expo: Stable reinforcement learning with expressive policies, 2026. URL https://arxiv.org/abs/2507.07986.

Yan Duan, Marcin Andrychowicz, Bradly C. Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning, 2017. URL https://arxiv.org/abs/1703.07326.

Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning, 2017. URL https://arxiv.org/abs/1709.04905.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning, 2021. URL https://arxiv.org/abs/2106.06860.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods, 2018. URL https://arxiv.org/abs/1802.09477.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645 (8081):633–638, September 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018. URL https://arxiv.org/abs/1801.01290.

Siddhant Haldar, Jyothish Pari, Anant Rai, and Lerrel Pinto. Teach a robot to fish: Versatile imitation from one minute of demonstrations, 2023. URL https://arxiv.org/abs/2303.01497.

Hengyuan Hu, Suvir Mirchandani, and Dorsa Sadigh. Imitation bootstrapped reinforcement learning, 2024. URL https://arxiv.org/abs/2311.02198.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025. URL https://arxiv.org/abs/2504.16054.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model, 2024. URL https://arxiv.org/abs/2406.09246.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success, 2025. URL https://arxiv.org/abs/2502.19645.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning, 2021. URL https://arxiv.org/abs/2110.06169.

Aviral Kumar, Anikait Singh, Frederik Ebert, Mitsuhiko Nakamoto, Yanlai Yang, Chelsea Finn, and Sergey Levine. Pre-training for robots: Offline rl enables learning new tasks from a handful of trials, 2023. URL https://arxiv.org/abs/2210.05178.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning, 2023. URL https://arxiv.org/abs/2306.03310.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning, 2022. URL https://arxiv.org/abs/2205.05638.

Jianlan Luo, Zheyuan Hu, Charles Xu, You Liang Tan, Jacob Berg, Archit Sharma, Stefan Schaal, Chelsea Finn, Abhishek Gupta, and Sergey Levine. Serl: A software suite for sample-efficient robotic reinforcement learning, 2025. URL https://arxiv.org/abs/2401.16013.

Max Sobol Mark, Tian Gao, Georgia Gabriela Sampaio, Mohan Kumar Srirama, Archit Sharma, Chelsea Finn, and Aviral Kumar. Policy agnostic rl: Offline rl and online rl fine-tuning of any class and backbone, 2024. URL https://arxiv.org/abs/2412.06685.

Mitsuhiko Nakamoto, Yuexiang Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning, 2024. URL https://arxiv.org/abs/2303.05479.

Soroush Nasiriany, Sepehr Nasiriany, Abhiram Maddukuri, and Yuke Zhu. Robocasa365: A large-scale simulation framework for training and benchmarking generalist robots, 2026. URL https://arxiv.org/abs/2603.04356.

Norman Di Palo and Edward Johns. Keypoint action tokens enable in-context imitation learning in robotics, 2024. URL https://arxiv.org/abs/2403.19578.

Chaoyi Pan, Giri Anantharaman, Nai-Chieh Huang, Claire Jin, Daniel Pfrommer, Chenyang Yuan, Frank Permenter, Guannan Qu, Nicholas Boffi, Guanya Shi, and Max Simchowitz. Much ado about noising: Dispelling the myths of generative robotic control, 2026. URL https://arxiv.org/abs/2512.01809.

Simone Parisi, Aravind Rajeswaran, Senthil Purushwalkam, and Abhinav Gupta. The unsurprising effectiveness of pre-trained vision models for control, 2022. URL https://arxiv.org/abs/2203.03580.

Sarvesh Patil, Mitsuhiko Nakamoto, Manan Agarwal, Shashwat Saxena, Jesse Zhang, Giri Anantharaman, Cleah Winston, Chaoyi Pan, Douglas Chen, Nai-Chieh Huang, Zeynep Temel, Oliver Kroemer, Sergey Levine, Abhishek Gupta, Hongkai Dai, Paarth Shah, and Max Simchowitz. Ogpo: Sample efficient full-finetuning of generative control policies, 2026. URL https://arxiv.org/abs/2605.03065.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL https://arxiv.org/abs/2103.00020.

Allen Z. Ren, Justin Lidard, Lars L. Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization, 2024. URL https://arxiv.org/abs/2409.00588.

Rutav Shah and Vikash Kumar. Rrl: Resnet as representation for reinforcement learning, 2021. URL https://arxiv.org/abs/2107.03380.

Zhanyi Sun and Shuran Song. From prior to pro: Efficient skill mastery via distribution contractive rl finetuning, 2026. URL https://arxiv.org/abs/2603.10263.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy, 2024. URL https://arxiv.org/abs/2405.12213.

Vitalis Vosylius and Edward Johns. Instant policy: In-context imitation learning via graph diffusion, 2025. URL https://arxiv.org/abs/2411.12633.

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, 2025. URL https://arxiv.org/abs/2506.15799.

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, 2026. URL https://arxiv.org/abs/2604.23073.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World action models are zero-shot policies, 2026. URL https://arxiv.org/abs/2602.15922.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023. URL https://arxiv.org/abs/2304.13705.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning, 2025a. URL https://arxiv.org/abs/2411.04983.

Xueyang Zhou, Yangming Xu, Guiyao Tie, Yongchao Chen, Guowen Zhang, Duanfeng Chu, Pan Zhou, and Lichao Sun. Libero-pro: Towards robust and fair evaluation of vision-language-action models beyond memorization, 2026. URL https://arxiv.org/abs/2510.03827.

Zhiyuan Zhou, Andy Peng, Qiyang Li, Sergey Levine, and Aviral Kumar. Efficient online reinforcement learning fine-tuning need not retain offline data, 2025b. URL https://arxiv.org/abs/2412.07762.

Appendices

Additional videos of rollouts and training are available here: coruscating-pika-920eda.netlify.app.

A Action Chunking

We apply action chunking (Zhao et al., 2023), where sequences of actions $u_{t:t+C-1}$ are predicted and executed in open-loop. Let $u_t \in \mathbb{R}^d$ denote a primitive robot action and let C denote the chunk horizon. We define the action space used by the policy as $\mathcal{A} \subseteq \mathbb{R}^{C \times d}$, so that each action $a_t \in \mathcal{A}$ corresponds to an entire action chunk, $a_t = (u_t, \ldots, u_{t+C-1})$. After sampling $a_t \sim \pi(\cdot \mid s_t)$, the robot executes the chunk open-loop for C environment steps before replanning from the resulting state s_{t+C}. For notational simplicity, we treat each chunk as a single action in the MDP, preserving the standard MDP notation above.

When using chunked actions, the reward associated with a transition is the discounted reward accumulated over the executed chunk, and bootstrapping occurs from s_{t+C} with discount γ^C. Thus, unless stated otherwise, a_t denotes an action chunk and r_t denotes the reward accumulated over the executed chunk, with bootstrapping discount γ^C. For our experiments, we use a reward of -1 for a chunk with effective discount of γ^C.

B Related Work

Generalist robot policies and VLAs. Large-scale robot pretraining has enabled generalist policies that can condition on language and visual observations to perform diverse manipulation tasks. Prior work has introduced transformer-based robot policies and vision-language-action models trained on broad robot and web-scale data, including RT-1 (Brohan et al., 2023b), RT-2 (Brohan et al., 2023a), Octo (Team et al., 2024), π_0 (Black et al., 2026), $\pi_{0.5}$ (Intelligence et al., 2025), world-action models (Ye et al., 2026), and efficient VLA fine-tuning methods (Kim et al., 2025). These models provide reusable perceptual representations and broad action priors, making them natural candidates for fast downstream adaptation. However, despite this progress, current generalist policies are still not reliable zero-shot learners in new environments: they may fail to reach sparse rewards or visit too narrow a set of task-relevant states for autonomous learning to bootstrap. In contrast, we study whether such pretrained policies can be made adaptable from minimal task-specific supervision.

Few-shot robot imitation and adaptation. A complementary line of work studies learning from very small numbers of demonstrations. One-shot and few-shot imitation learning methods (Duan et al., 2017; Finn et al., 2017) show that robots can infer new tasks from one or a few examples, often through meta-learning or in-context task conditioning (Palo and Johns, 2024). Other approaches study rapid adaptation from short demonstrations (Haldar et al., 2023) or in-context imitation policies (Vosylius and Johns, 2025). While appealing in theory, few shot and in-context imitations suffer from two structural limitations. First, context-conditioned policies are known to suffer form causal confusion or are prone to shortcut features (De Haan et al., 2019), and evidence from large-language-model literature shows that task-specific fine-tuning often outperforms in-context learning (Liu et al., 2022). Second, and more fundamentally, these methods do not have any mechanism for autonomously improving from their own experience. Motivated by these two observations, we study the MDA regime. With very few demonstrations for specifying the task, the MDA regime uses them as a scaffold to make autonomous interaction tractable and analyzes the components enabling this tractability.

Online and residual RL for robot policy adaptation. Online reinforcement learning and offline-to-online methods provide tools for improving robot policies after deployment. Prior work has studied offline RL pretraining (Kumar et al., 2023), calibrated offline-to-online fine-tuning (Nakamoto et al., 2024), efficient online RL with offline data (Ball et al., 2023), imitation-bootstrapped RL (Hu et al., 2024), and policy-agnostic RL for fine-tuning diverse policy classes (Mark et al., 2024). More closely related, recent methods use RL to improve expressive policies or pretrained robot policies, including diffusion-policy steering (Wagenmaker et al., 2025), diffusion policy optimization (Ren et al., 2024), RL-token adaptation (Xu et al., 2026), distribution-contractive fine-tuning (Sun and Song, 2026), and residual RL for refining imitation policies (Dong et al., 2026; Ankile et al., 2024,

2025). These methods provide foundations for improving pretrained robot policies through online interaction.

Building on this body of existing work, we study *minimal-data adaptation*: whether one or a few demonstrations can make autonomous improvement tractable when the pretrained policy is initially unreliable. Our emphasis is on isolating how pretraining, demonstrations, and online experience jointly enable adaptation, which algorithmic choices make learning effective in this regime, and what forms of generalization emerge or remain limited. From this study, we propose MIDAS, a representative practical recipe built on prior offline-to-online and residual RL approaches. Concurrent work, DICE-RL (Sun and Song, 2026), improves frozen generative policies frozen by learning value-based residual corrections, and combining demonstration data with online experience. DICE-RL shares key design principles underlying MIDAS and enables successful adaptation in the MDA regime. This provides further evidence that these shared principles enable effective learning from minimal task-specific supervision. Our contribution thus lies in establishing and analyzing the MDA regime and identifying the core algorithmic components required to learn within it.

C Residual Policy Parameterization

We parameterize the residual actor by conditioning it on $\mathbf{a}_t^{\text{base}}$. This exposes it directly to the action mode selected by π_{base}^K, so online RL refines a task-relevant VLA proposal rather than learning to represent a complex, multimodal action distribution from scratch. At the same time, predicting $\mathbf{a}_t^{\text{exec}}$ directly allows the policy to shift this proposal toward higher-value regions instead of being restricted to small fixed residual corrections around $\mathbf{a}_t^{\text{base}}$ that further limit the policy and are more common in the literature(Dong et al., 2026). Concretely, an additive parameterization would predict a bounded, additive correction $\mathbf{a}_t^{\text{exec}} = \mathbf{a}_t^{\text{base}} + \Delta_\theta(\mathbf{s}_t, \mathbf{a}_t^{\text{base}})$, constraining the executed action to a fixed neighborhood of the base proposal. Because minimal demonstrations do not guarantee that near-optimal actions lie close to $\mathbf{a}_t^{\text{base}}$ (Section 6), such a bounded residual can prevent the policy from reaching higher-value actions. Predicting $\mathbf{a}_t^{\text{exec}}$ directly instead lets the base proposal anchor exploration while leaving the policy free to move beyond its immediate neighborhood. Crucially, the amount of movement around the proposed action can now be heterogeneous across states, whereas the additive approach would bound the movement by a fixed constant at all states. We then evaluate this action by a critic ensemble $\{Q_{\psi_j}(\mathbf{s}_t, \mathbf{a}_t^{\text{exec}})\}_{j=1}^N$ trained with temporal difference learning in Eq. (3.1). We discuss implementation details for this residual parameterization in Appendix F.1.

For the state representation s_t, we use the output of the VLM head by taking a mean across all token positions. The residual actor is parameterized as a 3-layer MLP with a hidden dimension of 512 and ReLU activation.

D Algorithm

Algorithm 1 summarizes MIDAS using the residual policy in Eq. (4.1) and action chunks of length $H = C$ as in Appendix A. Hyperparameters used below are listed at the top of the algorithm; additional implementation settings are in Table 11.

Replay and sampling. Store $(\mathbf{s}_t, \mathbf{a}_t^{\text{base}}, \mathbf{a}_t^{\text{exec}}, r_t, m_t, \mathbf{s}_{t+H})$, where $\mathbf{s}_t = (\phi(o_t), q_t)$ combines mean-pooled VLM prefix features $\phi(o_t)$ of observation o_t and proprioception q_t. Set $(r_t, m_t) = (0, 0)$ for successful terminal chunks and $(-1, 1)$ otherwise; demonstrations use their recorded actions. The operator clip clips each action to the action bounds.

Update conventions. TD updates use Eq. (3.1) with ensemble-mean target values, next actions sampled from $\pi_\theta^{\text{res}}(\cdot \mid \mathbf{s}', \mathbf{a}'^{\text{base}})$, where $\mathbf{a}'^{\text{base}}$ is the frozen base policy's proposal at the next state. Use discount $\gamma^H m_t$, clip the sampled next action, and update targets after every critic update via $\bar{\psi} \leftarrow \tau\psi + (1-\tau)\bar{\psi}$. All TD and BC minibatches have size B; optimizer settings are in Appendix F.1.

E Simulation Experiment Details

We use the Libero and RoboCasa Benchmarks for our simulation experiments

Algorithm 1 MIDAS: Minimal-Data Adaptation Strategy

Require: Pretrained policy π_{base}, K successful demonstrations $\mathcal{D}^K_{\text{demo}}$, and target environment (Section 2)
Ensure: Adapted policy π^K_{RL}
Hyperparameters:
Replay and critics: chunk length H, batch size B, number of critics J, success fraction ρ_{succ}, discount γ, Polyak rate τ.
Schedule: demo warmup rounds per transition U_{demo}, warmup threshold W rounds, minimum replay size $n_{\min}$, update budget $T_{\max}$ rounds; critic/actor updates per round: warmup $(U^{\text{w}}_C, U^{\text{w}}_A)$, RL (U_C, U_A).
PA-RL: candidates N, elites M, ascent steps G, ascent step size η, actor learning rate α_π.

Stage I: Behavior cloning on K demonstrations
Fine-tune π_{base} on $\mathcal{D}^K_{\text{demo}}$ with flow matching $\to \pi^K_{\text{base}}$ (Appendix F.2)
Stage II: Residual reinforcement learning
Freeze π^K_{base}; initialize actor π^{res}_θ (Eq. (4.1)), critics $\{Q_{\psi_j}\}_{j=1}^J$ (Appendix F.1), and targets $\bar{\psi} \leftarrow \psi$
Define the mean critic $\bar{Q}(\mathbf{s}, \mathbf{a}) := J^{-1} \sum_{j=1}^J Q_{\psi_j}(\mathbf{s}, \mathbf{a})$
Use the state, transition, and clipping definitions on p. 16 and the TD/BC rules on p. 16
Initialize replay $\mathcal{B}$ and success replay $\mathcal{B}_{\text{succ}}$ with all stride-one demo transitions and base proposals sampled from π^K_{base}
Run $U_{\text{demo}}|\mathcal{B}|$ warmup rounds: U^{w}_C TD (Eq. (3.1)) and U^{w}_A BC (Eq. (4.2)) updates, sampling uniformly from $\mathcal{B}$
Set $i \leftarrow U_{\text{demo}}|\mathcal{B}|$, the number of completed update rounds
while $i \le T_{\max}$ **do**
  Collect an episode in H-step chunks (Appendix A): at each chunk, sample $\mathbf{a}_t^{\text{base}} \sim \pi^K_{\text{base}}(\cdot \mid o_t)$ with fresh flow-matching noise
    Execute the clipped base proposal if $i < W$; otherwise the clipped deterministic action $\tanh\big(\mu_\theta(\mathbf{s}_t, \mathbf{a}_t^{\text{base}})\big)$
  Add transitions to $\mathcal{B}$; if the episode succeeded, also add them to $\mathcal{B}_{\text{succ}}$
  if $|\mathcal{B}| \le n_{\min}$: **continue**
  Define $\mathcal{B}_{\text{mix}}$ as the mixture of the current $\mathcal{B}$ and $\mathcal{B}_{\text{succ}}$ in Eq. (F.1)
  for each chunk collected in the episode **do**
    $(u_C, u_A) \leftarrow (U^{\text{w}}_C, U^{\text{w}}_A)$ if $i < W$, else (U_C, U_A)
    Perform u_C TD updates (Eq. (3.1)) with minibatches from $\mathcal{B}_{\text{mix}}$
    if $i < W$: perform u_A BC updates (Eq. (4.2)) on uniform minibatches from $\mathcal{B}$
    else: perform u_A PA-RL updates as follows
      Sample minibatch $\mathcal{D} \sim \mathcal{B}_{\text{mix}}$, $|\mathcal{D}| = B$; for each $(\mathbf{s}, \mathbf{a}^{\text{base}}) \in \mathcal{D}$:
        Sample $\{\mathbf{a}^{(n)}\}_{n=1}^N$ using Eq. (4.1), conditioned on $(\mathbf{s}, \mathbf{a}^{\text{base}})$
        $\mathcal{C} \leftarrow \{\text{clip}(\mathbf{a}^{\text{base}})\} \cup \{\text{clip}(\mathbf{a}^{(n)})\}_{n=1}^N$
        $\mathcal{E} \leftarrow$ the M elements of $\mathcal{C}$ with highest $\bar{Q}(\mathbf{s}, \mathbf{a})$
        Refine each $\mathbf{a} \in \mathcal{E}$ for G steps using Eq. (3.2) with $\bar{Q}$; clip after each step
        $\mathbf{a}^\star(\mathbf{s}, \mathbf{a}^{\text{base}}) \leftarrow \arg\max_{\mathbf{a} \in \mathcal{E}} \bar{Q}(\mathbf{s}, \mathbf{a})$; stop gradients through this target
      $\mathcal{L}_{\text{PA}}(\theta) \leftarrow \mathbb{E}_{\substack{(\mathbf{s}, \mathbf{a}^{\text{base}}) \sim \mathcal{D} \\ \hat{\mathbf{a}} \sim \pi^{\text{res}}_\theta(\cdot \mid \mathbf{s}, \mathbf{a}^{\text{base}})}} \big[\|\hat{\mathbf{a}} - \mathbf{a}^\star(\mathbf{s}, \mathbf{a}^{\text{base}})\|_2^2\big]$
      Update θ with reparameterized $\nabla_\theta \mathcal{L}_{\text{PA}}$: Adam at α_π (Appendix F.1)
    end if
    $i \leftarrow i + 1$
  end for
end while
return π^K_{RL}: sample $\mathbf{a}^{\text{base}} \sim \pi^K_{\text{base}}$ and execute $\tanh\big(\mu_\theta(\mathbf{s}_t, \mathbf{a}_t^{\text{base}})\big)$

Table 2: **One demonstration adaptation results measured over 3 seeds.** Success rates (%) are mean ± std. Note that the zero-shot performance of the pretrained policy is 0%.

		1 Demo				
Task	Zero-Shot	BC	DSRL	Filt. BC	DiceRL	**MIDAS**
LIBERO-Long						
Alph. Soup + Cr. Cheese → Basket	0.0	12.7±1.2	12.0±2.0	14.0±2.0	98.0±2.0	**99.3±1.2**
Alph. Soup + Tom. Sauce → Basket	0.0	4.0±3.5	3.3±2.3	2.0±0.0	**92.0±4.0**	82.7±13.3
Black Bowl → Bottom Drawer	0.0	22.7±6.4	28.7±4.2	28.7±10.3	90.7±3.1	**98.7±1.2**
Book → Caddy	0.0	37.3±7.0	42.7±6.1	28.0±2.0	98.0±3.5	**98.7±2.3**
Both Moka Pots → Stove	0.0	13.3±3.1	14.7±4.6	19.3±7.0	74.7±9.5	**76.0±14.0**
Cr. Cheese + Butter → Basket	0.0	34.0±9.2	28.0±2.0	68.0±2.0	88.7±2.3	**96.0±0.0**
Moka Pot → Stove	0.0	84.0±0.0	92.0±0.0	85.3±8.3	**98.7±1.2**	95.3±3.1
White Mug + Choc. Pudding	0.0	33.3±11.7	45.3±6.4	42.0±4.0	**97.3±4.6**	96.7±5.8
White Mug + Plates	0.0	30.0±2.0	17.3±1.2	34.7±6.4	61.3±11.5	**79.3±11.0**
Yellow-White Mug → Microwave	0.0	64.0±2.0	50.7±4.6	68.0±7.2	**93.3±3.1**	89.3±8.1
Average	0.0	33.5±1.8	33.5±0.8	39.0±2.4	89.3±1.8	**91.2±4.2**

E.1 LIBERO AND ROBOCASA TASK DETAILS

We evaluate on two simulation benchmarks: LIBERO-Long (Liu et al., 2023) and RoboCasa-365 (Nasiriany et al., 2026). Below we describe each benchmark and the specific tasks used.

LIBERO-Long. LIBERO-Long consists of 10 long-horizon, language-conditioned manipulation tasks performed by a fixed-base Franka Panda arm. Each task requires completing multiple subtasks in sequence (e.g., picking two objects and placing them in a container). The 10 tasks are:

1. Put both the alphabet soup and the tomato sauce in the basket
2. Put both the cream cheese box and the butter in the basket
3. Turn on the stove and put the moka pot on it
4. Put the black bowl in the bottom drawer of the cabinet and close it
5. Put the white mug on the left plate and put the yellow and white mug on the right plate
6. Pick up the book and place it in the back compartment of the caddy
7. Put the white mug on the plate and put the chocolate pudding to the right of the plate
8. Put both the alphabet soup and the cream cheese box in the basket
9. Put both moka pots on the stove
10. Put the yellow and white mug in the microwave and close it

Each task has 50 human-teleoperated demonstrations available in the benchmark; we use only $K = 1$ for adaptation in our main experiments. The environment provides two camera views: an `agentview` third-person camera and a `robot0_eye_in_hand` wrist camera, both rendered at 224×224 pixels. The action space is 7-dimensional: 3D end-effector position, 3D end-effector rotation (axis-angle), and 1D gripper open/close. The episode horizon is 500 timesteps. The learning curves for all the tasks can be visualized in Fig. 9.

Libero-Long Results Discussion: We observe that MIDAS as well as Dice-RL (Sun and Song, 2026) enable successful autonomous improvement over the one demonstration warm start. Dice-RL has better sample-efficiency over MIDAS on 5/10 tasks while under-performing in 1 and equivalent on the remaining 4.

DICE-RL. We implement DICE-RL (Sun and Song, 2026) following the full set of design choices proposed in the original method, using the frozen VLM embedding as the state representation. DICE-RL is a particularly relevant comparison to MIDAS as it independently incorporates many of the ingredients we find important for stable minimal-data adaptation: a lightweight residual actor over a frozen base policy, demonstration data retained during online learning through offline data mixing, and value-based policy improvement. Beyond these shared ingredients, we retain DICE-RL's multi-sample expectation training over K policy samples for both the critic bootstrap and actor objective, and filtered TD3+BC regularization (Fujimoto and Gu, 2021) that selectively allows value-improving deviations from the base policy. This provides a strong residual-RL baseline that differs from MIDAS primarily in how value estimates are used for policy improvement.

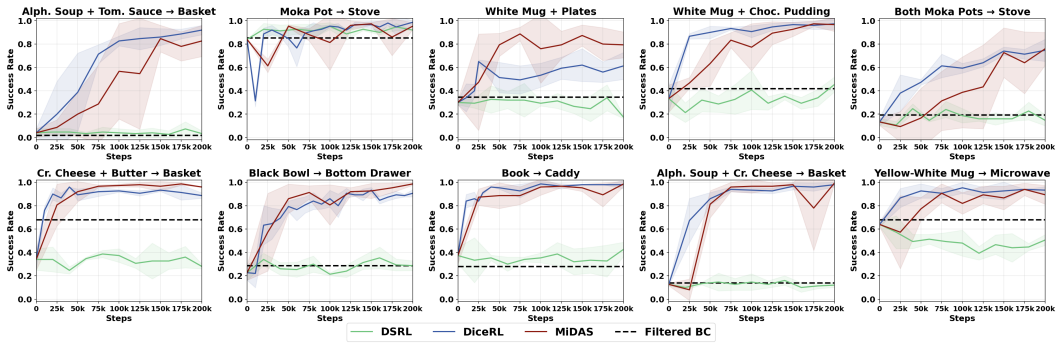


Figure 9: Libero-Long Learning Curves for MIDAS with baseline comparison. Solid line shows mean across 3 seeds and shaded region denotes standard deviation. Steps on X-axis indicate chunked environment steps (each chunked environment step is equivalent to C actual environment steps)

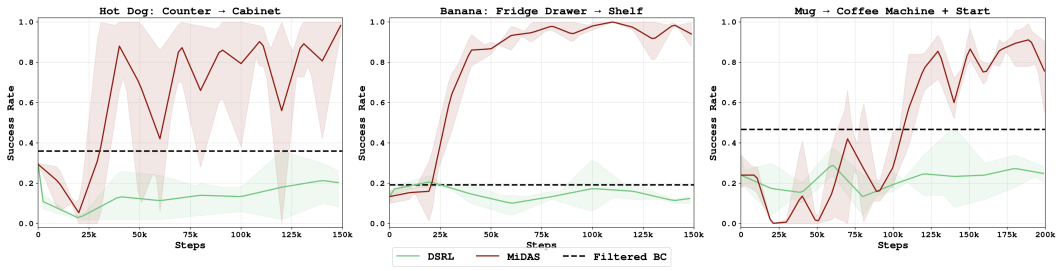


Figure 10: RoboCasa-365 Learning Curves for MIDAS with baseline comparison. Solid line shows mean across 3 seeds and shaded region denotes standard deviation. Steps on X-axis indicate chunked environment steps (each chunked environment step is equivalent to C actual environment steps)

RoboCasa-365. RoboCasa-365 (Nasiriany et al., 2026) provides long-horizon household manipulation tasks in diverse procedurally generated kitchen scenes. Unlike LIBERO, the robot is a mobile manipulator and tasks involve navigation, object retrieval, and multi-step interaction with kitchen appliances. We evaluate on four tasks:

1. **Banana: Fridge Drawer → Shelf** (`PickPlaceFridgeDrawerToShelf`): Pick a banana from a fridge drawer and place it on a shelf.
2. **Hot Dog: Counter → Cabinet** (`PickPlaceCounterToCabinet`): Pick a hot dog from the counter and place it in a cabinet.
3. **Mug → Coffee Machine + Start** (`PrepareCoffee`): Place a mug under the coffee machine and press the start button.
4. **Cup + Bowl → Dishwasher + Close** (`LoadDishwasher`): Load a cup and bowl into the dishwasher and close it.

Each task is instantiated in a specific kitchen layout and style that is fixed across training and evaluation. The environment provides two camera views: `robot0_agentview_left` and `robot0_eye_in_hand`, rendered at 256×256 pixels. The action space is 7-dimensional for the first three tasks (3D end-effector position, 3D rotation, 1D gripper), with episode horizons varying between 500 and 800 timesteps depending on task complexity (except for `LoadDishwashwer`). The learning curves for all the tasks can be visualized in Fig. 10.

Long Horizon Manipulation: Note that the `LoadDishwasher` task spans a horizon of 1100 timesteps and has an overall success rate of 0% across all baselines and MIDAS. This task shows the limitation of our proposed approach and provides motivation for future work to extend MIDAS to long-horizon, sparse-reward settings.

Shared observation and action details. In both benchmarks, the base policy ($\pi_{0.5}$) receives 224×224 images after resizing with padding. Actions are temporally chunked: the policy outputs a chunk of $C = 10$ primitive actions that are executed open-loop before replanning (see Appendix A). The residual learner receives the output of the VLM backbone as the state representation, along with the

base policy's proposed action chunk. Evaluation uses 50 rollouts per task from the canonical initial state distribution.

Details on environments. For all LIBERO tasks, episodes last at most 500 primitive environment steps. With action chunk length $H = C = 10$ (Table 11), the policy acts in a chunk-level MDP with at most $500/H = 50$ transitions per episode (Appendix A). After collecting each trajectory, once $|\mathcal{B}| > n_{\min}$, we perform one update round per collected transition using replay minibatches, as specified in Algorithm 1. Each round comprises U_C^{w} critic and U_A^{w} actor optimizer updates during warmup ($i < W$), or U_C critic and U_A actor updates thereafter, and increments i once after these updates. We use $(U_C^{\mathrm{w}}, U_A^{\mathrm{w}}) = (8, 8)$ and $(U_C, U_A) = (30, 10)$ (Table 11).

E.2 LIBERO PRO PERTURBATION DETAILS

We use the perturbation protocol from LIBERO-PRO to evaluate generalization under controlled distribution shifts. LIBERO-PRO defines five perturbation axes, of which we use four (excluding task perturbation, which changes the goal itself). We also introduce position perturbations at three magnitudes to test spatial generalization. Not all perturbation types apply to all 10 tasks; LIBERO-PRO defines perturbations only for tasks where a given shift is meaningful. Below we describe each perturbation type along with the specific tasks used.

Observation-level perturbations. These change task-irrelevant aspects of the observation while preserving the underlying state and goal.

- **Visual (object appearance)** (Fig. 6): Object colors, textures, and sizes are modified in the scene definition (e.g., an `akita_black_bowl` is replaced with a red or larger variant), while the task instruction and goal conditions remain unchanged. We evaluate on 5 tasks where LIBERO-PRO defines color or texture swaps that do not alter the object's shape or grasp affordance: *Both Moka Pots → Stove*, *Cr. Cheese + Butter → Basket*, *Moka Pot → Stove*, *White Mug + Choc. Pudding*, *White Mug + Plates*.
- **Language** (Fig. 6): The task instruction is replaced with a paraphrase that describes the same goal using different phrasing (e.g., "put both moka pots on the stove" → "place both moka pots on stove"). Object references and goal logic are preserved. We evaluate on all 10 LIBERO-Long tasks, as language paraphrases are defined for every task.

State-level perturbations. These change dynamically relevant quantities that may require different actions. LIBERO-PRO applies a single object-replacement perturbation per task, but the nature of the replacement varies: for some tasks the replacement changes only the object's shape while preserving its category, for others it substitutes a different object category entirely, and for others the two scene objects swap positions. We group results by the type of change each replacement induces.

- **Object swap** (Fig. 7): The initial positions of two objects are exchanged by swapping their spawn-region assignments in the scene definition. The policy must execute the task with objects in different spatial locations. We evaluate on the same 5 tasks as the visual perturbation: *Both Moka Pots → Stove*, *Cr. Cheese + Butter → Basket*, *Moka Pot → Stove*, *White Mug + Choc. Pudding*, *White Mug + Plates*.
- **Shape change** (Fig. 7): Target object meshes are replaced with geometrically different variants of the same category (e.g., a taller mug or wider bowl). If the new shape preserves similar grasp affordances, the required manipulation strategy remains the same; otherwise the policy must adapt its contact behavior. We evaluate on 4 tasks where the LIBERO-PRO replacement alters the object geometry: *Black Bowl → Bottom Drawer*, *Yellow-White Mug → Microwave*, *Alph. Soup + Cr. Cheese → Basket*, *Alph. Soup + Tom. Sauce → Basket*.
- **Object change** (Fig. 7): The target object is replaced with a different object category entirely, changing both visual appearance and manipulation affordances. We evaluate on 1 task where the replacement introduces a categorically different object: *Book → Caddy*.

Position perturbation. We evaluate position generalization by displacing target objects from their canonical initial positions at three levels:

- **Low:** 5 cm XY displacement, 0.1 radians yaw perturbation
- **Medium:** 10 cm XY displacement, 0.15 radians yaw perturbation

- **High:** 20 cm XY displacement, 0.25 radians yaw perturbation

XY offsets are sampled uniformly within a disk of the specified radius. After applying perturbations, objects are settled under physics simulation and validated: the perturbed object must remain visible in the camera frame, stay within 2 cm of its settled z-height (no falling off surfaces), not collide with other scene objects, and maintain an upright orientation (local up-axis within 18° of the world vertical). Invalid samples are re-drawn for up to 50 attempts per episode.

We evaluate position perturbation on 3 tasks: *Alph. Soup + Tom. Sauce → Basket*, *Black Bowl → Bottom Drawer*, *Both Moka Pots → Stove*. Perturbations are applied to the task-relevant target objects in each case (e.g., both moka pots, or the black bowl).

Evaluation protocol. All perturbation evaluations use 50 rollouts per task. For LIBERO-PRO perturbations (visual, language, swap, shape, object change), we evaluate from the perturbed initial configurations using the standard episode horizon. For position perturbations, we evaluate at each magnitude separately. Results are averaged across 3 seeds. Per-task breakdowns are provided in the tables below.

Visual perturbation results. Table 3 reports per-task success rates under visual (object appearance) perturbation.

	π^1_{base}		π^1_{RL}	
Task	ID	OOD	ID	OOD
Both Moka Pots → Stove	13.3±3.1	6.0±2.0	76.0±14.0	58.7±33.5
Cr. Cheese + Butter → Basket	34.0±9.2	38.0±9.2	96.0±0.0	98.7±2.3
Moka Pot → Stove	84.0±0.0	58.7±5.0	95.3±3.1	93.3±4.2
White Mug + Choc. Pudding	33.3±11.7	30.0±10.4	96.7±5.8	80.0±11.1
White Mug + Plates	30.0±2.0	24.0±2.0	79.3±11.0	79.3±13.6
Average	38.9±3.6	31.3±4.4	88.7±5.9	82.0±8.0

Table 3: Per-task success rate (%) under visual (object appearance) perturbation. OOD replaces object colors, textures, or sizes while preserving shape and task goal.

Language perturbation results. Table 4 reports per-task success rates under language perturbation.

	π^1_{base}		π^1_{RL}	
Task	ID	OOD	ID	OOD
Alph. Soup + Cr. Cheese → Basket	12.7±1.2	14.0±2.0	99.3±1.2	98.0±2.0
Alph. Soup + Tom. Sauce → Basket	4.0±3.5	4.7±3.1	82.7±13.3	78.7±11.0
Black Bowl → Bottom Drawer	22.7±6.4	23.3±9.2	98.7±1.2	96.7±5.8
Book → Caddy	37.3±7.0	33.3±8.3	98.7±2.3	96.7±4.2
Both Moka Pots → Stove	13.3±3.1	15.3±2.3	76.0±14.0	79.3±13.0
Cr. Cheese + Butter → Basket	34.0±9.2	40.0±14.0	96.0±0.0	96.0±2.0
Moka Pot → Stove	84.0±0.0	88.7±5.8	95.3±3.1	97.3±2.3
White Mug + Choc. Pudding	33.3±11.7	38.7±13.3	96.7±5.8	91.3±8.1
White Mug + Plates	30.0±2.0	33.3±5.8	79.3±11.0	78.0±2.0
Yellow-White Mug → Microwave	64.0±2.0	56.0±6.9	89.3±8.1	90.0±7.2
Average	33.5±1.8	34.7±4.8	91.2±4.2	90.2±3.3

Table 4: Per-task success rate (%) under language perturbation. OOD replaces the task instruction with a paraphrase.

E.3 LIBERO PRO STATE SHIFT PERTURBATION EXPERIMENTS

Here we provide further analysis of the state-shift generalization results from Section 7.

Object swap. When two objects exchange positions, the policy must approach the target from a different direction or reorder its subtask sequence. On the 5 evaluated tasks (*Both Moka Pots → Stove*, *Cr. Cheese + Butter → Basket*, *Moka Pot → Stove*, *White Mug + Choc. Pudding*, *White Mug + Plates*), swapping object positions changes the optimal grasp order and approach trajectory. Both π^1_{base} and π^1_{RL} achieve 0% success across all 5 tasks under swap perturbation (Table 5), despite strong ID performance. This indicates that swapping object positions produces configurations far enough from the training distribution that neither the BC warm start nor residual RL corrections can recover.

	π^1_{base}		π^1_{RL}	
Task	ID	OOD	ID	OOD
Both Moka Pots → Stove	13.3$_{\pm 3.1}$	0.0$_{\pm 0.0}$	76.0$_{\pm 14.0}$	0.0$_{\pm 0.0}$
Cr. Cheese + Butter → Basket	34.0$_{\pm 9.2}$	0.0$_{\pm 0.0}$	96.0$_{\pm 0.0}$	0.0$_{\pm 0.0}$
Moka Pot → Stove	84.0$_{\pm 0.0}$	0.0$_{\pm 0.0}$	95.3$_{\pm 3.1}$	0.0$_{\pm 0.0}$
White Mug + Choc. Pudding	33.3$_{\pm 11.7}$	0.0$_{\pm 0.0}$	96.7$_{\pm 5.8}$	0.0$_{\pm 0.0}$
White Mug + Plates	30.0$_{\pm 2.0}$	0.0$_{\pm 0.0}$	79.3$_{\pm 11.0}$	0.0$_{\pm 0.0}$
Average	38.9$_{\pm 3.6}$	0.0$_{\pm 0.0}$	88.7$_{\pm 5.9}$	0.0$_{\pm 0.0}$

Table 5: Per-task success rate (%) under object swap perturbation. OOD exchanges the initial positions of two scene objects.

Shape change. Shape perturbations test whether learned grasps transfer across object geometries. On the 4 evaluated tasks (*Black Bowl → Bottom Drawer*, *Yellow-White Mug → Microwave*, *Alph. Soup + Cr. Cheese → Basket*, *Alph. Soup + Tom. Sauce → Basket*), the replacement preserves the object category but alters its geometry. Table 6 shows that π^1_{RL} retains strong performance on tasks where the new shape preserves similar grasp affordances (e.g., Black Bowl at 90.7%), but drops sharply when the shape change alters the required contact geometry (e.g., Alph. Soup tasks at $<$3%).

	π^1_{base}		π^1_{RL}	
Task	ID	OOD	ID	OOD
Black Bowl → Bottom Drawer	22.7$_{\pm 6.4}$	20.7$_{\pm 6.4}$	98.7$_{\pm 1.2}$	90.7$_{\pm 8.1}$
Yellow-White Mug → Microwave	64.0$_{\pm 2.0}$	4.7$_{\pm 3.1}$	89.3$_{\pm 8.1}$	20.7$_{\pm 13.3}$
Alph. Soup + Cr. Cheese → Basket	12.7$_{\pm 1.2}$	0.0$_{\pm 0.0}$	99.3$_{\pm 1.2}$	2.7$_{\pm 4.6}$
Alph. Soup + Tom. Sauce → Basket	4.0$_{\pm 3.5}$	0.0$_{\pm 0.0}$	82.7$_{\pm 13.3}$	2.0$_{\pm 3.5}$
Average	25.8$_{\pm 0.8}$	6.3$_{\pm 1.0}$	92.5$_{\pm 5.4}$	29.0$_{\pm 5.8}$

Table 6: Per-task success rate (%) under shape change perturbation. OOD replaces the target object with a geometrically different variant of the same category.

Object change. Replacing the target with a different object category represents the most severe state shift. We evaluate this on *Book → Caddy*, where the book is replaced with a categorically different object. Table 7 shows that both π^1_{base} and π^1_{RL} drop to near 0% under this shift despite strong ID performance. The new object differs in size, shape, weight, and affordance, requiring manipulation strategies outside the behavioral support of π^1_{RL}. This category of shift cannot be addressed by online RL alone and requires additional demonstrations or interaction with the new object.

Position perturbation. Tables 8,9,10 reports per-task success rates under position perturbation at three displacement levels. We compare π^1_{base}, π^1_{RL}, and π^{50}_{base} (behavior-cloned on all 50 demonstra-

	π^1_{base}		π^1_{RL}	
Task	ID	OOD	ID	OOD
Book → Caddy	37.3±7.0	0.0±0.0	98.7±2.3	2.0±2.0

Table 7: Per-task success rate (%) under object change perturbation. OOD replaces the target with a categorically different object.

tions). π^1_{RL} improves over π^1_{base} at all levels but falls short of π^{50}_{base}, especially at larger displacements where the gap in state coverage becomes the binding constraint.

Task	π^1_{base}	π^1_{RL}	π^{50}_{base}
Alph. Soup + Tom. Sauce	4.0±2.0	52.7±5.8	94.7±5.8
Black Bowl → Drawer	24.0±7.2	88.7±5.0	59.3±10.1
Both Moka Pots → Stove	3.3±3.1	49.3±20.1	84.0±7.2
Average	10.4±2.5	63.6±7.3	79.3±6.4

Table 8: Per-task success rate (%) under low position perturbation of 5 cm.

Task	π^1_{base}	π^1_{RL}	π^{50}_{base}
Alph. Soup + Tom. Sauce	1.3±2.3	14.0±2.0	68.7±3.1
Black Bowl → Drawer	16.7±6.4	70.0±3.5	45.3±2.3
Both Moka Pots → Stove	1.3±1.2	22.7±8.3	76.0±7.2
Average	6.4±3.1	35.6±1.5	63.3±1.2

Table 9: Per-task success rate (%) under medium position perturbation of 10 cm.

F Implementation Details

F.1 MiDAS

State representation. The state input to both the residual actor and critic is derived from the frozen VLA backbone of π^K_{base} along with the proprioceptive state. At each timestep, we extract the VLM token embeddings and mean-pool across all token positions to produce a 2048-dimensional feature vector. This vector is then projected through a bottleneck layer (linear projection, layer normalization, and tanh activation) to yield a 200-dimensional latent state which is then concatenated with the proprioceptive state to get the state representation $\mathbf{s}_t$.

Actor inputs and architecture. The residual actor receives the state representation as the bottlenecked VLM embedding and proprioceptive state, concatenated with the flattened base action chunk $\mathbf{a}^{\text{base}}_t \in \mathbb{R}^{C\cdot d_a}$ proposed by π^K_{base}, where C=10 is the chunk length and d_a=7 is the per-step action dimension. As described in Section 4, the actor directly outputs the full executed action chunk $\mathbf{a}^{\text{exec}}_t$ rather than an additive residual correction. The actor is parameterized as a 3-layer MLP with 512 hidden units per layer and ReLU activations. The output distribution is a tanh-squashed diagonal Gaussian with learned mean and log-standard-deviation heads.

Critic inputs and architecture. The critic receives only the bottlenecked VLM embedding and proprioceptive state as its state input; the base action chunk is excluded. Given a candidate executed action $\mathbf{a}^{\text{exec}}_t$, each Q-network in the ensemble evaluates $Q_{\psi_j}(\mathbf{s}_t, \mathbf{a}^{\text{exec}}_t)$. We use an ensemble of 10 Q-networks, each parameterized as a 3-layer MLP with 512 hidden units and ReLU activations. The ensemble aggregates predictions via mean reduction during target computation.

Warmup Details. We collect warmup data by rolling out the base policy with an action chunk length of $C = 10$. We count collection steps at the chunk level: each step corresponds to executing one action chunk and collecting one chunked transition, spanning 10 primitive environment steps. We

Task	π^{1}_{base}	π^{1}_{RL}	π^{50}_{base}
Alph. Soup + Tom. Sauce	$0.0_{\pm 0.0}$	$2.7_{\pm 3.1}$	$25.3_{\pm 4.2}$
Black Bowl → Drawer	$6.7_{\pm 2.3}$	$37.3_{\pm 2.3}$	$23.3_{\pm 8.1}$
Both Moka Pots → Stove	$0.0_{\pm 0.0}$	$2.7_{\pm 3.1}$	$25.3_{\pm 9.5}$
Average	$2.2_{\pm 0.8}$	$14.2_{\pm 0.4}$	$24.7_{\pm 4.4}$

Table 10: Per-task success rate (%) under high position perturbation of 20 cm.

collect a total of 10,000 chunked transitions, corresponding to 100,000 primitive environment steps when all chunks are executed in full. For LIBERO, the maximum episode horizon is 500 primitive environment steps, or 50 chunks. Thus, this collection budget is equivalent to approximately 200 full-length episodes; collecting the same number of transitions requires more episodes if trajectories terminate early. During warmup, we perform eight actor optimizer updates and eight critic optimizer updates per collected chunked transition. The update-to-data (UTD) ratio is therefore 8 for each network when measured per chunked transition. Normalized by primitive environment steps, this corresponds to $8/C = 0.8$ updates per environment step for each network.

Policy update. At each actor update, a single base action $\mathbf{a}_t^{\text{base}}$ is sampled from the frozen π^{K}_{base} (in practice we use the cached base action from the replay buffer). The residual actor then proposes $N{=}16$ candidate actions conditioned on $\mathbf{s}_t$ and $\mathbf{a}_t^{\text{base}}$, and also includes the base action as a valid proposal. The top $M{=}8$ actions ranked by the critic ensemble are selected as elites. These elite actions undergo 30 steps of gradient ascent on the Q-ensemble with step size 10^{-3}. The actor is then updated via a distillation loss that regresses toward the refined elite action.

Optimization. Both actor and critic use the Adam optimizer with global gradient norm clipping at 1.0. The actor learning rate is 10^{-4} and the critic learning rate is 3×10^{-4}. For each chunked environment step (equivalent to C actual environment steps), we perform 30 critic updates and 10 actor updates. Training begins with a BC warmup phase as described in the warmup details above.

Key hyperparameters. Table 11 summarizes the hyperparameters used for all PARL training runs.

Table 11: PARL hyperparameters used for all training runs.

Hyperparameter	**Value**
Batch size (B)	64
Discount factor (γ)	0.999
Target network update rate (τ)	0.05
Actor learning rate (α_π)	10^{-4}
Critic learning rate	3×10^{-4}
Hidden dimensions	$(512, 512, 512)$
VLM embedding bottleneck dim	200
Number of Q-networks (J)	10
Critic ensemble reduction	mean
PA-RL candidate samples (N)	16
PA-RL elite samples (M)	8
PA-RL gradient steps (G)	30
PA-RL step size (η)	10^{-3}
Critic updates per chunked step (U_C)	30
Actor updates per chunked step (U_A)	10
Warmup collection steps (chunked)	10,000
BC warmup critic/actor updates ($U_C^{\text{w}}, U_A^{\text{w}}$)	8
Success buffer ratio (ρ_{succ})	0.20
Success buffer minimum size	20
Gradient norm clip	1.0
Optimizer	Adam
Action chunk length (C, H)	10

F.2 BC IMPLEMENTATION

For behavior cloning, we use a single demonstration out of the available 50 for LIBERO-Long and one demonstration for the particular scene and layout configuration for each RoboCasa task. We then fine-tune $\pi_{0.5}$ using the official OpenPI codebase on this demonstration for 16,000 gradient steps. Table 13 shows the performance on of behavior cloning on varying the number of demos $K = \{1, 3, 5\}$.

Trainable parameters. We apply LoRA adapters to the PaliGemma 2B vision-language-model (VLM) backbone, inserting low-rank updates into both the attention and feed-forward layers with rank 32 and scaling factor α=32. The 300M-parameter action head (Gemma 300M) and the SigLIP vision encoder are fully fine-tuned. All other backbone weights remain frozen. This selective fine-tuning keeps the majority of the VLM fixed while allowing the action head and vision encoder to specialize to the target task.

Loss and optimization. The model is trained with the flow-matching objective native to the $\pi_{0.5}$ architecture, predicting action chunks of horizon C=10. We use AdamW with β_1=0.9, β_2=0.95, and negligible weight decay (10^{-10}), with global gradient norm clipping at 1.0. The learning rate follows a cosine decay schedule: peak learning rate 2.5×10^{-5}, 50 warmup steps, decaying to 2.5×10^{-6} over 100,000 steps. Training is stopped at 16,000 gradient steps. The batch size is 32.

Key hyperparameters. Table 12 lists the hyperparameters used for all Stage I behavior cloning runs.

Table 12: Stage I behavior cloning hyperparameters.

Hyperparameter	**Value**
Number of demonstrations (K)	1
Training steps	16,000
Batch size	32
Optimizer	AdamW
Peak learning rate	2.5×10^{-5}
LR schedule	Cosine decay
LR warmup steps	50
Minimum learning rate	2.5×10^{-6}
β_1, β_2	0.9, 0.95
Gradient norm clip	1.0
LoRA rank	32
LoRA α	32
LoRA targets	Attention + FFN
Action expert	Fully fine-tuned
Vision encoder (SigLIP)	Fully fine-tuned
Action horizon (C)	10
Loss	Flow matching

F.3 SUCCESS BUFFER

The success buffer $\mathcal{B}_{\text{succ}}$ is a separate replay buffer that stores only transitions from successful episodes. Its purpose is to prevent actor and critic updates from being dominated by failed rollouts when the policy's success rate is low (2–5% early in training).

Initialization. Both the main replay buffer $\mathcal{B}$ and the success buffer $\mathcal{B}_{\text{succ}}$ are initialized with all transitions from the K demonstrations, since demonstration trajectories are successful by construction. The success buffer capacity is set to $\max(10{,}000,\ 0.5 \times |\mathcal{B}|_{\max})$.

Population during training. After each collected trajectory, all transitions are inserted into $\mathcal{B}$. If the episode was successful (sparse reward received), the same transitions are additionally inserted into $\mathcal{B}_{\text{succ}}$.

Sampling. Actor and critic update minibatches are sampled from a mixture of the two buffers:

$$\mathcal{B}_{\text{actor/critic}} \sim (1 - \rho_{\text{succ}}) \cdot \mathcal{B} + \rho_{\text{succ}} \cdot \mathcal{B}_{\text{succ}}, \tag{F.1}$$

Task	BC Success Rate (%) K=1	K=3	K=5
LIBERO-Long			
Alphabet Soup + Cream Cheese → Basket	12.7	39.3	**82.7**
Alphabet Soup + Tomato Sauce → Basket	4.0	32.7	**74.7**
Black Bowl → Bottom Drawer	22.7	90.0	**94.0**
Book → Caddy	37.3	26.7	**89.3**
Both Moka Pots → Stove	13.3	**51.3**	35.3
Cream Cheese + Butter → Basket	34.0	**90.7**	84.7
Moka Pot → Stove	84.0	86.0	**94.7**
White Mug + Chocolate Pudding	33.3	26.7	**66.0**
White Mug + Plates	30.0	48.7	**74.0**
Yellow-White Mug → Microwave	64.0	70.7	**86.0**
Average	33.5	56.3	**78.1**

Table 13: **Per-task BC success on LIBERO-Long.** Bold denotes the best result per task across $K \in \{1, 3, 5\}$ demonstrations.

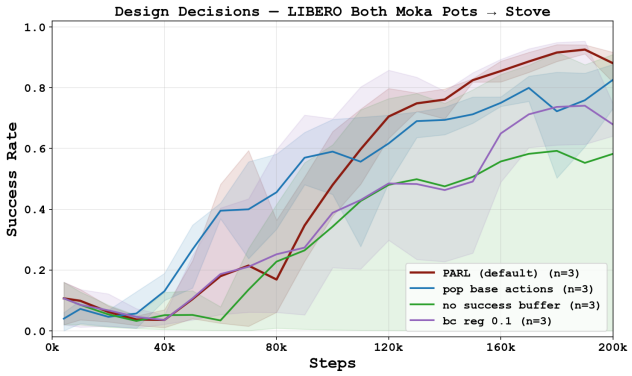


Figure 11: MIDAS ablations on both-mokapots task

where $\rho_{\text{succ}} = 0.2$ is the success buffer ratio. Concretely, each actor batch of size 64 contains 51 transitions sampled uniformly from $\mathcal{B}$ and 13 transitions sampled uniformly from $\mathcal{B}_{\text{succ}}$. This oversampling biases the actor toward states where successful behavior has been observed, steering the PA-RL elite selection and gradient ascent toward reward-bearing regions.

G Ablations

Design-decision ablations. Fig. 11 ablates three implementation choices in MIDAS on the `Both Moka Pots -> Stove` task. Each row changes a single knob while keeping the same PA-RL objective, VLM state representation, 10k-step BC warmup, and one demonstration initialization. First, removing the base-policy action from the residual actor nearly matches the default recipe. This suggests that the action head is most important for producing task-relevant warmup experience and distilling this information into the residual actor during the warmup; once this experience is available, online value learning from pretrained VLA features can recover much of the final performance without using the base action as an explicit actor input. Second, disabling the success buffer tests whether sparse successful transitions must be oversampled during actor updates, rather than being learned from uniformly mixed online replay. Third, adding a persistent success-only BC penalty tests whether the residual should remain behaviorally anchored after warmup, or whether continued RL updates should be allowed to move away from the demonstrated action distribution.

H CURRICULUM ENABLES STRONGER POSITION GENERALIZATION

Table 8, Table 9, Table 10 and Fig. 12b show that π^1_{RL} generalizes well to small position perturbations: at 5 cm displacement, average success reaches 63.6%, compared to 10.4% for π^1_{base}. Performance drops sharply at larger displacements, falling to 35.6% at 10 cm and 14.2% at 20 cm. By contrast, π^{50}_{base} maintains 79.3%, 63.3%, and 24.7% at the same levels, indicating that broader training coverage sustains robustness where π^1_{RL} does not.

A key observation is that larger position perturbations in our setting can be decomposed incrementally: a 10 cm displacement is reachable through a sequence of smaller shifts from the nominal configuration. If the policy can reliably solve the task under 5 cm perturbation, it should be able to extend to 10 cm once it has practiced under the smaller shift. This motivates a curriculum over the reset distribution: rather than training under the full perturbation range from the outset, we progressively widen the displacement radius as the policy masters each level.

Curriculum procedure. Starting from the trained π^1_{RL} checkpoint (after Stage II of MIDAS), we continue online RL under position-perturbed resets. The curriculum proceeds through three stages with increasing perturbation radii: 5 cm, 10 cm, and 20 cm. At each stage, the target objects are displaced uniformly at random within the specified radius from their default positions. The policy advances to the next stage once its evaluation success rate on the current perturbation level exceeds a threshold of 0.6. On the final stage, training continues until the step budget is exhausted. All other hyperparameters (critic/actor updates, learning rates, success buffer) remain identical to the standard MIDAS Stage II configuration (Table 11). No additional demonstrations are provided; the curriculum relies entirely on autonomous interaction.

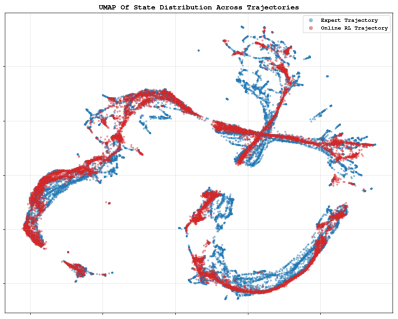
Task selection. We run and evaluate the curriculum on the three tasks used for position perturbation evaluation: *Alphabet Soup + Tomato Sauce → Basket*, *Both Moka Pots → Stove*, and *Black Bowl → Bottom Drawer*. For *Black Bowl*, π^1_{RL} already achieves 88.7% at 5 cm and 70.0% at 10 cm without curriculum (Table 8, Table 9), both exceeding the 0.6 advancement threshold. We therefore exclude it from the curriculum experiment, as it already demonstrates strong position generalization under the standard MIDAS recipe, and use it only for evaluation.

Results. Table 14 reports per-task success rates after curriculum training, compared against the non-curriculum π^1_{RL} and π^{50}_{base} baselines from Table 8,Table 9,Table 10. On *Both Moka Pots*, curriculum raises 5 cm success from 49.3% to 70.0% and 10 cm success from 22.7% to 75.3%, nearly matching π^{50}_{base} (76.0%) at the medium perturbation level. On *Alphabet Soup + Tomato Sauce*, 5 cm success improves from 52.7% to 68.7% and 10 cm success from 14.0% to 28.7%. The gains are consistent across seeds, with the largest improvements at the 10 cm level where the non-curriculum policy struggles most.

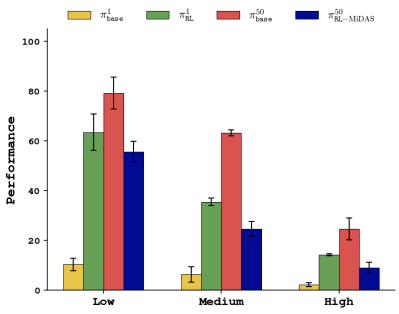
	π^1_{RL}		π^1_{RL} + Curriculum		π^{50}_{base}	
Task	5 cm	10 cm	5 cm	10 cm	5 cm	10 cm
Alph. Soup + Tom. Sauce	52.7$\pm$5.8	14.0$\pm$2.0	**68.7**$\pm$**5.0**	**28.7**$\pm$**8.1**	94.7$\pm$5.8	68.7$\pm$3.1
Both Moka Pots → Stove	49.3$\pm$20.1	22.7$\pm$8.3	**70.0**$\pm$**7.2**	**75.3**$\pm$**3.1**	84.0$\pm$7.2	76.0$\pm$7.2
Average	51.0$\pm$12.8	18.3$\pm$4.0	**69.3**$\pm$**1.2**	**52.0**$\pm$**5.3**	89.3$\pm$6.0	72.3$\pm$2.5

Table 14: Position perturbation success rate (%) before and after curriculum training. Curriculum closes much of the gap with π^{50}_{base} without additional demonstrations. Bold marks improvement over non-curriculum π^1_{RL}.

These results confirm that progressively expanding the reset distribution during online RL can close much of the position-generalization gap without collecting additional demonstrations. Each curriculum stage builds on the behavioral repertoire acquired at the previous level, allowing the policy to incrementally extend its reach to larger displacements. In Appendix I, we analyze what drives the position-generalization gap between π^1_{RL} and π^{50}_{base} and why the curriculum is effective.



(a) State-space coverage.



(b) Position generalization.

Figure 12: **(a)** π_{rl}^{1} covers a narrow region of state space around its warm-start demo; π_{base}^{50} visits a much broader distribution. **(b)** Success degrades with displacement magnitude; distilling π_{rl}^{1} rollouts into the base policy does not close the gap with π_{base}^{50}.

Table 15: Success rate (%) of flow-matching BC baselines on LIBERO-Long. Policies are evaluated at the final checkpoint on 50 canonical initial states.

Task	Image	State
LIBERO-Long		
Alph. Soup + Cr. Cheese → Basket	0.0	0.0
Alph. Soup + Tom. Sauce → Basket	0.0	0.0
Black Bowl → Bottom Drawer	14.0	0.0
Book → Caddy	0.0	0.0
Both Moka Pots → Stove	0.0	0.0
Cr. Cheese + Butter → Basket	0.0	0.0
Moka Pot → Stove	8.0	0.0
White Mug + Choc. Pudding	0.0	0.0
White Mug + Plates	0.0	0.0
Yellow-White Mug → Microwave	30.0	0.0
Average	**5.2**	**0.0**

I Position Generalization Analysis

Position perturbations expose a gap between ID success and state-shift robustness. We isolate this effect using position perturbations at different magnitudes. We compare π_{RL}^{1} to π_{base}^{1} and to π_{base}^{50}, a behavior-cloned policy trained on all 50 human-teleoperated demonstrations for the task. Although π_{RL}^{1} and π_{base}^{50} achieve similar in-distribution success, Fig. 12b shows that π_{base}^{50} generalizes much better as perturbations grow. *The gap is not explained by the residual parameterization alone.* One possibility is that π_{RL}^{1} generalizes poorly because the residual actor is less expressive than the pretrained policy architecture. To test this, we distill 50 successful rollouts from π_{RL}^{1} back into the base policy, yielding $\pi_{\text{RL}-\text{MIDAS}}^{50}$. This still does not recover the robustness of π_{base}^{50}; in fact, $\pi_{\text{RL}-\text{MIDAS}}^{50}$ performs worse than π_{RL}^{1} under position perturbations. This suggests that the gap is driven not by policy parameterization, but by differences in state coverage and motion priors between human expert demonstrations and autonomous data. We visualize this coverage in Fig. 12a, where the 50 human demonstrations span a broader region of state space than successful autonomous trajectories, further supporting this explanation.

J Baselines

J.1 Flow matching policy BC baseline

We include two flow-matching behavior-cloning baselines on LIBERO to separate the effect of the pretrained VLA from the effect of using an expressive action decoder. Both baselines use the same

flow-matching action generator and differ only in how observations are represented: one policy conditions on image observations and robot proprioception, while the other conditions on a privileged augmented state vector.

Flow-matching action generator. The action generator is a Multi-Modal Diffusion Transformer (MMDiT), following the joint-attention design used in Stable Diffusion 3-style generative models and adapted here from image generation to action-chunk prediction. We use this architecture as a design template: the flow policy itself is trained on LIBERO demonstrations, while the image-conditioned variant uses ImageNet-pretrained ResNet encoders for visual observations. The policy predicts a chunk of $H = 16$ future actions, where each action is a 7-dimensional end-effector command consisting of Δposition, Δorientation, and gripper control.

We train the policy with rectified-flow matching. Given an expert action chunk a and Gaussian noise $z \sim \mathcal{N}(0, I)$, we sample a flow time $t \sim \mathcal{U}[0, 1]$ and construct the linear interpolation

$$x_t = tz + (1 - t)a.$$

The network is trained to regress the conditional velocity field along this path using an action-padding-masked MSE loss. At inference time, actions are sampled by integrating the learned velocity field from Gaussian noise using 10 Euler steps, and we use EMA weights for evaluation.

Observation tokens and noised action tokens are linearly embedded to a width of 768 and augmented with frozen 1D sinusoidal positional embeddings. The MMDiT backbone then applies joint attention over observation and action tokens, with the action tokens allowed to attend to all observation tokens. The backbone has 12 transformer blocks, 12 attention heads, head dimension 64, and an FFN expansion factor of 4. Each block uses timestep-conditioned adaLN-Zero modulation, joint multi-head self-attention with RMS-normalized queries and keys, and a feed-forward layer. A final modulated projection maps the action tokens back to $\mathbb{R}^7$.

Table 16: Shared hyperparameters for the flow-matching BC baselines.

Hyperparameter	Value
Backbone	MMDiT
Transformer depth	12
Hidden size	768
Attention heads	12
Head dim.	64
FFN expansion	4
Obs. history	2
Action horizon	16
Action dim.	7
Flow path	rectified flow
Flow-time sampling	$\mathcal{U}[0, 1]$
Inference steps	10 Euler
Optimizer	AdamW
Learning rate	10^{-4}
Weight decay	0.01
EMA decay	0.9999
Normalization	z-score

Image-conditioned flow baseline. The image-conditioned baseline uses two camera streams: a third-person `agentview` camera and a wrist-mounted `eye-in-hand` camera. Each camera is processed by an independent ImageNet-pretrained ResNet-18 encoder. We replace BatchNorm layers with GroupNorm and truncate the ResNet at `layer4`, producing a $512 \times 3 \times 3$ feature map per frame. Input images are resized to 100×100, ImageNet-normalized, and cropped to 84×84, using random crops during training and center crops during evaluation.

Each camera feature map is converted into a single observation token using a learned spatial-embedding head, followed by a linear bottleneck to the MMDiT width of 768. The policy also receives robot proprioception, consisting of end-effector position, end-effector axis-angle orientation, and gripper joint positions. This 8-dimensional proprioceptive vector is linearly embedded as one additional observation token. Thus, for each observation timestep the image policy receives three tokens: one token for each camera and one proprioceptive token. With a two-step observation history, the policy receives six observation tokens total.

State-conditioned flow baseline. The state-conditioned baseline only conditions on a privileged augmented state vector. This vector is used identically when reconstructing observations from dataset MuJoCo states and during online evaluation. The state vector is constructed from four blocks: robot proprioception, object/fixture/site poses, BDDL goal-predicate indicators, and goal-target geometry.

The proprioceptive block contains end-effector position, end-effector axis-angle orientation, and gripper joint positions, giving 8 dimensions. The entity-pose block contains the position and quaternion of every object, fixture, and site, sorted by name. The goal-predicate block contains one binary indicator per BDDL goal sub-predicate. The goal-target geometry block contains the world position of each goal sub-predicate's target entity. For a task with N_{ent} entities and N_{goal} goal sub-predicates, the resulting state dimension is

$$D = 8 + 7N_{\text{ent}} + 4N_{\text{goal}}.$$

The state-conditioned policy receives a two-step history of this full augmented state vector and embeds it into observation tokens for the same MMDiT action generator used by the image baseline.

Table 17: Privileged-state dimensions for the LIBERO flow baseline.

Task	N_{ent}	N_{goal}	D
0	17	2	135
1	17	2	135
2	7	2	65
3	13	2	107
4	10	2	86
5	11	1	89
6	10	2	86
7	11	2	93
8	7	3	69
9	9	2	79

J.2 Diffusion-Steering RL (DSRL)

DSRL (Wagenmaker et al., 2025) trains a SAC agent that operates in the noise space of the frozen $\pi_{0.5}$ flow-matching policy. At each timestep, the SAC actor receives VLM features extracted by the frozen $\pi_{0.5}$ backbone and outputs a perturbation to the initial noise vector; the frozen denoising head then converts the perturbed noise into an executed action chunk. Because the base policy weights remain fixed, all task-relevant adaptation occurs through the learned noise perturbation. We use the official codebase of Wagenmaker et al. (2025) for this baseline.

The SAC actor and critic are three-layer MLPs with hidden dimension 128. Actions are predicted as chunks of length $C = 10$, and online updates begin after 500 initial environment steps. All other training infrastructure (evaluation protocol, reward definition, environment wrappers) matches the MIDAS Stage II setup (Appendix F).

We note that for RoboCasa tasks evaluated at 150k steps, we estimate DSRL performance at the reporting point by linearly interpolating the evaluations at 140k and 160k steps separately for each task and training seed. Each endpoint evaluation uses 50 rollouts. This is used for the numbers in Table 1.

J.3 Filtered BC

Filtered BC iteratively collects autonomous rollouts from π_{base}^{K}, retains only successful trajectories, and retrains the base policy on this filtered data. The model architecture and trainable parameters are identical to Stage I behavior cloning (Appendix F.2).

Procedure. Training proceeds in rounds. Each round consists of three phases: (1) collect N rollouts using the current policy, (2) filter for trajectories that reached the task goal, and (3) retrain the policy on the filtered set for M gradient steps. The updated checkpoint from each round initializes the next. We use cumulative filtering, where successful trajectories accumulate across rounds so the training set grows monotonically. The K expert demonstrations are included alongside the self-collected successes in every round. We run 20 rounds with 20 rollouts per round and 500 gradient steps per round, for a total of 400 collection rollouts and 10,000 gradient steps. We keep this setting as we

Table 18: DSRL hyperparameters.

Hyperparameter	Value
Batch size	64
Discount factor (γ)	0.999
Actor learning rate	10^{-4}
Critic learning rate	3×10^{-4}
Temperature learning rate	3×10^{-4}
Hidden dimensions	$(128, 128, 128)$
Update-to-data ratio	20
Action chunk length (C)	10
Action noise magnitude	1.0
Online update start	500 steps
Optimizer	Adam

find that the success rate saturates after the initial rise in our experiments and showed little further improvement within the evaluated budget.

Trainable parameters. The same parameter groups are updated as in Stage I: LoRA adapters (rank 32, $\alpha = 32$) on the PaliGemma 2B vision-language-model (VLM) backbone, full fine-tuning of the Gemma 300M action head, and full fine-tuning of the SigLIP vision encoder. All other backbone weights remain frozen.

Optimization. Each round trains with the flow-matching loss using the optimizer and learning-rate schedule from the $\pi_{0.5}$ training config (AdamW, $\beta_1 = 0.9$, $\beta_2 = 0.95$, peak learning rate 2.5×10^{-5}, cosine decay to 2.5×10^{-6}). The batch size is 16.

Table 19: Filtered BC hyperparameters.

Hyperparameter	Value
Number of rounds	20
Rollouts per round	20
Training steps per round	500
Total training steps	10,000
Batch size	16
Cumulative data	Yes
Include expert demos	Yes
Optimizer	AdamW
Peak learning rate	2.5×10^{-5}
LR schedule	Cosine decay
Minimum learning rate	2.5×10^{-6}
LoRA rank	32
LoRA α	32
LoRA targets	Attention + FFN
Action expert	Fully fine-tuned
Vision encoder (SigLIP)	Fully fine-tuned
Action horizon (C)	10
Loss	Flow matching

J.4 DICE-RL

DICE-RL (Sun and Song, 2026) trains a residual actor using a TD3+BC-style objective on top of the frozen $\pi_{0.5}$ base policy. The residual actor is parameterized as a tanh-Gaussian over flattened action chunks, and its output δ is added to the base action to form the executed action

$$a_{\text{exec}} = \text{clip}\big(\mathbf{a}^{\text{base}} + \alpha\,\delta,\, -1,\, 1\big)\,,$$

where α is a fixed residual scale. The base policy weights remain frozen; only the residual actor and a Q-critic are trained. All training infrastructure (reward definition, environment wrappers,

evaluation protocol) matches the MIDAS Stage II setup (Appendix F), and the policy conditions on the frozen $\pi_{0.5}$ VLM embedding rather than raw pixels.

Networks. The residual actor is an MLP tanh-Gaussian policy over the $C\times 7$ action chunk ($C = 10$), and the critic is an ensemble of 10 state–action Q-heads. Both use hidden width 512 and condition on the VLM embedding together with the base action. The critic is reduced by its ensemble mean, and a soft target critic is maintained by Polyak averaging ($\tau = 0.05$).

Critic update. The critic regresses an entropy-free, K-sample bootstrap target. At each next state, $K = 16$ residuals are sampled from the actor, composed into executed actions, scored by the target critic, and averaged before the ensemble reduction, giving the target $r + \gamma\,\mathbb{E}_K[Q']$ with a sparse reward and an MSE loss. No entropy term is added to the backup.

Actor update. The actor maximizes the mean of the critic over $K = 16$ action samples (a deterministic-policy-gradient-style objective with no entropy bonus), regularized by a *filtered* zero-residual penalty $\lambda\,\alpha^2\|\delta\|^2$. This anchor pulls the residual toward zero (i.e. toward the base action) except on transitions where the current action already improves on the base ($Q(s, a_{\text{cur}}) \geq Q(s, \mathbf{a}^{\text{base}})$) and is calibrated to not be incorrectly overestimating the true Q value with ($Q(s, a_{\text{cur}}) - \text{MC} \leq \epsilon$).

Data and warmup. We run in demo-buffer mode: the success buffer is seeded from the single expert demonstration and frozen as an offline buffer, while the main buffer becomes online-only after a demo BC warmup phase (UTD 10). Each minibatch mixes offline and online data, with the offline fraction decayed linearly from 0.5 to 0.1 over the first 500 gradient steps. Online updates begin after 500 chunked environment steps, and each gradient step applies 20 critic updates and 10 actor updates per chunked environment step.

Table 20: DICE-RL hyperparameters.

Hyperparameter	**Value**
Batch size	64
Discount factor (γ)	0.999
Actor learning rate	10^{-4}
Critic learning rate	3×10^{-4}
Hidden dimensions	512
Critic ensemble size	10
Critic reduction	mean
Target update rate (τ)	0.05
Optimizer	Adam
Max grad norm	1.0
Action chunk length (C)	10
Residual scale (α)	0.5
Gradient steps per env step (UTD)	1
Critic updates per step	20
Actor updates per step	10
K action samples	16
Zero-residual penalty coeff. (λ)	1.0
Filter threshold (ϵ)	-0.5
Online update start	500 steps
Offline buffer ratio (start $\rightarrow$ end)	$0.5 \rightarrow 0.1$
Ratio decay horizon	500 steps
Success buffer min. size	100
Demo BC warmup UTD	10
Number of demos	1
Conditioning	$\pi_{0.5}$ VLM embedding

K UMAP VISUALIZATION DETAILS

Given the demerits of UMAP based visualization, we also compute the distance between various action samples in the robot's original chunked action space.

Method	#Samples	Min. Dist. to Base Cloud
Base	33	0.0142 (within cloud)
FBC	33	0.1122
DSRL	33	0.0244
MiDAS	33	4.5371

Table 21: **Minimum Distance of action clouds of various methods from the base action cloud.**

In Table 21, we see that the L_2 distance of actions from π^1_{base} of MIDAS is substantially larger compared to that of DSRL and Filtered-BC, validating that the UMAP visualization in Fig. 5 remains faithful even in the high-dimensional action space.. These observations corroborate the hypothesis that the successful action does not lie within the base policy's effective support; sharpening or steering within this support cannot reach it. Breaking out of this support, therefore, requires online adaptation that can expand the action distribution beyond the effective support of π^1_{base} and MIDAS is able to do so by learning value-guided corrections.

L SHARED PRINCIPLES BETWEEN MIDAS AND DICE-RL

MiDAS and concurrent DICE-RL (Sun and Song, 2026) share several underlying principles for stable online adaptation of pretrained policies. Both retain a behavior-cloned policy and its observation encoder as frozen components, and learn lightweight action-space corrections using chunk-level critics. This reuses previously learned behavioral and perceptual structure while concentrating online learning on corrective control. Both also balance successful demonstration data with newly collected experience. MiDAS oversamples a success buffer initialized with demonstrations and expanded with successful online rollouts; DICE-RL mixes offline demonstrations with online replay and gradually reduces the demonstration fraction during an initial training period. These distinct replay strategies reflect a shared principle of retaining useful demonstration data while learning from autonomous experience, which is particularly relevant in our setting where initial success is rare. Finally, both methods balance behavioral anchoring with the freedom to learn corrections: MiDAS initializes its actor to reproduce base proposals before applying PA-RL updates, while DICE-RL selectively relaxes its residual regularization when value estimates and a Monte Carlo consistency check support improvement over the base action. These commonalities connect the two methods at the level of frozen representations, value-guided action correction, and careful use of demonstration data. Their strong performance in our one-demonstration experiments provides complementary evidence for the effectiveness of this combination in minimal-data adaptation.

M REAL WORLD IMPLEMENTATION DETAILS

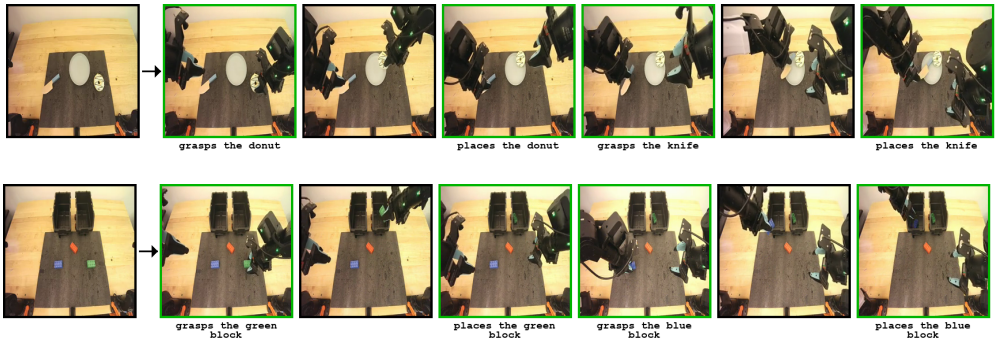


Figure 13: **Representative real-world rollouts.** Rollouts illustrating task progress on the knife-and-donut task (top) and the block placement task (bottom).

For our real world experiments, we use a bimanual YAM platform and evaluate MIDAS on two tasks: **(T1)** placing a green block in the right container and a blue block in the left container, and **(T2)** placing a knife and a donut on a plate. We fine-tune the $\pi_{0.5}$ base model for 20,000 gradient

steps using LoRA adapters for both the vision-language-model (VLM) backbone and the action head. The key differences from the simulation setup (Appendix F.2) are described below.

Observation space. The robot receives RGB images from three cameras: a top-down view and two wrist-mounted cameras (left and right). The state observation consists of the joint positions and gripper states of both arms.

Action space. Actions are predicted as chunks of horizon C=60, longer than the simulation horizon of 10 to accommodate the lower control frequency and extended task durations on hardware. Joint position targets use delta actions while gripper commands are absolute, matching the bimanual action space structure (6 joints + 1 gripper per arm).

Trainable parameters. Unlike the simulation configuration where the Gemma 300M action head is fully fine-tuned (Appendix F.2), the real-world setup applies LoRA adapters to both the PaliGemma 2B vision-language-model (VLM) backbone and the Gemma 300M action head. This reduces the number of trainable parameters and limits overfitting to the single available demonstration per task.

Optimization. The learning rate follows a cosine decay schedule with 1,000 warmup steps, a peak learning rate of 2.5×10^{-5}, and a minimum learning rate of 2.5×10^{-6} reached after 10,000 decay steps. EMA averaging is disabled. Table 22 lists all hyperparameters.

Training time and rollouts collected. We train our models with MIDAS for 5-6 hours, during the course of which we collect 350 trajectories for each task.

Table 22: Real-world behavior cloning hyperparameters for Stage I finetuning on the YAM platform.

Hyperparameter	**Value**
Number of demonstrations (K)	1
Training steps	20,000
Peak learning rate	2.5×10^{-5}
LR schedule	Cosine decay
LR warmup steps	1,000
Minimum learning rate	2.5×10^{-6}
LR decay steps	10,000
EMA decay	None
PaliGemma 2B backbone	LoRA
Action expert (Gemma 300M)	LoRA
Action horizon (C)	60
Action type	Delta (joints), absolute (grippers)
Cameras	Top, left wrist, right wrist
Checkpoint save interval	5,000

Stage II: Online residual RL. For Stage II, the residual actor-critic is trained on top of the frozen BC-fine-tuned policy using PARL (Appendix F.1). The architecture follows the simulation setup with adjustments for the bimanual action space (d_a=14, comprising 6 joints and 1 gripper per arm) and the longer action horizon. The actor and critic are 4-layer MLPs with 1024 hidden units per layer, wider than the 3-layer 512-unit networks used in simulation. The residual policy is queried every 30 steps within each 60-step base action chunk after which is new chunk is sampled.

Table 23: **One demonstration real-world adaptation results.** Success rates (%) are shown for BC and MIDAS computed over 15 rollouts varying the position and orientation of the target objects in a mix of ID/OOD settings.

	1 Demo	
Task	BC	**MIDAS**
Real World		
Placing a green block in the right container and a blue block in the left container	40.0	**67.0**
Placing a knife and a donut on a plate	27.0	**80.0**
Average	33.5	**73.5**

Several PARL hyperparameters are scaled relative to the simulation configuration (Table 11). The PA-RL procedure samples 32 candidates and retains 10 elites, and the critic and actor receive 40 and 20 updates per chunked environment step, respectively. A per-joint trust-region constraint of 0.2 rad limits the distance between refined actions and the base policy proposal for non-gripper dimensions; grippers are uncapped. Table 24 lists all Stage II hyperparameters.

Table 24: Real-world PARL hyperparameters for Stage II online RL on the YAM platform. Entries that differ from the simulation defaults (Table 11) are marked with †.

Hyperparameter	**Value**
Batch size (B)†	128
Discount factor (γ)†	0.9999
Target network update rate (τ)	0.05
Actor learning rate (α_π)	10^{-4}
Critic learning rate	3×10^{-4}
Hidden dimensions†	$(1024, 1024, 1024, 1024)$
VLM embedding bottleneck dim	200
Number of Q-networks (J)	10
Critic ensemble reduction	mean
PA-RL candidate samples (N)†	32
PA-RL elite samples (M)†	10
PA-RL gradient steps (G)	30
PA-RL step size (η)	10^{-3}
PA-RL trust-region radius†	0.2 rad (joints), uncapped (grippers)
Critic updates per chunked environment step (U_C)†	40
Actor updates per chunked environment step (U_A)†	20
Warmup chunked environment steps	10,000
BC warmup critic/actor updates per chunked environment step ($U_C^{\mathrm{w}}, U_A^{\mathrm{w}}$)	8
Success buffer ratio (ρ_{succ})†	0.20
Success buffer minimum size†	100
Gradient norm clip	1.0
Optimizer	Adam
Action chunk length (C, H)	60
Residual query frequency	30
Action dimension (d_a)†	14

Details on evaluation and resets.: For both online RL training and evaluations, the target objects are reset around the demonstration's target locations. We do two classes of evaluations: in-distribution, where the target objects are perturbed very mildly from their demonstration positions/orientations; out-of-distribution with larger perturbations. Videos of rollouts and training is available here: coruscating-pika-920eda.netlify.app.

N FUTURE WORK

Future work should study how to make exploration more effective. Our results show that a single demonstration can anchor exploration around a narrow tube of task-relevant behavior, within which autonomous RL can improve reliably. However, learning qualitatively new behaviors or reaching distinct behavioral modes remains substantially harder. Expanding the reset distribution provides one mechanism for broadening exploration, but our curriculum results also show that performance degrades as this distribution widens, and that reset expansion cannot recover behaviors that lie outside the support of the initial policy. This suggests that autonomous improvement will require multiple forms of supervision to work together, including demonstrations, reset curricula, language-based steering, and reward signals. Future work can characterize how these sources can be combined to discover new behavior while balancing human effort, computational cost, interaction efficiency, and final-policy robustness.

Second, MDA provides a setting for understanding what demonstrations are actually needed for. Given one demonstrated behavior, online RL can robustify and extend that behavior over a surrounding region of the task distribution, but there is an upper bound to what can be recovered from this initialization alone. This raises the question of when additional demonstrations become necessary: whether one demonstration is required for each distinct behavioral mode, what constitutes a mode in robot control, and which variations RL can generalize across without further task-specific supervision. The answer may also depend on how the demonstration is incorporated. We use behavior cloning followed by a memory-efficient residual policy, which may not extract all of the behavior that one trajectory makes available. More expressive adaptation, including larger task-specific modules or full-policy fine-tuning, may make better use of the same demonstration, but introduces different tradeoffs in compute, memory, stability, and sample efficiency.

Third, it remains important to determine how broadly the MDA regime extends beyond the behaviors studied here. Our experiments primarily consider pick-and-place tasks that are plausibly well represented in the pretraining distribution of $\pi_{0.5}$ (and other generalist policies of today). It is therefore unclear whether the same interaction between pretraining, minimal demonstrations, and autonomous RL will hold when adaptation requires greater policy capacity or more substantial behavioral change. Future work should test this regime across new embodiments and grippers, dexterous and precision-critical manipulation, assembly, tool use, and longer-horizon tasks. These settings can reveal whether the present findings reflect a general property of pretrained robot policies or a regime that is currently limited to behaviors already well supported by pretraining.

Finally, MDA exposes a distinction between memorizing task-specific behavior and generalizing across deployment variation. Our results suggest that different components of the pipeline support different forms of robustness: pretrained visual representations provide invariance to some observation-level changes, while demonstrations and online interaction determine which task-relevant states and behaviors the policy can reliably handle. Autonomous interaction therefore does not imply broader generalization by itself; the resulting robustness depends on the distribution of states encountered and the behaviors already available to the policy. Future work should characterize which forms of generalization arise from pretraining, which must be induced through task-specific demonstrations or online interaction, and how these components should be balanced to obtain policies that both adapt efficiently and generalize beyond the conditions encountered during adaptation.