COHERENT OFF-POLICY IMPROVEMENT OF LARGE BEHAVIOR MODELS WITH LEARNED REWARDS

Anonymous authors
Paper under double-blind review

ABSTRACT

Distilling expert demonstration data into large generative models using behavioral cloning is a scalable approach to learning capable policies for robotic control, particularly for dexterous manipulation. Reinforcement learning (RL) can be used as a means to finetune these policies further using additional experience. An open question is whether RL is more sample-efficient than collecting more expert demonstrations. Prior work has finetuned large pretrained policies in a scalable fashion by applying RL to a smaller residual policy that corrects the pretrained model. However, for typical sparse reward tasks, RL algorithms can struggle to optimize the behavior in a sample-efficient manner. We explore inverse reinforcement learning, where a dense reward function is learned from expert demonstrations, potentially reducing the challenge of RL finetuning. We specifically consider coherent imitation learning, an IRL method that facilitates improvement of the BC policy by using a specific learned reward formulation with theoretical guarantees for imitation learning. We show that our IRL method maintains or improves the performance of `pi-0.5` on all six sparse manipulation tasks and achieves a $\geq 90\%$ success rate on five out of six complex manipulation tasks, outperforming RL-based baselines using sparse rewards. We also demonstrate real-world learning on a robot manipulation task, finetuning `pi-0.5` to success with less than 30 minutes of environment interaction time. By ensuring our initial pretrained finetuning policy is optimal for our initial reward and critic, our method circumvents the initial drop commonly seen in RL finetuning and enables faster improvement.

1 INTRODUCTION

Distilling massive expert demonstration datasets into large behavior models (LBMs) using behavioral cloning (BC) has seen large success in robotic control, yielding highly capable policies for complex, long-horizon manipulation (Driess et al., 2023; Barreiros et al., 2026) given enough high-quality demonstration data. Despite these advances, covariate shift due to accumulating errors remains a fundamental limitation of any BC policy. Solutions such as collecting supplementary recovery demonstrations are fundamentally limited because they rely on teleoperators to anticipate out-of-distribution states. Since online agents will inevitably face novel states, online adaptation becomes indispensable.

Reinforcement learning (RL) (Sutton & Barto, 2018) is a natural candidate to improve BC policies through online experience. In particular, residual RL offers an attractive set of characteristics for online adaptation: freezing the base policy and learning a correction allows for parameter-efficient finetuning of even large models. Recent efforts to improve online adaptation have introduced fine-grained corrections for action chunks (Ankile et al., 2025b), structured exploration (Yuan et al., 2025), and more efficient value learning (Ma et al., 2026). Despite these advances, two major challenges remain for RL for foundation models: First, designing dense, task-specific rewards for diverse, open-world tasks is difficult and prone to misspecification (Eschmann, 2021). Second, even in parameter-efficient paradigms like residual RL (Xiao et al., 2026), where the base model is frozen, the framework remains vulnerable to initial performance degradation, as illustrated in Figure 1 and shown experimentally in Figure 6. Crucially, this unlearning directly degrades the sample efficiency of any RL finetuning.

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.

To achieve online adaptation without degrading the foundation model (see Figure 1), we build on CSIL (Watson et al., 2023), an entropy-regularized IRL algorithm. In contrast to other (deep) IRL algorithms, CSIL is designed to finetune a pretrained behavior policy using IRL due to its specially designed reward structure, as IRL methods with black-box reward models tend to unlearn the BC behavior rather than improve it (Watson et al., 2023). Our contributions are:

1. We propose using inverse RL, specifically CSIL, to finetune LBMs with a learned dense reward as an alternative to RL with challenging sparse rewards.
2. We provide an improved implementation of CSIL based on recent advances in deep actor-critic methods using batch normalization, weight normalization, and categorical critics (Palenicek et al., 2026).
3. We present an experimental benchmark using `pi-0.5` across six sparse, simulated manipulation tasks, showing benefits over RL-based finetuning baselines. We also demonstrate successful real-world learning on a robot manipulation task.

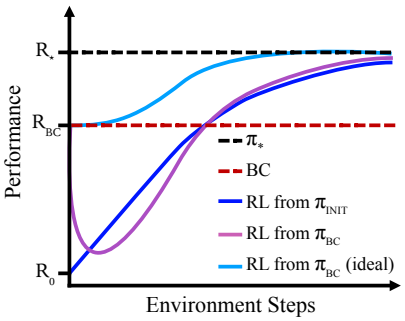

Figure 1: An illustration of RL finetuning of strong BC policies using residuals. Despite a strong initial performance by a BC policy, using function approximation in actor-critic methods means this performance is rapidly unlearned and relearned, so in practice, the BC initialization provides little benefit when running the RL method from scratch. We use coherent soft imitation learning (CSIL) for (inverse) RL finetuning without unlearning due to the learned coherent reward.

2 Related Work

While LBMs show strong performance, we seek to further improve them in an autonomous fashion using RL. The challenge here is that LBMs have incredibly large parameter spaces, which are tractable for supervised learning but are a challenge for reinforcement learning, where the gradient updates have higher variance.

Ren et al. (2025) refine a diffusion policy using on-policy RL by augmenting environment rollouts with those generated by the diffusion model, which allows them to optimize a subset of denoising steps rather than the entire model. Chen et al. (2026) propose a similar approach, but adapted for flow-matching policies. Ankile et al. (2025b) also use on-policy RL to improve a diffusion policy using a residual MLP policy focusing on precise assembly and sim-to-real transfer. On-policy methods are challenging to apply to real-world experience due to their poor sample efficiency. As a result, prior work has investigated off-policy RL. Wagenmaker et al. (2025) propose diffusion steering via RL, which applies off-policy RL to the latent space of the diffusion model rather than the model itself. For general LBMs, residual RL (Silver et al., 2018) is attractive to finetune an additional, smaller policy. Xiao et al. (2026) propose 'probe, learn, distill' (PLD), which combines regularized offline RL critics for off-policy RL with additional probing and distillation steps. Ankile et al. (2025a) propose ResFit, which also uses off-policy RL to finetune diffusion policies in a residual fashion. While these results validate the potential of residual RL finetuning, the presented learning curves consistently report severe initial performance drops during early finetuning. This phenomenon is due to the learning dynamics of actor-critic methods, where gradient updates from the initial suboptimal critic 'unlearn' the BC behavior rather than finetune it (Figure 1). BC unlearning has also been reported in the inverse RL setting due to the jointly learned reward function (Orsini et al., 2021; Watson et al., 2023).

In concurrent work, Sun & Song (2026) also apply off-policy RL to finetune residual policies, but use a heuristic and adaptive BC-based policy regularization using the learned Q-values. While their method, DICE-RL, does not exhibit unlearning due to the policy regularization, our proposed method appears x3 more sample-efficient on image-based Robomimic tasks. Zhong et al. (2026) improve LBMs using on-policy distillation from a provided expert policy. This approach strongly resembles CSIL, as the reward is a log-policy ratio, though the on-policy formulation is different.

3 Coherent Inverse Reinforcement Learning

In this section, we summarize the coherent soft imitation learning algorithm and describe several adaptations that scale the algorithm to the harder tasks where LBMs benefit from finetuning.

3.1 Coherent Soft Imitation Learning

We consider a Markov decision process (MDP) $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma, \mu_0 \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}^+$ is the transition model, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, γ is the discount factor, and μ_0 is the initial state distribution $\boldsymbol{s}_0 \sim \mu_0(\cdot)$. When the reward is unknown, imitation can be performed using a dataset $\mathcal{D}$ of transitions $(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')$ as demonstrations. The policy could be inferred directly or by inferring a reward and policy jointly, referred to as behavioral cloning and inverse reinforcement learning, respectively.

We consider the KL-regularized RL setting, where policy π is optimized to maximize the state-action value function Q subject to a soft KL constraint against a prior policy $p(\boldsymbol{a}|\boldsymbol{s})$ to promote exploration, $\max_\pi \mathbb{E}_{\boldsymbol{s},\boldsymbol{a}\sim\rho_q}[Q(\boldsymbol{s},\boldsymbol{a})] - \alpha\, \mathbb{D}_{\mathrm{KL}}[\pi(\boldsymbol{a}|\boldsymbol{s}) \,||\, p(\boldsymbol{a}|\boldsymbol{s})]$. The optimal policy update blends the exponentiated advantage function 'pseudo-likelihood' with the prior, as a form of regularized Boltzmann policy (Haarnoja et al., 2018) with temperature $\alpha > 0$ and soft value function $V(\boldsymbol{s})$,

$$\pi(\boldsymbol{a}|\boldsymbol{s}) \propto \exp((Q(\boldsymbol{s},\boldsymbol{a}) - V(\boldsymbol{s}))/\alpha)\, p(\boldsymbol{a}|\boldsymbol{s}),\ V(\boldsymbol{s}) = \alpha \log \textstyle\int \exp(Q(\boldsymbol{s},\boldsymbol{a})/\alpha)\, p(\boldsymbol{a}|\boldsymbol{s})\, \mathrm{d}\boldsymbol{a}. \quad (1)$$

In the function approximation setting, the update is performed in a variational fashion by minimizing the reverse KL divergence between the parametric policy $\pi_{\boldsymbol{\theta}}$ and the critic-derived update at sampled states $\boldsymbol{s}$, as done in the soft actor-critic (SAC) algorithm (Haarnoja et al., 2018),

$$\begin{aligned} \boldsymbol{\theta}_* &= \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{s}\sim\mathcal{B}}[\mathbb{D}_{\mathrm{KL}}[\pi_{\boldsymbol{\theta}}(\cdot \mid \boldsymbol{s}) \,||\, \exp((Q(\boldsymbol{s},\cdot) - V(\boldsymbol{s}))/\alpha)\, p(\cdot|\boldsymbol{s})]], \\ &= \arg\max_{\boldsymbol{\theta}} \mathcal{J}_\pi(\boldsymbol{\theta}),\ \mathcal{J}_\pi(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{s}\sim\mathcal{B},\, \boldsymbol{a}\sim\pi_{\boldsymbol{\theta}}(\cdot|\boldsymbol{s})} \left[Q(\boldsymbol{s},\boldsymbol{a}) - \alpha \left(\log \pi_{\boldsymbol{\theta}}(\boldsymbol{a}|\boldsymbol{s}) - \log p(\boldsymbol{a}|\boldsymbol{s})\right)\right]. \end{aligned} \quad (2)$$

The above objective $\mathcal{J}_\pi$ can be maximized using reparameterized gradients and minibatches from the replay buffer $\mathcal{B}$ (Haarnoja et al., 2018). In SAC, $p(\boldsymbol{a}|\boldsymbol{s})$ is a uniform distribution to provide maximum entropy regularization.

In coherent soft imitation learning (CSIL) (Watson et al., 2023), inverse RL is achieved by rearranging the policy update in Equation 1 and substituting it into the KL-regularized Bellman equation. This policy update inversion results in the log-policy ratio appearing as a reward term in the Bellman equation. By substituting a BC policy as the optimal policy, the log-policy ratio becomes a learned reward that is *coherent* with the BC policy π (Definition 1). Theorem 1 shows that this coherent reward is a shaped variant of the true reward and therefore shares the same optimal policy (Theorem 2,

Definition 1. *(Coherent reward, Watson et al. (2023)). For a BC policy π, prior $p(\boldsymbol{a} \mid \boldsymbol{s})$ and temperature $\alpha > 0$, the scaled log-policy ratio*

$$\tilde{r}(\boldsymbol{s},\boldsymbol{a}) = \alpha \left(\log \pi(\boldsymbol{a} \mid \boldsymbol{s}) - \log p(\boldsymbol{a} \mid \boldsymbol{s})\right), \quad (3)$$

is the coherent reward function, where π is the policy and p is a uniform prior.

Theorem 1. *(Coherent reward, Watson et al. (2023)) In the KL-regularized Bellman equation, the log-policy ratio (Definition 1) is derived as*

$$\alpha \log \frac{\pi(\boldsymbol{a} \mid \boldsymbol{s})}{p(\boldsymbol{a} \mid \boldsymbol{s})} = r(\boldsymbol{s},\boldsymbol{a}) + \gamma \mathbb{E}[V(\boldsymbol{s}')] - V(\boldsymbol{s}),$$

which shows it is a shaped reward (Theorem 2).

Theorem 2. *(Reward shaping, Ng et al. (1999)) For potential ψ, the shaped reward $\tilde{r}(\boldsymbol{s},\boldsymbol{a})$, where*

$$\tilde{r}(\boldsymbol{s},\boldsymbol{a}) = r(\boldsymbol{s},\boldsymbol{a}) + \gamma \mathbb{E}[\psi(\boldsymbol{s}')] - \psi(\boldsymbol{s})$$

has the same optimal policy as $r(\boldsymbol{s},\boldsymbol{a})$.

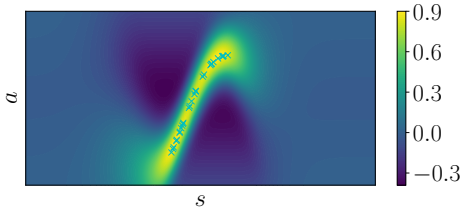

Figure 2: Given demonstration data (×), the coherent reward provides positive rewards for correct actions for observed states, negative rewards for incorrect actions in seen states, and zero rewards for any action under unseen states. This reward encourages the agent to stay (and return) to the demonstration distribution. In contrast to adversarial methods, no on-policy samples are needed to learn it. The contour plot is taken from Watson et al. (2023).

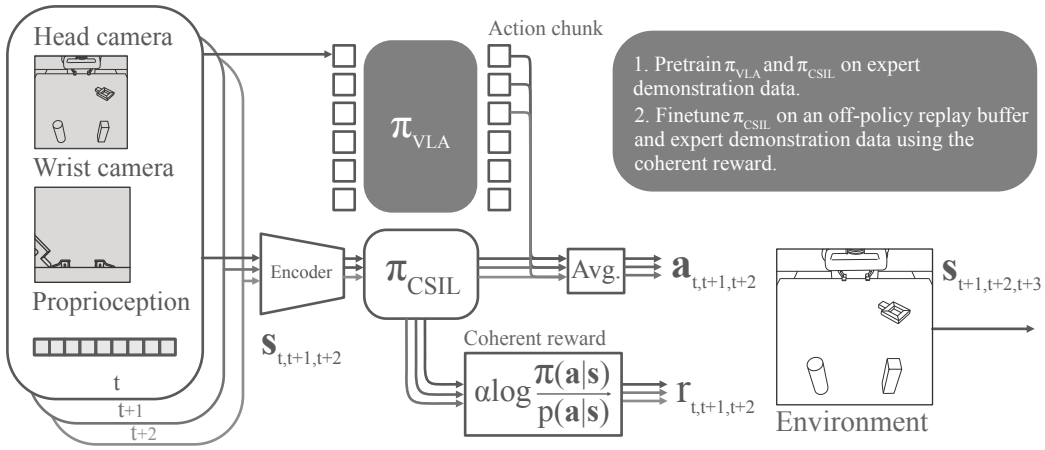

Figure 3: A system-level figure of CSIL-based finetuning of VLAs using averaged actions. The CSIL policy reacts to every observation and adapts the action chunk of the VLA, which executes at the action chunk frequency. CSIL has a dedicated encoder and does not use the VLA representation.

Ng et al. (1999)). The benefit of this approach to IRL is that BC policies are naturally finetuned with additional experience, in contrast to methods where the reward and policy are learned jointly, and BC behavior is quickly unlearned (Watson et al., 2023). The log-policy ratio can also be interpreted as a dense reward that encourages the agent to follow (and return to) the demonstration distribution (Figure 2).

CSIL requires a policy architecture that is *stationary*, such that its predictions revert to its prior $p(\boldsymbol{a} \mid \boldsymbol{s})$ outside the training distribution, ensuring the coherent reward provides a neutral reward of zero in unseen states. Following Watson et al. (2023), we use the HetStat (heteroscedastic stationary) architecture, which achieves stationarity via a periodic activation function in a wide final layer (Meronen et al., 2021), approximating a stationary kernel via the Wiener-Khinchin theorem (Rasmussen & Williams, 2006). Moreover, while CSIL is an actor-critic method that builds on SAC, it regularizes the policy against the BC policy rather than providing maximum entropy regularization. This KL regularization has a temperature β which does not have to match α. The latter is typically set to make the coherent reward scale invariant (Watson et al., 2023).

3.2 Scaling the Coherent Soft Imitation Learning Implementation

While Watson et al. (2023) demonstrate that CSIL achieves strong empirical results, the implementation struggles to solve more complex tasks in the vision-based setting in RoboMimic.

Actor-critic. To improve on CSIL's base performance, we integrated recent advances from the current state-of-the-art sample-efficient off-policy RL methods, which incorporate architectural regularization to facilitate more graceful scaling. In particular, following the results from XQC (Palenicek et al., 2026), we adapt the CSIL critic to a categorical critic and apply batch normalization (BN) (Ioffe & Szegedy, 2015) and weight normalization (WN) (Lyle et al., 2024) instead of layer normalization (Ba et al., 2016) to the critic, while applying layer normalization to the policy. Combining batch normalization and weight normalization improves the optimization landscape and helps maintain a steady 'effective' learning rate (Lyle et al., 2024; Palenicek et al., 2025). We observed that the CSIL implementation by Watson et al., which is based on SAC, suffered from exploding parameter values during training, which impedes learning as the 'effective' learning rate drops. Incorporating BN and WN into CSIL allowed us to mitigate these numerical issues and improve optimization and scaling. Incorporating this regularization also enables us to increase the update-to-data (UTD) ratio, where the critic is updated several times per environment step to accelerate learning. Following Fujimoto et al. (2018), we use policy delay and only update our policy once per environment step.

KL estimation. Additionally, while Watson et al. (2023) use a Monte Carlo KL estimator for the BC regularization term in the actor objective, we identify that this KL can be computed exactly between the latent Gaussian actions due to the shared bijection between policies (Polyanskiy, 2020).

Algorithm 1: CSIL-based finetuning of large behavior models.

Data: Demonstrations $\mathcal{D}$, initial temperature α, refinement temperature β, chunk horizon H, base policy π_{LBM}, parametric policy class $\pi_{\boldsymbol{\phi}}(\boldsymbol{a} \mid \boldsymbol{s})$, prior policy $p(\boldsymbol{a} \mid \boldsymbol{s})$, RL steps N
Result: $\pi_{\boldsymbol{\phi}_N}(\boldsymbol{a} \mid \boldsymbol{s})$, which, averaged with $\pi_{\text{LBM}}(\boldsymbol{a} \mid \boldsymbol{s})$, matches or improves the base policy
Finetune large foundation policy π_{LBM} from expert demonstrations and freeze it;
Train initial HetStat policy with BC, $\boldsymbol{\phi}_0 = \arg\max_{\boldsymbol{\phi}} \mathbb{E}_{\boldsymbol{s},\boldsymbol{a}\sim\mathcal{D}}[\log \pi_{\boldsymbol{\phi}}(\boldsymbol{a} \mid \boldsymbol{s})]$;
Define fixed shaped coherent reward $\tilde{r}(\boldsymbol{s}, \boldsymbol{a}) = \alpha\,(\log \pi_{\boldsymbol{\phi}_0}(\boldsymbol{a} \mid \boldsymbol{s}) - \log p(\boldsymbol{a} \mid \boldsymbol{s}))$;
Pretrain critic on demonstration data and coherent reward;
for $n \leftarrow 1$ **to** N **do**
  if $(n-1) \bmod H = 0$ **then**
    $\boldsymbol{a}_{\text{base},n}, \ldots, \boldsymbol{a}_{\text{base},n+H-1} \leftarrow \pi_{\text{LBM}}(\boldsymbol{s}_n)$;
  end
  $\boldsymbol{a}_{\pi,n} \sim \pi_{\boldsymbol{\phi}_{n-1}}(\cdot \mid \boldsymbol{s}_n)$;
  $\boldsymbol{a}_{\text{avg},n} \leftarrow \frac{1}{2}(\boldsymbol{a}_{\text{base},n} + \boldsymbol{a}_{\pi,n})$;
  Compute Q_n and $\pi_{\boldsymbol{\phi}_n}$ using soft off-policy RL, e.g. SAC or XQC, with temperature β.
end

Image encoder. To improve CSIL's performance in image-based settings, we adapted CSIL's original IMPALA-style encoder (Espeholt et al., 2018) by increasing its size, training an encoder per camera instead of one for all cameras, and utilizing spatial softmax pooling (Levine et al., 2016) at the end of the network to extract structural information efficiently.

We refer to this improved implementation of CSIL as CSIL++ in the subsequent text.

4 Coherent Finetuning of Pretrained Policies

This section outlines how CSIL is adapted to improve pretrained action-chunking policies, in particular large behaviour models (LBMs). A high-level description is provided in Figure 3 and Algorithm 1. An important design decision for LBM finetuning is the choice of optimized action. This action could be the environment action, a residual correction term to the LBM (Equation 4), or an averaged action between the base action and the predicted action (Equation 5):

$$\boldsymbol{a}_{\text{res}} = \boldsymbol{a}_{\text{base}} + \boldsymbol{a}_{\pi} \quad \text{(residual)}, \tag{4}$$

$$\boldsymbol{a}_{\text{avg}} = \tfrac{1}{2}(\boldsymbol{a}_{\text{base}} + \boldsymbol{a}_{\pi}) \quad \text{(average)}. \tag{5}$$

See Figure 4 for a visualization of both. Directly adapting the base policy to predict environment actions requires optimizing a model with billions of parameters via RL. Even with LoRA (Hu et al., 2022), this approach is computationally prohibitive due to a massive increase in forward passes of the LBM, as the model must be evaluated for every batch sample and environment step. To learn an additional, smaller corrective policy, we use a frozen version of the LBM to predict an action chunk every H steps, while the smaller model is trained to predict a full action every step. We use `pi-0.5` in our experiments, where $H = 10$. Using a residual action, i.e., predicting a correction (Silver et al., 2018) on top of the base LBM policy, performed worse (see Figure 6). For CSIL, we

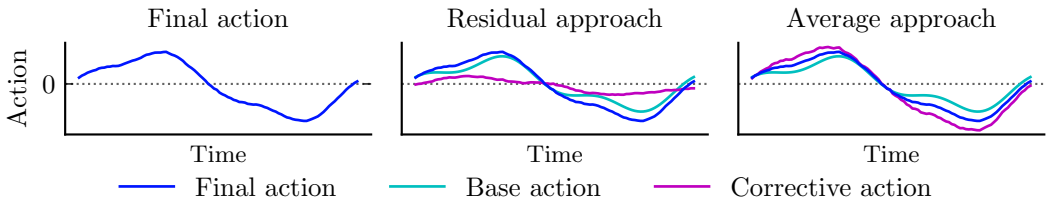

Figure 4: Visualization of the different action modalities. The cyan, magenta, and blue lines represent the actions generated by the base policy, the predictions of the trainable policy, and the final action executed in the environment, respectively. The final action for the residual approach is calculated using Equation 4, while the total action using the average approach is calculated via Equation 5. The average approach improves CSIL's coherent reward by differentiating between expert and non-expert actions more strongly.

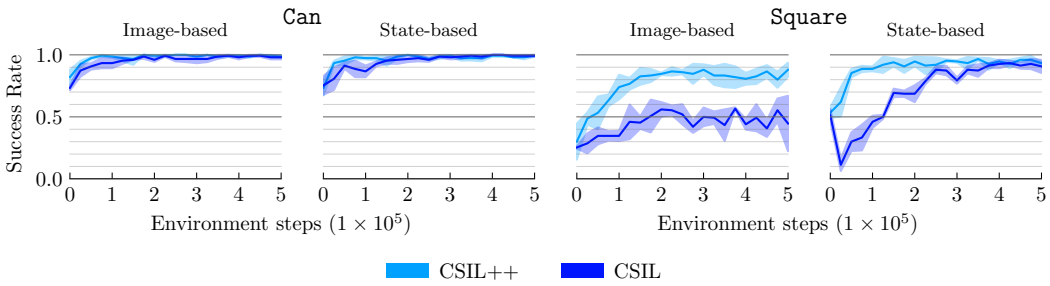

Figure 5: Performance of CSIL and our improved version CSIL++ on the `Can` and `Square` task. The 'image-based' run takes images and the robot's proprioceptive state as observations. The 'state-based' run operates on full state information comprising proprioception and ground-truth information of the object. In both settings, CSIL++ is more sample-efficient and improves performance significantly in the image-based setting.

found that a residual BC policy produced a poorly performing coherent reward. We attribute this result to the residual policy modeling function approximation errors rather than the demonstration behavior, which makes it harder to differentiate expert and non-expert behavior. We show this experimentally in Figure 8. To mitigate this issue, we take an averaging approach where the RL policy also learns the expert behavior and is averaged with the LBM action. Moreover, rather than using the LBM's internal representation for RL, we found it better and faster to let the RL policy learn its own observation encoder, which is smaller than the LBM's and therefore easier to finetune. The observation encoder is learned using BC and then kept frozen during RL.

5 Experimental Setup

We test our approach on tasks from RoboMimic (Mandlekar et al., 2021) and MimicGen (Mandlekar et al., 2023), which vary in terms of horizons, precision requirements, and complexity. Our RL baselines use a constant negative reward of -1 as the task feedback for MimicGen environments, and we wish to solve the task as quickly as possible. All our experiments utilize successful termination for bootstrapping. We train all methods across five independent seeds for 500k steps plus 10k warmup steps and evaluate the policy every 50k steps for 50 episodes. We use `pi-0.5` (Physical Intelligence et al., 2025) as our base model with an action chunk horizon of 10.

Tasks. We use the Robomimic and MimicGen benchmark suite to evaluate our method, due to their breadth in difficulty beyond straightforward pick-and-place tasks. All tasks use a Franka robot with a parallel-jaw gripper and have an action space dimension of 7: 6 for the end-effector pose delta and 1 for the gripper command. For `Square`, we use the released expert demonstration dataset with 200 episodes. For MimicGen tasks, we use their dataset of 200 episodes generated from a source dataset of 10 demonstrations. We use the environment's success flag to trigger an absorbing state.

Pretraining. For the simulation experiments, we finetune `pi-0.5` for 30k steps with a batch size of 128 for 13 hours on 4 H100 GPUs, combining all single-task datasets into one dataset. For the real-world experiments, we train 6 `pi-0.5` models in increments of 10k expert demonstration samples, starting at 10k samples using the same hyperparameters as for the simulation tasks. We use the model trained on 10k expert demonstration samples for our RL results.

Baselines and Ablations. Our primary policy improvement algorithm is CSIL++, as described in Section 3.2. As prior work uses state-of-the-art, sample-efficient, off-policy RL, we use XQC augmented with the expert offline data (XQC+OD) as an RL baseline and also an ablation of the coherent reward, as we provide XQC with the same HetStat policy, BC pretraining, and initial replay buffer as CSIL++, so only the reward and KL regularization are changed. We baseline our results with related works ResFit, policy decorator, and PLD. We adapt ResFit's and policy decorator's codebases to evaluate them on our chosen six environments and to use `pi-0.5` as the base model for fair comparison. Since there is no public code available for PLD, we implement the RL component of it in our codebase. We omit BC probing and VLA distillation to isolate the performance of the RL finetuning, since we believe it to be orthogonal to RL finetuning performance. Finally, to investigate the effect of the algorithm improvements introduced in Section 3.2, we compare the base version of CSIL with CSIL++ on state- and image-based settings of `Can` and `Square` without an LBM.

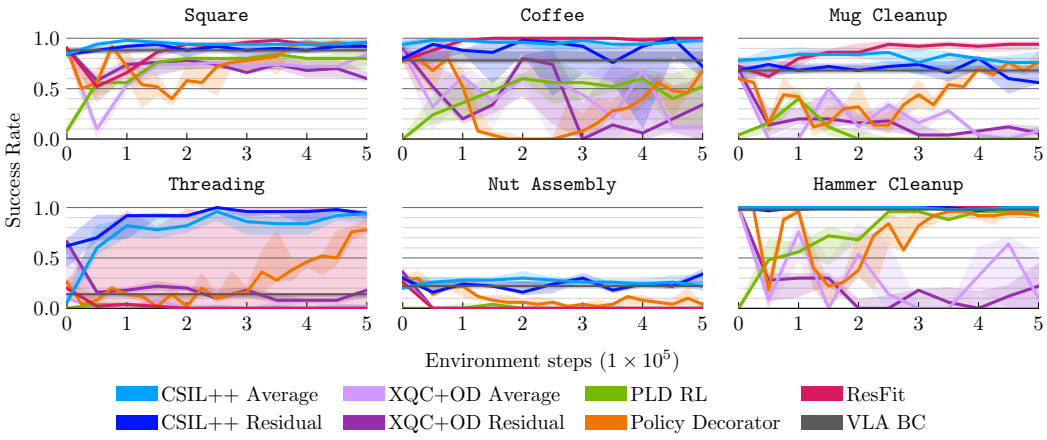

Figure 6: Performance of LBM finetuning on six simulated environments across five seeds. Success rates are reported as medians, and uncertainty ranges are quartiles on six sparse reward manipulation tasks and are computed over 50 evaluations every 50k steps.

Real-world setup and task. Our real-world setup uses a single Franka FR3 arm, a static and a wrist-mounted ZED camera, and a 3D-printed cylinder as its end-effector. The task is to push a cube over the finish line while staying inside the track markings, requiring reactivity and a high degree of error correction since the block can easily slide off the rounded end-effector. We collect expert demonstrations via teleoperation. The robot is controlled using 2D operational space control at 30 Hz. RL training is done using a separate data collection process and a learning process, which are run in parallel, similar to SERL (Luo et al., 2024). Only the entropy coefficient is changed from 0.3 to 0.1 in comparison to the hyperparameters of the simulation experiments.

Implementation details and hyperparameters. We summarize the specific architectural choices and hyperparameters used across our experiments in Table 2 in Appendix B.

6 Experimental Results

To evaluate CSIL++ finetuning of VLAs, our empirical evaluation is guided by four questions:

1. How does the performance of CSIL++ compare to CSIL?
2. Can CSIL++ improve the performance of LBMs better than RL with sparse rewards?
3. Is RL more sample efficient than BC with additional expert demonstrations?
4. How do residual and average policies compare across methods?

Comparing CSIL and CSIL++. While CSIL can be used in the pixel-based setting, we found it struggled to recover state-based performance in harder tasks. Figure 5 compares CSIL and CSIL++ on RoboMimic's `Can` and `Square` tasks. We can see that while CSIL and CSIL++ closely match

Method	`Square`	`Coffee`	`Mug Cleanup`	`Threading`	`Nut Assembly`	`Hammer Cleanup`
VLA	0.80 [0.70, 0.90]	1.10 [0.70, 1.20]	1.60 [1.40, 1.70]	4.30 [4.30, 4.40]	3.90 [3.70, 3.90]	0.10 [0.10, 0.10]
VLA (2K demos)	N/A	0.50 [0.30, 0.57]	1.20 [1.02, 1.40]	4.90 [4.83, 4.90]	2.30 [2.30, 2.40]	0.00 [0.00, 0.08]
CSIL++ Avg.	**0.29 [0.29, 0.32]**	**0.24 [0.12, 0.24]**	1.06 [0.75, 1.37]	1.04 [1.02, 1.12]	**3.60 [3.53, 3.83]**	0.03 [0.03, 0.08]
CSIL++ Res.	0.58 [0.44, 0.62]	0.94 [0.37, 1.56]	1.56 [1.37, 1.71]	**0.71 [0.38, 0.75]**	3.83 [3.77, 3.85]	**0.04 [0.03, 0.04]**
Policy decorator	1.31 [1.14, 1.34]	3.27 [3.18, 3.39]	2.83 [2.72, 2.89]	3.67 [3.64, 3.75]	4.50 [4.44, 4.55]	1.32 [1.13, 1.48]
ResFit	0.64 [0.59, 0.75]	0.17 [0.11, 0.25]	**0.73 [0.57, 0.87]**	4.90 [0.50, 4.93]	4.93 [4.92, 4.93]	0.04 [0.04, 0.05]
XQC+OD Avg.	1.79 [1.67, 1.88]	3.12 [2.55, 3.47]	4.12 [3.88, 4.16]	4.83 [4.75, 4.95]	4.93 [4.92, 4.94]	3.54 [3.50, 3.62]
XQC+OD Res.	1.46 [1.44, 1.48]	3.06 [2.12, 3.67]	4.10 [3.87, 4.46]	4.16 [4.01, 4.21]	4.92 [4.91, 4.92]	3.74 [3.58, 4.09]
PLD RL (Res.)	1.48 [1.39, 1.59]	2.75 [2.45, 2.85]	4.63 [4.63, 4.79]	4.96 [4.94, 4.98]	4.96 [4.96, 4.98]	1.19 [1.05, 1.19]

Table 1: LBM finetuning cumulative regret reported as median and quartiles on six sparse-reward manipulation tasks. The cumulative regret measures sample efficiency using the area between the success rate curve and that of a policy which succeeds in every episode over the first 5×10^5 environment steps in units of 10^5 steps, so it ranges from 0 (always successful) to 5 (never successful). Bold indicates the lowest upper quartile. Results aggregate five seeds per environment and method.

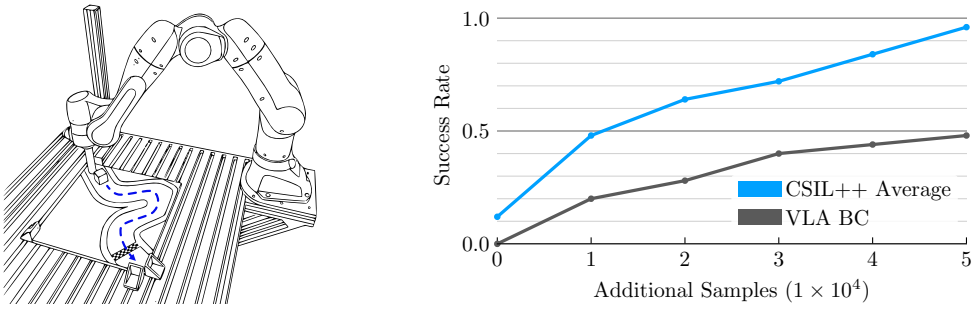

Figure 7: (left) Illustration of the real-world setup where the robot needs to push the cube over the checkered line without leaving the track markings. (right) Evaluation results of the real-world experiments with 25 evaluation episodes every 10k steps. CSIL++ Average achieves near-optimal performance within 50k environment steps. In contrast, `pi-0.5` only achieves roughly 50% success rate when trained on an additional 50k expert demonstration samples.

on the simpler `Can` task, there is a much larger gap on the `Square` task. CSIL++ is far more sample-efficient in the image-based setting, reflecting the findings of Palenicek et al. (2026).

Finetuning large behavior models. We now look at CSIL++ as a means of improving `pi-0.5` performance, baselining our results using XQC, PLD, policy decorator, and ResFit. Figure 6 shows that CSIL++ provides the best improvement in four of the six tasks. Most notably, CSIL++ improves success rates on `Threading` from 14% to 92%. Following CSIL's derivation for coherent improvement of the BC policy, CSIL++ is the only method that matches or improves on the base VLA's performance and does not exhibit initial performance drops across all tasks. For `Nut Assembly`, a challenging long-horizon task, while CSIL++ is not able to make significant progress on the task, it is the only method not to unlearn. ResFit is the strongest baseline, showing stable improvements and matching CSIL++'s performance on `Square`, `Coffee` and `Hammer Cleanup` while outperforming it on `Mug Cleanup`. It exhibits initial unlearning in all tasks but `Coffee` and `Hammer Cleanup`. Policy decorator's performance is unstable for all tasks except Threading. In particular, for the saturated task `Hammer Cleanup`, it exhibits jumps from nearly optimal performance to roughly 10% within 50k steps. The XQC baseline shows that residual policies are useful in the pure RL setting, where residual XQC outperforms average XQC on `Threading` and `Square`. The RL component of PLD exhibits immediate unlearning across all tasks, matching the results from Xiao et al. (2026). While it matches other baselines on `Hammer Cleanup` in its final performance, it requires $\sim$300k environment steps to recover VLA performance.

In the real-world setting, Figure 7 shows CSIL++ is able to reach near-optimal performance in the real world within 50k samples, which corresponds to around 28 minutes interaction time.

Evaluating demonstration efficiency. We collect 2K demonstrations using MimicGen's pipeline for all MimicGen tasks and finetune `pi-0.5` for 300k steps. 2K is x10 more data than the initial LBM and closely upper-bounds the data collected during RL. Table 1 shows that the additional demonstrations improve the performance for evaluated tasks except `Threading`. However, only on `Nut Assembly` do the additional demonstrations outperform the RL methods and CSIL++ average in particular. CSIL++ uses less than 2K total episodes for all tasks except `Threading` and `Hammer Cleanup`, where 2K total episodes are reached around 400K environment steps for both.

Comparing the real-world's CSIL++ performance against BC with equal demonstration data, Figure 7 supports the simulation results' findings of superior sample efficiency of RL data collection. Our finetuned VLA achieves 50% success rate when given the same amount of additional demonstration samples as CSIL++ uses to solve the task.

Comparing residual and average policies across methods. To get a clearer view of the benefits and drawbacks of the average approach for finetuning LBMs, we now compare its performance to the residual approach. Figure 6 shows CSIL++ Average to be stable across all six environments whereas CSIL++ residual's performance is unstable on `Coffee` and `Mug Cleanup`. In comparison, baselines such as the naive implementation of the average and residual approach as well as policy decorator are much more unstable, especially on `Hammer Cleanup` and `Coffee`. We find ResFit to be stable on all tasks except `Nut Assembly` and `Threading`. To understand the superior performance of the average approach over the residual approach on most of the tasks, we plot

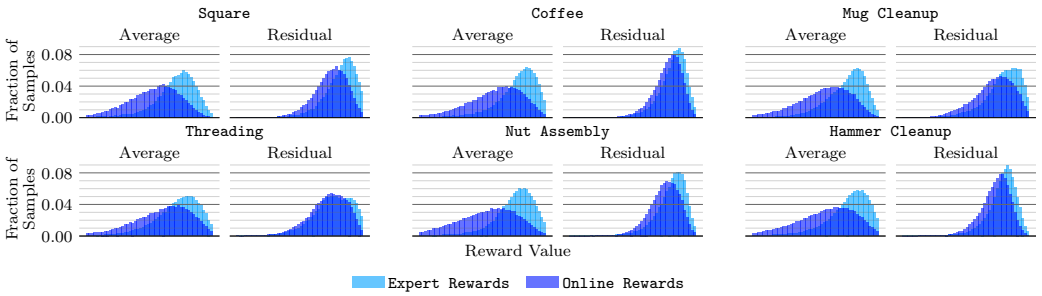

Figure 8: A comparison of the online and expert reward for the average and residual policy. The 'average' approach differentiates between online and expert samples much more strongly than the 'residual' approach, which we believe explains its coherent reward performance being better or on par with respect to downstream success rate. Online samples are collected per environment using a fixed HetStat policy, trained on expert demonstration data, in combination with a VLA.

the distribution of rewards for the expert demonstration samples and online samples per environment and approach collected using policies trained only via BC. Figure 8 shows that CSIL++ average differentiates more strongly than CSIL++ residual between online samples and expert samples across the environments despite having equal or higher success rates on them.

Visualizing refinement. In Figure 9, we visualize the effect of CSIL++ refinement on the VLA behavior, specifically focusing on the `Threading` task where CSIL++ had the largest impact. We see that `pi-0.5`'s main failure mode is high trajectory variance around insertion. CSIL++ refinement learns the alignment behavior pre-insertion that is required for consistent success.

7 Conclusion

We demonstrate that inverse reinforcement learning can be used to finetune large behavior models, as CSIL provides a coherent, dense reward that mitigates the sample inefficiency and instability typically associated with refining LBMs using RL with sparse rewards. By replacing sparse environment signals with a dense reward derived from expert demonstrations, we are able to achieve policy improvement without expensive unlearning. We have also shown that CSIL can be scaled to harder vision-based tasks by incorporating innovations in deep actor-critic methods. Our experimental study pushes LBM finetuning beyond pick-and-place tasks and demonstrates that CSIL-based finetuning can provide dramatic improvements in learning precise insertion tasks such as the `Threading` task, and also scales to real-world deployment. For future work, our results show there is still further research needed to achieve greater learning stability and sample-efficiency in LBM finetuning, in particular for long-horizon tasks as seen in the `Nut Assembly` task.

Limitations. A fundamental limitation of CSIL is how it leverages BC. If there are few demonstrations such that the BC policy is weak, CSIL's sample efficiency will also weaken. CSIL also optimizes an imitation objective, so it is susceptible to suboptimal demonstrations, as it will reinforce any suboptimal behaviors, e.g., hesitation, potentially impacting performance.

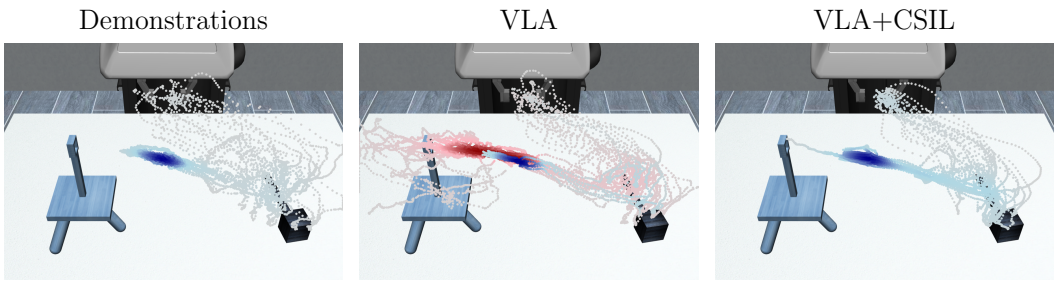

Figure 9: A visualization of 25 rollouts between the VLA and the refined policy on `Threading`, highlighting successful (blue) and failed (red) episodes. The base VLA fails the task due to high variance in the insertion trajectory. CSIL++ Average learns to align properly for precise insertion.

AI USE STATEMENT

In this work, we used generative AI tools for aiding with software development and providing feedback on research methodology, experiment design and this paper. We have not used generative AI tools for synthetic datasets, data processing, or data interpretation. Commenting on aid for novel mathematical results is not relevant to this work. Additionally, we used generative AI tools for help with creating the Figure 7 illustration from a photo reference. We have reviewed all AI-assisted work, primarily through inspection by the lead authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We use open-source simulation benchmarks with freely available demonstration data sets for reproducibility. Our code is anonymized and provided as supplementary material for reference. Our real-world experiment and demonstration data is not possible to open-source due to a bespoke lab setup, including a custom built table and camera rig, but is described as faithfully as possible.

REFERENCES

Lars Ankile, Zhenyu Jiang, Rocky Duan, Guanya Shi, Pieter Abbeel, and Anusha Nagabandi. Residual off-policy RL for finetuning behavior cloning policies. *arXiv preprint arXiv:2509.19301*, 2025a.

Lars Ankile, Anthony Simeonov, Idan Shenfeld, Marcel Torne, and Pulkit Agrawal. From imitation to refinement-residual RL for precise assembly. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2025b.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. In *NIPS 2016 Deep Learning Symposium*, 2016.

Jose Barreiros, Andrew Beaulieu, Aditya Bhat, Rick Cory, Eric Cousineau, Hongkai Dai, Ching-Hsin Fang, Kunimatsu Hashimoto, Muhammad Zubair Irshad, Masha Itkina, Naveen Kuppuswamy, Kuan-Hui Lee, Katherine Liu, Dale McConachie, Ian McMahon, Haruki Nishimura, Calder Phillips-Grafflin, Charles Richter, Paarth Shah, Krishnan Srinivasan, Blake Wulfe, Chen Xu, Mengchao Zhang, Alex Alspach, Maya Angeles, Kushal Arora, Vitor Campagnolo Guizilini, Alejandro Castro, Dian Chen, Ting-Sheng Chu, Sam Creasey, Sean Curtis, Richard Denitto, Emma Dixon, Eric Dusel, Matthew Ferreira, Aimee Goncalves, Grant Gould, Damrong Guoy, Swati Gupta, Xuchen Han, Kyle Hatch, Brendan Hathaway, Allison Henry, Hillel Hochsztein, Phoebe Horgan, Shun Iwase, Donovon Jackson, Siddharth Karamcheti, Sedrick Keh, Joseph Masterjohn, Masayuki Masuda, Jean Mercat, Patrick Miller, Paul Mitiguy, Tony Nguyen, Jeremy Nimmer, Yuki Noguchi, Reko Ong, Aykut Onol, Owen Pfannenstiehl, Richard Poyner, Leticia Priebe Mendes Rocha, Gordon Richardson, Christopher Rodriguez, Derick Seale, Michael Sherman, Mariah Smith-Jones, David Tago, Pavel Tokmakov, Matthew Tran, Basile Van Hoorick, Igor Vasiljevic, Sergey Zakharov, Mark Zolotas, Rareș Ambruș, Kerri Fetzer-Borelli, Benjamin Burchfiel, Hadas Kress-Gazit, Siyuan Feng, Stacie Ford, and Russ Tedrake. A careful examination of large behavior models for multitask dexterous manipulation. *Science Robotics*, 2026.

Kang Chen, Zhihao Liu, Tonghe Zhang, Zhen Guo, Si Xu, Hao Lin, Hongzhi Zang, Xiang Li, Quanlu Zhang, Zhaofei Yu, Guoliang Fan, Tiejun Huang, Yu Wang, and Chao Yu. $\pi_{\texttt{RL}}$: Online RL fine-tuning for flow-based vision-language-action models, 2026.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. PaLM-E: An embodied multimodal language model. In *International Conference on Machine Learning (ICML)*, 2023.

Jonas Eschmann. Reward function design in reinforcement learning. *Reinforcement learning algorithms: Analysis and Applications*, 2021.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In *International Conference on Machine Learning (ICML)*, 2018.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning (ICML)*, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, 2018.

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 (ICLR)*, 2022.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research (JMLR)*, 2016.

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. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024.

Clare Lyle, Zeyu Zheng, Khimya Khetarpal, James Martens, Hado van Hasselt, Razvan Pascanu, and Will Dabney. Normalization and effective learning rates in reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Guozheng Ma, Lu Li, Haoyu Wang, Zixuan Liu, Pierre-Luc Bacon, and Dacheng Tao. What makes value learning efficient in residual reinforcement learning? *arXiv preprint arXiv:2602.10539*, 2026.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning (CoRL)*, 2021.

Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. MimicGen: A data generation system for scalable robot learning using human demonstrations. In *Conference on Robot Learning (CoRL)*, 2023.

Lassi Meronen, Martin Trapp, and Arno Solin. Periodic activation functions induce stationarity. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning (ICML)*, 1999.

Manu Orsini, Anton Raichuk, Léonard Hussenot, Damien Vincent, Robert Dadashi, Sertan Girgin, Matthieu Geist, Olivier Bachem, Olivier Pietquin, and Marcin Andrychowicz. What matters for adversarial imitation learning? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Daniel Palenicek, Florian Vogt, Joe Watson, and Jan Peters. Scaling off-policy reinforcement learning with batch and weight normalization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Daniel Palenicek, Florian Vogt, Joe Watson, Ingmar Posner, and Jan Peters. XQC: Well-conditioned optimization accelerates deep reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2026.

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.

Yury Polyanskiy. Information theoretic methods in statistics and computer science: Lecture 1 — f-divergences, 2020.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Allen Ren, Justin Lidard, Lars Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. In *International Conference on Learning Representations (ICLR)*, 2025.

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

Zhanyi Sun and Shuran Song. From prior to pro: Efficient skill mastery via distribution contractive rl finetuning. *arXiv preprint arXiv:2603.10263*, 2026.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT press, 2018.

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

Joe Watson, Sandy H. Huang, and Nicolas Heess. Coherent soft imitation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Wenli Xiao, Haotian Lin, Andy Peng, Haoru Xue, Tairan He, Yuqi Xie, Fengyuan Hu, Jimmy Wu, Zhengyi Luo, Linxi Fan, et al. Self-improving vision-language-action models with data generation via residual RL. In *International Conference on Learning Representations (ICLR)*, 2026.

Xiu Yuan, Tongzhou Mu, Stone Tao, Yunhao Fang, Mengke Zhang, and Hao Su. Policy decorator: Model-agnostic online refinement for large policy model. In *International Conference on Learning Representations (ICLR)*, 2025.

Zhide Zhong, Haodong Yan, Junfeng Li, Junjie He, Tianran Zhang, and Haoang Li. VLA-OPD: Bridging offline sft and online RL for vision-language-action models via on-policy distillation. *arXiv preprint arXiv:2603.26666*, 2026.

A Coherent Soft Imitation Learning

For MDPs with success-based absorbing states, like many manipulation tasks, IRL methods can struggle if their rewards are positive. Agents are incentivized to 'stay alive' and accrue reward, and therefore never complete the task and enter the absorbing state.

To prevent this behavior in CSIL, the coherent reward can be bounded and its value subsequently shifted by its maximum reward, to ensure the coherent reward is always negative. To achieve this bounding of the maximum reward, the authors add a small bias term σ^2_{min} to the predictive variance that enables them to define an upper bound for the likelihood. As the $\tanh$ transformation is also bounded in practice to ensure numerical stability, the authors are then able to derive the following upper bound for the reward $r(\mathbf{s}, \mathbf{a})$, given action dimension d_a, a uniform prior across the action range $[-1, 1]^{d_a}$, and $\alpha = d_a^{-1}$

$$r(\mathbf{s}, \mathbf{a}) \leq -0.5 \cdot \log\left(\pi \sigma^2_{min}/2\right) + \tilde{c},$$

where $\tilde{c}$ depends on the $\tanh$ clipping term (Watson et al., 2023).

B Experimental Details

B.1 Methods, Baselines and Ablations

CSIL. We follow the default hyperparameters and architectural choices described in Watson et al. (2023), changing only the total number of environment steps to 510k.

CSIL++. We use the hyperparameters described in Table 2. In particular, we do not uniformly distribute the bins of the categorical critic. Instead, we calculate the Monte Carlo values for the demonstrations and then distribute the bins using quantiles. We normalize the reward and the entropy coefficient by the standard deviation of the Monte Carlo values. The residual is not conditioned on the base policy's actions, while the critic uses the current state as well as the combined action of the base policy and residual policy as input. We replace the upper bound of the coherent reward function with the maximum reward computed over the demonstration data and additionally clip the reward to ensure all reward values are negative.

XQC + OD. We use the same network sizes, learning rates, and pretraining hyperparameters as for CSIL++. We distribute the bins of the categorical critic uniformly. Since we pretrain the policy, we are able to initialize the replay buffer with on-policy actions without a loss of diversity in data. The residual is not conditioned on the base policy's actions, while the critic uses the current state as well as the combined action of the base policy and residual policy as input.

PLD. We closely follow the hyperparameters from Xiao et al. (2026). Our adjustments adapt the scaling of the residual actions to the range $[-0.1, 0.1]$ and use an initial entropy coefficient of 0.1.

ResFit. We use the public code base[1] of Ankile et al. (2025a), changing only the environment wrappers to run the experiments on RoboMimic and MimicGen environments and the base policy implementation to be able to use `pi-0.5` as a base model for fair comparison.

Policy decorator. Similar to ResFit, we adapt the public code base[2] of Yuan et al. (2025) to run the experiments on RoboMimic and MimicGen environments and use `pi-0.5` as a base model for fair comparison. We adapted the initial entropy coefficient from e^1 to $e^{0.2}$ as we observed large initial performance losses due to a too strong entropy regularization.

[1] https://github.com/amazon-far/residual-offpolicy-rl
[2] https://github.com/tongzhoumu/policy_decorator

B.2 HYPERPARAMETERS

Table 2: Core Hyperparameters for CSIL++

Hyperparameter	**Value**
Architecture & Optimization	
Batch Size	128
Critic Network	[1024, 1024]
HetStat Network	[1024, 1024, 48, 1024]
Learning Rates	3e-4
ResNet Channels	$[16, 32, 64, 128]$
Visual Pooling	Spatial Softmax
Critic Normalization	Batch Normalization
Policy Normalization	Layer Normalization
Categorical Critic Bins	101
Categorical Critic Range	$[-8, 0]$
Distribution of Bins	By Quantile
Training Dynamics	
Warmup Steps	10,000
Total Environment Steps	510,000
Update-To-Data (UTD) Ratio	4
Policy Delay	4
Entropy Coefficient	$0.3/$Std. of MC values
Replay Buffer Size	250,000
Pretraining	
Demonstrations per Env.	200
Policy Pretraining Steps (Image)	25,000
Policy Pretraining Steps (State)	50,000
Critic Pretraining Steps	20,000
Environment Horizon	Standard (Task-dependent)

C Additional Experimental Results

Method	Square	Coffee	Mug Cleanup	Threading	Nut Assembly	Hammer Cleanup
VLA	0.84 [0.82, 0.86]	0.78 [0.76, 0.86]	0.68 [0.66, 0.72]	0.14 [0.12, 0.14]	0.22 [0.22, 0.26]	0.98 [0.98, 0.98]
VLA (2K demos)	N/A	0.90 [0.89, 0.94]	0.76 [0.72, 0.80]	0.02 [0.02, 0.04]	0.54 [0.52, 0.54]	1.00 [0.98, 1.00]
CSIL++ Avg.	**0.96 [0.92, 0.98]**	0.98 [0.96, 1.00]	0.80 [0.68, 0.84]	0.92 [0.90, 0.94]	**0.27 [0.23, 0.32]**	**1.00 [0.98, 1.00]**
CSIL++ Res.	0.94 [0.90, 0.96]	0.78 [0.62, 0.94]	0.66 [0.52, 0.72]	**0.94 [0.92, 0.96]**	0.26 [0.20, 0.30]	**1.00 [1.00, 1.00]**
Policy decorator	**0.96 [0.96, 0.96]**	0.68 [0.56, 0.74]	0.76 [0.72, 0.86]	0.78 [0.76, 0.88]	0.04 [0.02, 0.04]	0.92 [0.92, 0.98]
ResFit	**0.96 [0.96, 0.98]**	**1.00 [0.98, 1.00]**	**0.94 [0.90, 0.96]**	0.00 [0.00, 0.96]	0.00 [0.00, 0.00]	0.98 [0.98, 1.00]
XQC+OD Avg.	0.82 [0.74, 0.88]	0.05 [0.00, 0.43]	0.10 [0.08, 0.12]	0.00 [0.00, 0.02]	0.00 [0.00, 0.00]	0.12 [0.00, 0.56]
XQC+OD Res.	0.66 [0.58, 0.74]	0.28 [0.00, 0.60]	0.12 [0.08, 0.18]	0.12 [0.10, 0.20]	0.00 [0.00, 0.00]	0.20 [0.00, 0.42]
PLD RL (Res.)	0.82 [0.78, 0.84]	0.56 [0.26, 0.62]	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]	0.96 [0.94, 0.98]

Table 3: LBM finetuning success rates reported as median and quartiles on six sparse-reward manipulation tasks. Bold indicates highest median seen over training. Success rates are computed from 5 evaluations with 50 episodes each, and aggregate results across five seeds per environment and method are concatenated.

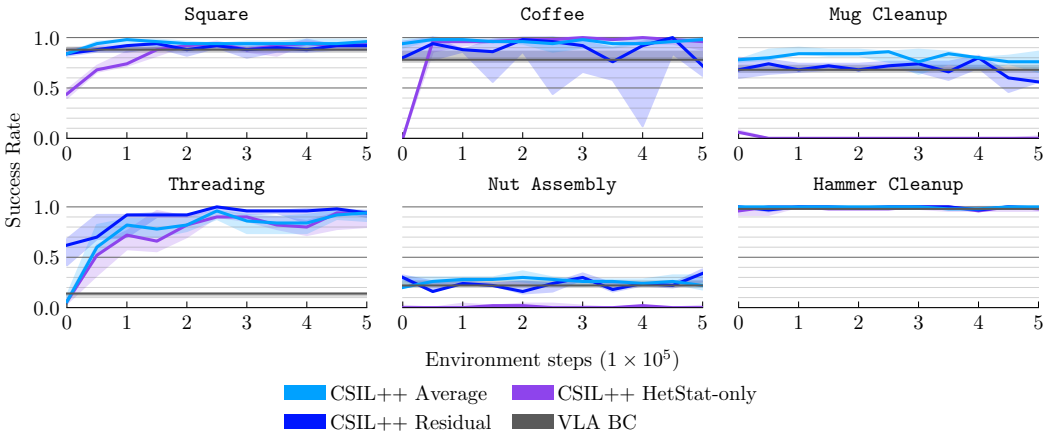

Figure 10: Comparison of CSIL++'s performance on LBM finetuning and HetStat-only finetuning across five seeds. Success rates are reported as medians, and uncertainty ranges are quartiles on six sparse reward manipulation tasks and are computed over 50 evaluations every 50k steps.