ABR^2: ADVANTAGE-BASED REGULARIZATION FOR ONLINE POLICY REPAIR

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-Language-Action (VLA) models are popular for zero-shot manipulation across diverse tasks, but their effectiveness is conditioned on exposure during training to **both** the deployment embodiment **and** the specific task domain. We observe that embodiment-grounded VLA policies frequently exhibit *partial competence* in unseen tasks, even in those closely related to tasks seen in training. These policies complete a task in some non-zero proportion of attempts but frequently make non-recoverable errors, preventing zero-shot deployment of the policy. To address this we propose *Advantage-based Regularization for Online Policy Repair* (ABR^2), a method that learns to correct the critical errors that lead to task failure when deploying VLA policies in unseen tasks. Our method requires neither expert demonstrations, reward shaping, nor VLA finetuning. ABR^2 bootstraps a residual action correction solely from the rollouts of a frozen base policy. An advantage regularizer grounds a critic on expected returns from the base policy before finding local corrective actions that would improve upon executing the VLA policy directly. We evaluate the ability of ABR^2 to repair policies on both small scale goal conditioned control environments and VLA policies in unseen manipulation and decision making tasks. Across five held-out LIBERO-90 tasks with 21.8–54.7% base success, ABR^2 improves final success to 55.6–96.3%, outperforming residual RL without advantage regularization on every task. ABR^2 excels at correcting acute critical errors that lead to policy failure, increases the sample efficiency of policy repair, and accommodates multi-step corrections for perturbed configurations of tasks seen during training.

1 INTRODUCTION

Vision-Language-Action (VLA) models are popular for learning to perform diverse manipulation tasks in a zero-shot manner (Brohan et al., 2023b;a; Ghosh et al., 2024; O'Neill et al., 2024). However, this ability requires sufficient exposure in training to **both** the particular deployment embodiment and task domain. Critically, satisfying both conditions is prohibitively expensive for general-purpose deployments. In particular, it is difficult to fully capture all potential tasks and task domains in training data. Prior work evaluating VLAs (Zhou et al., 2025), such as $\pi_{0.5}$ (Black et al., 2025), shows that despite task domain specific finetuning, competency does not transfer to tasks with different prompts or different environment configurations from the same domain (Table 2, 3).

We observe that embodiment finetuned VLAs frequently exhibit *partial competence* in these cases — when deployed on a related but unseen task they are able to complete it in some non-zero proportion of attempts, but frequently make unrecoverable errors that prevent deployment of the policy. Our key insight is that interaction data from a partially competent policy are valuable as the policy may visit task-relevant states, make contact with objects, and succeed occasionally. By leveraging these data, minor corrections to the policy increase its reliability enough to bootstrap a better policy without demonstrations or interventions.

This observation re-frames the role of the VLA or other base policies within a generalist robot system. An ideal data regime for general-purpose robotics now combines large-scale, embodiment-agnostic pretraining with a limited amount of embodiment-specific data across a collection of simple tasks. Downstream complex tasks become learnable online without additional supervision through policy repair guided by partially competent interaction data. The role of the VLA is no longer as a

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.

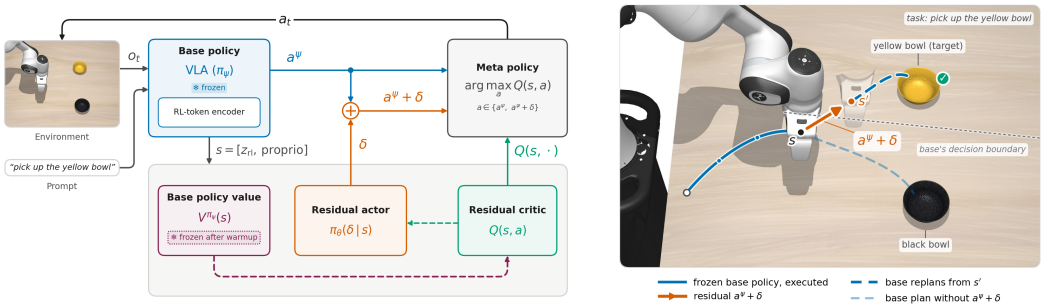


Figure 1: **(a)** System diagram for ABR^2. The base policy produces a candidate action a^ψ, then the action and its intermediate context is compressed using an encoder (Xu et al., 2026) and appended to proprioception before being fed to the residual policy. A value function trained on returns of the base policy regularizes the training of the critic and a residual actor trains against the critic to produce a correction δ that maximizes expected return. The critic is then used to decide between the VLA's candidate action and the corrected action. **(b)** An illustration of the corrections ABR^2 produces when a base policy is given a previously unseen task. The base policy makes a critical error. When prompted to pick up the yellow bowl, it picks up the black one. ABR^2 corrects the policy around this critical point so that, when the VLA replans, it moves to the yellow bowl.

universal zero-shot policy, but rather as (1) a strong warm start for learning new capabilities online, and (2) a regulator for unsafe or unproductive behavior during task learning.

When learning without labeled data, many approaches rely on reinforcement learning (RL) or behavior cloning BC. While approaches for sample-efficient RL exist (Haarnoja et al., 2018; Fujimoto et al., 2018), many utilize some form of distillation from expert data, most commonly human demonstrations (Nakamoto et al., 2023). These human demonstrations are particularly expensive to collect, making this approach difficult to scale. Due to the parameter size of VLAs, the computational and sample complexity required for improving the VLA directly online makes it prohibitively expensive and potentially dangerous for real world deployments, potentially corrupting the policy. Distilling or using behavior cloning to compress a base policy π_ψ to a more compact, easier to train representation, as in Jülg et al. (2025), struggles when the base policy π_ψ is partially competent. The policy π_θ initially approximates π_ψ, but in order to improve must diverge from π_ψ's failure modes. Prior work (Nikishin et al., 2022; Dohare et al., 2023; Lyle et al., 2023; Wang et al., 2023) identifies that this specific class of distribution shift harms the sample efficiency of policy learning as during behavior cloning, distillation, or other forms of offline training, the actor initially overfits to early data.

To avoid the limitations of behavior cloning using a weak policy prior, we instead focus on value learning where we can leverage the valuable interaction data generated by observing the base policy's trajectories. Using a residual formulation (Silver et al., 2018; Johannink et al., 2019; Xu et al., 2026; Ma et al., 2026) we train a critic based on the rollouts of π_ψ. However, unlike prior work, we do not use any sort of finetuning of the base policy, reward shaping, or human annotations in training the residual policy. We focus specifically on policy correction, repairing the specific failures that lead to task failure in unseen tasks while allowing the base policy to use its pretraining for covered sub-trajectories. Through the use of an advantage regularization term in the critic loss, the critic weighs proposed corrections against the expected returns of the base policy. This both better grounds proposed corrections as well as provides a mechanism to automatically hand off between the base policy and the residual policy over the course of online learning.

Our contributions are (1) **Base-policy value grounding as an objective.** We introduce an auxiliary loss term to the residual actor's critic objective to calibrate the critic against the expected performance of the base policy. This component improves success rate in tasks where the primary failure mode of the base policy is decision error. (2) **An evaluation of residual policy correction under suboptimal guidance**, showing that a base policy too unreliable to deploy zero-shot can be used to quickly bootstrap a policy in unseen tasks without demonstrations, reward shaping, or updates to π_ψ (3) **An analysis of the conditions where advantage regularization improves policy repair**. We identify base policy failure modes for which the advantage regularization term is most impactful.

2 Related Work

We study online adaptation when the only initial source of task-relevant experience is a pretrained policy that is itself only partially competent on the new task. We assume no expert demonstrations or human intervention, keep the policy frozen, and learn a lightweight correction policy that selectively improves upon its actions. Prior work examines aspects of this setting.

Vision-Language-Action Models Large-scale VLA models (Brohan et al., 2023b;a; Ghosh et al., 2024; O'Neill et al., 2024) have demonstrated impressive zero-shot generalization across diverse manipulation tasks by training on internet-scale data. Models like π_0 (Black et al., 2025) and OpenVLA (Kim et al., 2024) have further advanced robustness and controllability. However, robust zero-shot performance is conditioned on training data coverage: finetuning a VLA on a task set (e.g., LIBERO (Liu et al., 2023)) can saturate tasks seen in training while achieving only $\sim$20% success-rate (SR) on closely related, yet subtly different, tasks. While online finetuning of the base policy via reinforcement learning is the most direct approach to improving policy competency, the scale of VLAs means it is computationally intractable to do so for every task failure. Imitation learning or behavior cloning approaches can be more efficient but require demonstration of the task from an expert, incurring additional cost for each new task.

Offline and Offline to Online RL Offline RL methods (Nakamoto et al., 2023) and offline-to-online transitions (Nair et al., 2020; Zhou et al., 2024) have addressed the problem of using fixed datasets for policy improvement. Prior work has explored pretraining value representations from passive or cross-embodiment data before downstream RL (Bhateja et al., 2023; Ghosh et al., 2023), bootstrapping learning from diverse robot embodiments (Bhateja et al., 2023; Ghosh et al., 2023) or expert demonstrations (Wu et al., 2020). These approaches assume access to a pre-existing offline dataset that is sufficiently informative for the target task. In contrast, we assume that no task-specific offline dataset exists: the only source of task-relevant experience is the partially competent base policy itself. Other work has explicitly considered RL with suboptimal prior datasets (Nair et al., 2020). In policy repair, the offline data source is a generalist policy with sub-tasks capabilities such as failure recovery or regrasping not necessarily demonstrated in data collection. The base policy also regulates exploration, masking unproductive behavior. Access to these beneficial properties are not guaranteed to be preserved in offline to online RL.

Learning from Suboptimal Behavior and Value Learning To avoid the cost of finetuning a large VLA while using the base policy as a warm start in an unseen task, one may consider distilling or behavior cloning the base policy itself into a more parameter efficient form (Agarwal et al., 2022; Jülg et al., 2025) before refining with RL. This is appropriate when the offline data is high quality, where following the behavior of the demonstrator is likely to succeed without requiring extensive exploration. However, when a demonstrator is not an expert, e.g. at 20–60% SR, its rollouts include failure-causing actions that behavior cloning may reproduce. Behavior cloning initializes $\pi_\theta \approx \pi_\psi$ thereby incentivizing the selection of these failure-mode actions. Then, to improve beyond π_ψ, π_θ must diverge from the base policy, against the behavior cloning prior. Prior work (Nikishin et al., 2022; Dohare et al., 2023; Lyle et al., 2023; Wang et al., 2023) identifies that the actor initially overfits to early data, making transfer between the base policy's distribution and an effective policy's distribution difficult.

This motivates isolating the actor from directly replicating the policy of a suboptimal demonstrator while still extracting task relevant information from demonstrations, even partially competent ones. One method to do so is via value learning, using the offline demonstrations to bootstrap a critic which can guide the actor. Ma et al. (2026) examine the role of critic pretraining and normalization in residual reinforcement learning. They identify seeding base-policy transitions in the replay buffer as effective for addressing the critic's cold-start problem. They also find the use of normalization in the critic restores sensitivity for distinguishing relative values. We adopt these ideas in our approach, but additionally fit a value function estimating the return of the base policy, which is included in our objective for the critic to explicitly calibrate the critic against the expected returns of the base policy.

Residual RL and Online Adaptation of Pretrained VLAs Residual reinforcement learning freezes the base policy and learns an additive correction $\pi_\theta(\delta|s)$. For each produced action from the base policy $\pi_\psi(s) = a^\psi$, the policy executes $a = a^\psi + \delta$ (Silver et al., 2018; Johannink et al.,

2019; Yuan et al., 2025). Since $\pi_\theta(s) = 0$ recovers the base policy exactly, its behavior is available from initialization. Prior applications of this formulation include some sort of base policy finetuning as a precursor step to establish the base policy in an unseen task: Xu et al. (2026) adapt the base VLA on task demonstrations before freezing it, Ma et al. (2026) refine imitation policies trained on the same tasks they subsequently improve.

Xu et al. (2026) introduce the RL-token, an encoder which compresses the internal state of a VLA into a fixed sized vector. The vector and the VLA action are provided to a lightweight actor-critic policy that predicts refined action chunks (Li et al., 2025). However, Xu et al. (2026)'s adaptation protocol assumes one to ten hours of teleoperated demonstrations per task, finetuning of the VLA on them, human success labels as reward, and a human operator who selects the point at which control passes from base policy to residual. FORCE (Zhang et al., 2026) likewise finetunes the VLA and incorporates expert data during policy improvement, using value-calibrated warm-up and Q-based filtering to stabilize online finetuning. We adopt the RL-Token encoder and action chunking residual policy in our large scale experiments. However, we assume no access to offline data or human intervention in our approach. Additionally, we do not train the RL-Token encoder on deployment tasks as Xu et al. (2026) does, instead keeping it frozen during deployment. We collect task specific data online under the partially competent base policy and our method uses the learned critic to transition the agent onto the improved policy automatically rather than via operator intervention.

3 Advantage Based Regularization for Online Policy Repair

3.1 Problem Setting

Setup We are given (1) a base policy π_ψ that has been pretrained and then finetuned on a corpus of tasks $\mathcal{D}_p$, and (2) a *target task drawn from outside that corpus*, $t \in \mathcal{D}_d$ s.t. $\mathcal{D}_p \cap \mathcal{D}_d = \emptyset$ on which π_ψ is only partially competent. We assume (1) no external demonstrations of the target task, (2) the base policy is frozen, (3) task success is indicated with a reward signal, and (4) online interaction is the only source of task-specific data. We consider a base policy to be *partially competent* when it succeeds enough to be informative, but too rarely to be deployed as is. For the purposes of this work, we define this range as $\sim 20\% - 60\%$ per task. Our goal is to achieve task competency in a sample efficient manner using guidance from a partially competent π_ψ.

Formalization We formalize each task as an MDP $(\mathcal{S}, \mathcal{A}, P, r)$ with state space $\mathcal{S}$, action space $\mathcal{A}$, transition function P, and reward function r in which reward is 1 on a successful terminal transition and 0 elsewhere. The residual policy observes s_t. Transitions $(s_t, a_t^\psi, a_t, r_t, s_{t+1}, a_{t+1}^\psi, d_t)$ are stored in a replay buffer $\mathcal{B}$, where $a_t^\psi = \pi_\psi(s_t)$ is the base policy's proposal, a_t the executed action, and a_{t+1}^ψ the base policy's action at the successor state.

The residual formulation Rather than learning a policy over $\mathcal{A}$, we learn a correction to the base policy's proposal:

$$a_t^\psi = \pi_\psi(s_t), \qquad \mathcal{A} \subseteq \mathbb{R}^{d_a}, \qquad \pi_\theta : \mathcal{S} \to \mathbb{R}^{d_a}, \qquad \tilde{a}_t \;=\; a_t^\psi + \pi_\theta(s_t), \tag{1}$$

with the usual objective $\max_\theta \mathbb{E}\big[\mathbb{1}[\text{success}]\big]$.

3.2 Method

At each decision point the base policy proposes a_t^ψ, the residual actor proposes a correction $\pi_\theta(s_t)$, the critic scores both a_t^ψ and $\tilde{a}_t$, and the higher-valued proposal executes. Each correction is framed as local *both* in the action space and time. $\pi_\theta = 0$ recovers the base policy, so the residual actor starts from the base policy's competency level and regularizing the actor constrains corrections to be local in the action space. Similarly, because the target bootstraps from a_{t+1}^ψ, the corresponding Bellman fixed point is Q^{π_ψ}, i.e., the value of an action followed by deferral to π_ψ.

3.2.1 Critic Training

We train twin critics Q_{ϕ_1}, Q_{ϕ_2} with TD3 (Fujimoto et al., 2018) with prioritized replay (Schaul et al., 2016), making one substitution: the target bootstraps off the base policy's next action a_{t+1}^ψ

rather than the residual actor's next action. For brevity, we write $Q_\phi(s,a) = \min_{i\in\{1,2\}} Q_{\phi_i}(s,a)$ when scoring actions. Therefore,

$$y_t = r_t + \gamma\,(1-d_t)Q_\phi(s_{t+1}, a^\psi_{t+1}). \tag{2}$$

The Advantage Regularizer With a sub-optimal base policy, the role of the critic, already central in residual learning, is amplified. Whether a correction is worth making at s is determined by

$$A_\phi(s,\tilde{a}) = Q_\phi(s,\tilde{a}) - Q_\phi(s,a^\psi). \tag{3}$$

Writing the critic as $Q_\phi = Q^{\pi_\psi} + \varepsilon$, with ε the approximation error, and substituting into $A_\phi(s,\tilde{a}) = Q_\phi(s,\tilde{a}) - Q_\phi(s,a^\psi)$ gives equation 4, where $A^{\pi_\psi}(s,\tilde{a}) = \big[Q^{\pi_\psi}(s,\tilde{a}) - Q^{\pi_\psi}(s,a^\psi)\big]$ is the idealized advantage estimator:

$$A_\phi(s,\tilde{a}) = A^{\pi_\psi}(s,\tilde{a}) + \big[\varepsilon(s,\tilde{a}) - \varepsilon(s,a^\psi)\big]. \tag{4}$$

The decision therefore depends on the difference between the critic's error at the two actions, not only the magnitude of the error on expectation across the Q-function. Reducing overall critic error is therefore insufficient to guarantee correct action selection, since the decision depends on the difference in approximation error between the two candidate actions. Because behavior cloning regularization keeps $\tilde{a}_t$ close to a^ψ_t, the true action gap may be small, allowing even modest differential error to invert their ordering (Farahmand, 2011).

Ma et al. (2026) name equation 5 the *Q-value grounding error*,

$$E_{\text{grounding}} = \big|Q_\phi(s,a^\psi) - G^{\pi_\psi}\big|, \tag{5}$$

with G^{π_ψ} as the Monte-Carlo return. Since $\mathbb{E}[G^{\pi_\psi} \mid s, a^\psi] = Q^{\pi_\psi}(s,a^\psi)$, this provides a measure of the approximation error $\varepsilon(s,a^\psi)$. Ma et al. (2026) reduce it by seeding the replay buffer with base-policy transitions. This seeding places a^ψ in the replay distribution.

While Ma et al. (2026) use equation 5 only as a measurement of the quality of the replay buffer, we use it as part of the objective for training the critic. We separately fit a base-policy value function, $V_\mu^{\pi_\psi} \approx V^{\pi_\psi}$, by Monte-Carlo regression on realized returns by observing the behavior of π_ψ on the target task for a fixed number of episodes. $V_\mu^{\pi_\psi}$ is added as a grounding regularizer to the objective of the critic,

$$\mathcal{L}_{Q_i} = \mathbb{E}_{\mathcal{B}}\big[w_t\,(Q_{\phi_i}(s_t,a_t) - y_t)^2\big] + \lambda_v\,\mathbb{E}_{\mathcal{B}}\big[(Q_{\phi_i}(s_t,a^\psi_t) - V_\mu^{\pi_\psi}(s_t))^2\big], \tag{6}$$

where w_t denotes the prioritized-replay importance weights (Schaul et al., 2016), and λ_v weights the grounding regularizer. $V_\mu^{\pi_\psi}$ is fit during an initial base-policy data-collection phase and frozen before residual actor training begins. $V_\mu^{\pi_\psi}$ serves the role of anchoring $Q_\phi(s,a^\psi)$ to the base policy's expected returns from s. The critic's target uses the base policy's next action, so $Q_\phi(s,a)$ approximates the value of taking a at s and following the base policy thereafter, while $V^{\pi_\psi}(s)$ is the expected value of letting the base policy act.

$V^{\pi_\psi}(s)$ is particularly relevant in a suboptimal guidance setting. With binary terminal rewards, this value primarily reflects successful continuation under π_ψ. For a near-perfect base policy, it therefore varies relatively little in terms of success likelihood, whereas for a partially competent policy it distinguishes between states where the base policy is likely to succeed or fail. Any proposed correction is then scored against the value of deferring to the base policy with no correction.

3.2.2 Actor And Meta Policy

The actor π_θ maximizes Q_ϕ under a behavior-cloning regularizer (Xu et al., 2026) weighted by β,

$$\max_\theta\ \mathbb{E}_{\mathcal{B}}\Big[Q_\phi\big(s_t, a^\psi_t + \pi_\theta(s_t)\big)\Big] - \beta\,\mathbb{E}_{\mathcal{B}}\Big[\tfrac{1}{d_a}\|\pi_\theta(s_t)\|_2^2\Big]. \tag{7}$$

Since Q_ϕ estimates Q^{π_ψ}, equation 8 implements an approximate one-step policy-improvement operator over the candidate actions $\{a^\psi_t,\ \tilde{a}_t\}$

$$\hat{\pi}(s_t) = \arg\max_{a\in\{a^\psi_t,\ \tilde{a}_t\}} Q_\phi(s_t,a). \tag{8}$$

This allows for the handover between base and residual to be decided by the critic on a state-by-state basis rather than by an external operator, such as in prior work. Q_ϕ and π_θ continue updating online on executed transitions. As residual learning progresses, the meta policy can increasingly select corrected actions in states where the critic estimates them to achieve higher returns than the base policy, while retaining a^ψ where the base policy's action is strong.

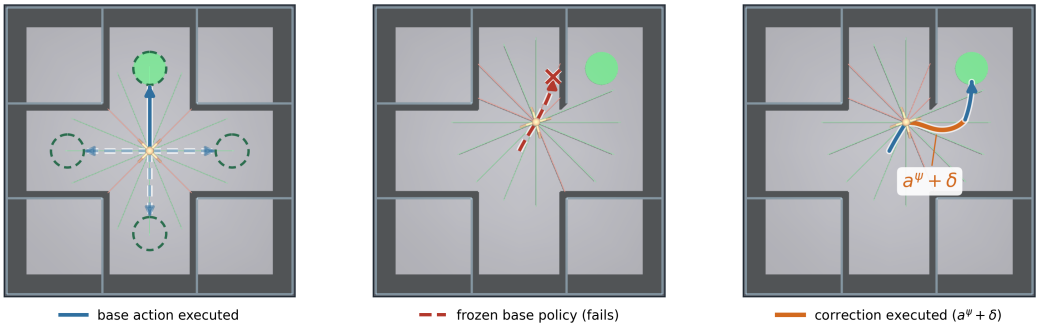
4 EXPERIMENTS

4.1 REPAIRING A SIMPLE CONTROL POLICY ONLINE

Our approach targets deficiencies that can be repaired by a small, state-dependent correction to the base policy's action, rather than by replacing the policy. In this section, we examine this by applying ABR^2 to repair a small goal conditioned control policy when given a previously unseen goal.

Experimental Setup Using the DeepMind Control Suite (Tunyasuvunakool et al., 2020) we create a pair of environments: *Grid Plus* (Figure 2a) where goals are sampled from the four cardinal directions and *Grid Corner* (Figure 2c) where goals are sampled from one of the corners. The observation is $o_t = [\text{proprio}_t, \text{lidar}_t, \text{goal}_t]$, a 171-dimensional vector of 153 proprioceptive values, 16 lidar returns, and the goal position, and the actions a_t are 8 joint torques in $[-1, 1]$. The critic Q_ϕ, base value function $V_\mu^{\pi_\psi}$, and residual actor π_θ are all two-layer MLPs. We train a goal-conditioned base policy π_ψ with Soft Actor-Critic (Haarnoja et al., 2018) on *Grid Plus* that succeeds on *Grid Corner* 33%–75% of the time (Table 1).

Online training runs for 1500 episodes split in two phases. In the warmup phase (500 episodes) the frozen π_ψ acts while Q_ϕ, and $V_\mu^{\pi_\psi}$ train on the rollouts of π_ψ. In the deployment phase $V_\mu^{\pi_\psi}$ is frozen and the meta policy is enabled, scoring $\{a_t^\psi, \tilde{a}_t\}$ at each step and executing the higher-valued action. Q_ϕ and π_θ then update for the remainder of the episode budget. The environment reward is sparse, returning 1 when the goal position is achieved, 0 otherwise. We compare against three baselines: the residual policy without the advantage objective, online finetuning of π_ψ, and standard online Soft Actor-Critic. Full environment, architecture, and optimization details are given in Appendix B.



(a) Grid Plus base policy training

(b) Grid Corner [Up + Right] frozen base policy fails

(c) Grid Corner [Up + Right] repaired by the residual

Figure 2: Grid Plus is an environment where the base policy is trained using a uniform sampling from the cardinal directions. Grid Corner is then used to train the residual and evaluate performance.

Table 1: Success rate on *Grid Corner*, mean over 100 evaluation episodes across 5 seeds with standard deviation. We ablate components in each column to isolate contributions: the residual actor alone editing each action, then under meta-policy selection, then the residual trained using advantage regularization using the meta-policy to select actions.

GOAL	FROZEN BASE	RESIDUAL	+ ADV REG	+ META	+ META & ADV REG
Up + Right	54.0%	51.8 ± 4.1	63.2 ± 2.6	63.6 ± 2.6	**71.4 ± 4.4**
Up + Left	33.0%	27.6 ± 1.1	31.6 ± 1.5	34.0 ± 3.3	33.8 ± 3.6
Down + Right	75.0%	74.4 ± 2.2	79.2 ± 3.3	75.4 ± 5.1	**81.2 ± 1.9**
Down + Left	55.0%	60.8 ± 2.3	62.6 ± 3.0	61.2 ± 2.8	61.2 ± 3.8

Results We ablate each component of ABR^2 in repairing π_ψ. We find that the standard residual policy alone is not sufficient to learn competent policies on Grid Corner Tasks (Table 1). Adding the advantage regularizer can improve performance over residual repair. The combination of the advantage regularizer and the meta policy results in the best performance. The meta policy alone can exceed the performance of standalone residual repair as well. Post training π_ψ leads to collapse of the policy due to transferring between the dense and sparse reward settings of *Grid Plus* and *Grid Corner* respectively. Under sparse reward, online learning from scratch is not effective. Even when provided dense reward (same as Grid Plus), ABR^2 still exceeds the performance of online learning from scratch: 43% in *Up + Right* vs 71.4% for ABR^2 in a 1500 episode budget. (See Appendix G).

4.2 Repairing Pre-Trained VLA Policies Online

Experimental Setup We use $\pi_{0.5}$-LIBERO (Black et al., 2025) as the base policy π_ψ. This model has been finetuned on subsets of the LIBERO suite (Liu et al., 2023), in particular *LIBERO Spatial, LIBERO Object, LIBERO Goal*, and *LIBERO-10*. Notably, tasks from *LIBERO-90* are unseen tasks for π_ψ. An RL-token encoder (Xu et al., 2026) is trained to compress the intermediate state of $\pi_{0.5}$ to a fixed sized representation, z^{rl}. In contrast to Xu et al. (2026), **the encoder is pretrained on the same LIBERO task corpus used to finetune** $\pi_{0.5}$ and is frozen during experiments. Hence, no novel information is leaked via the context encoder. The state passed to the residual policy is $s_t = [z_t^{\text{rl}}, \text{proprio}_t]$. $\pi_{0.5}$ produces action chunks of size $K = 10$. As such the residual policy produces a vector of corrections $\delta \in \mathbb{R}^{K\times 7}$.

We parameterize the critic $Q_\phi(s, a)$ as an MLP with 2 hidden layers operating on a $[2048 + 8 + 70]$ concatenation of s_t and the base policy's action chunk $a^\psi \in \mathbb{R}^{K\times 7}$. Similarly, the value network $V_\mu^{\pi_\psi}$ is also a two-hidden layer MLP using the same $[2048 + 8 + 70]$ input vector. Finally, for the residual policy $\pi_\theta(\delta \mid s)$ network we use another two layer MLP. The network takes s_t and produces a correction for the action chunk $\delta \in \mathbb{R}^{1\times 70}$, which is added to the base action a^ψ. The meta policy then selects between a^ψ and $a^\psi + \delta$. The first half of the action chunk is executed before re-planning, corresponding with the behavior of $\pi_{0.5}$. Exact network architectures are described in Appendix B.

The environment returns sparse reward of 1 at the terminal step of a successful episode and 0 otherwise; no shaped or intermediate reward is used. Online training is structured in two phases. For the first phase, π_ψ ($\pi_{0.5}$) produces each executed action chunk and transitions are written to a replay buffer. The critic Q_ϕ is updated according to equation 6 and the value function $V_\mu^{\pi_\psi}$ is updated by regressing over Monte-Carlo returns. The residual policy is updated using equation 7 against Q_ϕ, however, these corrections are not executed. At the end of phase one the residual is deployed and $V_\mu^{\pi_\psi}$ is frozen. Thereafter, the executed action is selected at each step using equation 8 between a_t^ψ and $\tilde{a}_t$. The critic and the residual continue to train for the remainder of deployment, but $V_\mu^{\pi_\psi}$ is frozen, as new transitions are obtained under the meta policy, not the base policy. We find two relevant hyperparameters which influence training, the length of the first phase and the weight of the behavior cloning regularizer which constrains the magnitude of the correction δ. We report the best configuration for each task in our results.

4.2.1 Unseen Tasks

From the unseen LIBERO-90 task set we identify low-competency tasks, where π_ψ has a $20\% - 60\%$ success rate (Table 2). Training proceeds over total budget 1000 online episodes split in two phases. We report the best configuration of phase one length and behavior cloning regularizer weight in our results. We compare our proposed method against two baselines: training the residual policy without the advantage regularizer, and finetuning the base policy π_ψ online, reporting the final success rate over 100 episodes using 3 seeds per task and experiment configuration after 1000 episodes.

Results We examine the benefit of using ABR^2 to learn corrections for the base $\pi_{0.5}$ policy across a range of baseline success rates. All evaluations are conducted on a fixed set of 100 episodes for consistency across methods, and we find that ABR^2 achieves a higher final success rate than a residual policy without the advantage regularizer (Table 2). We see that when using the advantage regularization term the residual policy is able to correct the critical irrecoverable errors that lead to task failure when the base policy is deployed in tasks not seen in training (Figure 3). When comparing the sample efficiency to finetuning the base policy directly using a LoRA (Hu et al.,

Table 2: Success rate after a 1000 episode budget (including both replay buffer filling and online learning) on the unseen LIBERO-90 tasks, average success rate over 100 episodes following training for 3 seeds with standard deviation. $\pi_{0.5}$ is frozen throughout; the residual is the only learned policy.

TASK	FROZEN $\pi_{0.5}$	RESIDUAL	+ ADV REG (Ours)
Task 77 (Book to Caddy)	21.8%	89.7 ± 4.2	$\mathbf{96.3 \pm 0.6}$
Task 59 (Sauce to Tray)	26.6%	33.0 ± 5.9	$\mathbf{55.6 \pm 6.3}$
Task 82 (Book to Left Caddy)	36.7%	50.0 ± 3.6	$\mathbf{65.0 \pm 6.2}$
Task 14 (Middle Bowl to Plate)	41.1%	65.0 ± 10.6	$\mathbf{81.3 \pm 7.6}$
Task 60 (Left Bowl to Tray)	54.7%	86.7 ± 5.5	$\mathbf{92.3 \pm 5.1}$

2022) and online behavior cloning, we find that it takes behavior cloning twice as many episodes to achieve comparable success rate to what ABR^2 achieves in 1000 episodes (see: Appendix E).

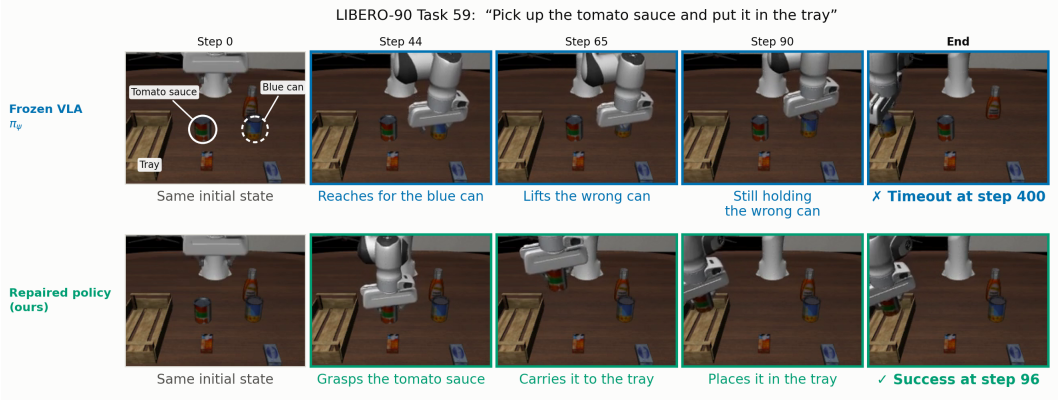


Figure 3: Comparison of the base policy π_ψ ($\pi_{0.5}$) and the corrected policy on LIBERO-90 Task 59 - *pick up the tomato sauce and put it in the tray*. Highlighted frames indicate when the policies diverge. Corrected policy avoids the incorrect can selection.

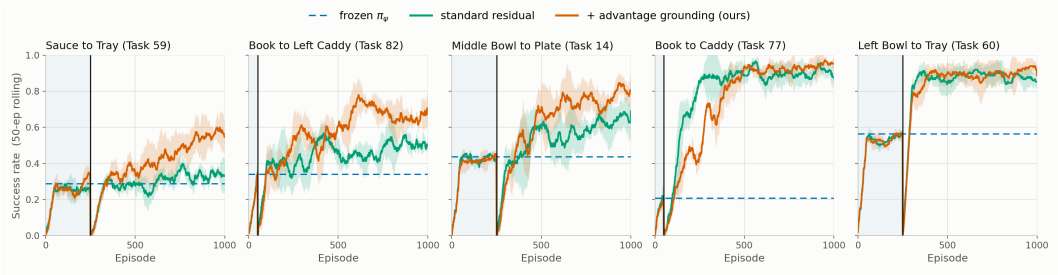


Figure 4: Comparison of policy competency as measured by rolling success rate over 50 episodes versus number of online training episodes on various LIBERO-90 tasks over 3 seeds with variance reported. The blue area denotes the warm up phase where the base policy acts, and rollouts are collected to fill the replay buffer before the residual policy controlled by the meta policy is allowed to correct the actions of the base policy. Success rates start from 0 and reset between phases to separate the effects of acting under the different policies. The blue dotted line indicates the base policy success rate on the task.

4.2.2 Seen Tasks but Unseen Configurations

We also examine the case of a task having been seen in training by π_ψ but with perturbed placement of objects in the scene. Zhou et al. (2025) provides augmentations of *LIBERO-Object* tasks, where

Table 3: Mean success rate on LIBERO-PRO with scene perturbation over 100 deployment episodes on Task 0 (*pick up the alphabet soup and place it in the basket*) and Task 8 (*pick up the chocolate pudding and place it in the basket*). Deleted objects are noted in the perturbation column. The residual penalty $\lambda_\delta = 3$ for all settings except for Task 8 + 0.28m displacement, where $\lambda_\delta = 1$.

TASK	PERTURBATION	FROZEN $\pi_{0.5}$	RESIDUAL	+ ADV REG (Ours)
T0	+0.07 m	87.6%	94.8 ± 1.9	$\mathbf{95.2 \pm 2.7}$
	+0.14 m, *milk* deleted	34.5%	$\mathbf{85.0 \pm 11.5}$	83.7 ± 4.0
T8	+0.07 m	97.3%	$\mathbf{97.6 \pm 2.3}$	97.0 ± 2.3
	+0.14 m	70.9%	$\mathbf{77.6 \pm 9.0}$	73.2 ± 5.7
	+0.28 m, *ketchup* deleted	13.3%	$\mathbf{48.3 \pm 27.8}$	16.7 ± 17.0

object placements in the scene are sampled within x meters of the position in the source task. In cases where the placement of objects is nonviable, the object may be removed. Zhou et al. (2025) show that such displacements are sufficient to disrupt π_ψ, causing the success rate on solved tasks to drop. We evaluate our method on these tasks, specifically two tasks in which the success rate of π_ψ scales predictably with the displacement of the target objects. We consider each pair of task and amount of displacement from the original task as distinct new tasks to evaluate. In this experiment we consider tasks both inside and outside the region of partial competency. We follow the same general training structure as the prior experiment (Section 4.2). However, we extend the total episode budget to 1500 and length of the warm up phase to 500 episodes to allow for enough episodes to accurately estimate $V_\mu^{\pi_\psi}$. Otherwise, the procedure remains the same using the same architectures as above for the critic, advantage anchor, and residual actor policies.

Results We do not find a significant performance difference between ABR^2 and not using the advantage regularizer (Table 3). We hypothesize this is due to policy failures stemming from continuous displacement errors instead of acute, critical errors such as choosing the wrong target object. The base policy requires continuous intervention to compensate for the displacement in the target objects, whereas ABR^2 is most effective at making selective fixes to an otherwise performant policy. As such the benefit of grounding around the expected returns of the base policy is less important. In situations where the base policy performs very poorly, such as in the most perturbed instance of *pick up the chocolate pudding and place it in the basket* (T8 with 0.28m displacement, Table 3), ABR^2 pulls performance towards this under-performing base policy, hurting final performance, as the base policy is too weak to provide a useful value reference within the episode budget.

5 Conclusion

In this paper, we motivate a new role for generalist, pretrained robot policies π_ψ such as VLAs. Rather than requiring these policies to solve every new task zero-shot, we propose using them as behavioral priors for learning new capabilities online. To this end, we present ABR^2, a method for repairing partially competent policies on unseen tasks using only their own online interaction experience. ABR^2 freezes the base policy and learns selective action corrections that improve expected return. By leveraging the generalist policy as a behavioral prior, ABR^2 improves policies with low initial success rates on unseen tasks without target-task demonstrations, human intervention, reward shaping, or modifications to π_ψ. Rather than asking generalist robot policies to solve every new task zero-shot, ABR^2 turns partial competence into a foundation for learning—preserving what the policy already knows while repairing the decisions that keep it from succeeding.

AI use statement

In this work, we used generative AI tools for managing experiment execution on our compute cluster, the writing of training/experiment code, and code to generate tables and figures. We have reviewed all AI-assisted work. All experiments are logged in an experiment tracking system including hyperparameters and links to code revisions and tested for correctness and reproducibility. Authors audited all AI generated code. AI was not used to do independent statistical analysis or mine for results, only to parse experiment data from logged experiments to construct figures. Generative AI

was used to generate feedback on drafts on structure and phrasing. Generative AI was not used to generate synthetic datasets, to develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in writing proofs, clean and reformat datasets, or interpret results. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Reproducibility statement

Source code for replicating the results including training residual policies for all three sets of experiments and the creation of the base goal conditioned ant policy will be open sourced. The data used for offline training of the RL token encoder is provided by Black et al. (2025). Offline finetuning data of $\pi_{0.5} - LIBERO$ used for baselines was generated by sampling trajectories of $\pi_{0.5} - LIBERO$ on target LIBERO tasks. The training procedure for the RL Token encoder we used is detailed in Appendix C. The finetuning of $\pi_{0.5}$ on *LIBERO Spatial, LIBERO Object, LIBERO Goal*, and *LIBERO-10* is also provided by Black et al. (2025). Online behavior cloning finetuning procedure is detailed in Appendix D. The methodology used to train the base policy in 4.1 is documented in Appendix B as well as hyperparameter selections and training details for the residual policies.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Reincarnating reinforcement learning: Reusing prior computation to accelerate progress. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 28955–28971. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/ba1c5356d9164bb64c446a4b690226b0-Paper-Conference.pdf.

Chethan Bhateja, Derek Guo, Dibya Ghosh, Anikait Singh, Manan Tomar, Quan Vuong, Yevgen Chebotar, Sergey Levine, and Aviral Kumar. Robotic offline rl from internet videos via value-function pre-training. *arXiv preprint arXiv:2309.13041*, 2023.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert 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. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 27–30 Sep 2025. URL https://proceedings.mlr.press/v305/black25a.html.

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, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil 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 S. Ryoo, Grecia Salazar, Pannag R. 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. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023b. doi: 10.15607/RSS.2023.XIX.025.

Shibhansh Dohare, J Fernando Hernandez-Garcia, Parash Rahman, A Rupam Mahmood, and Richard S Sutton. Maintaining plasticity in deep continual learning. *arXiv preprint arXiv:2306.13812*, 2023.

Amir-massoud Farahmand. Action-gap phenomenon in reinforcement learning. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, NIPS'11, pp. 172–180, Red Hook, NY, USA, 2011. Curran Associates Inc. ISBN 9781618395993.

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

Dibya Ghosh, Chethan Anand Bhateja, and Sergey Levine. Reinforcement learning from passive data via latent intentions. In *International Conference on Machine Learning*, pp. 11321–11339. PMLR, 2023.

Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Quan Vuong, Ted Xiao, Pannag R. Sanketi, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.090.

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.

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1026–1034, 2015. URL https://api.semanticscholar.org/CorpusID:13740328.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 international conference on robotics and automation (ICRA)*, pp. 6023–6029. IEEE, 2019.

Tobias Jülg, Wolfram Burgard, and Florian Walter. Refined policy distillation: From vla generalists to rl experts. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 11677–11684, 2025. doi: 10.1109/IROS60139.2025.11246761.

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.

Qiyang Li, Zhiyuan Zhou, and Sergey Levine. Reinforcement learning with action chunking. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=XUks1Y96NR.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, qiang liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=xzEtNSuDJk.

Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Avila Pires, Razvan Pascanu, and Will Dabney. Understanding plasticity in neural networks. In *International Conference on Machine Learning*, pp. 23190–23211. PMLR, 2023.

Guozheng Ma, Lu Li, Haoyu Wang, Zixuan Liu, Pierre-Luc Bacon, and Dacheng Tao. What makes value learning efficient in residual reinforcement learning?, 2026. URL https://arxiv.org/abs/2602.10539.

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

Mitsuhiko Nakamoto, Simon 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. In *Advances in Neural Information Processing Systems*, 2023.

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *International Conference on Machine Learning*, 2022.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay, 2016. URL https://arxiv.org/abs/1511.05952.

Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.

Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020. ISSN 2665-9638. doi: 10.1016/j.simpa.2020.100022.

Hang Wang, Sen Lin, and Junshan Zhang. Warm-start actor-critic: From approximation error to sub-optimality gap. 2023. URL https://arxiv.org/abs/2306.11271.

Bohan Wu, Feng Xu, Zhanpeng He, Abhi Gupta, and Peter K Allen. Squirl: Robust and efficient learning from video demonstration of long-horizon robotic manipulation tasks. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9720–9727. IEEE, 2020.

Charles Xu, Jost Tobias Springenberg, Michael Equi, Ali Amin, Adnan Esmail, Sergey Levine, and Liyiming Ke. Precise manipulation with efficient online rl. Technical report, Physical Intelligence, March 2026. URL https://www.pi.website/download/rlt.pdf. RL Tokens (RLT).

Xiu Yuan, Tongzhou Mu, Stone Tao, Yunhao Fang, Zhang Zhang, and Hao Su. Policy decorator: Model-agnostic online refinement for large policy model. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 28129–28164, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/45c361d4117d598d4bb6568b407e9ac9-Paper-Conference.pdf.

Shuyi Zhang, Yunfan Lou, Hongyang Cheng, Yichen Guo, Chuyao Fu, Yaoxu Lyu, Xiaojie Zhang, Haoran Li, Pengwei Wang, Zhongyuan Wang, and Shanghang Zhang. Force: Efficient vla reinforcement fine-tuning via value-calibrated warm-up and self-distillation, 2026. URL https://arxiv.org/abs/2606.26006.

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. *arXiv preprint arXiv:2510.03827*, 2025.

Zhiyuan Zhou, Andy Peng, Qiyang Li, Sergey Levine, and Aviral Kumar. Efficient online reinforcement learning fine-tuning need not retain offline data. 2024. URL http://arxiv.org/abs/2412.07762.

A Appendix: Base task success rates

Table 4: Frozen $\pi_{0.5}$ success rates on a selection of LIBERO-90 Tasks where performance is non zero. These tasks are not included in the data used to finetune $\pi_{0.5}$. $\pi_{0.5}$ fails to reliably execute these tasks in a zero shot manner. We observe in many cases that failures are due to short critical errors that lead to task failure.

TASK	SCENE	INSTRUCTION	BASE SR
t77	Study2	pick up the book and place it in the back compartment of the caddy	21.8%
t59	LivingRoom3	pick up the tomato sauce and put it in the tray	26.6%
t82	Study3	pick up the book and place it in the left compartment of the caddy	36.7%
t14	Kitchen2	put the middle black bowl on the plate	41.1%
t60	LivingRoom4	pick up the black bowl on the left and put it in the tray	54.7%

B Appendix: Experiment Specifics

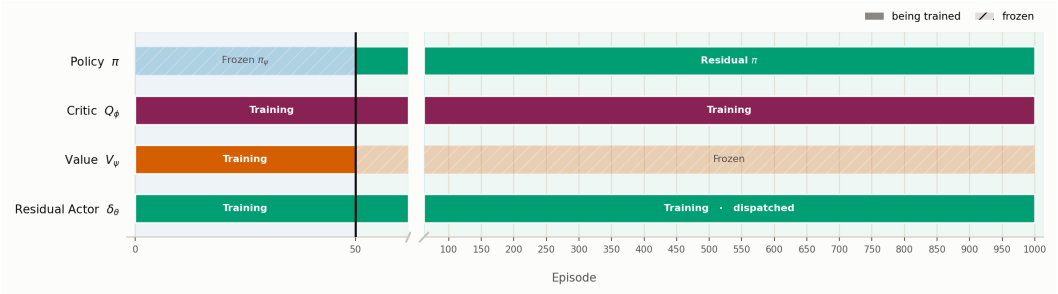


Figure 5: Illustrative timeline of the training procedure for ABR^2 on an unseen task

Experiment 4.1 - Repairing Goal Conditioned Control Policies Both environments, Grid Plus and Grid Corner, utilize the DeepMind Control Suite (Tunyasuvunakool et al., 2020). The policy is tasked with controlling the ant embodiment and guiding it to a goal position. In addition to proprioception, we use an equidistant 16-ray lidar array for sensing the walls of the environment as part of the observation provided by the environment $o_t = [\text{proprio}_t, \text{lidar}_t, \text{goal}]$. The policy executes actions $a \in [-1, 1]^8$. All networks, π_ψ, value function $V_\mu^{\pi_\psi}$, critics Q_ϕ use the same two-layer MLP architecture of 128 wide Linear-ReLU blocks.

Base policy training The base policy π_ψ is trained on Grid Plus with a dense reward function $r_t = \left(1 - \frac{1}{3}\left|v_t^g - 3\right|\right)_+ + 10\,\mathbb{1}_{\text{success}}$, with v_t^g the walker's speed toward the goal in m/s with a target speed of 3 m/s. Goals are sampled from the cardinal directions, and the walker must get within 1 m of the goal to accomplish the task. The policy network is trained using Soft-Actor Critic (Haarnoja et al., 2018) for 1000 training episodes and then the base policy is frozen.

Policy Repair π_ψ is deployed in the Grid Corner environment with one of the goal positions sampled and fixed. The environment only indicates task success with a reward of 1 at the terminal step if successful and 0 otherwise. For each episode, one of the two walls blocking the goal is removed.

Training Procedure Training proceeds over 1500 online episodes in two phases. For the first 500 episodes the frozen base policy π_ψ produces every action. Transitions are written to a replay

buffer of capacity 10^5, and gradient updates begin once the buffer holds one batch of 256 transitions ($\sim$ episode 1). One gradient step is taken per environment step, with learning rate $3\text{x}10^{-4}$, $\gamma = 0.99$, and the target networks updated at $\tau = 0.005$. The critic Q_ϕ is updated according to equation 6 and the value function $V_\mu^{\pi_\psi}$ by regressing over Monte-Carlo returns. The residual actor is updated using equation 7 against Q_ϕ. After the first phase, $V_\mu^{\pi_\psi}$ is frozen and the meta policy is then turned on, deciding the executed action by scoring $\{\tilde{a}, a^\psi\}$, executing the higher value one. The critic and residual actor continue to update until the episode budget is exhausted.

Experiment 4.2 - Repairing VLA Policies on Unseen Tasks & Seen Tasks, Unseen Configurations All tasks are drawn from LIBERO suite (Liu et al., 2023). The policy is tasked with manipulating objects in a tabletop scene according to a language instruction. The state passed to the residual policy is $s_t = [z_t^{\text{rl}}, \text{proprio}_t$ where z_t^{rl} is the compressed intermediate state of the base policy produced by a frozen RL-token encoder (Xu et al., 2026) (Appendix C). The base policy produces action chunks of size $K = 10$, $a^\psi \in \mathbb{R}^{K\times 7}$, of which the first half is executed before re-planning. The critic Q_ϕ operates on the $[2048 + 8 + 70]$ concatenation of s_t and a^ψ, projecting it to a 256-dimensional hidden representation with a Linear-LayerNorm-ELU block following Ma et al. (2026), followed by two further Linear-LayerNorm-ELU blocks with skip connections and a final linear layer to a scalar Q-value clipped to $[-10, 10]$ for numerical stability. The value function $V_\mu^{\pi_\psi}$ takes the same input and uses two 256-dimensional hidden Linear-LayerNorm-ReLU blocks with a final linear layer to a scalar state value. The residual actor π_θ is a two-layer MLP with Linear-LayerNorm-ELU blocks taking s_t and producing a correction to the base action chunk. All linear layers are initialized using Kaiming normal initialization (He et al., 2015).

BASE POLICY TRAINING The base policy π_ψ is $\pi_{0.5}$-LIBERO (Black et al., 2025), finetuned on subsets of the LIBERO suite, in particular *LIBERO Spatial*, *LIBERO Object*, *LIBERO Goal*, and *LIBERO-10*. The RL-token encoder is pretrained on the same LIBERO task corpus used to finetune $\pi_{0.5}$ and is frozen during experiments, so no novel information is leaked through it.

POLICY REPAIR π_ψ is deployed on tasks outside the corpus it was finetuned on. The environment only indicates task success with a reward of 1 at the terminal step if successful and 0 otherwise. Two settings are evaluated. In the first, we draw *LIBERO-90* tasks that have been identified as low-competency where π_ψ has a $20\% - 60\%$ success rate. In the second, tasks are augmentations of *LIBERO-Object* provided by (Zhou et al., 2025), in which object placements are sampled within x meters of the position in the source task and objects may be removed where placement is nonviable; each pair of task and displacement is treated as a distinct task.

TRAINING PROCEDURE Training proceeds over 1000 online episodes in two phases. For the first phase the frozen base policy π_ψ produces every expected action chunk. Transitions are written to a replay buffer of capacity 10^5, and gradient updates begin once the buffer holds at least one batch of 256 transitions ($\sim$ episode 4). The critic Q_ϕ is updated according to equation 6 and the value function $V_\mu^{\pi_\psi}$ by regressing over Monte-Carlo returns. The residual actor is updated using equation 7 against Q_ϕ, but these corrections are not executed. After the first phase, $V_\mu^{\pi_\psi}$ is frozen and the meta policy is then turned on, deciding the executed action by scoring $\{\tilde{a}, a^\psi\}$ using equation 8 and executing the higher value one. The critic and residual actor continue to update until the episode budget is exhausted. Hyperparameters used for the LIBERO experiments are listed in 5

Table 5: Hyperparameters used for Experiment 4.2; both with and without the advantage regularizer use the same values on each task. Warm-up is the number of episodes the frozen base policy acts before the residual is deployed. λ_δ is the weighting of the behavior cloning regularizer

Task	Warm-up (episodes)	λ_δ
Task 77 (Book to Caddy)	50	3
Task 59 (Sauce to Tray)	250	1
Task 82 (Book to Left Caddy)	50	3
Task 14 (Middle Bowl to Plate)	250	1
Task 60 (Left Bowl to Tray)	250	1

For the LIBERO-PRO experiments, the budget is expanded to 1500 episodes where uniformly 500 episodes are used for phase 1 and the remainder for phase 2. $\lambda_\delta = 3$ for all settings except for Task 8 + 0.28m displacement, where $\lambda_\delta = 1$

C Appendix: RL-Token Encoder Pretraining

Purpose The critic and residual actor operate on a fixed-size state vector, whereas the VLA exposes a variable-length sequence of prefix hidden states (one per prefix token). Following Xu et al. (2026) we learn an encoder-decoder that compresses this sequence into a single *RL token* z^{rl}, and use $s_t = [z_t^{\text{rl}}, \text{proprio}_t]$ as the RL state.

Embedding source and training domain We run the frozen $\pi_{0.5}$ LIBERO checkpoint forward on frames from `physical-intelligence/libero` — the *same* 40-task corpus (the four standard LIBERO suites, ten tasks each) on which that checkpoint was itself finetuned and extract the prefix hidden states $z \in \mathbb{R}^{N \times 2048}$, where N is the prefix length and 2048 the Gemma-2B backbone width. Sequences are zero-padded to the batch maximum. In contrast to Xu et al. (2026) the VLA is never updated, and the encoder is trained once and reused without modification across every task and perturbation condition we report.

Divergence from RL-Token We adopt Xu et al. (2026)'s architecture: the encoder is trained by reconstruction loss with the VLA held fixed with respect to that objective, and both the VLA and the encoder are frozen before online RL begins.

However, to better investigate the behavior of our additions to the residual policy and the capability for an autonomous agent to independently adapt to an OOD task we change *what data the encoder is trained on*. In Xu et al. (2026), the base VLA policy is finetuned on 1–10 hours of collected teleoperated demonstrations of that task, and the RL-token encoder is trained on the same data. Its z^{rl} is therefore learned by reconstructing the embeddings of a VLA that is *itself being adapted to the target task*, so target-task structure enters the representation through the embeddings it compresses. This is well matched to their setting, where the objective is to smooth a policy that already performs the task.

Our setting provides no such data. We assume a target task for which no demonstrations exist and on which the base policy was never trained; all task-specific adaptation must occur online, through the residual alone. The encoder consequently cannot be task-specific, and is trained on the shared 40-task corpus that the VLA has been finetuned on.

No additional target-task training data Because the encoder's training domain is the base policy's finetuning domain, z^{rl} can carry no information about the evaluation condition that π_{VLA} did not already have. Whatever is out-of-distribution for the base policy is equally out-of-distribution for the encoder. An improvement therefore cannot be attributed to the residual observing a better-adapted representation than the policy it is correcting. Concretely:

- **Held-out tasks (LIBERO-90).** The evaluation tasks are absent from the 40-task corpus, hence unseen by both the VLA and the encoder.
- **Perturbed initial configurations (LIBERO-PRO).** The task *is* in-distribution for both (`libero_object` is one of the four suites); what is held out is the object placement. LIBERO-PRO displaces objects by a controlled magnitude, and no displaced configuration appears in the corpus. The symmetry is unchanged: the encoder saw the nominal placements and no more than the VLA did.

Architecture A learned query token $q_{\text{rl}} \in \mathbb{R}^{1 \times 2048}$ (initialized $\mathcal{N}(0, 0.02^2)$) is appended to the prefix sequence, which is given sinusoidal positional encodings. A 2-layer Transformer encoder (8 heads, FFN width 2048, dropout 0.1) processes the sequence, and the output at the query position is taken as z^{rl}. A 2-layer Transformer decoder reconstructs the prefix sequence $\hat{z}$ from z^{rl}. Following Xu et al. (2026) we set $\dim(z^{\text{rl}}) = 2048$, equal to the backbone width, so the projection layers reduce to the identity and compression is over *sequence length* ($N \to 1$) rather than width. The encoder–decoder has 138.6M parameters (50.4M encoder, 84.0M decoder, 4.2M output head); only the encoder is retained at deployment.

Training The encoder is trained with a reconstruction loss using Adam, a learning rate of 10^{-4}, batch size 8, and 10,000 steps.

D Appendix: Online BC-Finetuning $\pi_{0.5}$

While ABR^2 is ideal for a compute-constrained setting, we include a comparison to a finetuned VLA to establish an upper-bound of performance if significantly more compute is available. For this setting, we iteratively finetune $\pi_{0.5}$ against successful rollouts for each task. Concretely, given a target task (e.g., "Place the soup in the basket"), we run 128 episodes on the task with $\pi_{0.5}$. We inject additional noise into the flow model to produce varied trajectories with some successes and some failures. We retain all successes in a replay buffer and use them to finetune the model for 50 optimizer steps, and then the entire process repeats.

It was crucial to retain old successes in the replay buffer rather than clearing the buffer from scratch for each new batch of 128 rollouts. Specifically, we found that retaining successes for 16 iterations (for a maximum of $16 \times 128 = 2048$ possible successful rollouts in the buffer) struck the best balance between improving performance and preventing collapse.

We finetuned only the flow-matching action expert (the Gemma-300M transformer model) with a LoRA adapter (Hu et al., 2022). This choice ensured that the action distribution changed without affecting the model's vision or language processing capabilities. The LoRA adapter is set to rank $r = 16$ and $\alpha = 16$, and applied to all attention and MLP layers of the expert. Finally, the input and output MLPs are unfrozen. All experiments were run on an 8x80GB A100 GPU node, with 8 concurrent LIBERO processes, completing in about 12 hours per run (for a total of about 94 GPU hours).

E Appendix: Comparison vs. Behavior Cloning

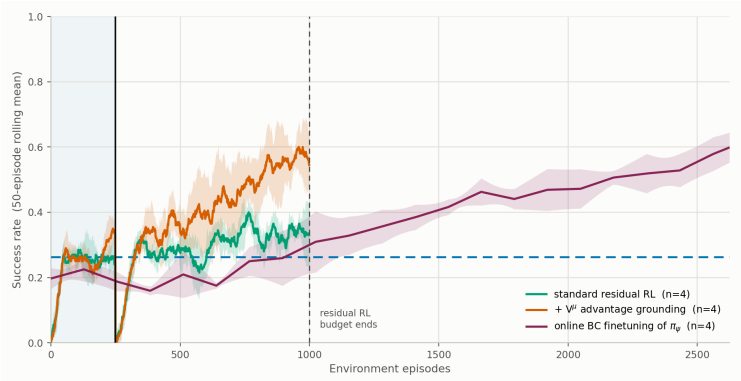


Figure 6: Comparison of method sample efficiency when learning to complete LIBERO-90 Task 59 - *pick up the tomato sauce and put it in the tray*. The purple curve shows success rate during online BC finetuning when finetuning the base policy π_ψ ($\pi_{0.5}$) on the task directly. The blue dotted line indicates the base policy success rate on the task

F Appendix: Illustrative Recovery Combining Policy Repair and VLA Robustness



Figure 7: An illustrative example demonstrating how policies repaired by ABR^2 retain the generalist skills of the base policy. The residual actor corrects the selection of the tomato can vs the blue can which is then grasped. However, the grasping fails, causing the can to be dropped. The base policy is able to re-grasp the object and complete the task without interference from the residual actor.

G Appendix: DeepMind Control Success Rates

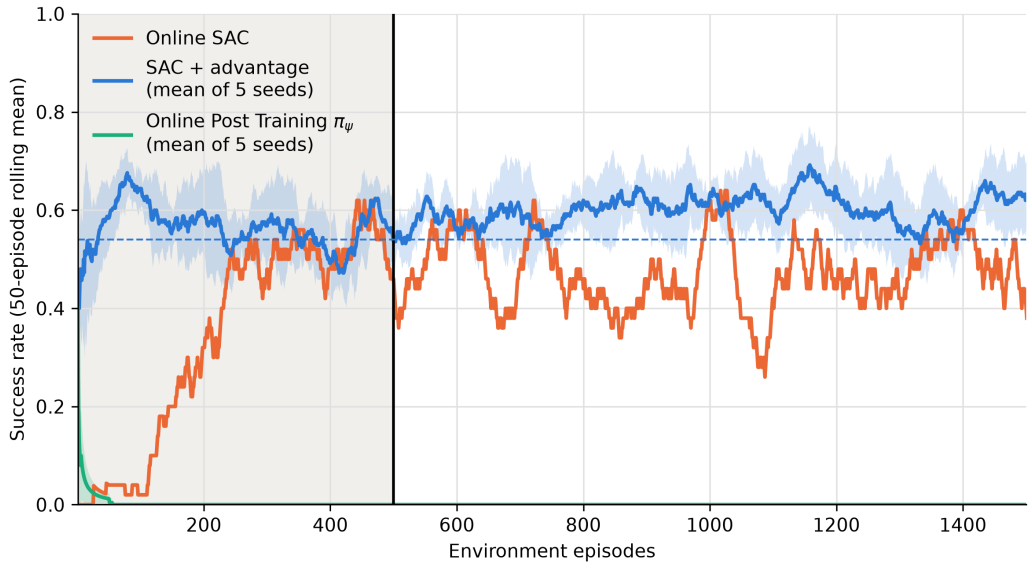


Figure 8: Comparison of success rates in the Grid Corner [Up + Right] task across 3 different models. The green curve shows the success rate of online post training attempting to directly update π_ψ using SAC which quickly collapses to 0 when it encounters a new task in a sparse reward environment. The orange line shows the success rate of a brand new Online SAC learning the task from scratch in a dense reward environment. The blue line shows the success rate of the base policy combined with meta selection and the advantage term finetuning on the task. The dotted blue line indicates the base policy's success rate on the task.