Extended Abstract

Motivation Low-cost arms such as the SO-101 make robot learning more accessible, but they also make learned manipulation unforgiving: policies must be compact, fast, and robust to shifts in object pose, target pose, lighting, viewpoint, appearance, and dynamics. The central problem is transfer. Direct image-to-action imitation can be brittle because sparse action labels may encourage copying expert motions without learning the underlying state or transferable object/target geometry. When the environment changes, such a policy may not know where the object and target are in the sense needed for control. We ask whether a compact SO-101 student can learn both action and perception, so transfer depends on estimated scene state rather than hidden visual shortcuts.

Method We introduce the *Perception-Factored Policy* (PFP), a compact architecture that separates scene understanding from control. The student receives RGB and proprioception, while a privileged simulator teacher supplies action labels and hidden scene-state labels. PFP predicts object pose, target pose, object-target displacement, end-effector-object displacement, scalar reach/transport distances, and grasp/held state, then routes control only through this predicted bottleneck and proprioception. Object/target information therefore cannot bypass the supervised scene-state interface.

Implementation We evaluate on a hardened MuJoCo SO-101 pick-and-place benchmark designed to require visual scene understanding. The student observes a 96×96 overview RGB image and an 11-D proprioceptive vector; the privileged bottleneck is supervised with a 15-D simulator state. A scripted privileged oracle provides reactive pick-and-place actions and state labels. We compare PFP against same-scale direct image-to-action policies, ACT, and a visual-bypass student (DextrAH-G-inspired), and we use filtered DAgger to relabel student-visited states while filtering unstable rollout data.

Results On 600-episode/domain evaluations, PFP with filtered DAgger reaches 0.96 train success and 0.33 target success, outperforming Concat with DAgger ($0.65/0.01$) and ACT with DAgger ($0.45/0.23$). Without DAgger, PFP reaches $0.84/0.32$, so filtered relabeling mainly improves source-domain recovery while leaving target transfer nearly flat. Mechanism studies support the architecture: removing privileged-observation supervision lowers train success to 0.53 and raises cube error to 176 mm, while the forced bottleneck keeps cube error to 25 mm. A learned-blind poster sanity check confirms that the task cannot be solved from proprioception alone.

Discussion Filtered DAgger improves source-domain recovery but leaves target transfer nearly flat (0.32 to 0.33), so aggregation alone does not solve held-out transfer. Prediction overlays expose cube/target estimates, and failures correlate with larger perception error, making the learned bottleneck inspectable. PFP is small enough to be practical for low-cost hardware: the main policy has 5.2M parameters and available CPU timing artifacts around 1.5–1.9 ms/step. An 8-bit weight-only quantization sweep preserves success while shrinking the model from 20.9 MB to 5.2 MB; 4-bit compression breaks both perception and control.

Conclusion The central lesson is architectural. The best result does not come from the largest model or from giving the controller more raw visual features. It comes from forcing visual information through a supervised, inspectable scene-state bottleneck before control, pointing toward efficient manipulation policies whose internal perception can be measured, debugged, and improved rather than hidden inside a monolithic image-to-action mapping.

PFP: A Perception-Factored Policy for Robust and Efficient SO-101 Manipulation

CS224R Spring 2026

Amirreza Zeinali
Department of Computer Science
Stanford University
azeinali@stanford.edu

Shobhit Agarwal
Department of Computer Science
Stanford University
shobhit1@stanford.edu

Abstract

We study compact vision-based manipulation on the SO-101 arm. The problem is transfer: a policy that maps images directly to actions can imitate expert motions on familiar scenes while failing to learn the object/target state needed in a shifted environment. We ask whether a small student policy transfers better if it is forced to predict task-relevant scene geometry before control. We propose the *Perception-Factored Policy* (PFP), which predicts a privileged scene-state bottleneck from RGB and proprioception and conditions the controller only on that prediction. The privileged state is available from simulation during training and contains object pose, target pose, relative geometry, distance features, and grasp state; at deployment the policy receives only image and proprioception. On a hardened pick-and-place benchmark, PFP with filtered DAgger reaches 0.96 train and 0.33 target success, outperforming same-scale direct image-to-action and larger ACT baselines. A learned-blind sanity check confirms that the benchmark requires visual scene understanding, while a lambda sweep and visual-bypass ablation (DextrAH-G-inspired) show that privileged-state supervision and forced bottleneck routing are both important. Prediction overlays and perception-error analyses make the learned geometry inspectable, and an 8-bit weight-only quantization sweep preserves success at $4\times$ smaller model size. These results suggest that perception-factored architectures are a promising path toward efficient, interpretable visual manipulation on low-cost hardware.

1 Introduction

Low-cost robot arms such as the LeRobot SO-101 create an attractive setting for learned manipulation: the hardware is inexpensive, the action space is modest, and image-based behavior cloning can be trained without elaborate model-based pipelines. The difficulty is transfer. A policy must be small enough to run quickly on consumer hardware, yet robust to changes in object pose, target pose, lighting, texture, viewpoint, and object dynamics. In such settings, direct image-to-action imitation can succeed for the wrong reasons: sparse action labels may be sufficient to imitate an average trajectory on familiar scenes, while providing little pressure to learn the underlying object/target state. We believe this missing state grounding is one reason translation to a new environment fails: the policy copies expert motions without learning transferable geometry. We ask whether a compact SO-101 student can learn both action and perception.

This paper studies that failure mode through a compact architecture for simulated SO-101 pick-and-place. We evaluate only on a hardened benchmark: cube and target locations are randomized by 5 cm,

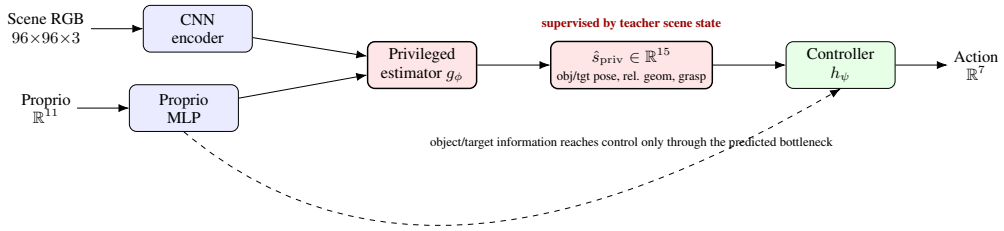


Figure 1: **Perception-Factored Policy.** PFP predicts a privileged scene-state bottleneck from image and proprioception, then acts through the predicted state. The dashed arrow is proprioception only; raw visual object/target features are not allowed to bypass the bottleneck.

grasp and placement tolerances are 4.5 cm, and policies must use a full-scene camera to localize both object and target. A learned-blind Concat control with RGB zeroed fails in a smaller poster sanity check, so the learned policies cannot solve the task from proprioception alone.

Our method, shown in Figure 1, is the *Perception-Factored Policy* (PFP). Instead of asking one network to infer perception and control implicitly, PFP predicts a compact privileged scene state from image and proprioception, then routes the controller through that prediction. The predicted state includes object pose, target pose, object-target displacement, end-effector-object displacement, and grasp state. This is supervised by a privileged simulator teacher, whose ground-truth state and action labels are free in simulation. Crucially, visual object/target information is not allowed to bypass the bottleneck. The controller sees proprioception and the predicted scene state, not raw visual features.

The core claim is that this architectural factorization matters more than raw capacity. On the hardened benchmark, PFP outperforms a same-scale direct image-to-action network, ACT, and a visual-bypass variant (DextrAH-G-inspired). The latter is especially important: we give the student the original image features together with proprioception and an estimated privileged state, using our own architecture for a fair comparison. This baseline resembles prior privileged-distillation manipulation systems in spirit, but allowing a raw visual pathway into the action head does not improve performance. The forced bottleneck is the stronger design choice.

Our contributions are:

1. **A perception-factored policy class.** PFP turns privileged simulator state into the deployment-time interface between vision and control: RGB and proprioception are used to predict scene geometry, and actions are conditioned on that prediction.
2. **A hardened SO-101 manipulation benchmark.** The benchmark includes a learned-blind control, which establishes that the learned policies cannot succeed by ignoring visual information.
3. **A controlled architecture study.** We compare PFP to direct image-to-action policies, ACT, and a visual-bypass baseline (DextrAH-G-inspired), plus a sweep over the privileged-observation loss weight.
4. **Interpretability and efficiency analyses.** We visualize student predictions of cube and target locations, connect failures to larger perception error, and show that 8-bit weight-only quantization preserves success while reducing model size by $4\times$.

2 Related Work

Privileged information for visuomotor control. Learning with privileged information studies settings where additional state or teacher explanations are available during training but absent at deployment [1]. In embodied control, Learning by Cheating trains a privileged agent with access to ground-truth layout and traffic state, then distills it into a vision-only sensorimotor agent [2]. Asymmetric Actor-Critic similarly exploits simulator state, but only asymmetrically during reinforcement learning: the critic receives full state, while the actor is trained from rendered observations and is deployed without privileged state [3]. PFP differs in how privileged information enters the final

policy. It does not merely use privileged state to improve a teacher, critic, or training signal; it trains the deployable student to explicitly predict the task-relevant privileged scene state and then conditions control on that prediction.

Teacher-student distillation and auxiliary state prediction. Recent sim-to-real systems often distill privileged policies into sensor-based students. Policy distillation [4, 5] and DAgger-style relabeling [6] provide general tools for transferring behavior from stronger supervisors to weaker deployed policies. Robust perceptive locomotion, for example, trains a teacher with privileged simulation data and distills a student that imitates actions while reconstructing ground-truth environment state from noisy observations [7]. DextrAH-G is the closest manipulation prior: it trains a privileged fabric-guided policy in simulation and distills it into a depth-based student that imitates actions and predicts object position [8]. DextrAH-RGB extends the recipe to RGB dexterous grasping [9]. These works demonstrate the power of privileged distillation, but the learned object estimate is auxiliary or used by a downstream controller, while raw visual features can still directly inform the action policy. PFP makes a different architectural commitment: object and target information reach control only through a compact predicted scene-state interface, making perception a supervised, inspectable, and causal bottleneck rather than an auxiliary loss on a pixel-to-action policy.

Direct visuomotor imitation. End-to-end imitation methods such as ACT, Behavior Transformer, and Diffusion Policy have improved visuomotor behavior cloning by modeling temporal action chunks, multimodal demonstrations, or action distributions directly from observations [10, 11, 12]. These methods build on a long line of visuomotor imitation and large-scale robot learning systems [13, 14, 15]. ACT was introduced for low-cost bimanual hardware and reduces effective horizon by predicting chunks of actions. Behavior Transformer models multimodal continuous behavior from demonstrations, and Diffusion Policy represents robot behavior as conditional action diffusion. PFP is complementary to these action-modeling advances: rather than increasing the capacity of a direct image-to-action learner, it factors the policy into supervised perception and compact control. This factorization is especially relevant for low-cost arms, where appearance shift, camera shift, limited demonstrations, and runtime constraints can make monolithic visual policies brittle.

Domain randomization and transfer. Domain randomization is a standard strategy for making policies robust to nuisance variation in simulation before deployment [16, 17]. We use it in a narrower way: train and target simulators define a controlled sim-to-sim transfer test, while a learned-blind control verifies that the learned baselines cannot succeed from proprioception alone. This lets us isolate the architectural question before claiming real-world robustness.

Positioning. Overall, prior work uses privileged information mainly as a training-time teacher, a critic-side value signal, or an auxiliary prediction target for a visual student. PFP instead turns privileged state into the deployment-time interface between vision and control. The student predicts object pose, target pose, relative geometry, and grasp state from RGB and proprioception, and the controller acts only through this predicted bottleneck. This gives PFP a distinct role among privileged-learning approaches: it is not just distillation from a better teacher, but a perception-factored policy class designed for compactness, interpretability, and transfer on resource-constrained manipulation hardware.

3 Method

3.1 Task, observations, and teacher

We use the official SO-101 MuJoCo model for tabletop pick-and-place. Each episode randomizes the cube and target poses; success requires the cube to remain within the placement tolerance for five consecutive simulator steps. The student observes a 96×96 RGB image from an overview camera and an 11-dimensional proprioceptive vector containing joint positions/velocities and gripper opening. The student never receives object or target positions directly.

The teacher is a scripted privileged oracle that has access to simulator state: true object pose, target pose, robot state, relative geometry, and grasp/held status. It executes a reactive pick-and-place sequence: move above the cube, descend, close/activate the grasp, lift, move above the target, lower,

and release when the object is within tolerance. The teacher provides two kinds of supervision: the action target and the hidden scene-state target used to train PFP's bottleneck.

3.2 Perception-Factored Policy

The privileged state missing from the deployment observation is a 15-dimensional vector

$$s_{\text{priv}} = [p_{\text{obj}}^{(3)},\ p_{\text{tgt}}^{(3)},\ (p_{\text{obj}} - p_{\text{tgt}})^{(3)},\ (p_{\text{ee}} - p_{\text{obj}})^{(3)},\ \|p_{\text{ee}} - p_{\text{obj}}\|^{(1)},\ \|p_{\text{obj}} - p_{\text{tgt}}\|^{(1)},\ g^{(1)}],$$

where p_{obj}, p_{tgt}, and p_{ee} are the 3D world-frame positions of the object, placement target, and robot end-effector, respectively, and g is the grasp/held flag. A direct policy would regress $a = f_\theta(\text{image}, \text{proprio})$. PFP instead decomposes the computation into perception and control:

$$z = [\text{CNN}(I),\ \text{MLP}(q)], \tag{1}$$

$$\hat{s}_{\text{priv}} = g_\phi(z), \tag{2}$$

$$a = h_\psi(\hat{s}_{\text{priv}}, q). \tag{3}$$

Here I is the RGB image and q is proprioception. The controller receives object/target information only through $\hat{s}_{\text{priv}}$; visual features do not bypass the bottleneck. We train with behavior cloning plus observation distillation,

$$\mathcal{L} = \underbrace{\|a^{\text{cont}} - \hat{a}^{\text{cont}}\|^2 + \text{BCE}(a^g, \hat{a}^g)}_{\text{action distillation}} + \lambda_{\text{priv}} \underbrace{\left(\|s_{\text{priv}}^{\text{cont}} - \hat{s}_{\text{priv}}^{\text{cont}}\|^2 + \text{BCE}(g, \hat{g})\right)}_{\text{observation distillation}}.$$

We use $\lambda_{\text{priv}} = 1$ unless otherwise noted. The CNN is a compact NatureCNN-style encoder, and the full PFP has 5.2M parameters with roughly 1.5–1.9 ms CPU inference in our available timing artifacts.

3.3 Baselines and controls

We compare against three policy families. **Concat image-to-action** uses the same image encoder but feeds visual and proprioceptive features directly to an action MLP, removing the bottleneck constraint. **ACT** is a stronger temporal image-to-action baseline with more parameters. **Visual bypass (DextrAH-G-inspired)** is our bypass ablation: it predicts $\hat{s}_{\text{priv}}$ but also gives the controller the raw image feature vector, proprioception, and estimated privileged state. In the experiment artifacts this appears as the `bottleneck_bypass` / `pfp_bypass_lambda_1` policy. We use our own architecture for this baseline so the comparison is fair: the main difference is whether visual information can bypass the supervised scene-state interface.

We also sweep $\lambda_{\text{priv}} \in \{0, 0.1, 0.5, 1, 2, 5\}$ to measure how strongly observation distillation should be weighted. This tests whether PFP's gains come from the privileged-state supervision, the bottleneck routing constraint, or both.

4 Experimental Setup

4.1 Data and evaluation domains

Training data consists of oracle demonstrations in the randomized training simulator. Each sample contains RGB, proprioception, teacher action, and privileged-state labels. We also evaluate policies trained with and without filtered DAgger: student-visited states are relabeled by the privileged teacher, but off-distribution or low-quality rollout data is filtered before aggregation to reduce the influence of unstable student behavior. We present this only as a with/without comparison, not as an iterative training story.

The target simulator is a held-out domain with shifted camera pose/FOV, lighting, object/table colors, cube size, mass, and friction. The action task remains the same, so target success measures transfer under visual and physical shift. Each reported policy setting was run with multiple random seeds. All headline success rates use 600 episodes per domain and Wilson 95% confidence intervals unless otherwise noted.

Table 1: Architecture comparison on the hardened benchmark. Success rates use 600 episodes per domain unless noted; brackets are Wilson 95% confidence intervals. The learned-blind row is a 100-episode/domain poster sanity check with RGB zeroed.

Policy	Training	Params	Train succ.	**Target succ.**
PFP (ours)	w/ DAgger	5.2M	**0.96** [0.94, 0.97]	**0.33** [0.29, 0.37]
PFP (ours)	w/o DAgger	5.2M	0.84 [0.81, 0.87]	0.32 [0.29, 0.36]
Visual bypass (DextrAH-G-inspired)	w/o DAgger	5.7M	0.80 [0.77, 0.83]	0.26 [0.22, 0.29]
Concat image-to-action	w/o DAgger	5.9M	0.53 [0.49, 0.57]	0.16 [0.13, 0.19]
Concat image-to-action	w/ DAgger	5.9M	0.65 [0.61, 0.68]	0.01 [0.01, 0.03]
ACT	w/o DAgger	9.2M	0.28 [0.25, 0.32]	0.19 [0.16, 0.22]
ACT	w/ DAgger	9.2M	0.45 [0.41, 0.49]	0.23 [0.20, 0.27]
Learned blind Concat	w/o DAgger	5.9M	0.00 [0.00, 0.04]	0.00 [0.00, 0.04]

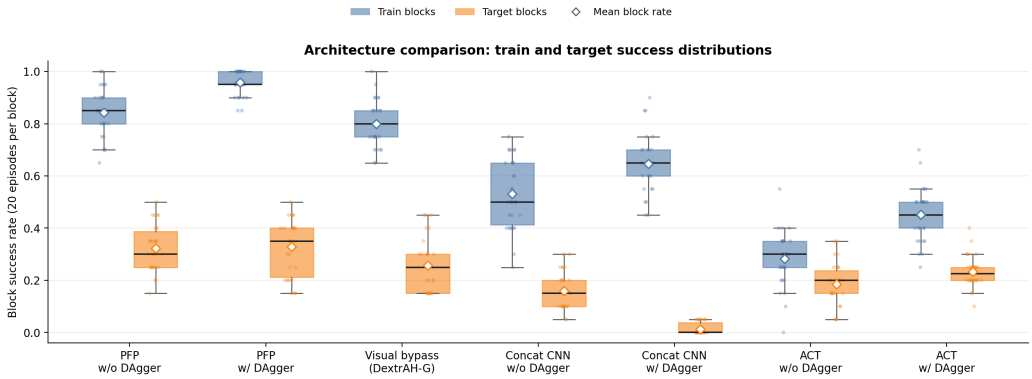


Figure 2: **Train/target success distributions.** PFP has the highest train success and the strongest target result among learned policies. The box-and-whisker plot uses 20-episode blocks for policies with per-episode data.

5 Results

5.1 Quantitative Evaluation

5.1.1 Architecture comparison on the hardened benchmark

Table 1 and Figure 2 summarize the main result in a fixed order: headline PFP performance, same-regime baselines, the effect of filtered DAgger, and the visual-dependence sanity check. The best PFP model, trained with filtered DAgger, reaches 0.96 train success and 0.33 target success, outperforming Concat with DAgger (0.65/0.01) and ACT with DAgger (0.45/0.23). Even without DAgger, PFP reaches 0.84/0.32, ahead of the same-scale Concat image-to-action baseline (0.53/0.16), ACT (0.28/0.19), and the visual-bypass student (DextrAH-G-inspired, 0.80/0.26). The small target change from 0.32 to 0.33 is a useful negative result: filtered relabeled data improves source-domain recovery but does not close the held-out target gap. Finally, the learned-blind sanity check verifies that the benchmark requires visual scene understanding. The architecture, not the data aggregation loop, is the strongest signal.

5.1.2 Does the bottleneck do real work?

Table 2 shows that privileged-observation supervision is not cosmetic. With $\lambda_{\text{priv}} = 0$, train success drops to 0.53 and cube prediction error rises to 176 mm. Moderate supervision improves both control and scene estimation, with $\lambda_{\text{priv}} = 1$ performing best overall. Larger values improve neither target transfer nor perception error, suggesting that over-weighting state prediction can compete with action learning.

The architecture comparison above also makes the visual-bypass result informative. That baseline receives image features, proprioception, and the estimated privileged state in the action pathway, matching the common pattern in privileged-distillation systems where raw visual observations still directly condition actions. Its lower target success and larger cube error (47 mm vs. 25 mm) suggest

Table 2: Observation-distillation lambda sweep. All rows use the same 600-episode/domain protocol. Cube error is mean predicted cube-position error. The visual-bypass architecture baseline is shown in Table 1.

Policy	Params	Train succ.	**Target succ.**	Cube err.
PFP, $\lambda_{\text{priv}} = 0$	5.2M	0.53 [0.49, 0.56]	0.26 [0.22, 0.29]	176 mm
PFP, $\lambda_{\text{priv}} = 0.1$	5.2M	0.78 [0.74, 0.81]	0.29 [0.25, 0.33]	38 mm
PFP, $\lambda_{\text{priv}} = 0.5$	5.2M	0.81 [0.78, 0.84]	0.27 [0.24, 0.31]	28 mm
PFP, $\lambda_{\text{priv}} = 1$	5.2M	**0.84** [0.81, 0.87]	**0.32** [0.29, 0.36]	**25** mm
PFP, $\lambda_{\text{priv}} = 2$	5.2M	0.79 [0.76, 0.82]	0.23 [0.20, 0.27]	28 mm
PFP, $\lambda_{\text{priv}} = 5$	5.2M	0.77 [0.74, 0.81]	0.20 [0.17, 0.23]	32 mm

Table 3: Post-training weight-only quantization of PFP. This is a 40-episode/domain sweep used to assess compression sensitivity.

Bits	Size	Compression	Train succ.	Target succ.	Cube err.
FP32	20.9 MB	1.0×	0.70	0.425	63.6 mm
8-bit	5.2 MB	4.0×	0.70	0.425	64.9 mm
6-bit	3.9 MB	5.3×	0.60	0.425	72.0 mm
4-bit	2.6 MB	8.0×	0.15	0.20	110.9 mm

that forcing object/target information through the supervised bottleneck is better than allowing an unconstrained visual shortcut.

We also ran a poster-only learned-blind sanity check in which a Concat policy receives proprioception with RGB frames zeroed. On 100 episodes per domain it reaches 0.00 success on both train and target, reinforcing that the policy comparisons are testing visual use rather than proprioceptive memorization. Because this control uses a smaller evaluation budget than the main 600-episode protocol, we treat it as supporting evidence rather than a headline table row.

5.1.3 Efficiency and quantization

The PFP architecture is compact: 5.2M parameters, with available CPU timing artifacts around 1.5–1.9 ms/step. ACT is larger (9.2M parameters) and has a measured latency of about 4.7 ms/step in the available ACT evaluation artifact. We also test post-training weight-only quantization of PFP. Table 3 and Figure 3 show that 8-bit quantization preserves success while shrinking the model from 20.9 MB to 5.2 MB. At 6-bit, train success drops but target success is unchanged in the small sweep; at 4-bit, both success and perception error degrade sharply.

5.2 Qualitative Analysis

5.2.1 What does the student perceive?

Because PFP predicts an explicit scene state, we can inspect its internal beliefs. Figure 4 overlays predicted and true cube/target locations. In train scenes, the predictions are often within 10–30 mm; in target scenes, errors increase under the shifted camera, lighting, and dynamics. Figure 5a shows the same trend quantitatively: failures have substantially larger object/target perception error than successes. This is exactly the behavior we want from an interpretable bottleneck: when control fails, we can often see that the scene estimate was wrong.

6 Discussion

The main lesson is that architecture matters. PFP is not the largest model we test, and it is not the only model trained with privileged supervision. Its advantage comes from the forced interface between perception and control: the controller must act through a predicted scene state. This makes the policy more interpretable and appears to reduce reliance on brittle visual shortcuts.

At the same time, target transfer remains modest. PFP improves target success over Concat and ACT, but 0.32–0.33 target success is not robust deployment. The filtered DAgger comparison reinforces this point: relabeled student states improve train success but do not meaningfully improve target

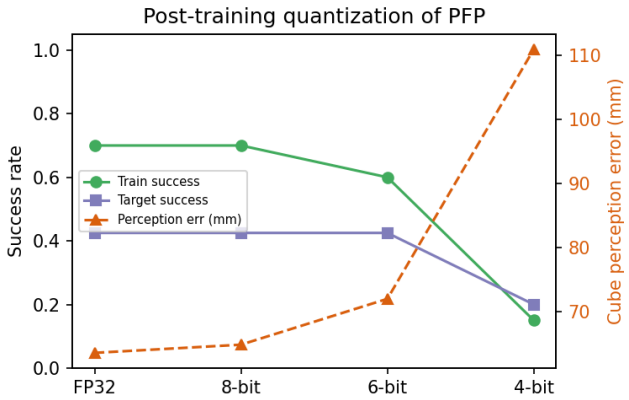
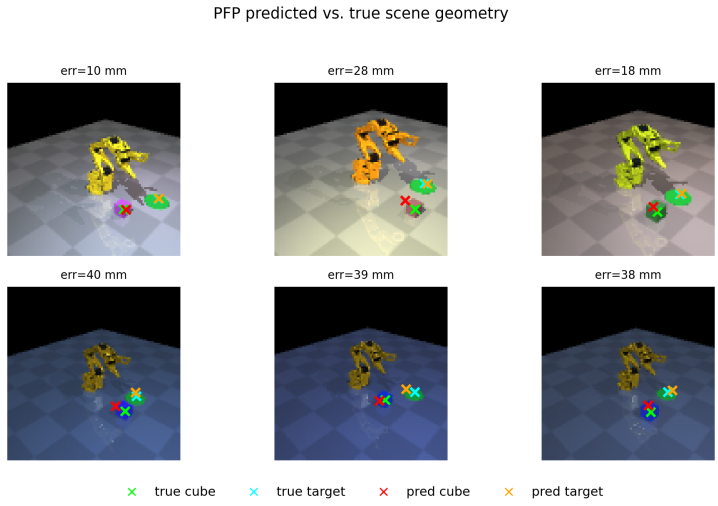
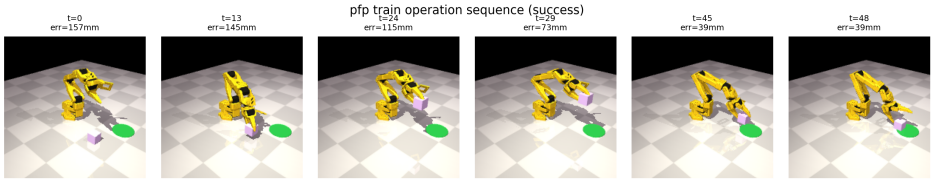


Figure 3: **Quantization sensitivity.** PFP is effectively lossless at 8-bit in this quantization sweep, while 4-bit compression breaks both control and perception.



(a) Predicted vs. true cube and target locations.

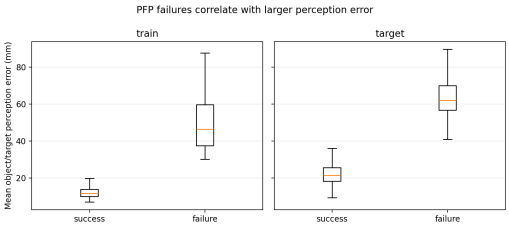


(b) A successful PFP rollout.

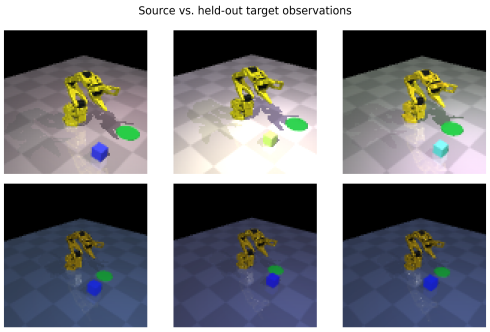
Figure 4: **PFP visualizations.** The learned bottleneck exposes what the student believes about the scene, and rollout frames show how those predictions support closed-loop pick-and-place.

transfer. Future work should stabilize aggregation further with better sampling schedules, source-data reweighting, or trust-region-style re-distillation.

There are also important scope limits. The privileged labels are easy to obtain in simulation but not in real-only data. The current bottleneck covers a single cube and target; clutter, occlusion, multiple objects, deformables, and contact-rich tasks will require richer scene representations. Finally, the quantization experiment is weight-only; hardware-specific int8 export and latency should be measured before making deployment claims. Real SO-101 validation remains future work.



(a) Failures correlate with larger perception error.



(b) Source vs. held-out target observations.

Figure 5: **Analysis visualizations.** Target scenes shift viewpoint, lighting, color, and dynamics; PFP failures correspond to larger predicted-geometry errors.

7 Conclusion

We presented PFP, a perception-factored architecture for compact SO-101 manipulation. PFP predicts task-relevant privileged scene state from RGB and proprioception, then routes control through that prediction. On a hardened benchmark with a learned-blind control verifying visual dependence, this architecture outperforms direct image-to-action baselines, ACT, and a visual-bypass student (DextrAH-G-inspired). Ablations show that both privileged-state supervision and forced bottleneck routing matter, while prediction visualizations expose what the student believes about object and target locations. PFP remains small and quantization-tolerant, making it a promising direction for efficient visual manipulation on low-cost hardware. The next step is real SO-101 validation and richer bottlenecks for cluttered, multi-object manipulation.

8 Team Contributions

Changes from Proposal. In the proposal, Amirreza Zeinali was assigned simulation environment setup (MuJoCo/LeRobot sim), domain randomization, Stage A RL training, sim evaluation infrastructure, and quantitative analysis of iterative improvement curves. Shobhit Agarwal was assigned teacher-model integration (Octo/privileged oracle), the Stage B–C distillation pipeline, real-robot teleoperation and deployment on the SO-101, real-world evaluation, and final report writing. We changed this breakdown because real-robot access, teleoperation data collection, and reliable hardware evaluation carried higher cost and schedule risk than expected. The final project therefore pivoted to a controlled simulation study of the core architectural question: whether forcing visual information through a predicted scene-state bottleneck improves compact manipulation policies.

Amirreza Zeinali led the MuJoCo/LeRobot SO-101 simulation environment setup, hardened benchmark design, domain randomization implementation, and evaluation infrastructure. He implemented and analyzed the train/target simulator split, learned-blind validation, quantitative evaluation protocol, and the plots/tables used for architecture comparison, lambda sweeps, and quantization analysis. Relative to the proposal, his role expanded from Stage A training and improvement-curve analysis into benchmark construction, simulation reliability, and quantitative analysis of PFP and baselines.

Shobhit Agarwal led the privileged-teacher integration, PFP architecture design, observation-distillation objective, visual-bypass baseline (DextrAH-G-inspired), ACT/direct-policy comparisons, and paper writing. He also developed the student prediction and rollout visualizations used to interpret the learned bottleneck. Relative to the proposal, his real-robot teleoperation/deployment and real-world evaluation responsibilities shifted into PFP architecture development, privileged-state supervision, sim-to-sim evaluation, and paper synthesis.

9 Implementation Details

The implementation uses the official LeRobot/TheRobotStudio SO-101 MuJoCo model with kinematic stepping for data collection and evaluation. The image policy observes the overview camera

because it keeps both the randomized cube and target in frame; proprioception contains arm joint positions, joint velocities, and gripper opening. The scripted teacher is reactive rather than phase-indexed, so it can relabel student-visited DAgger states. The main report artifacts are produced from the evaluation summaries in `outputs/feedback_experiments/`, the quantization sweep in `outputs/v2_quant_bottleneck.json`, and the figures in `report/figures/`.

References

[1] V. Vapnik and A. Vashist. A new learning paradigm: Learning using privileged information. *Neural Networks*, 2009.

[2] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl. Learning by Cheating. *CoRL*, 2019.

[3] L. Pinto, J. Davidson, and A. Gupta. Asymmetric Actor Critic for Image-Based Robot Learning. *RSS*, 2018.

[4] G. Hinton, O. Vinyals, and J. Dean. Distilling the Knowledge in a Neural Network. 2015.

[5] A. A. Rusu et al. Policy Distillation. *ICLR*, 2016.

[6] S. Ross, G. Gordon, and D. Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. *AISTATS*, 2011.

[7] J. Lee et al. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 2020.

[8] Tyler Ga Wei Lum, Martin Matak, Viktor Makoviychuk, Ankur Handa, Arthur Allshire, Tucker Hermans, Nathan D. Ratliff, and Karl Van Wyk. DextrAH-G: Pixels-to-Action Dexterous Arm-Hand Grasping with Geometric Fabrics. *CoRL*, 2024.

[9] Ritvik Singh, Arthur Allshire, Ankur Handa, Nathan Ratliff, and Karl Van Wyk. DextrAH-RGB: Visuomotor Policies to Grasp Anything with Dexterous Hands. 2024.

[10] T. Zhao, V. Kumar, S. Levine, and C. Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. *RSS*, 2023.

[11] N. M. M. Shafiullah et al. Behavior Transformers: Cloning k modes with one stone. *NeurIPS*, 2022.

[12] C. Chi et al. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. *RSS*, 2023.

[13] S. Levine et al. End-to-End Training of Deep Visuomotor Policies. *JMLR*, 2016.

[14] A. Mandlekar et al. What Matters in Learning from Offline Human Demonstrations for Robot Manipulation. *CoRL*, 2021.

[15] A. Brohan et al. RT-1: Robotics Transformer for Real-World Control at Scale. 2022.

[16] J. Tobin et al. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. *IROS*, 2017.

[17] X. B. Peng et al. Sim-to-Real Transfer of Robotic Control with Dynamics Randomization. *ICRA*, 2018.

[18] E. Todorov, T. Erez, and Y. Tassa. MuJoCo: A physics engine for model-based control. *IROS*, 2012.