KEEP: KNOWLEDGE-PRESERVING EXPERT DISTILLATION FOR EMBODIED POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

We seek to adapt a pre-trained VLA policy for novel manipulation tasks while preserving previously acquired knowledge. This is a non-trivial task, as naive fine-tuning causes catastrophic forgetting, and existing remedies, such as co-training with replayed pre-training data or merging the task experts with the generalist in the weight space, usually trade retention for acquisition. To address these challenges, we introduce KEEP, a method for novel skill acquisition in generalist VLA policies via the distillation of task experts in the flow-field space. The method proceeds by (i) fine-tuning a lightweight LoRA-based task expert, (ii) computing for every training sample a competence gate that compares the expert's and the frozen generalist's own flow-matching errors, and (iii) training a student, initialized from the generalist, to match the expert's velocity field where the expert is competent and the generalist's field everywhere else. We thoroughly evaluate KEEP on a real robot with 5,150 rollouts, covering generalist evaluations, three single-task settings (bottle pouring, plate stacking, breaker box flipping), and a two-task setting with joint and sequential curricula. Every model is scored on its task suites and on a 100-episode generalist suite against the pre-trained $\pi_{0.5}$ policy, LoRA task experts, full fine-tuning with weight merging, and co-training with replayed pre-training data followed by weight merging. KEEP matches the task experts on the novel tasks while keeping generalist performance within the noise of the pre-trained policy, whereas the strongest baseline loses close to fifteen points. In real-to-sim validated simulation, we further ablate every component of KEEP over more than 250,000 episodes, including the gate, the anchor, and the data mixture. Taken together, our results indicate that the flow field, rather than the weight space, is the more favorable place to compose a task expert with a generalist policy.

1 INTRODUCTION

Large-scale robotic foundation models (Ye et al., 2026a; Black et al., 2025a;b; Kim et al., 2025) promise to solve many tasks within a single model. But in practice, they still need task-specific adaptation for skills that are hard or underrepresented in the pre-training data, and the standard recipes for it, full fine-tuning and LoRA (Hu et al., 2022), degrade the model's generalist capabilities.

This degradation, called catastrophic forgetting (McCloskey & Cohen, 1989; Kirkpatrick et al., 2017), arises because fine-tuning on a narrow task updates shared parameters that the generalist's other skills also depend on, and the objective contains no term that protects them. The two prevailing remedies are replay, which co-trains on a mixture of the new task and a slice of the model's original training data (Robins, 1995; Rebuffi et al., 2017), and weight-space merging, which interpolates the adapted model back toward the generalist (Yadav et al., 2026; Yu et al., 2024). Both fix the balance between acquisition and retention before any input is seen, replay through its mixing ratio and merging through its coefficients (Appendix A.1), with no notion of where the new skill should overwrite the generalist and where the generalist should be preserved. Replay also supervises both halves of the mixture only at noised demonstrations, never at the policy's own actions.

Our key observation is that this balance can be struck per sample and without task labels. A task-specialized expert is reliable where it was trained and unreliable elsewhere, and its flow-matching error relative to the generalist's reveals which is which. We exploit this in KEEP (**K**nowledge-Preserving **E**xpert Distillation for **E**mbodied **P**olicies), which adapts a flow-based VLA to a new

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

skill while preserving its generalist competence. KEEP trains on the same new-task and replay mixture as standard co-training but replaces ground-truth regression with a competence-gated distillation objective over two frozen teachers: a single-task expert and the base generalist. A gate g, computed by comparing the two teachers' flow-matching errors on each sample, pulls the student toward the expert's velocity field where the expert is the better predictor ($g \to 1$, acquiring the skill) and anchors it to the frozen base where it is not ($g \to 0$, preventing forgetting).

We make three contributions: **(i)** we introduce KEEP, a distillation objective that adds a skill to a flow-based VLA by weighting two frozen teachers per sample according to their flow-matching errors, without task labels, injecting the skill along the student's own denoising trajectories and anchoring it to the base policy on the training data; **(ii)** we record 1,220 teleoperated demonstrations of three skills on which $\pi_{0.5}$ largely fails and evaluate KEEP in 5,150 real-robot rollouts. Averaged over tasks, it matches the LoRA experts on the novel skills, exceeds the strongest merging baseline by 24 points of novel-task and 13 points of generalist success rate, and retains generalist success within two points of the base policy; and **(iii)** we ablate every component of KEEP and compare it against the baselines over more than 250,000 episodes in the real-to-sim aligned REALM simulator, finding that the acquired skill resides mostly in the LLM backbone, and repeat the single-task comparison on the LIBERO benchmark with a different base checkpoint, where KEEP again achieves the highest average novel-task success at base-level generalist success.

2 Related Work

Distillation for the acceleration and compression of sampling. A large body of work distills a generative teacher into a student that is sampled in one or a few network evaluations; we group it by the compression mechanism. A first family directly learns the ODE's solution operator that maps noise to data ($x_1 \to x_0$), rather than the per-step velocity that must be numerically integrated, by halving the step count (Salimans & Ho, 2022), directly regressing the noise-to-data map by enforcing self-consistency along the sampling trajectory (Song et al., 2023; Boffi et al., 2025; Tong et al., 2026), or straightening the probability-flow path (Liu et al., 2023b; 2024a). A second family matches the teacher's output distribution using its score as the training gradient by minimizing a score-based divergence to the teacher (Wang et al., 2023; Luo et al., 2023; Yin et al., 2024b;a; Zhou et al., 2024), extending it to flow-matching backbones (Huang et al., 2024; Zhou et al., 2025), or adding an adversarial term (Sauer et al., 2025). Most related to our setting, Score-of-Mixture Training (Jayashankar et al., 2025) estimates the score of a mixture of the real and generated distributions, a quantity close to the responsibility our gate assigns to each teacher but used there for one-step generation, while π-Flow (Chen et al., 2026) distills the teacher into a network-free step policy. A third family applies these mechanisms to robot control, accelerating the diffusion policies (Prasad et al., 2024; Wang et al., 2025) and VLAs, via one-step action generation (Luan et al., 2026), parallel decoding (Song et al., 2025b), or layer skipping in a lighter student (Ye et al., 2026b). Across all three families, the teacher is single, and the goal is to have fewer evaluations or a smaller student. We reuse the same machinery for a different end, blending two teachers so the student retains what one knows rather than sampling one faster.

On-policy vs. off-policy distillation. A second line distills a teacher's behavior into a student rather than compressing its sampler, and we organize it by whose data the student learns from. The on/off-policy axis, which goes back to DAgger (Ross et al., 2011), was established in language-model distillation, where off-policy training trains the student on the teacher's own rollouts (Kim & Rush, 2016), while on-policy training instead lets the student roll out and has the teacher relabel its trajectories (Agarwal et al., 2024; Gu et al., 2024), and the trade-off between the two has been studied directly (Ko et al., 2024). Recent VLA and flow-matching works have largely adopted the on-policy regime, supervising student rollouts with teacher-relabeled actions (Zhong et al., 2026) or advantage-guided self-distillation (Wang et al., 2026). Orthogonal to this axis, a separate line studies what is transferred between teacher and student: consolidating several teachers into one student (Rusu et al., 2016), distilling a specialist into a generalist via RL-generated data (Xu et al., 2025), or specializing a generalist into an expert (Jülg et al., 2025). Closest to us, the concurrent DanceOPD (Zhou et al., 2026), DiffusionOPD (Li et al., 2026) and Flow-OPD (Fang et al., 2026) distill several frozen teachers into one student at states obtained by rolling out the student's own generator to compose capabilities in image generation. All three assign every sample to exactly one teacher based on the task or data partition from which it was drawn, and DanceOPD argues that mix-

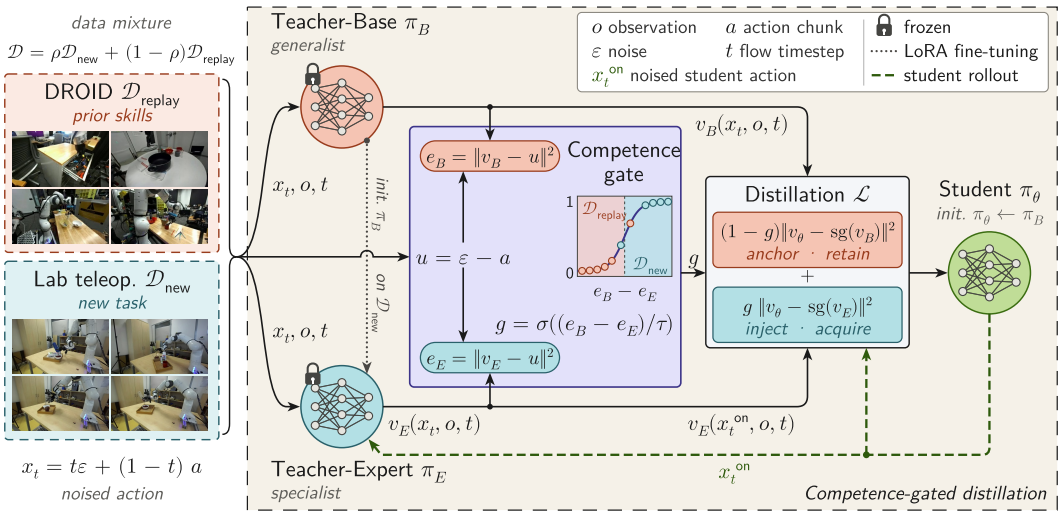

Figure 1: **Method overview.** KEEP trains on a mixture of new-task demonstrations and replayed pre-training data (left). Each noised sample x_t is passed to two frozen teachers, the base generalist π_B and a LoRA task expert π_E fine-tuned from it on $\mathcal{D}_{\text{new}}$. The competence gate compares their flow-matching errors against the demonstration target u and yields $g \to 1$ on new-task samples and, on most replayed ones, $g \to 0$. The student π_θ, initialized from π_B, is then pulled toward the expert's velocity on its own samples x_t^{on} (inject) and toward the generalist's velocity on the training data (anchor), weighted by g and $1-g$.

ing fields within a sample can yield supervision matching no single capability. KEEP instead infers the assignment per sample from the teachers' own errors rather than from the partition. It therefore needs no task labels, and the assignment can cut across the partition, routing replayed samples to the expert where it fits them better. Like these methods, and unlike the VLA distillers above, it needs no environment interaction since its on-distribution inputs come from the student's own probability flow.

Continual learning and knowledge preservation in robotics. Continual learning for robot policies has been pursued largely without teacher distillation, through task-specific adapters (Liu et al., 2024b), autonomous adapter routing and expansion (Römer et al., 2026), knowledge-guided expert routing (Wu et al., 2026), temporal replay with extendable action prompts (Zheng et al., 2025), task-prompt graph evolution (Song et al., 2025a), unsupervised skill discovery (Wan et al., 2024), or dual-critic advantage regulation (Zeng et al., 2026). Teacher distillation is rarely used in this setting, a notable exception being multimodal distillation for lifelong imitation (Roy et al., 2025). A separate line questions whether any special mechanism is needed at all, as large pretrained VLAs appear surprisingly resistant to forgetting under simple replay or sequential RL fine-tuning (Liu et al., 2026; Hu et al., 2026). In contrast to these works, we distill from two teachers at once, a generalist and a specialist, blending them per sample by comparing their flow-matching errors, and without any environment interaction. Our real-robot results also speak to the last line of work above, which draws its evidence from simulation, mainly LIBERO, since on $\pi_{0.5}$ and narrow real-world skills, replay with weight merging still costs about twenty points of generalist success on average.

3 Method

Given a pre-trained flow-based Vision-Language-Action (VLA) generalist and a small set of demonstrations of a new skill, our goal is to obtain a single policy that performs the new skill while retaining the generalist's existing capabilities. Fine-tuning on the narrow new task pulls the shared backbone away from the representations that the generalist relies on. The standard remedies, replay and weight merging, set the balance once and independently of the input, whether as a mixing ratio over the data or as coefficients over the weights. Yet whether the new skill or the generalist should win varies from sample to sample. Comparing the flow-matching error of a task expert with that of the generalist reveals which one is competent on a given sample, and KEEP exploits this by training

a student policy, initialized from the generalist, against two frozen teachers rather than ground-truth actions: the generalist itself and a lightweight single-task expert obtained by LoRA fine-tuning. A competence weight, computed from the teachers' flow-matching errors, splits each training sample between them, so an **inject** term acquires the skill where the expert is reliable, and an **anchor** term holds the student to the generalist everywhere else. Figure 1 gives an overview.

Flow-matching policies. We consider VLA policies that generate actions by flow matching (Lipman et al., 2023; Liu et al., 2023b; Black et al., 2025a). Given an observation o (camera views, language instruction, and proprioceptive state of the robot), the policy produces an action chunk $a \in \mathbb{R}^{N\times D}$ (N steps of D-dimensional actions) by integrating a learned velocity field $v_\theta(x_t, o, t)$ from Gaussian noise at $t = 1$ to a clean action at $t = 0$, where $x_t \in \mathbb{R}^{N\times D}$ is the partially denoised action chunk at time t. Training regresses this field onto the straight-line flow between noise and data. For a demonstration (o, a), noise $\epsilon \sim \mathcal{N}(0, I)$, and time $t \sim p(t)$, with $p(t)$ the generalist's timestep sampler, we define the noised sample as

$$x_t = t\,\epsilon + (1 - t)\,a, \tag{1}$$

with $x_0 = a$ being the original action. The policy is then trained with the objective

$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}_{(o,a)\sim\mathcal{D},\,\epsilon,\,t}\left[\|v_\theta(x_t, o, t) - (\epsilon - a)\|_2^2\right] \tag{2}$$

to regress the velocity field $v = \epsilon - a$. This is both the generalist pre-training objective and the objective used by standard fine-tuning.

Problem setup. We write π_B for the pre-trained generalist with velocity field v_B, and $\mathcal{D}_{\text{new}}$ for the demonstrations of the new skill. Fine-tuning π_B on $\mathcal{D}_{\text{new}}$ with Equation 2 forgets the generalist's capabilities, and co-training on a mixture with the original data only partially recovers them since both halves are regressed with the same objective, and the mixing ratio alone decides where the new skill overwrites the generalist. We seek an objective that makes this decision per sample.

The `KEEP` objective. Instead of regressing the demonstration target, `KEEP` starts the student from the generalist's weights, $v_\theta \leftarrow v_B$, and trains it to reproduce the velocities of two teachers on each sample: the generalist v_B itself and the LoRA expert v_E fine-tuned on $\mathcal{D}_{\text{new}}$. A competence weight $g \in [0, 1]$, computed from the teachers' own flow-matching errors, sets how much each of them supervises the student. Each term evaluates the student and its own teacher at the same noised input, giving

$$\mathcal{L} = \mathbb{E}_{(o,a)\sim\mathcal{D},\,\epsilon,\,t}\Big[\underbrace{g\,\big\|v_\theta(x_t^{\mathrm{on}}, o, t) - \mathrm{sg}\,v_E(x_t^{\mathrm{on}}, o, t)\big\|_2^2}_{\text{inject (acquire)}} + \underbrace{(1 - g)\,\big\|v_\theta(x_t, o, t) - \mathrm{sg}\,v_B(x_t, o, t)\big\|_2^2}_{\text{anchor (retain)}}\Big], \tag{3}$$

where sg is the stop-gradient, so the frozen teachers and the gate act as constants, and gradients flow only through the student. The anchor term is evaluated at the demonstration-noised x_t of Equation 1, whereas the inject term is evaluated at x_t^{on}, which mixes the same noise with the student's own rollout (Equation 7). The gate g is computed at x_t for both terms. We motivate this choice of inputs below. Because g and $1 - g$ sum to one, the two terms share a fixed budget on every sample, and the pull shifts toward whichever teacher fits it better (Appendix A.1). On inputs the expert fits worse than the generalist, $g \to 0$, so its errors there do not reach the student. The demonstration enters only through the gate and, off-distribution, through the noised input x_t, never as a direct regression target, so the student is supervised purely by the two teacher velocities. The rest of this section specifies the teachers, the gate, and the inputs on which each term is evaluated.

Teachers and data mixture. `KEEP` concentrates the new skill in a separate expert whose reliability can be measured per sample. We obtain it by LoRA (Hu et al., 2022) fine-tuning π_B on $\mathcal{D}_{\text{new}}$ and write v_E for its velocity field. The expert forgets much of the generalist's competence in the process, but the gate suppresses it precisely where that shows, so only its skill on the new task reaches the student. The generalist v_B is the second teacher and needs no training. The student is trained on the co-training mixture $\mathcal{D} = \rho\,\mathcal{D}_{\text{new}} + (1 - \rho)\,\mathcal{D}_{\text{replay}}$, where $\mathcal{D}_{\text{replay}}$ is a slice of the generalist's original training data and ρ is the novel-task proportion, so that the student is supervised on both the new task and the states the generalist was trained on. Both teachers are held frozen throughout. The alternative of anchoring to a moving target, such as an exponential moving average of the student, lets the anchor drift along with the student and lose the very behavior it is meant to preserve; Section 4.2 confirms that retention degrades as the anchor is allowed to move.

Competence gating. The expert is reliable on the inputs it was trained on and unreliable elsewhere, and its per-sample flow-matching error identifies which is which. Evaluating the noised sample x_t against the target $u = \epsilon - a$, we compute each frozen teacher's error as

$$e_E = \left\| v_E(x_t, o, t) - u \right\|_2^2, \qquad e_B = \left\| v_B(x_t, o, t) - u \right\|_2^2. \tag{4}$$

What matters for the gate is not how well the expert fits the sample in absolute terms but whether it fits it better than the generalist, so we gate on the comparison of the two errors,

$$g = \operatorname{softmax}\big(\big[-e_E,\, -e_B\big]/\tau\big)_E = \frac{e^{-e_E/\tau}}{e^{-e_E/\tau} + e^{-e_B/\tau}} = \sigma\left(\frac{e_B - e_E}{\tau}\right), \tag{5}$$

where σ is the logistic function and the temperature τ sets how sharply the gap $e_B - e_E$ is resolved. As $\tau \to 0$ the gate becomes the hard indicator $\mathbf{1}[e_E < e_B]$, and Section 4.2 shows that retention is flat and acquisition varies without trend for $\tau \leq 10^{-2}$, while both axes degrade above it. Since g depends only on the frozen teachers and the data, it carries no gradient to the student and measures what each teacher knows about the input rather than the student's own uncertainty.

Routing the two terms. The two inputs in Equation 3 differ only in the action with which the noise is mixed. **Off-distribution**, x_t is the noised demonstration of Equation 1, so the teachers are queried on the data manifold. **On-distribution**, the student first integrates its own probability-flow ODE from the same noise sample ϵ down to $t = 0$,

$$a_\theta = \epsilon - \int_0^1 v_\theta(x_s, o, s)\, \mathrm{d}s, \qquad x_1 = \epsilon, \quad \dot{x}_s = v_\theta(x_s, o, s), \tag{6}$$

and x_t is built from that self-generated action instead,

$$x_t^{\text{on}} = t\,\epsilon + (1 - t)\,\operatorname{sg} a_\theta, \tag{7}$$

so the teachers are queried along the straight path from ϵ to the student's own endpoint, at the cost of one ODE integration per sample. Here, on-distribution refers to the action space only: the observation o always comes from the training data. The gate is always computed at the demonstration-noised x_t, since it requires the target $u = \epsilon - a$, so it never sees the student's own action. The two terms call for different choices. The skill is best injected on inputs the student actually visits, since that is where the student's errors on the new task appear, so `KEEP` evaluates the inject term on-distribution. The generalist is most trustworthy on the inputs it was trained on, so the anchor term is evaluated off-distribution. Section 4.2 ablates all four assignments and finds that the other three trade worse.

Multi-task adaptation. The same objective extends to more than one new skill in three ways, all evaluated in Section 4.1. Each uses Equation 3 unchanged over a pair of frozen teachers, and only which models fill the two roles differs. In the **sequential** case, per-task LoRA experts are distilled sequentially, each into the student produced by the previous round, so skills are added one at a time and every round anchors to the model that already carries the earlier ones. In the **joint, multi-task teacher** case, a single LoRA expert is trained on both skills and distilled in one pass, at the cost of a less sharp, mode-averaged teacher. In the **joint, distilled teachers** case, each skill is first distilled into its own student, and those two students then fill the teacher roles in a second on-distribution pass, the one carrying the first skill as v_B and the one carrying the second as v_E. The student again starts from the base policy.

4 Experiments

We evaluate `KEEP` along the two axes it balances, novel skill acquisition and generalist retention. We adapt $\pi_{0.5}$ (Black et al., 2025a), a flow-matching VLA post-trained on DROID, to three real-world skills on which it largely fails: pouring, stacking, and breaker box flipping. After describing the setup, we report real-world results for single- and two-skill adaptation (Section 4.1), then ablate every component of `KEEP` in REALM (Sedlacek et al., 2026), a real-to-sim aligned simulator (Section 4.2). To test whether the results depend on our platform, we also repeat the single-task study on the LIBERO benchmark (Liu et al., 2023a), with a different environment and base policy, where `KEEP` again matches the base policy on generalist tasks, while exceeding the LoRA expert and every RETAIN configuration on average over the novel tasks (Appendix A.2).

Datasets and metrics. We use DROID (Khazatsky et al., 2024), a large in-the-wild manipulation dataset collected on a Franka-based platform, on which $\pi_{0.5}$ is post-trained. It is therefore both the source of the generalist's prior skills and, through a uniform random subset, the replay data in our training mixture. To study skill acquisition, we record demonstrations on the same platform for three tasks that are rare in DROID and on which the pre-trained policy largely fails: bottle pouring, plate stacking, and breaker box flipping (≈50,000 frames per task after idle-chunk filtering, see Appendix A.3 for more details). We report the success rate (SR) on two axes: task SR on the novel skill and generalist SR on a suite representative of the base policy's original capabilities (Appendix A.4). A rollout succeeds if the full manipulation sequence is completed within 30 seconds.

Real-world and simulated evaluation. Main comparisons are run on the real robot, where each policy is evaluated with 50 rollouts on its novel task and 100 on the generalist suite, on a fixed set of scene configurations shared by all methods, with success judged by a human operator. Ablations require far more rollouts than are feasible physically, so we run them in REALM (Sedlacek et al., 2026), a real-to-sim aligned simulator of the DROID platform whose success rates track ours in the real world ($r = 0.77$–0.99 across tasks and the generalist suite, see Appendix A.5 for details). Each simulated policy is evaluated with 450 novel-task rollouts for pouring, 675 for stacking, and 1,000 on the REALM generalist benchmark, sweeping scene configurations and perturbation axes as detailed in Appendix A.5. Flipping, whose spring-loaded breaker dynamics are difficult to simulate, is evaluated in the real world only.

Baselines. We compare against the two standard remedies for forgetting, replay and weight-space merging. The single-task LoRA expert (Hu et al., 2022), which also serves as the `KEEP` teacher, is the acquisition reference and the forgetting worst case. Full fine-tuning (FFT) adapts all weights on the novel task, while co-training (Co-Train) does the same on the novel task and the DROID replay mixture that `KEEP` uses. Both are merged back into the base policy by weight interpolation following RETAIN (Yadav et al., 2026), the strongest retention mechanism we found: in simulation, alternative merging rules and classic regularization baselines all lie inside the front that RETAIN alone defines (Appendix A.6). We run each under our training recipe (R1) and under the recipe of the RETAIN paper (R2), so that the baselines are evaluated both on equal footing with `KEEP` and on the settings reported in the RETAIN paper. All methods train on the same pre-mixed data shards in the same deterministic order, which rules out data ordering as a source of the differences we report. On flipping, we re-tune the peak learning rate of KEEP and the R2 baselines by one rule in simulation. Full training recipes are shown in Appendix A.7.

4.1 Real-world results

We report every policy as a point in the plane spanned by novel-task SR and generalist SR, so a method is judged by the trade-off it strikes rather than by either axis alone. We first ask whether one novel skill can be added without cost, then whether one policy can carry two.

Single task. Figure 2 shows one panel per novel task and their average. The base policy solves almost none of the novel tasks (2–16% task SR) at 61% generalist SR. The LoRA experts solve them (62–86% task SR), but forget, and their generalist SR drops to 7–33%. RETAIN merging brings the generalist SR back up to 27–51%, but the merged policies lose most of the skill on pouring and flipping, where none exceeds 48% task SR. Stacking is the exception since every method learns it (78–90%). `KEEP` does not face this trade-off. On all three tasks, it matches or exceeds the expert (72%, 94%, and 78% task SR) and keeps 58–60% generalist SR, within three points of the base policy. In the Pareto view, this places `KEEP` outside the region reachable by any mixture of the baselines on every task. Averaged over the three tasks, `KEEP` reaches 81% task SR at 59% generalist SR, while the best merging baseline reaches 57% at 46%. Across the robot, simulation, and LIBERO, `KEEP` exceeds its own teacher on average over the novel tasks, consistent with the expert's velocity being a lower-variance target than raw demonstrations (`gt/off`, Figure 4).

Two tasks. Figure 3 tests whether one policy can carry two novel skills. We use pouring and stacking, which share a training recipe (Appendix A.7), so that schedule and data split are the only variables that change. We use Co-Train + RETAIN as the baseline, since FFT cannot replay the first task. Each method is trained sequentially in both orders and jointly, with the two novel tasks and DROID replay split 40/40/20 or 80/20. We report each task's SR and their mean against the

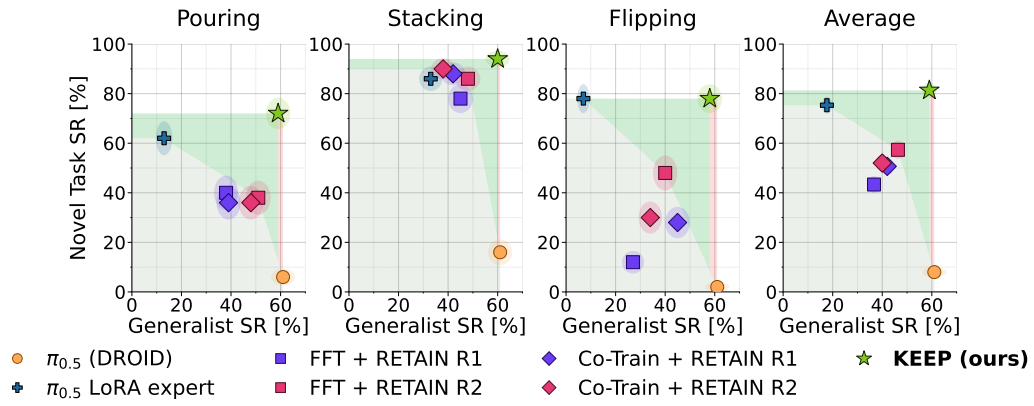

Figure 2: **Real-world single-task results.** Each panel plots novel-task SR against generalist SR on the DROID-inspired generalist suite for one novel task; the rightmost panel averages over the three. Every point is one policy evaluated on the fixed real scene set ($N = 50$ for novel tasks and $N = 100$ for generalist retention), with standard-error ellipses. $\pi_{0.5}$ (DROID) is the base $\pi_{0.5}$ policy post-trained on the DROID dataset, $\pi_{0.5}$ LoRA experts are single-task experts used for distillation by KEEP. FFT + RETAIN and Co-Train + RETAIN fine-tune the full policy on the novel task (alone, or co-trained with DROID replay) and merge the result with the base policy following RETAIN (Yadav et al., 2026). R1 and R2 denote two different training recipes, detailed in Appendix A.7. The shaded area is the set of operating points reachable by any mixture of the baselines. Green marks what KEEP adds beyond it and red marks where a baseline still dominates. KEEP alone matches or exceeds the experts on the novel task while staying close to the base policy's generalist SR.

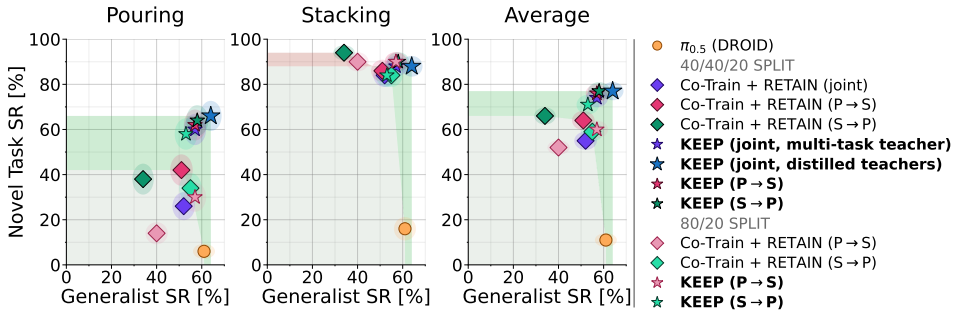

Figure 3: **Real-world multi-task results.** Success rate on each novel task (left, middle) and their mean (right) against success rate on the DROID-inspired generalist suite, for policies trained sequentially (pouring to stacking, denoted by P→S, and stacking to pouring, denoted by S→P) or jointly on both tasks, grouped by the data split between the novel tasks and DROID replay. Co-Train + RETAIN uses the R2 recipe described in Appendix A.7. Joint KEEP uses either a single multi-task LoRA teacher or the two single-task KEEP policies of Figure 2 as teachers. Baselines, evaluation protocol and shading follow Figure 2. KEEP improves on every Co-Train + RETAIN model for a given training schedule at comparable or better generalist success rate, and the joint model with distilled teachers matches the base policy on the generalist suite.

generalist SR. Co-Train + RETAIN varies strongly with these choices (52–66% mean task SR, 34–55% generalist SR), and its best acquisition (66%) comes with its worst retention (34%). KEEP is stable: its four 40/40/20 configurations lie within 74–77% mean task SR and 57–64% generalist SR, and the best of them, the joint model whose teachers are the two single-task KEEP policies of Figure 2, reaches 77% at 64%, with generalist SR indistinguishable from the base policy's. The split matters in one way: with 40/40/20, which replays the earlier task, the order is irrelevant for KEEP; with 80/20, which replays only DROID, the task learned first degrades for both methods (for P→S, pouring falls from 62% to 30% for KEEP and from 42% to 14% for Co-Train + RETAIN), while KEEP's generalist SR holds at 53–57%. Retaining an earlier skill therefore requires replaying it, which 40/40/20 provides at no cost to generalist SR, and we use it as the default.

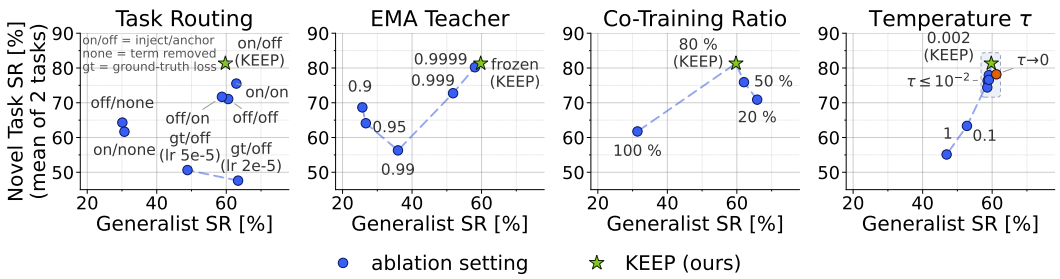

Figure 4: **Design choices, in the retention-acquisition plane.** Mean success rate over pouring and stacking against generalist success rate. Axes are shared and equal-aspect, so one point on x is one point on y and up-and-right is better. The lime star is the `KEEP` setting used everywhere else in the paper, and the dashed line joins each sweep in parameter order. **Task routing** asks where each distillation term is evaluated, on inputs the student itself visits (`on`) or on inputs drawn from the training batch (`off`). The panel is the 2×2 factorial over the two terms at the full recipe, labeled `inject/anchor`, together with the two runs that remove the anchor term altogether (`none`) and train on novel-task data only, and two runs that keep the off-distribution anchor but replace the expert's velocity with the ground-truth flow-matching target on novel-task samples (`gt/off`, a DER-style objective of Buzzega et al. (2020)), at the learning rates of KEEP and of R1. **EMA teacher** sweeps the decay of an EMA-of-the-student teacher with `KEEP`'s frozen base policy as the decay limit. **Co-Training ratio** sweeps the novel/DROID mixture, where retention trades gently against skill from 20% to 80% novel and both axes collapse at 100%, once replay is removed entirely. **Temperature** sweeps τ over five orders of magnitude, where everything at $\tau \leq 10^{-2}$ lands in the boxed region (58.6–59.8% generalist, 74.4–81.3% novel task) with no monotone ordering inside it, so the cluster carries one label instead of five.

4.2 Ablation Studies and Analysis

Figure 4 isolates the design choices behind `KEEP`'s trade-off, one choice per panel. All ablations run in simulation and report the mean over pouring and stacking (N=450 and N=675 novel-task episodes, N=1000 generalist episodes per policy). Simulated and real success rates track each other closely, with $r = 0.93$ on pouring, 0.99 on stacking, and 0.77 on generalist retention (Appendix A.5), so we treat REALM as a proxy for the robot in what follows.

Both loss terms are necessary, and each needs its own routing. Dropping the anchor term outright is not viable (Figure 4, task routing). The student still learns the skill (64.3% against 4.6% for the unmodified base policy), but the generalist collapses from 60.4% to 30.1%, and since those runs train on new-task data only, the drop reflects the missing anchor and the missing replay together. With both terms present, routing the inject term on-distribution moves the novel task from 71.1% to 81.3% at a fixed anchor setting, at no measurable cost to the generalist (-0.7 points). Routing the anchor term on-distribution does the reverse, gaining 3.1 points of generalist but giving up 5.8 points on the novel task. The inject term also cannot be replaced by the demonstrations themselves. Keeping the anchor but fitting the ground-truth flow-matching target instead of the expert (`gt/off`, at the KEEP and R1 learning rates) holds the generalist at 48.9–63.4%, but reaches only 47.6–50.6% on the novel tasks against 81.3% for `KEEP`. The skill is best injected where the student actually is, with the base policy anchored where it was trained.

The teacher must be frozen, not an EMA of the student. Replacing the frozen base policy with an exponential moving average of the student degrades retention monotonically in the decay rate (EMA teacher), from 59.8% at the frozen limit to 58.1% at 0.9999 (within noise), 51.8% at 0.999, and 25.7% at 0.9, and novel-task success does not compensate: no decay setting reaches the frozen teacher's 81.3%. An EMA drifts with the student, and the frozen base policy, the decay $\to 1$ limit of the sweep, dominates every point on it.

Replay is not only about retention. Between 20% and 80% novel data, the mixture trades gently (co-training ratio), giving up 6.0 points of the generalist for 10.4 points of the novel task. Removing replay altogether is different in kind: at 100% novel, the generalist falls to 31.4%, roughly half of the base policy, and the novel task falls with it to 61.7%. Replay therefore not only protects what the

Table 1: **The gate does not need the task label.** The softmax gate, which compares the two teachers' flow-matching errors, against a data oracle given each sample's true source, in simulation. The generalist SR is the mean over the two single-task models ($N = 2000$).

Gate	Pouring SR [%]	Stacking SR [%]	Generalist SR [%]
Task label	68.2 ± 2.2	87.4 ± 1.3	59.1 ± 1.1
Softmax (KEEP)	$\mathbf{71.1 \pm 2.1}$	$\mathbf{91.6 \pm 1.1}$	$\mathbf{59.8 \pm 1.1}$

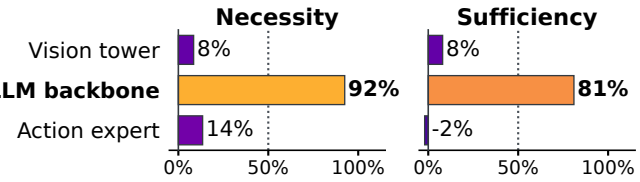

Figure 5: **The novel skill lives in the LLM backbone.** We split KEEP's weight update ΔW by block and graft it onto the base policy in simulation, reporting the fraction of the KEEP gain, $(\mathrm{SR} - \mathrm{SR}_{\mathrm{base}})/(\mathrm{SR}_{\mathrm{KEEP}} - \mathrm{SR}_{\mathrm{base}})$, averaged over pouring and stacking. **Necessity**: skill lost when that block's update is removed. **Sufficiency**: skill kept when only that block's update is added.

policy already knew but also stabilizes acquisition of the new skill. We use 80/20, which matches the base policy on retention (59.8% vs. 60.4%).

The gate needs no task labels. The softmax gate weights the two terms by comparing the teachers' flow-matching errors; it saturates on over 99% of new-task samples and is near zero on most replayed ones, routing roughly a fifth of the replay to the expert where its error is lower. Table 1 shows it loses nothing to a data oracle given each sample's true source: averaged over the two tasks, retention is 59.8% against the oracle's 59.1%, and the novel tasks favor the softmax gate by 3.5 points on average. Retention is also insensitive to the temperature: $\tau \leq 10^{-2}$ lands in the same cluster (temperature panel), and both axes degrade only once τ is large enough to blend the teachers.

The new skill lives in the LLM backbone. Figure 5 grafts individual blocks of the weight delta onto the base policy and reports how much of the novel-task skill returns, averaged over the two tasks. Keeping only the LLM backbone's delta recovers 81% of the skill and dropping it leaves 8%, so that block is both necessary and nearly sufficient on its own. Neither the vision tower nor the action expert is necessary, since dropping either still recovers 86–92%, and neither is sufficient, since keeping only one recovers at most 8%, though the backbone holds most of the parameters.

Limitations. Three limitations persist: (i) KEEP relies on a teacher that already solves the new skill, so each skill requires training a LoRA expert, and a training step costs $2.6\times$ a co-training step (Appendix A.7); (ii) the anchor term needs a slice of the generalist's training data, which is not available for closed models, and Figure 4 shows that both axes collapse without it; and (iii) we study up to two new skills in one policy on $\pi_{0.5}$, the same scope as RETAIN (Yadav et al., 2026), so scaling to longer sequences of skills and to VLAs with other action heads, where the flow-matching error that drives the gate would need to be replaced, remains to be explored.

5 Conclusion

We present KEEP, a task-label-free distillation approach for adding new skills to a pre-trained flow-based VLA without forgetting the skills it already has. Weighting a skill teacher and the frozen base policy per sample by their flow-matching errors, with the skill injected on the student's own distribution and the base policy anchored on the training data, lets a single policy acquire a new skill while retaining its generalist behavior. On a real robot, on three skills on which $\pi_{0.5}$ largely fails, KEEP matches or exceeds single-task experts while staying close to the base policy, and carries two skills in one policy, whereas replay and weight merging trade one objective against the other. The same holds on LIBERO. KEEP is thus a step toward VLAs that keep improving after deployment without losing what they already know.

AI USE STATEMENT

We used Claude and Claude Code (Anthropic) throughout this project. The idea of composing a task expert with a generalist by distilling in the flow-field space is ours. At the start we described it to Claude and asked it to find related work and write up what it found; we then read the papers ourselves. The KEEP objective, the stop-gradient teachers, the competence gate, the on- and off-distribution routing of the two terms, and the co-training mixture were all formulated by us; Claude Code was used to implement them. The geometric analysis in Appendix A.1 was derived and drafted with the help of Claude Code. The first implementation of the objective and the training pipeline was written by Claude Code from our specification. During development we used Claude Code to write out the training configurations we asked for (step counts, batch size, learning rate); we ran the sweeps in REALM ourselves and selected the final recipes (Appendix A.7) ourselves. We did not use AI tools to generate data or to clean or reformat datasets, and the other tasks on the required-disclosure list do not apply to this work.

Beyond that, Claude helped write the plotting scripts and was used actively on the manuscript itself. We wrote each section paragraph by paragraph, sometimes as full text and sometimes as a bullet list of what the paragraph should say, and iterated on it with Claude, which drafted, rewrote, or expanded the text under our guidance and helped reorganize sections and the order in which arguments and results are presented. We accepted, rejected, or rewrote every proposed change.

Everything produced with AI assistance was reviewed by us. We read the code before using it, designed, ran, and scored every real-robot evaluation ourselves, and checked every number, claim, and citation in the paper against the experiments and the sources. We take responsibility for the final content of this work, including anything produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

`KEEP` is implemented on top of the publicly available `openpi` codebase, and every model in the paper starts from a public $\pi_{0.5}$ checkpoint: $\pi_{0.5}$-DROID in Section 4 and the $\pi_{0.5}$ base checkpoint in Appendix A.2. The objective, gate, and routing are fully specified by Equations 3–7, and Appendix A.7 lists the complete recipe behind every reported number, including both RETAIN recipes and the flipping learning-rate rule. All methods read the same frozen, deterministically ordered training streams (Appendix A.3). The real-world protocol, with the fixed scene suites and the full generalist instruction list, is given in Appendix A.4, the simulated protocol in Appendix A.5, and all ablations run in the publicly available REALM simulator. Code, trained checkpoints, and the novel-task demonstration datasets will be released upon acceptance.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-Policy Distillation of Language Models: Learning from Self-Generated Mistakes. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=3zKtaqxLhW`.

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory Aware Synapses: Learning What (not) to Forget. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss (eds.), *Computer Vision – ECCV 2018*, pp. 144–161, Cham, 2018. Springer International Publishing. ISBN 978-3-030-01219-9.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a Vision-Language-Action Model with Open-World Generalization. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 27–30 Sep 2025a. URL `https://proceedings.mlr.press/v305/black25a.html`.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A Vision-Language-Action Flow Model for General Robot Control. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025b. doi: 10.15607/RSS.2025.XXI.010.

Nicholas Matthew Boffi, Michael Samuel Albergo, and Eric Vanden-Eijnden. How to build a consistency model: Learning flow maps via self-distillation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=Di5apl8HSH.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark Experience for General Continual Learning: a Strong, Simple Baseline. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 15920–15930. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/b704ea2c39778f07c617f6b7ce480e9e-Paper.pdf.

Hansheng Chen, Kai Zhang, Hao Tan, Leonidas Guibas, Gordon Wetzstein, and Sai Bi. pi-Flow: Policy-Based Few-Step Generation via Imitation Distillation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=1vAte8dsap.

Zhen Fang, Wenxuan Huang, Yu Zeng, Yiming Zhao, Shuang Chen, Kaituo Feng, Yunlong Lin, Lin Chen, Zehui Chen, Shaosheng Cao, and Feng Zhao. Flow-OPD: On-Policy Distillation for Flow Matching Models, 2026. URL https://arxiv.org/abs/2605.08063.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge Distillation of Large Language Models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=5h0qf7IBZZ.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Jiaheng Hu, Jay Shim, Chen Tang, Yoonchang Sung, Bo Liu, Peter Stone, and Roberto Martín-Martín. Simple Recipe Works: Vision-Language-Action Models are Natural Continual Learners with Reinforcement Learning. *Reinforcement Learning Journal*, 7, 2026. URL https://rlj.cs.umass.edu/2026/papers/Paper84.html.

Zemin Huang, Zhengyang Geng, Weijian Luo, and Guo-jun Qi. Flow Generator Matching, 2024. URL https://arxiv.org/abs/2410.19310.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. RLBench: The Robot Learning Benchmark & Learning Environment. *IEEE Robotics and Automation Letters*, 5(2): 3019–3026, 2020. doi: 10.1109/LRA.2020.2974707.

Tejas Jayashankar, Jongha Jon Ryu, and Gregory W. Wornell. Score-of-Mixture Training: One-Step Generative Model Training Made Simple via Score Estimation of Mixture Distributions. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=zk5k2NQcEA.

Tobias Jülg, Wolfram Burgard, and Florian Walter. Refined Policy Distillation: From VLA Generalists to RL Experts. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 11677–11684, 2025. doi: 10.1109/IROS60139.2025.11246761.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree

Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Youngwoon Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pannag R Sanketi, Archit Sharma, Cody Simpson, Quan Vuong, Homer Rich Walke, Blake Wulfe, Ted Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, Christopher Agia, Rohan Baijal, Mateo Guaman Castro, Daphne Chen, Qiuyu Chen, Trinity Chung, Jaimyn Drake, Ethan Paul Foster, Jensen Gao, David Antonio Herrera, Minho Heo, Kyle Hsu, Jiaheng Hu, Donovon Jackson, Charlotte Le, Yunshuang Li, Roy Lin, Zehan Ma, Abhiram Maddukuri, Suvir Mirchandani, Daniel Morton, Tony Nguyen, Abigail O'Neill, Rosario Scalise, Derick Seale, Victor Son, Stephen Tian, Emi Tran, Andrew E. Wang, Yilin Wu, Annie Xie, Jingyun Yang, Patrick Yin, Yunchu Zhang, Osbert Bastani, Glen Berseth, Jeannette Bohg, Ken Goldberg, Abhinav Gupta, Abhishek Gupta, Dinesh Jayaraman, Joseph J Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, and Chelsea Finn. DROID: A Large-Scale In-The-Wild Robot Manipulation Dataset. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.120.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An Open-Source Vision-Language-Action Model. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

Yoon Kim and Alexander M. Rush. Sequence-Level Knowledge Distillation. In Jian Su, Kevin Duh, and Xavier Carreras (eds.), *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1317–1327, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1139. URL https://aclanthology.org/D16-1139/.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. doi: 10.1073/pnas.1611835114. URL https://www.pnas.org/doi/abs/10.1073/pnas.1611835114.

Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. DistiLLM: Towards Streamlined Distillation for Large Language Models. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=lsHZNNoC7r.

Quanhao Li, Junqiu Yu, Kaixun Jiang, Yujie Wei, Zhen Xing, Pandeng Li, Ruihang Chu, Shiwei Zhang, Yu Liu, and Zuxuan Wu. DiffusionOPD: A Unified Perspective of On-Policy Distillation in Diffusion Models, 2026. URL https://arxiv.org/abs/2605.15055.

Xuhong Li, Yves Grandvalet, and Franck Davoine. Explicit Inductive Bias for Transfer Learning with Convolutional Networks. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2825–2834. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/li18a.html.

Zhizhong Li and Derek Hoiem. Learning without Forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2018. doi: 10.1109/TPAMI.2017.2773081.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow Matching for Generative Modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023a. URL https://openreview.net/forum?id=xzEtNSuDJk.

Huihan Liu, Changyeon Kim, Bo Liu, Minghuan Liu, and Yuke Zhu. Pretrained Vision-Language-Action Models are Surprisingly Resistant to Forgetting in Continual Learning. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=VzdSHEab4G.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=XVjTT1nw5z.

Xingchao Liu, Xiwen Zhang, Jianzhu Ma, Jian Peng, and Qiang Liu. InstaFlow: One Step is Enough for High-Quality Diffusion-Based Text-to-Image Generation. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=1k4yZbbDqX.

Zuxin Liu, Jesse Zhang, Kavosh Asadi, Yao Liu, Ding Zhao, Shoham Sabach, and Rasool Fakoor. TAIL: Task-specific Adapters for Imitation Learning with Large Pretrained Models. In *The Twelfth International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=RRayv1ZPN3.

Wuyang Luan, Junhui Li, Weiguang Zhao, Wenjian Zhang, Tieru Wu, and Rui Ma. SnapFlow: One-Step Action Generation for Flow-Matching VLAs via Progressive Self-Distillation, 2026. URL https://arxiv.org/abs/2604.05656.

Weijian Luo, Tianyang Hu, Shifeng Zhang, Jiacheng Sun, Zhenguo Li, and Zhihua Zhang. Diff-Instruct: A Universal Approach for Transferring Knowledge From Pre-trained Diffusion Models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=MLIs5iRq4w.

Michael McCloskey and Neal J. Cohen. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. *Psychology of Learning and Motivation*, 24:109–165, 1989. ISSN 0079-7421. doi: 10.1016/S0079-7421(08)60536-8. URL https://www.sciencedirect.com/science/article/pii/S0079742108605368.

Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, and Jeannette Bohg. Consistency Policy: Accelerated Visuomotor Policies via Consistency Distillation. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.071.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental Classifier and Representation Learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5533–5542, 2017. doi: 10.1109/CVPR.2017.587.

Anthony Robins. Catastrophic Forgetting, Rehearsal and Pseudorehearsal. *Connection Science*, 7(2):123–146, 1995. doi: 10.1080/09540099550039318. URL https://doi.org/10.1080/09540099550039318.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík (eds.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL https://proceedings.mlr.press/v15/ross11a.html.

Kaushik Roy, Akila Dissanayake, Brendan Tidd, and Peyman Moghadam. M2Distill: Multi-Modal Distillation for Lifelong Imitation Learning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1429–1435, 2025. doi: 10.1109/ICRA55743.2025.11128857.

Andrei A. Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy Distillation, 2016. URL https://arxiv.org/abs/1511.06295.

Ralf Römer, Yi Zhang, Yuming Li, and Angela P. Schoellig. CLARE: Continual Learning for Vision-Language-Action Models via Autonomous Adapter Routing and Expansion. *IEEE Robotics and Automation Letters*, 11(7):8331–8338, 2026. doi: 10.1109/LRA.2026.3693992.

Tim Salimans and Jonathan Ho. Progressive Distillation for Fast Sampling of Diffusion Models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=TIdIXIpzhoI.

Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial Diffusion Distillation. In Aleš Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol (eds.), *Computer Vision – ECCV 2024*, pp. 87–103, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-73016-0.

Martin Sedlacek, Pavlo Yefanov, Georgy Ponimatkin, Jai Bardhan, Simon Pilc, Mederic Fourmy, Evangelos Kazakos, Cees G. M. Snoek, Josef Sivic, and Vladimir Petrik. REALM: A Real-to-Sim Validated Benchmark for Generalization in Robotic Manipulation. *IEEE Robotics and Automation Letters*, 11(7):8315–8322, 2026. doi: 10.1109/LRA.2026.3692088.

Mingchen Song, Xiang Deng, Guoqiang Zhong, Qi Lv, Jia Wan, Yinchuan Li, Jianye Hao, and Weili Guan. Few-Shot Vision-Language Action-Incremental Policy Learning, 2025a. URL https://arxiv.org/abs/2504.15517.

Wenxuan Song, Jiayi Chen, Pengxiang Ding, Yuxin Huang, Han Zhao, Donglin Wang, and Haoang Li. CEED-VLA: Consistency Vision-Language-Action Model with Early-Exit Decoding, 2025b. URL https://arxiv.org/abs/2506.13725.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency Models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 32211–32252. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/song23a.html.

Shangyuan Tong, Nanye Ma, Saining Xie, and Tommi Jaakkola. Flow Map Distillation Without Data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 33973–33984, June 2026.

Weikang Wan, Yifeng Zhu, Rutav Shah, and Yuke Zhu. LOTUS: Continual Imitation Learning for Robot Manipulation Through Unsupervised Skill Discovery. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 537–544, 2024. doi: 10.1109/ICRA57147.2024.10611129.

Kejing Wang, Toan Nguyen, Minh Hoang Nguyen, Simon Khan, and Flora D. Salim. ROAD-VLA: Robust Online Adaptation via Self-Distillation for Vision-Language-Action Models, 2026. URL https://arxiv.org/abs/2606.25800.

Zhendong Wang, Max Li, Ajay Mandlekar, Zhenjia Xu, Jiaojiao Fan, Yashraj Narang, Linxi Fan, Yuke Zhu, Yogesh Balaji, Mingyuan Zhou, Ming-Yu Liu, and Yu Zeng. One-Step Diffusion Policy: Fast Visuomotor Policies via Diffusion Distillation. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=E2VsqgKNlr.

Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. ProlificDreamer: High-Fidelity and Diverse Text-to-3D Generation with Variational Score Distillation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=ppJuFSOAnM.

Yuxuan Wu, Guangming Wang, Zhiheng Yang, Tianchen Deng, Maoqing Yao, Brian Sheil, and Hesheng Wang. Continually Evolving Skill Knowledge in Vision Language Action Model, 2026. URL https://arxiv.org/abs/2511.18085.

Charles Xu, Qiyang Li, Jianlan Luo, and Sergey Levine. RLDG: Robotic Generalist Policy Distillation via Reinforcement Learning. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.028.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. TIES-Merging: Resolving Interference When Merging Models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=xtaX3WyCj1.

Yajat Yadav, Zhiyuan Zhou, Andrew Wagenmaker, Karl Pertsch, and Sergey Levine. Robust Fine-tuning of Vision-Language-Action Robot Policies via Parameter Merging. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=uWJwQ5SZoM.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World Action Models are Zero-shot Policies, 2026a. URL https://arxiv.org/abs/2602.15922.

Wencheng Ye, Tianshi Wang, Lei Zhu, Fengling Li, Guoli Yang, and Hengtao Shen. ActDistill: General Action-Guided Self-Derived Distillation for Efficient Vision-Language-Action Models, 2026b. URL https://arxiv.org/abs/2511.18082.

Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Frédo Durand, and William T. Freeman. Improved Distribution Matching Distillation for Fast Image Synthesis. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a. URL https://openreview.net/forum?id=tQukGCDaNT.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Frédo Durand, William T. Freeman, and Taesung Park. One-step Diffusion with Distribution Matching Distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6613–6623, June 2024b.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language Models are Super Mario: Absorbing Abilities from Homologous Models as a Free Lunch. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=fq0NaiU8Ex.

Qixin Zeng, Shuo Zhang, Hongyin Zhang, Renjie Wang, Han Zhao, Libang Zhao, Runze Li, Donglin Wang, and Chao Huang. CRL-VLA: Continual Vision-Language-Action Learning, 2026. URL https://arxiv.org/abs/2602.03445.

Zexin Zheng, Jia-Feng Cai, Xiao-Ming Wu, Yi-Lin Wei, Yu-Ming Tang, Ancong Wu, and Wei-Shi Zheng. iManip: Skill-Incremental Learning for Robotic Manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 13890–13900, October 2025.

Zhide Zhong, Haodong Yan, Junfeng Li, Junjie He, Tianran Zhang, and Haoang Li. VLA-OPD: Bridging Offline SFT and Online RL for Vision-Language-Action Models via On-Policy Distillation, 2026. URL https://arxiv.org/abs/2603.26666.

Mingyuan Zhou, Huangjie Zheng, Zhendong Wang, Mingzhang Yin, and Hai Huang. Score identity Distillation: Exponentially Fast Distillation of Pretrained Diffusion Models for One-Step Generation. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=QhqQJqe0Wq.

Mingyuan Zhou, Yi Gu, Huangjie Zheng, Liangchen Song, Guande He, Yizhe Zhang, Wenze Hu, and Yinfei Yang. Score Distillation of Flow Matching Models, 2025. URL https://arxiv.org/abs/2509.25127.

Wei Zhou, Xiongwei Zhu, Zelin Xu, Bo Dong, Lixue Gong, Yongyuan Liang, Meng Chu, Leigang Qu, Lingdong Kong, Wei Liu, and Tat-Seng Chua. DanceOPD: On-Policy Generative Field Distillation, 2026. URL https://arxiv.org/abs/2606.27377.

APPENDIX CONTENTS

A Appendix **16**

A.1 Geometric intuition: stitching velocity fields instead of averaging weights 16

A.2 Evaluation on LIBERO 19

A.3 Datasets and metrics 20

A.4 Evaluation Protocols 22

A.5 Simulation Environments 24

A.6 Baseline Selection 26

A.7 Training Recipes 27

A APPENDIX

A.1 GEOMETRIC INTUITION: STITCHING VELOCITY FIELDS INSTEAD OF AVERAGING WEIGHTS

This section gives a geometric reading of the KEEP objective in action space and contrasts it with weight-space merging. The argument has three steps, and Figure 6 summarizes it. First, under the linear interpolant of Equation 1, every velocity that a policy predicts points to an action chunk, so the expert and the generalist can be compared through the motions they predict. Second, the target towards which the student is trained is a point on the straight segment between these two motions, at a fraction set by the gate. Because the temperature is small compared with the typical gap between the two teachers' errors, the gate is nearly binary and this point is in practice one of the two endpoints: the student follows whichever teacher predicts the demonstrated motion better, the expert on essentially every new-task input and the generalist on most replayed ones, which is the situation that Figure 6 depicts. Third, a first-order expansion shows that weight merging places every input at the same fraction along its segment, fixed before any input is seen, which is why it trades acquisition against retention.

A velocity is a pointer to an action chunk. A flow-matching policy does not output an action directly. At a noised action chunk x_t, the noised sample of Equation 1, it outputs a velocity, and the action emerges only after the velocity field has been integrated. Under the linear interpolant of Equation 1, however, every velocity carries an explicit estimate of the clean action chunk,

$$\hat{a} \;=\; x_t - t\, v(x_t, o, t), \tag{8}$$

where $x_t \in \mathbb{R}^{N\times D}$ is the noised action chunk, $t \in (0, 1]$ is the flow time, with $t = 1$ being pure noise ϵ and $t = 0$ a clean action a, o is the observation, $v(x_t, o, t)$ is the velocity that the network predicts at this input, and $\hat{a} \in \mathbb{R}^{N\times D}$ is the resulting estimate of the clean action chunk. Along the linear interpolant, the velocity is the constant direction from the clean action to the noise, so the clean action is found by walking from x_t against the velocity for the remaining time t. Equation 8 is thus a single extrapolation step to $t = 0$. The estimate is exact for the regression target $\epsilon - a$ of Equation 2, since $x_t - t\,(\epsilon - a) = a$. In words, a network that fits the demonstration perfectly points from every noised action straight back to the demonstrated action. The vector $-t\, v$ therefore points from the noised action to the clean action chunk that the network currently predicts, that is, to an N-step sequence of robot commands. When the expert and the generalist disagree at x_t, they point to two different robot motions, $\hat{a}_E = x_t - t\, v_E(x_t, o, t)$ and $\hat{a}_B = x_t - t\, v_B(x_t, o, t)$, and this disagreement can be drawn directly in action space.

KEEP interpolates between the two predictions, separately for every input. For a fixed input, Equation 3 is a weighted least-squares problem with two targets, and its minimizer is their weighted mean,

$$v^\star(x_t, o, t) \;=\; g\, v_E(x_t, o, t) \;+\; (1 - g)\, v_B(x_t, o, t), \tag{9}$$

where $v_E(x_t, o, t)$ and $v_B(x_t, o, t)$ are the velocities of the frozen expert and of the frozen generalist at the input (x_t, o, t), $g \in [0, 1]$ is the competence gate of Equation 5, and $v^\star$ is the velocity towards

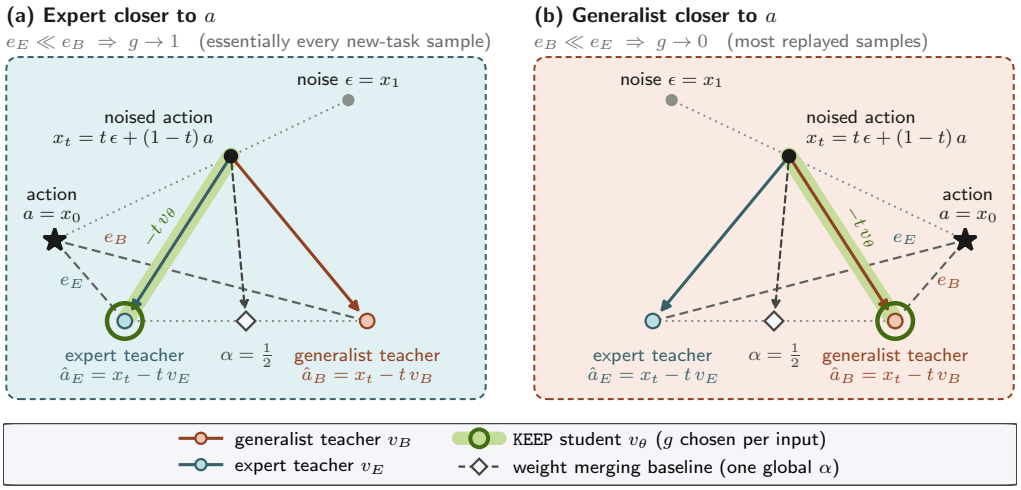

Figure 6: **Geometric intuition.** Both panels show a 2-D slice of the action-chunk space $\mathbb{R}^{N\times D}$. The dotted line through x_t is the interpolation path of Equation 1, $x_t = t\,\epsilon + (1-t)\,a$, from the noise $\epsilon = x_1$ to the demonstrated action chunk $a = x_0$, and the noised action x_t is the point on this path at which both frozen teachers are evaluated. A predicted velocity v implies an estimate of the clean action chunk, $\hat{a} = x_t - t\,v$ (Equation 8), so each arrow points from x_t to the motion that the corresponding model is steering towards: $\hat{a}_E$ for the expert teacher and $\hat{a}_B$ for the generalist teacher. The dashed segments from $\hat{a}_E$ and $\hat{a}_B$ to a are the distances whose comparison sets the gate (Equation 13). The minimizer of Equation 3 places the student's estimate on the dotted chord between $\hat{a}_E$ and $\hat{a}_B$, at the fraction set by g; the chord is thus the set of all targets the student can be given, one point per value of $g \in [0, 1]$. **(a)** When the expert's estimate is the closer one, $g \to 1$ and the student (green band and ring) follows the expert: the skill is acquired. **(b)** When the generalist's estimate is the closer one, $g \to 0$ and the student stays with the generalist: the prior behavior is retained. Case (a) holds on essentially every new-task sample and case (b) on most replayed ones (Section 4.2); on the remaining replayed samples the expert is the closer one and the student follows it there too. The weight merging baseline (gray diamond) acts, to first order, as the same blend but with a single coefficient α fixed before any input is seen. It lands at the same interior point of the chord in both panels, losing part of the skill in (a) and part of the generalist in (b), whereas KEEP chooses the position on the chord for each input, an endpoint wherever the two teachers disagree.

which the student v_θ is trained at this input. Intuitively, the two loss terms act as two springs that pull the student's velocity towards the two teachers with stiffness g and $1-g$, and $v^\star$ is the point at which the two pulls balance. It coincides with the expert for $g = 1$, with the generalist for $g = 0$, and lies halfway between them for $g = 1/2$. In Figure 6, $v^\star$ corresponds to the dotted chord between the two teachers' estimates, drawn in action space by Equation 11 below. The figure shows only the endpoints of this chord, because the gate of KEEP is nearly binary (Equation 13): on every new-task sample and most replayed ones, $v^\star$ coincides with one of the two teachers. The interior of the chord still matters, since it is where weight merging ends up (Equation 15). Equation 9 follows from the identity

$$g\,\|v - v_E\|_2^2 + (1-g)\,\|v - v_B\|_2^2 \;=\; \|v - v^\star\|_2^2 \;+\; g(1-g)\,\|v_E - v_B\|_2^2, \tag{10}$$

where v is any candidate value of the student's velocity at this input, the left-hand side is the KEEP loss of Equation 3 for a single sample, the first term on the right is the squared distance of v to the blend $v^\star$, and the second term measures the disagreement between the two teachers and depends neither on v nor, therefore, on the student's weights θ. Put differently, distilling from two teachers with weights g and $1-g$ is the same as distilling from a single virtual teacher $v^\star$, and this virtual teacher changes from one input to the next. Substituting Equation 9 into Equation 8 expresses the same blend in action space,

$$\hat{a}_\theta \;=\; g\,\hat{a}_E \;+\; (1-g)\,\hat{a}_B, \tag{11}$$

where $\hat{a}_E$ and $\hat{a}_B$ are the action chunks predicted by the expert and by the generalist, $\hat{a}_\theta$ is the action chunk that the student is trained to predict, all three obtained from Equation 8 at the same noised action x_t, and g is the gate as above. Geometrically, the student is trained to point to a location on the

straight segment, or chord, that joins the two teachers' predicted motions, and the gate g determines where on this chord the location lies. [1]

The gate selects the teacher whose prediction matches the demonstration. Equation 8 also gives the flow-matching errors of Equation 4 a meaning in action space,

$$e_E \;=\; \frac{\|\hat{a}_E - a\|_2^2}{t^2}, \qquad e_B \;=\; \frac{\|\hat{a}_B - a\|_2^2}{t^2}, \tag{12}$$

where e_E and e_B are the flow-matching errors of the expert and of the generalist on the sample, $\hat{a}_E$ and $\hat{a}_B$ are their predicted action chunks, a is the demonstrated action chunk, and the factor $1/t^2$ converts a squared distance between action chunks into a squared distance between velocities. Intuitively, the flow-matching error of a teacher measures how far the motion it predicts lies from the motion that was demonstrated. Inserting Equation 12 into the gate of Equation 5 gives

$$g \;=\; \sigma\!\left(\frac{\|\hat{a}_B - a\|_2^2 \;-\; \|\hat{a}_E - a\|_2^2}{\tau\, t^2}\right), \tag{13}$$

where σ is the logistic function, τ is the temperature, t is the flow time, and $\|\hat{a}_B - a\|_2$ and $\|\hat{a}_E - a\|_2$ are the distances of the generalist's and of the expert's predicted action chunks to the demonstrated one. In words, the gate asks a simple question: which teacher predicts an action chunk closer to the demonstrated one? It tends to one when the expert is closer, to zero when the generalist is closer, and a small temperature τ makes the transition between the two sharp.

On a novel-task sample, the expert's prediction lies close to the demonstration, whereas the generalist, which largely fails on the task, predicts a different motion, so $g \to 1$ (Figure 6a). On most replayed samples, the roles are reversed. The expert has forgotten much of the prior skill, the generalist has not, and $g \to 0$ (Figure 6b). At the low temperature we use, the gate is nearly binary, so the student does not average the two teachers but follows one of them on nearly every input. Its velocity field coincides with the expert's where the expert is competent and with the generalist's everywhere else. In this sense, the student's field is *stitched* together from the two teacher fields along the boundary of the expert's competence.

That the gate selects rather than averages is essential. A point in the interior of the chord of Equation 11 is the mean of two different action chunks, which need not be a meaningful motion at all. This is the mode-averaging failure familiar from behavior cloning. Consistent with this view, both novel-task and generalist success degrade once the gate is softened ($\tau > 10^{-2}$ in Figure 4).

Weight merging makes a single choice for all inputs. Weight-space merging, as in RETAIN (Yadav et al., 2026), interpolates parameters rather than velocities,

$$\theta_{\text{merge}} \;=\; (1-\alpha)\,\theta_B \;+\; \alpha\,\theta_{\text{FT}}, \tag{14}$$

where θ_B are the weights of the generalist, θ_{FT} are the weights obtained by fine-tuning the generalist on the new task, θ_{merge} are the weights of the merged policy, and $\alpha \in [0,1]$ is the merging coefficient, which is chosen once for the whole model, before any input is seen. Merging thus slides the weights along the straight line from the generalist ($\alpha = 0$) to the fine-tuned model ($\alpha = 1$). A first-order expansion around θ_B shows what this does to the velocity field,

$$v_{\theta_{\text{merge}}}(x_t, o, t) \;\approx\; (1-\alpha)\, v_B(x_t, o, t) \;+\; \alpha\, v_{\text{FT}}(x_t, o, t), \tag{15}$$

where $v_{\theta_{\text{merge}}}$, v_B and v_{FT} are the velocity fields of the merged, the generalist and the fine-tuned policies, and the approximation holds to first order in the weight change $\Delta\theta = \theta_{\text{FT}} - \theta_B$. Writing

[1] Equation 3 evaluates the two terms at different inputs: with the routing used by KEEP, the inject term is evaluated at x_t^{on} (Equation 7) and the anchor term at x_t, whereas the gate of each sample is always computed at its x_t, since it requires the target u, which the student does not observe. At the population level, the minimizer at any input therefore remains a convex combination of the two teacher velocities, with g replaced by an effective gate

$$\bar{g}(x_t, o, t) \;=\; \frac{\sum_{i\in\mathcal{I}} g_i}{\sum_{i\in\mathcal{I}} g_i + \sum_{i\in\mathcal{A}} (1-g_i)} \;\in\; [0,1],$$

where $\mathcal{I}$ and $\mathcal{A}$ are the sets of training samples whose inject and anchor terms, respectively, are evaluated at this input, and g_i is the gate of sample i. Since the gate is nearly binary, $\bar{g} \approx 1$ on the student's own rollouts of new-task samples and $\bar{g} \approx 0$ on most replayed inputs, which is the situation Figure 6 depicts.

$J_B(x_t, o, t)$ for the Jacobian of the generalist's velocity with respect to its weights, and expanding both policies around θ_B, where they differ from it by $\alpha\,\Delta\theta$ and $\Delta\theta$, gives $v_{\theta_{\text{merge}}} \approx v_B + \alpha\, J_B\,\Delta\theta$ and $v_{\text{FT}} \approx v_B + J_B\,\Delta\theta$, and eliminating $J_B\,\Delta\theta$ between the two gives Equation 15. Moving a fraction α of the way in weight space moves the predicted velocity roughly a fraction α of the way between the two policies, and it does so by the same fraction at every input. To first order, merging therefore performs the same interpolation as Equation 9, but with one coefficient shared by all inputs and with higher-order terms that are left uncontrolled. Here v_{FT} is the fine-tuned model from which merging starts, not the LoRA expert v_E of `KEEP`; Figure 6 identifies the two so that both blends can be drawn on the same chord. In Figure 6, the merged policy sits at the same interior point of the chord in both panels. Changing α moves all inputs along their chords together, so every gain on the novel task is paid for on the prior skills, which is the trade-off exhibited by the merged baselines in Figure 2. The temperature sweep connects the two regimes: as $\tau \to \infty$, the gate of Equation 13 tends to $g = 1/2$ on every input, which is exactly such an input-independent blend. `KEEP` removes the coupling between inputs by choosing the position on the chord for each input separately.

What remains at test time. The gate requires the demonstrated action and therefore exists only during training. At test time, the student is a single network with the inference cost of the base policy, and the decision of which teacher to follow has been distilled into its weights. Note also that the anchor term constrains the function computed by the student on the replayed data, not its parameters, in the spirit of learning without forgetting (Li & Hoiem, 2018). The shared backbone remains free to change wherever this does not alter the generalist's behavior, which is consistent with Figure 5, where the new skill is written mostly into the LLM backbone.

A.2 Evaluation on LIBERO

To test whether the results of Section 4.1 depend on the DROID platform, we repeat the single-task study on the LIBERO (Liu et al., 2023a) simulation benchmark. We use the LIBERO evaluation code released with RETAIN (Yadav et al., 2026), including its scene splits.

Setup. RETAIN reports its LIBERO results on π_0 (Black et al., 2025b). To stay consistent with the rest of the paper, we start from the public $\pi_{0.5}$ base checkpoint and pre-train it on the RETAIN pre-training mixture of LIBERO-90, LIBERO-Goal, LIBERO-Spatial, and LIBERO-Object. The resulting policy plays the role that $\pi_{0.5}$-DROID plays on the real robot. It is the base policy we aim to preserve, and every method in this section starts from it. As novel skills, we take three long-horizon tasks from LIBERO-10 that the base policy cannot solve and that were used in RETAIN. In **Basket**, the robot must put both the alphabet soup and the cream cheese box in the basket. In **Mugs**, it must place two mugs on two different plates. In **Stove**, it must turn on the stove and then put the moka pot on it. Each task is trained and evaluated with a single fixed LIBERO instruction.

Evaluation. Each novel task is evaluated on the scene splits defined by RETAIN. We hold out `OOD_HARD-set-0` (N=50 episodes) as a validation set, used only for model selection. Novel-task SR pools the remaining three OOD splits, `OOD-easy`, `OOD-medium`, and `OOD_HARD-set-1`, into N=150 episodes per task. Following RETAIN, we additionally report each split separately, including the in-distribution split (ID, N=20 per task). Generalist SR is measured on the `GENERALIST` split, which covers 20 tasks spread over four pre-training suites (five each from LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-90) with 10 episodes each (N=200), using the LIBERO evaluation protocol.

Methods. All methods start from the $\pi_{0.5}$ LIBERO base policy that we pre-train, and their recipes are listed in Appendix A.7. `KEEP` follows its DROID recipe with a batch size of 128 and replays the LIBERO pre-training mixture in a 50/50 split with the novel task, following RETAIN. The LoRA expert, which is also the `KEEP` teacher, is trained with a batch size of 128 for 250 steps. For RETAIN, we train FFT and Co-Train under two recipes: ours and the one from the RETAIN paper, and merge each into the base policy with weight α on the fine-tuned model. The paper recipe uses the merge weights from the RETAIN paper, and ours selects $\alpha \in \{0.7, 0.8, 0.9\}$ per task on validation. `KEEP` is run in a single configuration without per-task tuning.

Baseline selection. With our recipe, we select the merge weight with the highest validation SR for each method and task and break ties by generalist SR (Figure 7). Since the tie-breaker uses

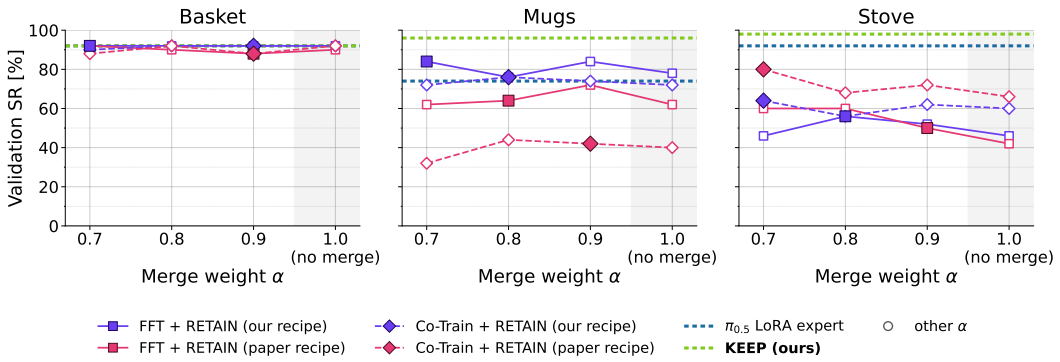

Figure 7: **RETAIN tuning on LIBERO.** Validation SR (`OOD_HARD-set-0`, N=50) against the RETAIN merge weight α for each task, method, and recipe. Filled markers are the merge weights used, selected on validation for our recipe and taken from the RETAIN paper for the paper recipe. The shaded column at $\alpha = 1$ is the unmerged fine-tuned model, shown for reference and never used. Dotted lines show the LoRA expert and `KEEP` on the same split.

the reported generalist split, this selection, if anything, favors the baselines. Basket is saturated on validation, where every model reaches 88–92%, so the tie-breaker decides. On Mugs and Stove, `KEEP` reaches 96% and 98% validation SR, compared to at most 84% and 80% for any RETAIN model.

Results. Figure 8 (top) shows the evaluation results. The base policy retains 92.0% generalist SR but solves almost none of the novel tasks (1.6% on average). The LoRA experts learn them (78.4% on average) at the cost of 9.7 points of generalist SR. `KEEP` reaches 88.0%, 78.0%, and 88.0% novel-task SR on the three tasks and 84.7% on average, the highest of all methods on Mugs, Stove, and on average, while keeping 91.7% generalist SR, within 0.3 points of the base policy. The strongest baseline, Co-Train + RETAIN with our recipe, reaches 76.0% novel-task SR at 93.3% generalist SR, and the other three RETAIN models stay between 66.9% and 70.4%. `KEEP` thus improves on the LoRA expert and on every RETAIN model on the novel tasks on average, while its generalist SR stays close to that of the base policy and of both Co-Train + RETAIN models, and above both FFT + RETAIN models. It exceeds its own teacher on the novel tasks by 6.3 points while retaining 9.4 more points of generalist SR. Figure 8 (bottom) breaks novel-task SR down by split. Every fine-tuned model reaches 93–100% on ID, where KEEP solves every episode, and the splits separate the methods only as they get harder: KEEP has the highest average SR on every split, and its margin over the next-best model grows from 0.6 points on `OOD-easy` to 5.3 on `OOD-medium` and 10.6 on `OOD_HARD-set-1`.

A.3 Datasets and metrics

This section provides the data behind Section 4.1: the novel-task demonstrations, the DROID replay subset, how both are transformed into the fixed training streams that all methods share, and the metrics.

Novel-task demonstrations. We record teleoperated demonstrations on the same Franka-based platform as DROID, with object poses, distractors (and their placement), and camera viewpoints randomized during collection. Recordings are chunked, and idle chunks, in which the robot does not move, are removed before training. Table 2 lists the filtered data. The three tasks differ almost fivefold in episode count, yet each contributes close to 50,000 valid frames, or 55 minutes of motion at the 15 Hz control rate, so no task is over-represented in training by data volume alone.

DROID replay buffer. The replay data are a uniform random subset of DROID (Khazatsky et al., 2024), about 5 % of the dataset, to which we apply the same idle-chunk filter as to the novel-task demonstrations before mixing.

Training streams. Every dataset, each novel task, and the DROID replay subset is materialized once into a frozen stream of $5{,}000 \times 256 = 1{,}280{,}000$ samples in a fixed random order. A training

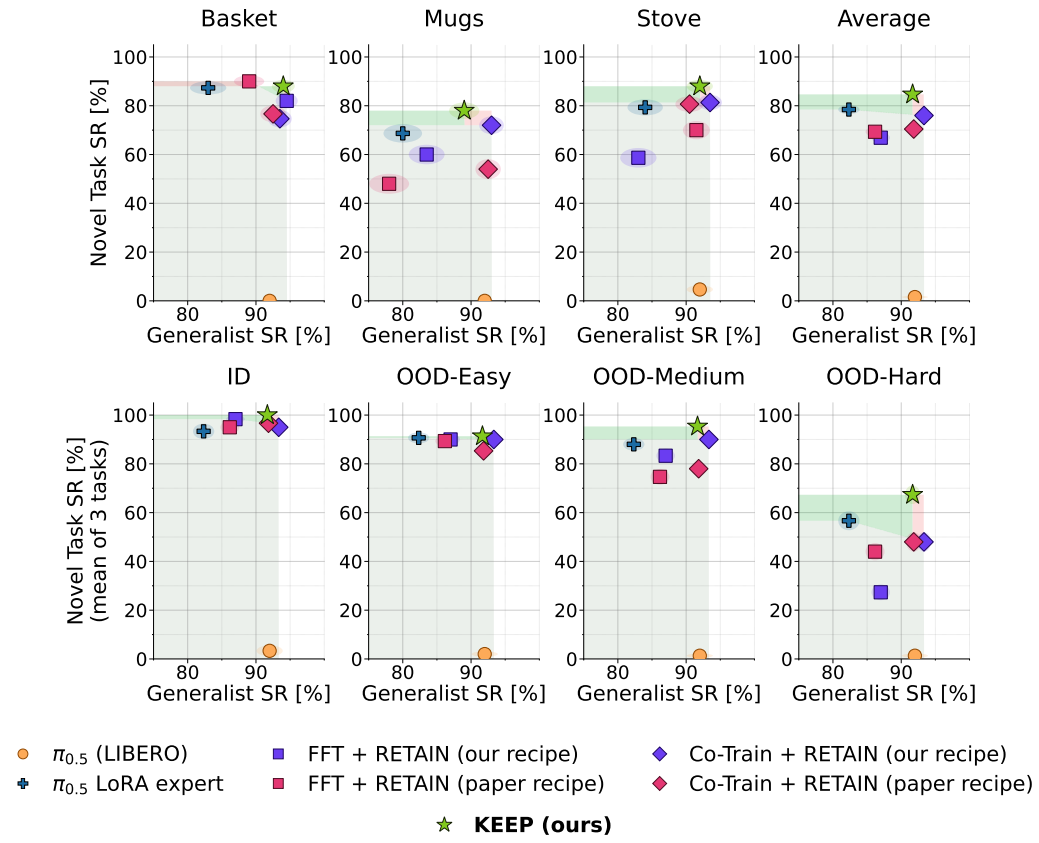

Figure 8: **LIBERO results. Top:** novel-task SR per task on the pooled OOD test splits (`OOD-easy`, `OOD-medium`, and `OOD_HARD-set-1`, N=150 per task) against generalist SR on the `GENERALIST` split (N=200), with the rightmost panel averaging the three tasks. **Bottom:** novel-task SR per split, averaged over the three tasks, including the in-distribution split (ID, N=20 per task), reported separately as in RETAIN; OOD-Hard denotes `OOD_HARD-set-1`. Generalist SR does not depend on the split, so each model keeps the same horizontal position across the bottom row. RETAIN is shown for FFT and Co-Train under both recipes, ours and the one of the RETAIN paper, each at the merge weights of Figure 7. Shading and error ellipses follow Figure 2.

Table 2: **Novel-task demonstrations after idle-chunk filtering.** Time is valid frames at 15 Hz, and the last column is the average length of the valid episode.

Task	Episodes	Valid frames	Time [min]	Per episode [s]
Pouring	150	49,560	55.1	22.0
Stacking	350	50,002	55.6	9.5
Flipping	720	51,737	57.5	4.8
Total	1,220	151,299	168.1	

mixture is then assembled from these streams by drawing each batch's samples from the constituent streams in the mixture ratio, so an 80/20 run reads the same novel-task and DROID samples in the same order regardless of the method trained on it. The stream length is a training budget, not a dataset size: it covers about 5 % of the filtered DROID frames once and a novel task roughly 26 times. Runs with fewer steps or a smaller batch size read the mixture from the start in the same order and stop when their budget is spent, so two runs see identical samples up to the point where the shorter one ends.

Metrics. We report the success rate (SR), the fraction of rollouts that complete the full manipulation sequence within 30 seconds, with the standard error as the uncertainty everywhere. Task SR is

measured on the novel skill and generalist SR on the suite of Appendix A.4. Averages over tasks are plain means of the per-task success rates.

A.4 Evaluation Protocols

Real-world scenes. All real-world evaluations use fixed scene suites created once and replayed identically for every method, so that two policies are always compared on the same scenes. Each scene fixes the objects, their positions, and the distractors. Before each rollout, the operator arranges the scene against a reference frame captured the first time that scene was evaluated, so every method sees the same physical layout. The external camera is mounted at one of two predefined viewpoints, fixed across the whole evaluation, and each scene is assigned one of them. A rollout succeeds if the full manipulation sequence is completed within 30 seconds, as judged by the operator.

Generalist suite. The suite covers 50 DROID-style instructions, listed in Table 3, each evaluated twice in a different layout, for $N{=}100$ rollouts per policy. Instructions span pick-and-place, wiping, pushing, tool use, opening, and stacking over 72 household and workshop objects. The two layouts of an instruction differ in object poses, the number of distractors present (two to five) and their placement, and the camera viewpoint, so no scene is repeated. Figure 9 shows example rollouts of `KEEP` on ten instructions from the suite.

Table 3: **Generalist suite.** The 50 instructions, each evaluated twice with a different object layout and distractors, for $50 \times 2 = 100$ rollouts per policy.

lay the tape on the notebook	put the marker in the holder
move the electric switch to the left	put the mustard bottle on the right side of the table
move the marker from the holder to the notebook	put the pliers in the holder
move the notebook to the right side of the table	put the scissors in the holder
move the sponge from the mixing bowl to the tray	put the screwdriver in the tin can
move the tennis ball from the box to the tray	put the spam can and the red can in the tray
move the tomato from the plate to the bowl	put the spatula in the pan
open the book	put the sponge in the mixing bowl
pick up the ethernet switch	put the spoon in the tray
pick up the pringles container from the tray	put the stapler on the notebook
pick up the red mug	put the tape measure in the tray
pick up the small plastic bottle	put the tennis ball in the pan
push the cheezit box to the right	put the tin can in the box
push the cube	put the tin can next to the cheezit box
put all the objects in the box	put the tomato on the plate
put the baseball into the bowl	rotate marker on the notebook
put the can on top of the boxes	separate the duct tapes
put the cube on the book	squeeze the cleanser on the plate
put the cubes into the bowl	stack the cubes
put the fork in the mug	throw the softball in the box
put the glass in the bowl	use the hammer
put the glass on the left side of the table	use the power drill
put the hammer in the tray	use the screwdriver
put the knife on the plate	wipe the table with the kitchen towel
put the ladle in the pot	wipe the table with the sponge

Novel-task suites. Each task has $N{=}50$ scenes, with the same fixed instruction per task in training and evaluation ("pour the bottle into the bowl", "stack the plates", "flip the breaker"). **Pouring** pairs one of five bottles, filled to one third, two thirds, or full, with one of six bowls and containers. **Stacking** pairs two of seven plates, with the target occasionally raised on a box or tray. **Flipping** presents one of 18 breaker-panel configurations over five breaker models, most at three panel positions, with the target being the only breaker in the ON state. Pouring and stacking scenes add one to four distractors drawn from a pool of 18 household and workshop objects. As in the generalist suite, scenes differ in object poses, distractor selection and placement, and camera viewpoint; for flipping, the panel position and the surrounding breakers play these roles. Figures 10–12 show example rollouts from each suite.



Figure 9: **Example generalist evaluations with `KEEP` rollouts.** Ten evaluation scenes, one per row, with time running left to right. Scenes cover different instructions from the generalist suite (Table 3) and differ in the objects, their placement, and the distractors on the table.



Figure 10: **Example pouring evaluations with `KEEP` rollouts.** Three evaluation scenes, one per row, with time running left to right. Scenes differ in the bottle and bowl, their placement, and the distractors on the table.

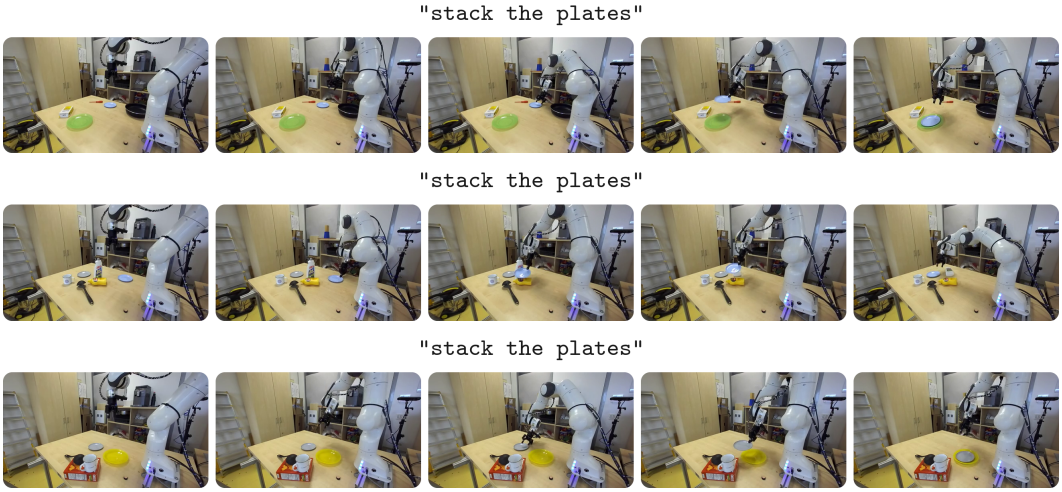

Figure 11: **Example stacking evaluations with `KEEP` rollouts.** Three evaluation scenes, one per row, with time running left to right. Scenes differ in the plate pair, their placement, and the distractors on the table.

A.5 Simulation Environments

All ablations run in REALM (Sedlacek et al., 2026), a real-to-sim aligned simulator of the DROID platform, with simulated counterparts of the real pouring and stacking tasks. Flipping is not simulated, since the spring-loaded breaker dynamics are difficult to reproduce. This section first verifies that simulated success rates predict real ones and then describes the simulated evaluation protocol.

Simulation-to-real alignment. Figure 13 compares simulated and real success rates for every single-task policy of Section 4.1, each evaluated with 50 novel-task and 100 generalist rollouts on the robot, and with 450 (pouring) or 675 (stacking) novel-task and 1,000 generalist rollouts in simulation. On the novel tasks, the two are strongly correlated, with Pearson $r{=}0.93$ on pouring and $r{=}0.99$ on stacking ($N{=}7$ policies each, the base policy and the six trained methods). The generalist comparison pools the policies of all three tasks, since the generalist suite can be simulated for flipping policies even though flipping itself cannot, and counts the task-independent base policy once, giving $N{=}19$ policies. On these suites, which contain different tasks in simulation and in the real

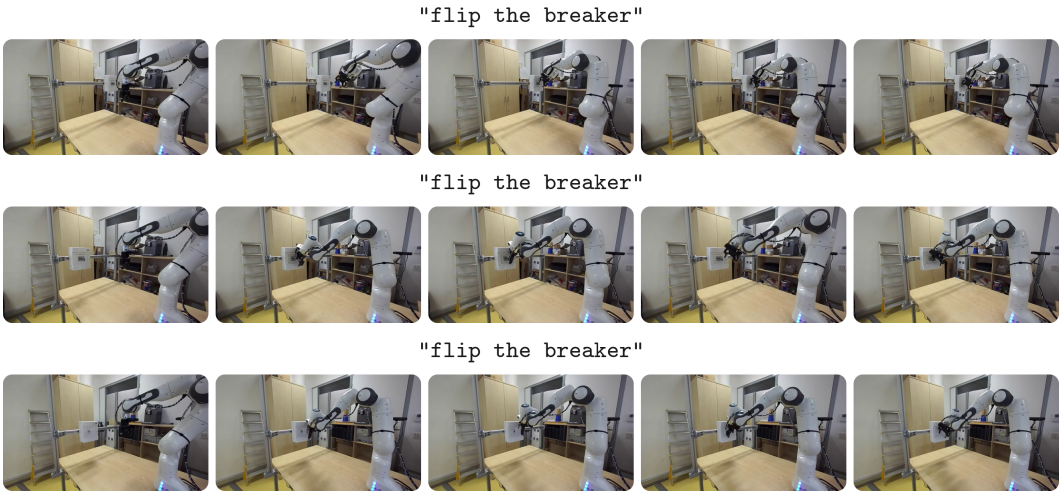

Figure 12: **Example flipping evaluations with KEEP rollouts.** Three evaluation scenes, one per row, with time running left to right. Scenes differ in the panel configuration and its position; the target is the only breaker in the ON state.

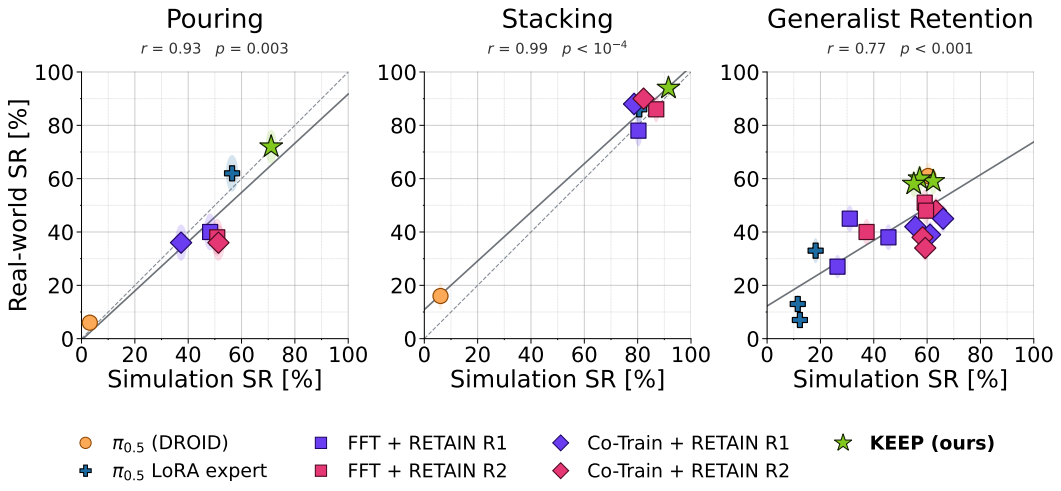

Figure 13: **Simulated against real success rate.** One point per single-task policy of Figure 2. On the novel tasks, simulation recreates the real task, so the dashed diagonal marks perfect transfer. On generalist retention, the suites contain different tasks, so no diagonal is drawn. Every method except the base policy is trained separately for each novel task, so it contributes one generalist point per task. Flipping policies appear only in the generalist panel, since the flipping task itself is not simulated.

world, the correlation is r=0.77. The largest deviations are on the generalist axis of the co-trained baselines, whose simulated retention exceeds the real one by 14 to 25 points on all three tasks, while KEEP transfers within four points on every axis. Because simulation overestimates the co-trained baselines' retention but not KEEP's, the simulated comparisons of Section 4.2 and Appendix A.6 are, if anything, conservative with respect to KEEP's advantage. For the same reason, baseline settings selected in simulation need not be optimal on the robot, which is why we also compare the two RETAIN reciples on the robot (Figure 15).

Simulation protocol. REALM evaluates each policy under perturbations applied independently to a clean scene. **Viewpoint** resamples and jitters the camera pose. **Distractors** places distractors at new positions and swaps each for an object of a different category. **Object pose** resamples the placement of the manipulated objects. **Instruction** replaces the command with one of ten pre-

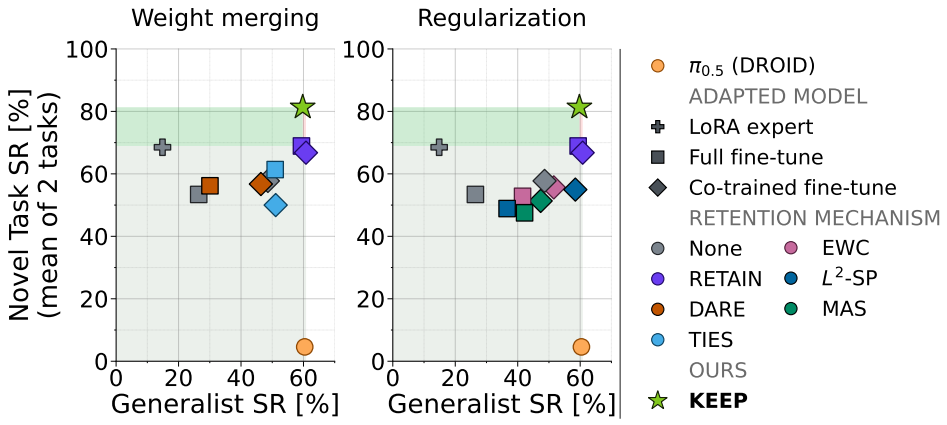

Figure 14: **Baseline selection in simulation.** Mean success rate over pouring and stacking against generalist success rate. **Weight merging** replaces RETAIN's uniform interpolation with DARE and TIES, applied to the same FFT and Co-Train checkpoints. **Regularization** trains FFT and Co-Train with an EWC, L^2-SP or MAS penalty, shown at the best penalty weight of each. In each panel the shaded front is built from every baseline shown, and in both it is defined by the two RETAIN models alone. Shading follows Figure 2.

authored paraphrases. Table 4 lists which conditions each suite uses and the resulting number of rollouts. Success is binary per episode and pooled over all episodes of a suite.

Table 4: **Simulated evaluation suites.** Each policy is evaluated on every scene under every condition, with 25 rollouts per scene and condition on the generalist suite and 75 on the novel tasks.

Suite	Scenes	Conditions	N
Generalist	8 tasks[a]	clean, viewpoint, distractors, object pose, instruction	1,000
Pouring	2 variants[b]	viewpoint, distractors, object pose	450
Stacking	3 variants[b]	viewpoint, distractors, object pose	675

[a]Block into bowl, banana into box, rotate marker, rotate mug, pick spoon, pick bottle, stack cubes, push switch.
[b]Variants differ in the object models: bottle and bowl for pouring, plate pair for stacking.

A.6 Baseline Selection

The real-world comparison uses RETAIN (Yadav et al., 2026) as the only retention baseline, for two reasons. To our knowledge, it is the only retention method that has been demonstrated on the DROID platform with a π_0-family policy (Black et al., 2025b). The continual-learning methods of Section 2 are evaluated on other benchmarks and platforms (e.g., LIBERO, RLBench; James et al., 2020), and the adapter-based ones among them change the architecture or require a task identifier at test time, whereas KEEP targets a single unmodified network with the inference cost of the base policy. Within the family of methods that keep the architecture fixed, this section justifies the choice in simulation, first by comparing RETAIN against alternative merging rules and parameter regularizers, then by showing that RETAIN itself is run at the best setting we found.

RETAIN is the strongest retention baseline we found. Before the real-world comparison, we evaluated the two families of alternatives to RETAIN in simulation, holding the adapted models and the data mixture fixed (Figure 14). **Weight merging** keeps RETAIN's FFT and Co-Train checkpoints and replaces its uniform interpolation with DARE (Yu et al., 2024) or TIES (Yadav et al., 2023), with a drop rate of p=0.9 for DARE and the default density of d=0.2 for TIES. **Regularization** replaces merging altogether and trains FFT and Co-Train with an EWC (Kirkpatrick et al., 2017), L^2-SP (Li et al., 2018) or MAS (Aljundi et al., 2018) penalty toward the base weights. We swept each penalty weight over $\lambda \in \{3{\times}10^{-7}, \ldots, 3{\times}10^{-3}\}$ and show, per method and adapted model, the λ with the highest sum of the generalist and novel-task success: for FFT, $\lambda = 3 \times 10^{-5}$ for all three; for Co-Train, $\lambda = 3 \times 10^{-5}$ for EWC and L^2-SP and 3×10^{-7} for MAS. The latter lies at

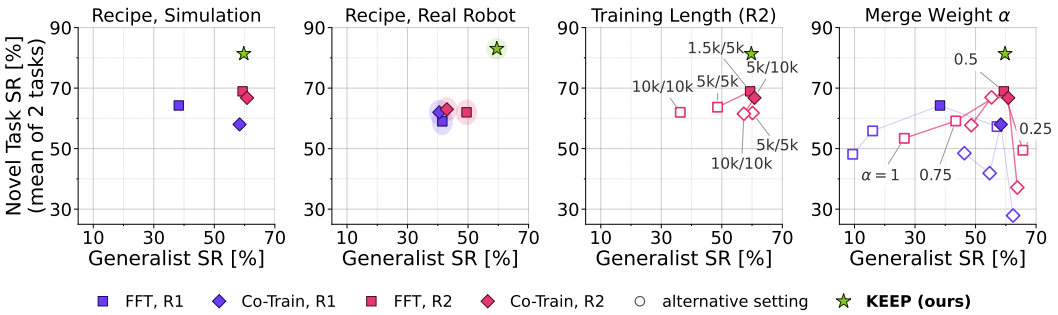

Figure 15: **Tuning RETAIN.** Mean success rate over pouring and stacking against generalist success rate. **Recipe** compares our training recipe (R1) with that of the RETAIN paper (R2), in simulation and on the real robot. **Training length** labels each R2 model by steps trained over schedule length. A line joins the stopping point used in the paper to the end of the same run, and a run on a different schedule is shown as a separate point. **Merge weight** sweeps α in $\theta = (1-\alpha)\,\theta_{\text{base}} + \alpha\,\theta_{\text{ft}}$ from 0.25 to the unmerged model at 1. Filled markers are the settings used in the paper, and hollow markers are the alternatives. The base policy lies below the plotted range (60.4% generalist and 4.6% novel-task SR in simulation).

the lower end of the sweep, but as $\lambda \to 0$ the penalty vanishes and the model approaches the unregularized one, which RETAIN already dominates. Every hyperparameter is selected on REALM, which serves as our validation benchmark for all methods, KEEP included. None of the alternatives improves on RETAIN: every merge and every regularized model is matched or beaten by one of the two RETAIN models on both success rates at once. The same holds for plain co-training without any retention mechanism (**None** in Figure 14), the replay-only setting of Liu et al. (2026). We therefore carry only RETAIN into the real-world comparison.

RETAIN is run at the best setting we found. We tune RETAIN on the same validation benchmark as KEEP and confirm the result on the robot (Figure 15). Every RETAIN model is trained both under our recipe (R1) and under the recipe of the RETAIN paper (R2). R2 matches or improves on R1 in both success rates, in simulation and on the robot, for FFT and Co-Train alike. The gap is largest for FFT, where R2 retains 21 points more of the generalist in simulation and 8 points more on the robot. R2 also stops training early, FFT at 1,500 steps of a 5,000-step schedule and Co-Train at 5,000 steps of a 10,000-step schedule. Continuing either run to the end of its schedule lowers both success rates, and training on a shorter schedule to completion does no better, so early stopping is the right choice. We then sweep the merge weight α from 0.25 to 1, which traces the trade-off each model makes between retention and acquisition. The value of 0.5 used in all DROID experiments gives FFT R2, the strongest baseline, its highest novel-task success of any setting, at near-base retention. Smaller weights raise generalist success up to five points above the base policy but cost 20 to 30 points on the novel task, and no weight brings RETAIN within 12 points of KEEP on the novel task. The flipping learning rate follows the single rule described in Appendix A.7.

A.7 Training Recipes

All methods read the same frozen training streams (Appendix A.3), and Tables 5–9 list the recipe behind every reported number. Settings are shared across pouring and stacking. Flipping uses the same recipes except for the peak learning rate of KEEP and Co-Train R2, selected as described below.

Training cost. All timings are on 8×H200 GPUs at batch size 256 with the same data pipeline, so they isolate the objective; we compare against the R1 baselines, which share this batch size. A plain flow-matching step, as used by FFT and Co-Train, takes 1.66 s. Adding the two frozen teachers, evaluated without gradients at the demonstration-noised input, raises this to 2.39 s (1.44×), and the 10-step Euler rollout of the student for the on-distribution inject term of Equation 3 raises it to 4.38 s (2.64×). The rollout is also where the gain comes from: without it, the novel task falls from 81.3% to 71.1% at the same retention (Figure 4, off/off), so the extra 1.8× buys ten points of skill. KEEP further requires its teacher, and a LoRA expert is not much cheaper per step than full fine-

tuning, since the forward and backward passes still traverse the whole 3.3B-parameter model and the savings are in optimizer state and memory: 1.55 s per step, or 1.9 h for its 4,500 steps. End to end, adding one skill with KEEP takes 5.6 h (≈45 GPU-hours), or 3.9 h with both terms off-distribution, against 2.1 h (≈17 GPU-hours) for a Co-Train R1 run and its free merge. At test time the student is a single network with the inference cost of the base policy.

Flipping learning rate. On the robot, the default recipes undertrain flipping for KEEP and the RETAIN baselines, whereas the LoRA expert learns it at its default recipe. We therefore re-select the peak learning rate of KEEP and of both R2 baselines by the same rule, applied in simulation: the largest value before simulated generalist success degrades. Flipping itself is not simulated, so the selection does not use flipping success, and it is the same for our method and the baselines. For KEEP, this is 7×10^{-5}: simulated generalist success stays between 55 and 59% up to this rate and falls to 49% at 8×10^{-5}. For Co-Train + RETAIN, it is 5×10^{-5} (59%, falling to 45% at 6×10^{-5}). For FFT + RETAIN, retention already degrades above the default rate of 3×10^{-5}, so the default is kept. The recipes for pouring and stacking were not adjusted on the robot. The KEEP design choices follow the simulated ablations of Section 4.2, the RETAIN merge weights and schedules follow the simulated tuning in Appendix A.6, and all were transferred to the robot unchanged. The R1 baselines keep their default learning rate on flipping, as R2 is the stronger RETAIN recipe (Appendix A.6).

Two-task training. Two-task models learn pouring and stacking either sequentially, in both orders, or jointly. In sequential training, the model that has already acquired the first skill becomes the starting point of the second stage and takes the role of the base policy. For KEEP, it initializes the student and serves as the anchor teacher, while the LoRA expert of the second task provides the inject term. Every LoRA expert is fine-tuned from the base policy, so the second-task expert is the one of Figure 2. For Co-Train + RETAIN, it is fine-tuned on the second task and the result is merged back into it. The second stage otherwise uses the single-task recipe of each method and differs only in its data split, which is either 80/20 between the new task and DROID or 40/40/20 between the new task, the first task, and DROID, so that the first task is replayed. Joint models instead train on both tasks at once with the 40/40/20 split, again using the single-task recipe of each method. Joint KEEP replaces the single-task LoRA expert with either a LoRA expert trained on both tasks or the two single-task KEEP policies of Figure 2, distilled into the base policy. The two-task LoRA expert sees twice as much task data as a single-task expert, so we double its training budget to 9,000 steps and train it on an even mix of pouring and stacking (Table 6). Joint Co-Train + RETAIN is merged into the base policy once, after training. All Co-Train + RETAIN two-task models use the R2 recipe.

LIBERO recipes. Tables 10–14 list the LIBERO recipes. The base policy is trained from the $\pi_{0.5}$ base checkpoint on the RETAIN pre-training mixture, and every other model starts from it. KEEP and the LoRA expert keep their DROID recipes up to batch size, step budget, and data split, and replay the pre-training mixture in place of DROID. FFT and Co-Train are trained under our recipe and under that of the RETAIN paper, and for our recipe, the merge weights are selected per task on the validation split (Appendix A.2).

Table 5: **LoRA expert.**

Hyperparameter	Value
Trainable parameters	LoRA
Batch size	256
Learning rate schedule	linear warmup with cosine decay
Peak learning rate	1×10^{-4}
End learning rate	1×10^{-5}
Warmup steps	450
Gradient steps	4,500
Co-training mix	100 % task

Table 6: **LoRA two-task expert.**

Hyperparameter	Value
Trainable parameters	LoRA
Batch size	256
Learning rate schedule	linear warmup with cosine decay
Peak learning rate	1×10^{-4}
End learning rate	1×10^{-5}
Warmup steps	900
Gradient steps	9,000
Co-training mix	50 % pouring, 50 % stacking

Table 7: **KEEP.**

Hyperparameter	Value
Trainable parameters	full model
Teachers	frozen base policy, LoRA expert
Gate	softmax, $\tau = 0.002$
Routing	inject on-distribution, anchor off-distribution
Batch size	256
Learning rate schedule	linear warmup with cosine decay
Peak learning rate	2×10^{-5} (7×10^{-5} on flipping)
End learning rate	2×10^{-6} (7×10^{-6} on flipping)
Warmup steps	300
Gradient steps	3,000
Co-training mix	80 % task, 20 % DROID

Table 8: **FFT + RETAIN.**

Hyperparameter	R1	R2
Trainable parameters	full model	full model
Batch size	256	32
Learning rate schedule	linear warmup with cosine decay	
Peak learning rate	5×10^{-5}	3×10^{-5}
End learning rate	5×10^{-6}	2×10^{-6}
Warmup steps	450	500
Decay steps	4,500	5,000
Gradient steps	4,500	1,500
Co-training mix	100 % task	100 % task
Merge weight α	0.5	0.5

Table 9: **Co-Train + RETAIN.** On flipping, R2 uses a peak learning rate of 5×10^{-5} decaying to 2×10^{-6}.

Hyperparameter	R1	R2
Trainable parameters	full model	full model
Batch size	256	32
Learning rate schedule	linear warmup with cosine decay	
Peak learning rate	5×10^{-5}	3×10^{-5}
End learning rate	5×10^{-6}	2×10^{-6}
Warmup steps	450	1,000
Decay steps	4,500	10,000
Gradient steps	4,500	5,000
Co-training mix	80 % task, 20 % DROID	80 % task, 20 % DROID
Merge weight α	0.5	0.5

Table 10: $\pi_{0.5}$ **LIBERO base.**

Hyperparameter	Value
Trainable parameters	full model
Batch size	128
Learning rate schedule	linear warmup with constant schedule
Peak learning rate	2×10^{-5}
Warmup steps	1,000
Gradient steps	10,000

Table 11: **LIBERO LoRA expert.**

Hyperparameter	Value
Trainable parameters	LoRA
Batch size	128
Learning rate schedule	linear warmup with cosine decay
Peak learning rate	1×10^{-4}
End learning rate	1×10^{-5}
Warmup steps	25
Gradient steps	250
Co-training mix	100 % task

Table 12: **LIBERO `KEEP`.**

Hyperparameter	Value
Trainable parameters	full model
Teachers	frozen base policy, LoRA expert
Gate	softmax, $\tau = 0.002$
Routing	inject on-distribution, anchor off-distribution
Batch size	128
Learning rate schedule	linear warmup with cosine decay
Peak learning rate	2×10^{-5}
End learning rate	2×10^{-6}
Warmup steps	300
Gradient steps	3,000
Co-training mix	50 % task, 50 % pretrain

Table 13: **LIBERO FFT + RETAIN.**

Hyperparameter	Ours	Paper
Trainable parameters	full model	full model
Batch size	64	64
Learning rate schedule	linear warmup with cosine decay	
Peak learning rate	2.5×10^{-5}	2.5×10^{-5}
End learning rate	2.5×10^{-6}	2.5×10^{-6}
Warmup steps	100	1000
Decay steps	1,000	30,000
Gradient steps (mugs)	1,000	1,000
Gradient steps (stove)	1,000	500
Gradient steps (basket)	1,000	500
Co-training mix	100 % task	100 % task
Merge weight α (mugs)	0.7	0.8
Merge weight α (stove)	0.8	0.9
Merge weight α (basket)	0.7	0.9

Table 14: **LIBERO Co-Train + RETAIN.**

Hyperparameter	Ours	Paper
Trainable parameters	full model	full model
Batch size	64	64
Learning rate schedule	linear warmup with cosine decay	
Peak learning rate	2.5×10^{-5}	2.5×10^{-5}
End learning rate	2.5×10^{-6}	2.5×10^{-6}
Warmup steps	100	1,000
Decay steps	1,000	30,000
Gradient steps (mugs)	1,000	1,000
Gradient steps (stove)	1,000	1,000
Gradient steps (basket)	1,000	1,000
Co-training mix	50 % task, 50 % pretrain	50 % task, 50 % pretrain
Merge weight α (mugs)	0.8	0.9
Merge weight α (stove)	0.7	0.7
Merge weight α (basket)	0.9	0.9