Post-Training $\pi_{0.5}$ for Real-Robot Bimanual Manipulation

Technical report for the RSS 2026 Post-Training for Robotics Foundation Models Challenge

Team UTokyo

Haruki Abe (abe@mi.t.u-tokyo.ac.jp) Takehiro Hara (t-hara@mi.t.u-tokyo.ac.jp)
Tatsuya Harada (harada@mi.t.u-tokyo.ac.jp)
Machine Intelligence Laboratory (MIL), The University of Tokyo
posttraining-for-robotics.github.io

Key takeaways.

- Phase 1: first place in both the Single-Task and Multi-Task tracks.
- Core recipe: full-weight fine-tuning of the $\pi_{0.5}$ backbone with global batch size 128 in Phase 1 and 256 in Phase 2, using all available non-failure data without data-type-specific loss weighting.
- HIL recovery data and task-specific reset removal appeared useful in development; the evaluated generic curation methods did not produce large gains.

Abstract. We report our solution for the RSS 2026 Post-Training for Robotics Foundation Models Challenge. We retained the official $\pi_{0.5}$-style architecture and used full-weight fine-tuning with a uniform imitation-learning loss on all available non-failure data. Phase 1 used four 80 GB NVIDIA A100 GPUs and global batch size 128; our submission ranked first in both tracks, with average progress scores of 82.0 (Single-Task) and 68.7 (Multi-Task), compared with 60.3 and 52.3 for the official `pi05` reference policy [11]. HIL recovery behavior and removal of post-success reset segments appeared useful in our development, whereas the curation methods we evaluated did not yield large gains under limited evaluation. In Phase 2, we continued full-weight fine-tuning from the Phase 1 checkpoints with eight 80 GB A100 GPUs and global batch size 256, using all 808 released rollout episodes. The `seal-water-bottle-cap` data received the same reset removal, while the `insert-mouse-battery` policy remained unchanged because its Phase 2 training did not finish before submission. Phase 2 evaluation yielded average progress scores of 67.3 for the generalist policy and 74.3 for the specialist policies.

1. Problem Setup

1.1 Problem setting and challenge phases

The Post-Training for Robotics Foundation Models Challenge studies how a pretrained robot policy can be improved using real interaction data, including successes, failures, and human corrections. Policies are evaluated on standardized real-robot bimanual manipulation tasks. The challenge has two rounds: in **Phase 1**, teams train offline on the released dataset and submit either task-specific policies to the Single-Task track or one shared policy to the Multi-Task track. In **Phase 2**, the top three teams from Phase 1 deploy their own policies, collect additional rollouts with organizer support, and iteratively post-train improved policies. Ranking is by Average Success Rate first, with Average Progress Score as the tie-breaker [1].

1.2 Benchmark tasks

The benchmark contains three bimanual manipulation tasks. Figure 1 shows the final frame of each official task video.

1.3 Released dataset

The standardized Phase 1 dataset is provided in LeRobot format for offline training and contains four complementary sources: expert demonstrations, baseline success rollouts, baseline failure rollouts, and human-in-the-loop (HIL) data. Their roles are summarized in Table 2.

Phase 1 contains three task directories, each split into `expert-data`, `failure-data`, and `success-and-hil-data`. The release contains 3,209 episodes, 8,635,032 frames, and approximately 39.9 hours of synchronized 60 Hz data. Each frame includes a 14-dimensional state and action, three RGB camera streams, and an `observation.commander_state` mode label. HIL and autonomous successes are stored together and distinguished by the per-frame mode [2].

Phase 2 consists of 23 independent LeRobot v2.1 subdatasets across the same tasks. It contains 808 episodes and 4,289,763 frames, covering generalist and specialist policies. Per-frame `observation.commander_state` labels distinguish human teleoperation from autonomous policy inference [3].

This mixture makes the benchmark a post-training problem rather than conventional expert-only imitation learning: a method must decide which rollout behavior should be reinforced, corrected, or removed. The Phase 2 release then provides newly collected on-policy and HIL rollouts from the selected teams.

2. Phase 1 Results

Our submissions ranked first in both Phase 1 tracks (Table 3). Scores are average progress scores; baseline and submission results are from the official leaderboard [11].

3. Method Overview

Summary. We did not introduce a new architecture. We used a $\pi_{0.5}$-style baseline and focused on data mixture, fine-tuning stability, HIL recovery data, and removing `seal-water-bottle-cap` reset behavior.

3.1 Architecture

We followed the organizers' OpenPI baseline and used its $\pi_{0.5}$ configuration rather than introducing a new backbone [5, 6]. The implementation details below are taken from the official baseline [4].

Table 1: Benchmark tasks and progress scoring.

Task	Goal and progress scoring
`insert-mouse-battery`	Position the mouse correctly, insert the battery with the correct polarity, and fully seat it. Partial progress is credited for correct positioning and partial insertion.
`tower-of-hanoi-game`	Move the small ring to the opposite peg, move the large ring to the middle peg, and finally stack the small ring on the large ring. Intermediate placements receive partial credit.
`seal-water-bottle-cap`	Insert the cap's straw into the bottle, place the cap, and tighten it. The final tightening score increases with the number of turns, up to a fully tightened cap.

(a) `insert-mouse-battery`. **(b)** `tower-of-hanoi-game`. **(c)** `seal-water-bottle-cap`.

Figure 1: Final frames from the official top-view challenge videos [1].

3.2 Data mixture

We used all available non-failure data: expert demonstrations, baseline success rollouts, and HIL trajectories. We excluded baseline failure rollouts and one expert trajectory that contained failure behavior. All retained samples used the same imitation-learning objective; we did not assign different loss weights by data type.

3.3 Optimization

We inherited the official OpenPI challenge baseline optimization recipe except for global batch size and training steps [4]. Both generalist and specialist policies were full-weight fine-tuned. Training durations, checkpoint initialization, batch sizes, and hardware are listed in Sections 4.1 and 8.2.

4. What Worked

4.1 Large-batch fine-tuning

We used relatively large-batch fine-tuning. Heterogeneous rollout data produces high-variance updates when batches are small, especially when a batch overrepresents a particular task phase or recovery pattern. Larger batches provide a more stable gradient estimate and make it easier to mix expert, success, and HIL behavior in each update.

The official baseline uses global batch size **32**. Phase 1 used **128** (a **4× increase**) on four 80 GB NVIDIA A100 GPUs. Phase 2 used **256** (an **8× increase**) on eight 80 GB A100 GPUs. All policies used full-weight fine-tuning.

Table 5: Batch size and training resources.

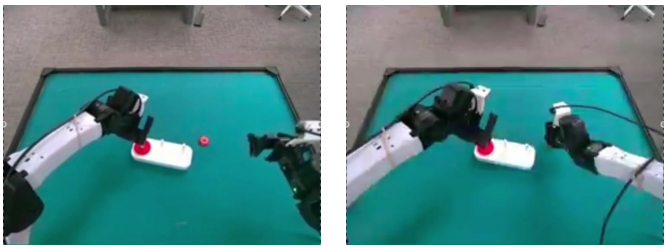
Recipe	Batch	Rel.	GPUs
Official baseline	32	1×	–
Phase 1	128	4×	4×A100
Phase 2	256	8×	8×A100

This choice is consistent with prior work. Large-batch studies show that batch size can increase without necessarily sacrificing data efficiency up to a task-dependent critical batch size [9], and that increasing batch size can play a role similar to reducing learning rate by lowering gradient noise [10]. HPT reports that increasing batch size generally improved heterogeneous robot-policy pre-training until convergence and used batch sizes from 256 to 2,048 [8]. The RLDX-1 ablation compares 64, 256, and 1,024: RoboCasa Kitchen performance rises from 66.9 to 69.6 to 70.6, while GR-1 Tabletop rises from 36.8 to 53.2 to 58.7 [7].

For Phase 1, we full-weight fine-tuned the generalist for **190K steps**, the `insert-mouse-battery` and `tower-of-hanoi-game` specialists for **50K steps** each, and the `seal-water-bottle-cap` specialist for **100K steps**. We treat the batch-size choice as an empirical observation rather than a universal claim: useful batch size should depend on dataset size, model scale, learning rate, and action representation.

4.2 HIL data for recovery

HIL data appeared useful in development when it contained behavior after realistic rollout errors. `tower-of-hanoi-game` is the clearest example. In clean demonstrations, the ring is ideally inserted onto the peg. In real rollouts, insertion can fail, the ring can fall or roll, and the robot must recover by re-grasping and retrying. Training on HIL trajectories exposed the policy to these recoverable states (Figure 3).



(a) Failure state: the ring is displaced after an unsuccessful interaction. **(b)** Recovery: the policy re-engages with the displaced ring.

Figure 3: Example failure and recovery states in a `tower-of-hanoi-game` rollout.

Table 2: Released data components and their roles in post-training.

Component	Description	Role in post-training
Expert data	High-quality human teleoperation demonstrations.	Clean task-completion behavior.
Baseline success rollouts	Trajectories in which the baseline completes the task.	Successful behavior close to the deployed-policy distribution.
Baseline failure rollouts	Trajectories in which the baseline fails.	Available as a negative or filtering signal, but not used in our training mixture.
HIL data	Human interventions, corrections, and preference labels collected during baseline rollouts.	Corrective and recovery behavior after policy errors.

Table 3: Official Phase 1 results.

Track	Rank	Our score	Official `pi05`	Gain	Success rate
Single-Task	1st	82.0	60.3	+21.7	76.7%
Multi-Task	1st	68.7	52.3	+16.4	66.7%

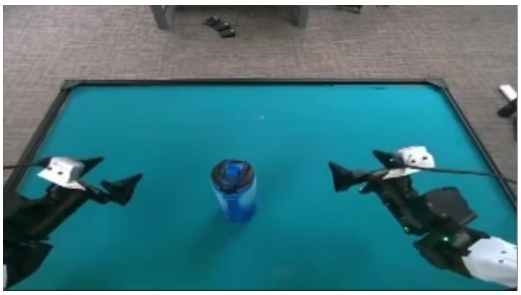
This suggests that HIL data should not be viewed only as "more demonstrations." Its value is highest when it covers the state distribution induced by the imperfect policy itself.

4.3 Removing reset behavior for `seal-water-bottle-cap`

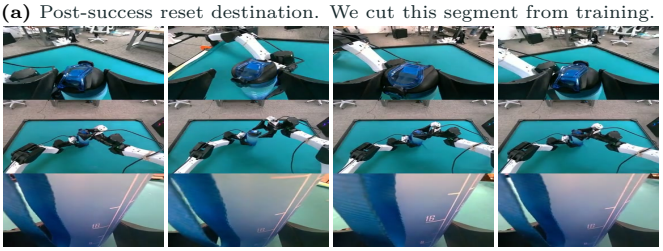
The `seal-water-bottle-cap` task rewards inserting the straw/cap and then tightening the cap, with the final score increasing with the number of tightening turns. From the camera observation alone, however, it is difficult to determine how tightly the cap has already been fastened: the cap, gripper, and bottle appearance changes only subtly across turns and is partially occluded (Figure 4).

Figure 4: Visual ambiguity in `seal-water-bottle-cap`: the image provides little direct evidence of current cap tightness.

In the original trajectories, cap tightening is followed by a post-success reset behavior. Because the policy cannot reliably infer tightness from the image, this transition can teach it to stop tightening or undo progress.



(a) Post-success reset destination. We cut this segment from training.



(b) After reset removal, the policy continues rotating the cap until the environment forces a reset.

Figure 5: Task-specific removal of post-success reset behavior. In the longest observed case, tightening continued for up to eight minutes.

We removed the post-success reset segment from the training data. The resulting policy continued tightening until the evaluator or environment reset the episode. This is not a general-purpose manipulation rule; it is a task- and metric-specific alignment decision.

5. What Was Less Effective

5.1 Splitting HIL trajectories at action discontinuities

Baseline-collected HIL trajectories contained many abrupt action changes. For each timestep, we measured the maximum absolute change across action dimensions, $\max_d |a_{t+1,d} - a_{t,d}|$. The time series contained numerous spikes (Figure 6), which we hypothesized could provide harmful temporal supervision.

#	Team / Model	insert-mouse-battery		tower-of-hanoi-game		seal-water-bottle-cap		Average	
		Score	SR	Score	SR	Score	SR	Score	SR
1	Haruki U-Tokyo trials: 10 / 10 / 10	90	90%	66	50%	90	90%	82	76.7%
2	PengfangQian SII trials: 10 / 10 / 10	90	90%	33	30%	80	60%	67.7	60%
3	VLAlab-JP trials: 10 / 10 / 10	100	100%	40	40%	30	30%	56.7	56.7%
4	Zhangyu Huazhong trials: 10 / 10 / 10	86	70%	59	50%	38	30%	61	50%
5	TongJiang cityu trials: 4 / 5 / 5	70	50%	25	20%	70	60%	53.9	42.9%

(a) Single-Task track: first with average score 82.0.

#	Team / Model	insert-mouse-battery		tower-of-hanoi-game		seal-water-bottle-cap		Average	
		Score	SR	Score	SR	Score	SR	Score	SR
1	Haruki U-Tokyo trials: 10 / 10 / 10	80	80%	66	60%	60	60%	68.7	66.7%
2	Zecheng ICL trials: 10 / 10 / 10	80	80%	52	40%	55	50%	62.3	56.7%
3	VLAlab-JP trials: 13 / 13 / 13	98.5	92.3%	48.5	46.2%	34.6	23.1%	60.5	53.8%
4	PengfangQian SII trials: 10 / 12 / 10	96	80%	44.2	41.7%	40	40%	59.1	53.1%
5	TongJiang cityu trials: 12 / 13 / 13	71.7	58.3%	23.1	23.1%	69.2	61.5%	54.2	47.4%

(b) Multi-Task track: first with average score 68.7.

Figure 2: Official Phase 1 leaderboards [11].

Table 4: Policy input, output, and initialization details inherited from the baseline.

Item	Configuration
Initialization	$\pi_{0.5}$ base, gs://openpi-assets/checkpoints/pi05_base/params
Challenge configs	`pi05_insert-mouse-battery`, `pi05_seal-water-bottle-cap`, and `pi05_tower-of-hanoi-game`
Action horizon	50 steps, inherited from the default `Pi0Config`.
Observation cameras	Three RGB views: overhead `cam_high`, left wrist `cam_left_wrist`, and right wrist `cam_right_wrist`; resized to 224×224.
State/action dimensions	14 physical dimensions: six joints plus one gripper value for each arm. OpenPI pads tensors to the model's 32-dimensional action space. Joint actions are deltas; gripper commands remain absolute.

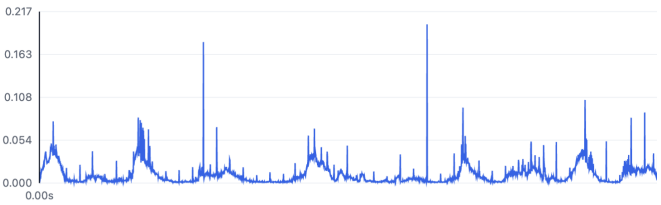
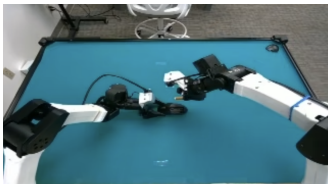


Figure 6: Action discontinuities in baseline-collected HIL data. The horizontal axis is time; the vertical axis is $\max_d |a_{t+1,d} - a_{t,d}|$. Large spikes indicate abrupt changes in at least one action dimension.



(a) Right-side insertion: the dominant strategy retained for training.

(b) Left-side insertion: the rarer strategy removed from the strategy-unified dataset.

Figure 7: Two expert-data strategies for `insert-mouse-battery`.

We split trajectories at detected spikes and trained with the resulting segments. Evaluation performance did not change, so we did not use this curation in the final recipe. The model may have tolerated these discontinuities, or splitting may have removed useful temporal context without addressing task-level causes of errors.

5.2 Extracting human-correction segments from HIL trajectories

In an HIL trajectory, the autonomous policy typically makes a mistake before the human operator takes over. We hypothesized that behavior cloning on this pre-intervention model behavior could be harmful, even though the subsequent teleoperation contains a useful correction. We extracted only the `teleop` intervals and trained on those segments instead of full trajectories. Evaluation performance became slightly worse, so the final model used the full retained HIL trajectories.

5.3 Strategy unification for `insert-mouse-battery`

The original expert data contained 832 trajectories, but one trajectory included failure behavior and was removed, leaving 831. The retained trajectories used two insertion strategies: from the right or from the left. A rule-based classifier estimated that **62 of 831** retained trajectories used the rarer left-side strategy. We removed those trajectories to reduce multimodality.

Evaluation performance did not change. Although the filtered dataset was more behaviorally consistent, strategy unification did not improve the evaluated policy. We therefore treat this as a task-specific negative result rather than a general curation principle.

6. Limitations

- **No controlled ablation study.** Competition-time constraints prevented systematic ablations that isolate each design choice while holding all other variables fixed.
- **High evaluation variance.** Each setting was evaluated with only a small number of real-robot trials. Techniques reported as not effective may still be beneficial under more extensive evaluation.
- **Training was not run to full loss convergence.** The effect of a technique may differ after longer optimization; for example, data curation could become useful once competing runs are fully converged.

7. Practical Takeaways

1. Strong post-training does not always require changing the VLA backbone. A strong pretrained policy plus the right data mixture can be highly competitive.
2. HIL recovery data appeared useful in our development. The most useful trajectories covered states produced by imperfect policies, not only clean task completions.

3. The evaluated curation heuristics did not yield large gains. Action-discontinuity splitting, teleoperation-only HIL filtering, and strategy unification did not measurably improve performance under limited evaluation.
4. Large-batch fine-tuning was practical in development. It can make each update reflect the intended mixture of tasks, sources, and behavior modes.

8. Phase 2 Method and Results

Phase 2 extends the Phase 1 policy through organizer-supported deployment, additional on-policy rollout collection, and iterative post-training. We report the rollout composition, post-training method, and evaluation results below.

8.1 Additional rollout collection

The released Phase 2 dataset contains 23 LeRobot v2.1 sub-datasets, 808 episodes, and 4,289,763 frames recorded at 60 Hz, corresponding to approximately 19.86 hours. These rollouts cover generalist and specialist policies across the three tasks [3].

The release contains no separate failure-data subset. For each policy that completed Phase 2 post-training, we combined all **808 episodes** released across participating teams with the corresponding non-failure Phase 1 mixture. All samples used the same imitation-learning objective without data-type-specific loss weights.

8.2 Phase 2 post-training method

Before Phase 2, we continued Phase 1 training beyond the checkpoints used for the Phase 1 submission. These later checkpoints were used only to initialize Phase 2 post-training. We retained the architecture and used full-weight fine-tuning. We changed the data, increased global batch size from 128 to 256, and used eight 80 GB NVIDIA A100 GPUs. The checkpoint step denotes the source Phase 1 checkpoint; subsequent steps are additional Phase 2 post-training steps.

Table 6: Phase 2 initialization, training, and policy-specific data treatment.

Policy	Phase 2 update and treatment
Generalist	Phase 1 350K checkpoint → 60K additional steps; no additional processing.
`tower-of-hanoi-game`	Phase 1 190K checkpoint → 40K additional steps; no additional processing.
`seal-water-bottle-cap`	Phase 1 100K checkpoint → 60K additional steps; remove post-success reset behavior as in Phase 1.
`insert-mouse-battery`	Phase 2 training did not finish; submit the Phase 1 policy unchanged.

8.3 Phase 2 evaluation

Each task was evaluated with 10 trials. The reported value is the average progress score across the three benchmark tasks.

Table 7: Phase 2 evaluation.

Policy	Iteration	Score
Generalist	3	**67.3**
Specialists	2	**74.3**

Despite additional Phase 2 post-training, the generalist and specialist scores were 1.4 and 7.7 points lower than their respective Phase 1 results. Given the limited number of trials and lack of controlled ablations, we cannot determine whether these differences were caused by evaluation variance, the Phase 2 data mixture, or the training configuration.

9. Future Work

We would like to investigate offline reinforcement learning methods that can use the baseline failure trajectories excluded from our behavior-cloning mixture. Such methods may extract learning signals from unsuccessful interactions without directly imitating the actions that caused the failures.

Acknowledgments

This work was partially supported by JST Moonshot R&D Grant Number JPMJPS2011.

References

[1] Post-Training for Robotics Foundation Models Challenge website. https://posttraining-for-robotics.github.io/

[2] Post-Training for Robotics Foundation Models Challenge. *Challenge Phase 1 Dataset*. https://huggingface.co/datasets/Posttraining-RFM-RSS2026/Challenge-phase1-dataset

[3] Post-Training for Robotics Foundation Models Challenge. *Challenge Phase 2 Rollouts Dataset*. https://huggingface.co/datasets/Posttraining-RFM-RSS2026/Challenge-phase2-rollouts-dataset

[4] Post-Training for Robotics Foundation Models Challenge. *OpenPI Baseline*, commit `3cf4747fcc2d9b8023eb43b681b141f93527d65e`. https://github.com/posttraining-for-robotics/openpi-baseline/tree/3cf4747fcc2d9b8023eb43b681b141f93527d65e

[5] Physical Intelligence. $\pi_{0.5}$*: A Vision-Language-Action Model with Open-World Generalization*. arXiv:2504.16054, 2025.

[6] Physical Intelligence. *OpenPI repository*. https://github.com/Physical-Intelligence/openpi

[7] Kim et al. *RLDX-1 Technical Report*. arXiv:2605.03269, 2026.

[8] L. Wang, X. Chen, J. Zhao, and K. He. *Scaling Proprioceptive-Visual Learning with Heterogeneous Pre-trained Transformers*. NeurIPS, 2024.

[9] S. McCandlish et al. *An Empirical Model of Large-Batch Training*. arXiv:1812.06162, 2018.

[10] S. L. Smith et al. *Don't Decay the Learning Rate, Increase the Batch Size*. ICLR, 2018.

[11] Post-Training for Robotics Foundation Models Challenge. *Phase 1 leaderboard*, website commit `fc94bc171414584e22a956b4494a65b6c0e33341`. https://github.com/posttraining-for-robotics/posttraining-for-robotics.github.io/blob/fc94bc171414584e22a956b4494a65b6c0e33341/index.html