Vision-Language-Action Models: Adaptive and Uncertainty-Gated Policy Distillation for Robotic Manipulation

Nicolas Mello[1]

[1]Instituto de Matemática e Estatística (IME) – Universidade de São Paulo (USP)
Rua do Matão, 1010 – 05508-090 – São Paulo – SP – Brazil

n.mello@ime.usp.br

Abstract. *The application of transformer-based models to robotics has led to Vision-Language-Action (VLA) models: networks that take camera images and natural-language instructions as input and output robot actions directly, and that generalize remarkably well across tasks. Their large size relative to conventional approaches, however, makes them costly for real-time control and hard to parallelize. A novel improvement is to distill the knowledge of a large VLA into a small expert policy invoked per task, a technique known as Refined Policy Distillation (RPD). RPD itself has two limitations: it weights the teacher's influence equally throughout training regardless of how competent the student has become, and it trusts every teacher prediction equally even when the teacher is uncertain. We address both limitations with an adaptive schedule that reduces the teacher's influence as the student improves, and an uncertainty-gated loss that discounts unreliable teacher predictions.*

1. Introduction

Teaching robots contact-rich manipulation has traditionally followed two paradigms. The first is **imitation learning through teleoperation**, where a human operator guides the robot through each trajectory and the recorded data trains a policy via supervised learning. This produces high-quality demonstrations but scales poorly: every new task requires fresh human effort, and the policy inherits the operator's biases. The second is **Reinforcement Learning (RL)** in simulation, where an agent learns by trial and error to maximize a reward. On-policy methods such as Proximal Policy Optimization (PPO) [Schulman et al. 2017], which constrains updates to a trusted region for stability, produce competent task-specific policies given a well-shaped reward and enough experience. RL avoids the data-collection bottleneck, but the resulting policies are narrow: one trained to pick up a cube generally cannot push it, and rarely transfers across environments.

A third approach applies transformer architectures to robotics, in the form of **Vision-Language-Action Models (VLAs)**. A VLA integrates a visual encoder that turns camera images into tokens, a language model that turns text instructions into tokens, and an *action expert* that consumes both streams and produces action vectors. Trained on internet-scale vision-language data and large-scale robotics datasets, models such as Octo [Ghosh et al. 2024], OpenVLA [Kim et al. 2024], and $\pi_{0.5}$ [Black et al. 2025] generalize across tasks and embodiments in a way pure RL policies cannot. The cost is twofold. First, VLAs are large (Octo has 93 M parameters, OpenVLA 7 B), making them expensive at the rates required for closed-loop control. Second, they typically need task-specific **Supervised Fine-Tuning (SFT)** on a new environment, inheriting the weakness

of behavioral cloning [Ross et al. 2011]: performance is bounded by the demonstration data, and small errors compound as the robot drifts into unseen states. Figure 1 shows the inputs and outputs of a VLA.

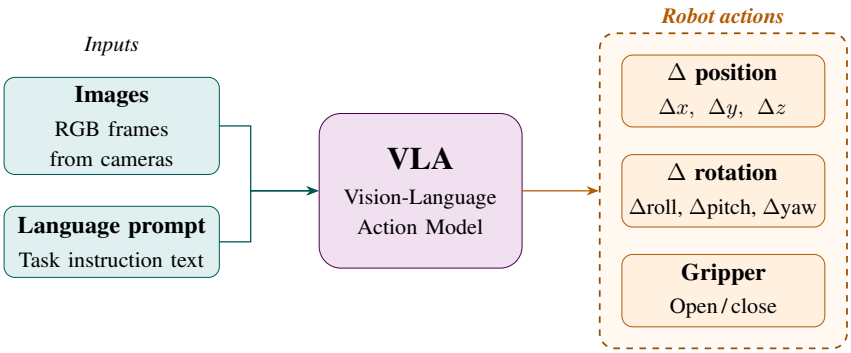


Figure 1. Vision-Language-Action model: inputs (images and language prompts) and outputs (robot control actions) i.e., the robot end-effector actions (△ position, △ rotation, gripper).

Refined Policy Distillation (RPD). A natural way to combine RL and VLAs is distillation, using a fine-tuned VLA as behavioral guidance for an RL policy [Julg et al. 2025]. In this teacher–student setting the *teacher* is the large VLA whose behavior we wish to transfer to a much smaller *student* policy trained with PPO. Given a new task, one keeps the VLA frozen and *distills* its knowledge into a compact student through policy distillation [Rusu et al. 2016]. The student can be orders of magnitude smaller (millions versus millions-to-billions of parameters) and, once trained, runs independently of the teacher at inference, eliminating the latency cost. In RPD the student is trained with a composite objective: a PPO term that maximizes reward, plus a Mean Squared Error (MSE) term penalizing the student's predicted action mean for diverging from the teacher's. The MSE term constrains only the mean, not the variance, so the student retains exploration capacity and can ultimately surpass the teacher, a regime the authors call *policy refinement*.

We identify two structural limitations in RPD. First, the distillation weight λ balancing the RL and MSE terms is a fixed scalar throughout training [Julg et al. 2025]: early on the student needs strong guidance because its policy is essentially random, while late in training, after it has matched or surpassed the teacher, the same fixed λ pulls it back toward a worse policy. Second, RPD treats every teacher prediction as equally informative, yet VLA action distributions are multimodal in ambiguous states, where the predicted mean is a poor imitation target. Distillation injects this noise directly into the student without regard for the uncertainty magnitude.

We propose two modifications to the RPD process [Julg et al. 2025] and reimplement RPD in a parallelized simulation ecosystem. Our contributions are:

1. **Adaptive λ schedule.** We replace RPD's fixed distillation weight with a schedule that decays as a function of the student's success rate, comparing a linear and a sigmoid-shaped decay that holds teacher influence high until the student crosses a competence threshold.
2. **Uncertainty-gated distillation.** We weight each distillation update inversely by the teacher's predicted action variance, replacing the uniform MSE of base RPD.

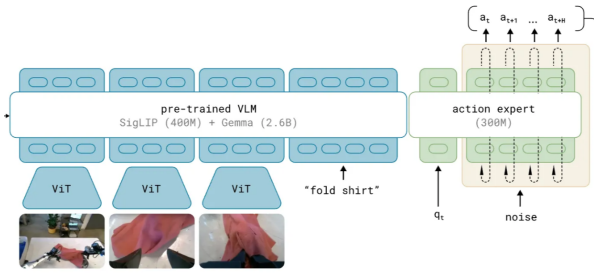


Figure 2. VLA architecture. Image and language tokens are fused through a transformer backbone and decoded into actions by the action expert. (Extracted from [Black et al. 2024].)

Because Octo returns variance alongside the mean in a single forward pass, this adds no inference cost.

3. **Simulator and pipeline.** We replace ManiSkill3 [Tao et al. 2025] with NVIDIA Isaac Lab [Mittal et al. 2025] on a Franka Panda arm. Lacking a scripted expert planner, we replace teleoperated fine-tuning data with successful rollouts from an intermediate PPO policy, as in prior work on PPO-based VLA post-training [Wang et al. 2025], making the pipeline self-contained in simulation.

Section 2 covers background on VLAs and RPD; Section 3 presents our proposal; Section 4 describes the environment and fine-tuning pipeline; Section 5 reports the evaluation; and Section 6 discusses implications and limitations.

2. Background and Related Work

2.1. Vision-Language-Action Models

A VLA maps camera images and a natural-language instruction to robot actions. Figure 2 shows the typical architecture: visual encoders (ViT) that tokenize images, a language model that tokenizes the instruction, and an *action expert* that consumes both streams and outputs action vectors (typically end-effector deltas plus a gripper command). The shared backbone lets the action expert attend jointly across visual and linguistic context, the source of VLAs' cross-task generalization. The visual and language components are usually initialized from a pre-trained Vision-Language Model and fine-tuned via behavioral cloning on datasets such as Open X-Embodiment [O'Neill et al. 2024]. Such models generalize across tasks but underperform task-specific policies on novel environments, so an extra round of in-domain SFT is usually required [Julg et al. 2025].

A detail that matters for our method is that VLAs typically output a *distribution* over actions. Octo, for instance, returns both an action mean and a variance from a single forward pass, the variance reflecting model confidence; this dual output exists because action distributions are often multimodal in ambiguous states.

2.2. Refined Policy Distillation

The RPD distillation loss is the squared error between the student's predicted action mean and the teacher's, over a minibatch of B on-policy transitions:

$$\hat{\mathcal{L}}_{\text{MSE}}(\theta) = \frac{1}{B}\sum_{b=1}^{B}\left\|\mu_\theta(s_b) - a_b^{\text{VLA}}\right\|_2^2, \tag{1}$$

where $\mu_\theta(s_b)$ is the student's predicted mean and a_b^{VLA} the teacher's. The full objective combines this with the PPO clipped surrogate loss $\mathcal{L}_{\text{RL}}$ via a scalar weight λ:

$$\mathcal{L}_{\text{RPD}}(\theta) = \mathcal{L}_{\text{RL}}(\theta) - \lambda \cdot \mathcal{L}_{\text{MSE}}(\theta). \tag{2}$$

Because (2) constrains only the action mean and not its variance, the student retains exploration capacity and can refine past the teacher's mean through the RL term (*policy refinement*).

2.3. Related Work on RL with VLAs

A parallel line of work [Liu et al. 2025, Wei et al. 2025] fine-tunes the VLA itself with online GRPO or PPO over its token output distribution, and RLDG [Xu et al. 2024] uses RL-trained specialists to generate fine-tuning data for the VLA, distilling RL knowledge *into* the generalist. Both improve the VLA but inherit its full inference cost at deployment. RPD makes the opposite trade: it freezes the VLA and trains a separate compact student, specialized to a narrow task family but cheap enough for real-time control.

3. Proposal: RPD with Adaptive Schedule and Uncertainty-Gated Distillation

We propose two modifications to the RPD objective of (2), each targeting one limitation from Section 1. The first replaces the fixed distillation weight with a success-rate-dependent schedule; the second replaces the uniform MSE loss with a variance-weighted version that suppresses ambiguous teacher predictions. The two are independent: the schedule operates on the global loss weight λ, while the gate operates on the per-sample loss term, so they can be applied together or in isolation.

3.1. Adaptive λ Schedule

A fixed scalar λ is mismatched to the training dynamics. Early on the student is essentially random and a moderate λ may be too weak to discover task-relevant behavior; late in training, once the student has matched or surpassed the teacher, the same λ becomes harmful, pulling the student back toward a policy *worse* than its own. We replace the fixed weight with a schedule $\lambda(t)$ that decays as a function of the student's empirically measured success rate $\text{SR}(t) \in [0, 1]$, since the success rate is itself a measurement of how much guidance the student still needs.

Estimating $\text{SR}(t)$. With N parallel environments running asynchronously, episodes finish at different times, so there is no well-defined "last episode." We maintain a rolling buffer of the most recent $K = 128$ completed episodes and define $\text{SR}(t)$ as the fraction of successes in it, updated on every episode termination.

Schedule functions. We compare two forms. The first is linear:

$$\lambda_{\text{lin}}(t) = \lambda_{\max} \cdot \big(1 - \text{SR}(t)\big), \tag{3}$$

which decays teacher influence in direct proportion to competence and adds no hyperparameters beyond $\lambda_{\max}$, but begins reducing influence immediately, even at small success rates that may reflect lucky rollouts. We therefore also evaluate a sigmoid schedule that holds λ near $\lambda_{\max}$ until the student crosses a competence threshold $\text{SR}^\star$:

$$\lambda_{\text{sig}}(t) = \lambda_{\max} \cdot \Big(1 - \sigma\big(k \cdot (\text{SR}(t) - \text{SR}^\star)\big)\Big), \tag{4}$$

where $\sigma(x) = 1/(1 + e^{-x})$ and $k > 0$ controls transition sharpness. We use $\mathrm{SR}^\star = 0.5$ and $k = 10$ in our main experiments. The full adaptive objective, for either schedule, is:

$$\mathcal{L}_{\text{RPD}}^{\text{adapt}}(\theta) = \mathcal{L}_{\text{RL}}(\theta) - \lambda(t) \cdot \mathcal{L}_{\text{MSE}}(\theta). \tag{5}$$

3.2. Uncertainty-Gated Distillation

The MSE loss in (1) treats every teacher prediction as equally informative. This is appropriate when the teacher's distribution is unimodal and sharply peaked, but VLA distributions are multimodal in ambiguous states—the assumption motivating Octo's diffusion-based action head [Ghosh et al. 2024]. In PickCube, for instance, the cube can be grasped from several directions; a teacher that has seen all of them may place mass on each, yielding a mean that lies between modes and corresponds to no valid action. Distilling from such predictions injects noise rather than guidance.

The teacher itself signals when this happens: Octo returns a predicted variance $\sigma^{\text{VLA}} \in \mathbb{R}^d$ alongside the mean $a^{\text{VLA}} \in \mathbb{R}^d$ in one forward pass ($d = 7$: three position deltas, three orientation deltas, gripper command). Variance is elevated when the state is genuinely ambiguous or outside the fine-tuning distribution. We weight each sample's contribution to the loss inversely by the teacher's variance:

$$\mathcal{L}_{\text{UG}}(\theta) = \frac{1}{B} \sum_{b=1}^{B} \frac{\left\| \mu_\theta(s_b) - a_b^{\text{VLA}} \right\|_2^2}{\bar{\sigma}_b^{\text{VLA}} + \varepsilon}, \tag{6}$$

where $\bar{\sigma}_b^{\text{VLA}} = \frac{1}{d} \sum_{i=1}^{d} \sigma_{b,i}^{\text{VLA}}$ averages the predicted variance across action dimensions and $\varepsilon > 0$ is a small stability constant. Confident predictions ($\bar{\sigma}^{\text{VLA}}$ small) approach the standard MSE up to $1/\varepsilon$; uncertain ones are suppressed.

Design choices and cost. A natural alternative is per-dimension gating, but the seven action dimensions are coupled at execution: a confident position command paired with an unconstrained orientation can cause collisions, so we scale each sample as a whole. Cost is negligible—Octo emits mean and variance in the same forward pass, so gating adds only a per-sample average and a division.

3.3. Combined Objective

The two contributions compose into a single objective:

$$\mathcal{L}_{\text{Ours}}(\theta) = \mathcal{L}_{\text{RL}}(\theta) - \lambda(t) \cdot \mathcal{L}_{\text{UG}}(\theta), \tag{7}$$

where $\lambda(t)$ follows either schedule and $\mathcal{L}_{\text{UG}}$ is the uncertainty-gated loss. The schedule controls *when* the teacher is influential during training, while the gate controls *which* of the teacher's predictions are influential at all. Figure 3 illustrates the full pipeline. Isaac Lab provides a reward signal, a low-dimensional state observation (end-effector pose, cube position, goal), an RGB image, and a language instruction. The agent has two components: the frozen VLA teacher (Octo) and the trainable RL student. Only the teacher receives the camera image and language prompt; the student operates exclusively on the low-dimensional state vector. At each step the student outputs a 7-DoF action sent back to the environment while simultaneously querying the frozen teacher. The combined loss $\mathcal{L}_{\text{Ours}}$ is computed from the PPO RL loss, the adaptive $\lambda(t)$ weight, and the uncertainty-gated distillation loss; only the student's parameters θ are updated.

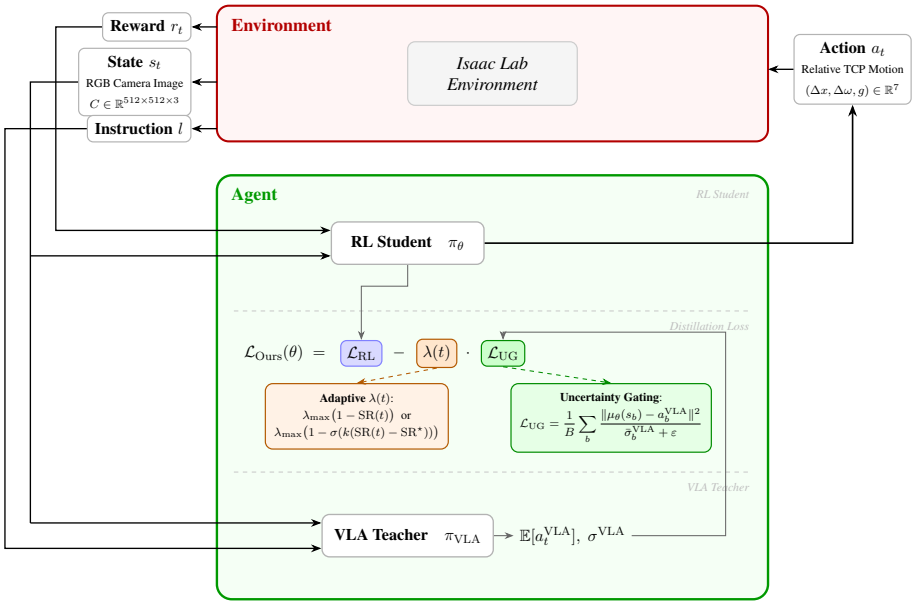


Figure 3. Training pipeline. The Isaac Lab environment provides reward r_t, state s_t, and instruction l. The RL Student π_θ consumes r_t, s_t and acts; the frozen VLA Teacher π_{VLA} consumes s_t, l. The objective $\mathcal{L}_{\text{Ours}}$ combines the RL loss with an adaptive $\lambda(t)$ schedule and an uncertainty-gated loss $\mathcal{L}_{\text{UG}}$.

4. Experimental Setup

4.1. Isaac Lab Environment

We use NVIDIA Isaac Lab [Mittal et al. 2025], which provides GPU-parallelized rigid-body physics on top of Isaac Sim with tightly integrated RL infrastructure, allowing thousands of environment instances on a single GPU. The arm is a Franka Emika Panda 7-DoF arm, modeled through its **Unified Robot Description Format (URDF)** file—the standard XML description of link masses, joint limits, inertial parameters, and torque constraints. Using the same URDF in simulation and on hardware keeps the simulated dynamics close to the real robot (Figure 4).

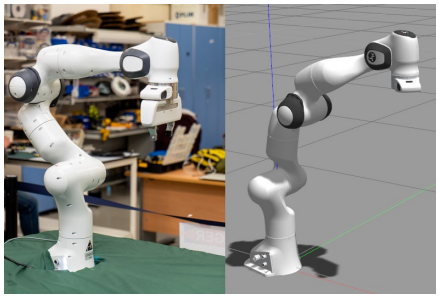


Figure 4. The Franka Emika Panda arm and its URDF representation. Extracted from [Pezzato et al. 2020].

Tasks. We implement three tabletop tasks in Isaac Lab's Manager-Based framework. **PickCube** requires gripping a cube and lifting it to a target in free space, holding for three seconds before success. **PushCube** requires pushing a cube along the tabletop to a target without lifting it. **PullCube** requires pulling a cube toward the arm base to a target. PickCube is the most complex, involving two phases (approach-and-grasp, then lift) that must be chained. Cube and target positions are randomized each episode.

Observations and actions. The robot is controlled through 7-DoF end-effector delta actions $(\Delta\mathbf{p}, \Delta\boldsymbol{\omega}, g) \in \mathbb{R}^7$, where $\Delta\mathbf{p} \in \mathbb{R}^3$ is a Cartesian position delta, $\Delta\boldsymbol{\omega} \in \mathbb{R}^3$ an axis-angle orientation delta, and $g \in [-1, 1]$ the gripper. Observations include the end-

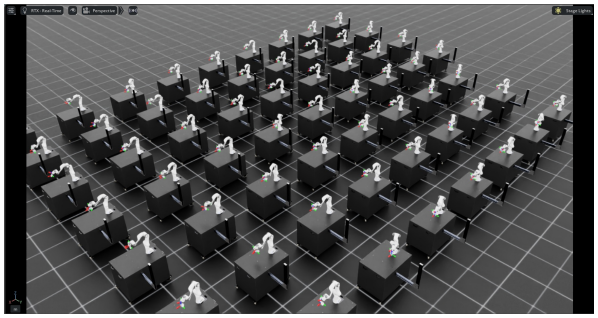


Figure 5. Parallelized simulation environment running many Franka Panda arms in parallel during training. Each arm operates in its own physically isolated workspace.

effector pose, cube pose, goal position, and a 256×256 RGB image from a fixed camera. The image is used only for VLA queries; the student operates on the low-dimensional state vector and never sees the camera image directly. Figure 5 shows the simulation environment with multiple parallel arms during training.

4.2. Octo Fine-Tuning Pipeline

Our teacher is Octo [Ghosh et al. 2024], a 93 M-parameter generalist policy pre-trained on Open X-Embodiment [O'Neill et al. 2024]. To make it useful in Isaac Lab, we fine-tune it on task-specific simulated data.

Fine-tuning data without a scripted planner. Earlier RPD work [Julg et al. 2025] fine-tuned Octo on the ManiSkill3 dataset generated by a scripted planner with privileged state. Isaac Lab ships no such planner, so we instead generate data from intermediate PPO rollouts, following prior work on PPO-based VLA post-training [Wang et al. 2025]: we train a vanilla PPO policy until it reaches a 20–40% success rate, then roll it out across many environments and collect roughly 500 successful episodes per task. Trajectories are converted to the **Reinforcement Learning Datasets (RLDS)** format expected by Octo (each episode a sequence of observation, action, reward, discount steps; the standard format across Open X-Embodiment). Images are downsampled to 256×256, actions expressed in the 7-DoF Cartesian delta format, and Octo is fine-tuned for 50,000 gradient steps.

Docker-based VLA server. Octo runs on JAX while Isaac Lab and our PPO code are PyTorch-based; to avoid dependency and GPU-memory conflicts we serve Octo from a dedicated Docker container exposing `/inference` and `/finetune` REST endpoints, isolating the frameworks and letting the VLA be updated or swapped between phases without restarting Isaac Lab.

Sim-to-sim domain gap. Even after fine-tuning, Octo's action quality is sensitive to differences in lighting, texture, and camera optics relative to its original training data. We mitigate this with Isaac Lab's RTX path-tracing renderer for more photorealistic observations and by fixing the camera pose to match the fine-tuning pose.

4.3. Training Protocol

The student is a small MLP with two hidden layers of 256 units, taking the low-dimensional state and producing a Gaussian over the 7-DoF action space. The PPO setup follows the default Isaac Lab configuration with 1024 parallel environments and a rollout length of

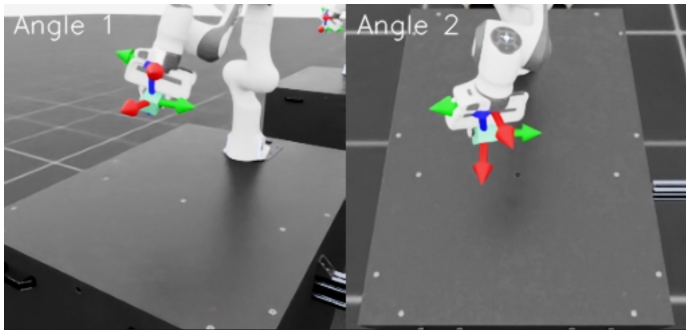


Figure 6. Camera viewpoints from simulation, kept fixed across Octo fine-tuning and inference to minimize the sim-to-sim domain gap.

16 steps ($1024 \times 16 = 16{,}384$ transitions per rollout, split into 8 minibatches of 2,048). The student is trained for 5×10^7 environment steps per task. Figure 6 shows the two fixed camera angles used during Octo inference. For both schedule variants we sweep $\lambda_{\max} \in \{0.5, 1.0, 2.0, 4.0\}$; the sigmoid schedule's threshold $\mathrm{SR}^\star$ and steepness k are set to 0.5 and 10 and swept separately in the ablations. The success buffer holds 128 episodes, and the stability constant is swept over $\varepsilon \in \{10^{-3}, 10^{-2}, 10^{-1}\}$.

5. Experiments

We evaluate the two contributions on PickCube, PushCube, and PullCube. Section 5.1 reports the main comparison against vanilla PPO, base RPD, and the Octo-only baseline; Section 5.2 disentangles the adaptive schedule and the uncertainty gate; and Section 5.3 reports a cross-task generalization observation.

5.1. Main Comparison

We compare six conditions: **Vanilla PPO** (pure on-policy RL, no teacher); **Octo-VLA+SFT** (the fine-tuned Octo evaluated standalone, establishing the ceiling distillation must surpass); **RPD-MSE** (base RPD with fixed λ and uniform MSE [Julg et al. 2025]); **RPD+AL** (base RPD with the sigmoid schedule (4) and standard MSE); **RPD+UG** (fixed λ with the uncertainty-gated loss (6)); and **RPD+AL+UG** (our full method, combining the sigmoid schedule and uncertainty gating (7)). Each variant runs for five seeds on PickCube and two on PushCube and PullCube, due to compute constraints. The primary metric is success rate at convergence, averaged over the final 5×10^6 steps (the last 10% of training) and reported as mean $\pm$ standard deviation across seeds.

Our full method achieves 64.9% on PickCube, a 13.4-point improvement over base RPD (51.6%) and 35.7 points over vanilla PPO (29.2%); consecutive method variants are separated by non-overlapping 95% confidence intervals (Table 1). All RPD variants exceed the Octo SFT ceiling (20.1% on PickCube), confirming that the policy refinement property reported for base RPD [Julg et al. 2025] also holds in our Isaac Lab setup. The teacher–student gap is largest on PickCube (44.8 points) and smallest on PushCube (39.4 points), consistent with the intuition that more complex tasks have more room for refinement beyond the teacher's coverage.

5.2. Ablations

The two contributions are orthogonal: one acts on the loss weight, the other on the per-sample loss. Unless noted, ablation runs use $\lambda_{\max} = 1.0$ and $\varepsilon = 10^{-2}$, near-optimal in the

Table 1. Main comparison across three Isaac Lab tasks. SR (%) at convergence, mean ± std. PickCube uses five seeds; PushCube and PullCube use two. Best per column in bold. 95% confidence intervals (Student's t, five PickCube seeds): RPD-MSE $[50.0, 53.1]$, +Adaptive λ $[55.4, 57.3]$, +Unc. gating $[58.1, 59.2]$, RPD (Ours) $[64.2, 65.7]$; consecutive intervals do not overlap.

Condition	PickCube	PushCube	PullCube
Vanilla PPO	29.2 ± 0.8	30.3 ± 0.6	27.2 ± 1.6
Octo SFT only	20.1 ± 0.9	22.0 ± 0.5	21.4 ± 1.6
RPD-MSE	51.6 ± 1.3	51.8 ± 3.5	49.8 ± 1.1
+ Adaptive λ	56.4 ± 0.8	56.7 ± 2.1	53.1 ± 0.5
+ Unc. gating	58.6 ± 0.5	59.5 ± 1.1	56.0 ± 3.1
RPD (Ours)	$\mathbf{64.9 \pm 0.6}$	$\mathbf{61.4 \pm 2.1}$	$\mathbf{62.4 \pm 1.3}$

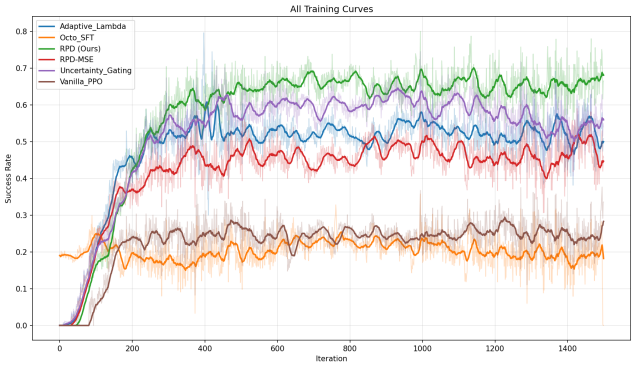


Figure 7. Training curves (success rate vs. training iteration) for all six conditions on PickCube. Shaded regions show range across seeds. Octo SFT is shown as a non-learning baseline since its weights are frozen.

sweeps below.

Adaptive λ versus fixed. The schedule alone improves PickCube from 51.6% to 56.4% (+4.8 points). The gain concentrates late in training, where down-weighting the teacher lets the student refine past its ceiling; early learning is roughly unchanged.

Uncertainty gating versus uniform MSE. Gating alone improves PickCube from 51.6% to 58.6% (+7.1 points). The improvement appears earlier than the schedule's, consistent with the gate filtering noise at every step rather than releasing the student late. Gating alone outperforms the schedule alone, suggesting that filtering ambiguous predictions matters more than timing teacher release in our setup.

Combined effect. The full method reaches 64.9% (+13.4 over base RPD), with both single ablations falling between, consistent with the orthogonality argument of Section 3.3; the five-seed PickCube intervals support the additive decomposition, though it is still interpreted cautiously.

Schedule shape. On PickCube the two schedules perform comparably (linear 56.5 ± 0.8 at two seeds; sigmoid 56.4 ± 0.8 at five seeds), so we claim no performance advantage for either. We adopt the sigmoid schedule on design grounds: it holds teacher influence high through early learning instead of down-weighting immediately at small,

Table 2. Ablation grid on PickCube. SR (%), mean ± std. Dense-reward results use five seeds (linear-schedule row, two seeds); all sparse-reward results use two seeds.

Condition	Dense SR (%)	Sparse SR (%)
Vanilla PPO	29.2 ± 0.8	11.4 ± 1.3
RPD-MSE (fixed λ)	51.6 ± 1.3	39.9 ± 0.4
+ Adaptive λ (linear)	56.5 ± 0.8	41.0 ± 1.5
+ Adaptive λ (sigmoid)	56.4 ± 0.8	43.3 ± 2.2
+ Unc. gating only	58.6 ± 0.5	46.8 ± 0.9
RPD (Ours, full)	$\mathbf{64.9 \pm 0.6}$	$\mathbf{49.5 \pm 1.7}$

possibly lucky success rates, a property we expect to matter more on harder or longer-horizon tasks.

Sensitivity. Sweeping $\lambda_{\max} \in \{0.5, 1.0, 2.0, 4.0\}$ gives $34.8/64.9/54.1/49.8\%$ and $\varepsilon \in \{10^{-3}, 10^{-2}, 10^{-1}\}$ gives $58.8/64.9/59.1\%$ (off-peak points two-seed); both are inverted-U, peaking at $\lambda_{\max} = 1.0$ and $\varepsilon = 10^{-2}$.

Reward density. Under sparse rewards the RPD (Ours)–PPO gap widens from 35.7 to 38.1 points (both drop; Table 2), consistent with teacher guidance mattering more when environment feedback is scarce.

5.3. Cross-Task Generalization

As a preliminary investigation, we ran the full method (RPD+AL+UG) on PullCube using a version of Octo fine-tuned exclusively on PushCube, a related but distinct task, rather than on PullCube itself. The student achieved 64.3% final success, compared to 62.4% for the in-distribution setup where Octo was fine-tuned on PullCube directly. Convergence was noticeably slower but final performance comparable. We report this as a single-seed observation to be interpreted cautiously, but believe it warrants further investigation: the uncertainty gate may suppress enough of the out-of-distribution teacher signal for the student to learn effectively regardless, with the adaptive schedule compensating for weaker early guidance. A controlled multi-seed study is left to future work.

5.4. Computational Cost

On an RTX 4090 a single student forward pass takes 4–6 ms against 20–25 ms for Octo ($\sim$40–50 Hz), so the student is roughly 4–$5\times$ faster. This gap matters only during training: at deployment the teacher is discarded and the student runs standalone, inheriting none of the VLA's inference cost—the central trade against methods that fine-tune the VLA itself [Liu et al. 2025, Wei et al. 2025, Xu et al. 2024] and pay Octo-scale latency at every control step. During distillation, 80–90% of per-step wall-clock is spent querying the frozen teacher rather than the student, making it the dominant training cost; fine-tuning Octo beforehand takes 5 h 40 min on the same GPU. The uncertainty gate adds nothing to either budget—Octo emits the variance in the same forward pass, so gating is one average over the seven action dimensions plus one division, applied only to the training loss and never at deployment.

6. Discussion

Cross-task generalization and primitive reuse. The preliminary result from Section 5.3, where a teacher fine-tuned on PushCube guided a student to near in-distribution performance on PullCube, points to a useful design pattern: rather than fine-tuning a VLA separately for every task, one could fine-tune it on a small set of *primitive* tasks (reach, grasp, push, pull, lift) and rely on uncertainty-aware distillation to handle the teacher–target mismatch. The gate naturally suppresses high-variance out-of-distribution predictions, and the schedule compensates by maintaining teacher influence longer. Whether this scales to larger task families or longer horizons remains open and is the most immediate direction for future work.

Toward a closed, autonomous VLA-RL loop. We fine-tune a VLA on rollouts from an intermediate PPO policy, then distill it into a refined RL student. Superficially circular, yet the final student substantially outperforms both the VLA teacher and the PPO policy: the VLA contributes visual and linguistic priors RL cannot acquire from sparse rewards, while RL refinement contributes environment-specific corrections. Closing this loop iteratively—using the refined student to generate fine-tuning data for the next VLA—is a natural next step. The pipeline is then close to fully autonomous in simulation, the only human inputs being reward shaping and environment design; the main remaining obstacle is the sim-to-real gap, and systems that automatically generate diverse environments and adapt VLAs to them (e.g. RoboCasa [Nasiriany et al. 2024]) are a promising route to scalable physical intelligence.

Limitations. We evaluate three tabletop tasks with a single embodiment; findings may not generalize to more complex contact-rich settings or other robot types. We use Octo as the only teacher, and results with larger models or different action representations may differ. PickCube results use five seeds; PushCube and PullCube use two due to compute constraints. On PickCube, 95% confidence intervals are non-overlapping between consecutive method variants, and the linear and sigmoid schedules are comparable on PickCube (the linear row is two-seed), so we make no performance claim between them. All experiments are in simulation, and sim-to-real transfer of the resulting policies is important future work. Finally, the cross-task result is a single qualitative observation rather than a dedicated study.

7. Conclusion

We presented two targeted improvements to Refined Policy Distillation: an adaptive λ schedule that anneals teacher influence as the student gains competence, and an uncertainty gate that suppresses ambiguous teacher predictions. Both are simple, add no inference overhead, and address structural limitations in base RPD. On three Isaac Lab tasks with Octo as teacher, the full method achieves 64.9% on PickCube against 51.6% for base RPD and 29.2% for vanilla PPO, with similar gains on PushCube and PullCube. Ablations confirm the schedule and gate contribute independently and compose roughly additively, and results under sparse rewards suggest teacher guidance buffers the student against reward sparsity. Together these results support VLA-RL distillation as a practical pipeline for autonomous policy improvement in simulation, with cross-task generalization, iterative training, and sim-to-real transfer as natural next steps.

References

Black, K. et al. (2024). π_0: A vision-language-action flow model for general robot control. arXiv:2410.24164.

Black, K. et al. (2025). $\pi_{0.5}$: A vision-language-action model with open-world generalization. Physical Intelligence, arXiv:2504.16054.

Ghosh, D., Walke, H., Pertsch, K., et al. (2024). Octo: An open-source generalist robot policy. In Robotics: Science and Systems (RSS).

Julg, T., Burgard, W., and Walter, F. (2025). Refined policy distillation: Distilling task-specific expert policies from vision-language-action models. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

Kim, M. J. et al. (2024). OpenVLA: An open-source vision-language-action model. In Conference on Robot Learning (CoRL).

Liu, H. et al. (2025). VLA-RFT: Vision-language-action reinforcement fine-tuning with verified rewards in world simulators. arXiv:2510.00406.

Mittal, M. et al. (2025). Isaac Lab: A GPU-accelerated simulation framework for multi-modal robot learning. NVIDIA, arXiv:2511.04831.

Nasiriany, S., Maddukuri, A., Zhang, L., Parikh, A., Lo, A., Joshi, A., Mandlekar, A., and Zhu, Y. (2024). RoboCasa: Large-scale simulation of everyday tasks for generalist robots. In Robotics: Science and Systems (RSS).

O'Neill, A. et al. (2024). Open X-Embodiment: Robotic learning datasets and RT-X models. In IEEE International Conference on Robotics and Automation (ICRA).

Pezzato, C., Ferrari, R., and Corbato, C. H. (2020). A novel adaptive controller for robot manipulators based on active inference. IEEE Robotics and Automation Letters, 5(2):2973–2980.

Ross, S., Gordon, G., and Bagnell, D. (2011). A reduction of imitation learning and structured prediction to no-regret online learning. In International Conference on Artificial Intelligence and Statistics (AISTATS), pages 627–635.

Rusu, A. A. et al. (2016). Policy distillation. In International Conference on Learning Representations (ICLR).

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. arXiv:1707.06347.

Tao, S., Xiang, F., Shukla, A., et al. (2025). ManiSkill3: GPU parallelized robotics simulation and rendering for generalizable embodied AI. In Robotics: Science and Systems (RSS).

Wang, S.-C., Xiang, T.-Y., Zhou, X.-H., Gui, M.-J., Xie, X.-L., Liu, S.-Q., Wang, S.-Y., Jin, A.-Q., and Hou, Z.-G. (2025). VLA model post-training via action-chunked PPO and self behavior cloning. arXiv:2509.25718.

Wei, K. et al. (2025). π_{RL}: Online RL fine-tuning for flow-based vision-language-action models. arXiv:2510.25889.

Xu, C., Li, Q., Luo, J., and Levine, S. (2024). RLDG: Robotic generalist policy distillation via reinforcement learning. arXiv:2412.09858.