OBJECT COUNT GENERALIZATION IS AN INTERFACE PROBLEM: WARPING A FROZEN SKILL INSTEAD OF RETRAINING IT

Paul Mattes[1,2,*] **Ayman Abdeldayem**[1,2] **Denis Shcherba**[1,2]
Wojciech Samek[1,2,3] **Marc Toussaint**[1,2,4]

[1]Fraunhofer Heinrich Hertz Institute, Berlin, Germany [2]Technische Universität Berlin, Germany
[3]BIFOLD – Berlin Institute for the Foundations of Learning and Data, Germany
[4]Robotics Institute Germany [*]Corresponding author, paul.mattes@hhi.fraunhofer.de

ABSTRACT

Reinforcement learning policies for robot manipulation degrade when the number of objects N in the scene grows beyond the training range. This is a covariate shift that also changes the dimension of the state. We propose WAVE (Warping Actions and Views of an Expert), which extends a pretrained single-object skill to many objects without updating its weights. WAVE has three parts. First, a selection policy picks the one object the robot works on next. Second, a warping policy reads a local view of the scene around that object, whose size does not depend on N. It emits a multiplier and an offset for every quantity the skill reads and every action it outputs. Third, the skill acts under this affine warp. It is shown a scene that holds a single object, while the warp steers it around the objects it is not shown and edits the actions that are sent to the robot. Both the selection and warping policies are trained with PPO on the multi-object reward and use the skill as a black box, so WAVE is task-general and skill-agnostic. It requires only that the skill was trained on the same embodiment and that every object has a goal pose. We also bound the number of attempts needed to place all N objects, assuming positive expected net progress per attempt, and measure this progress for each object count. We evaluate WAVE on cube arrangement tasks with a gantry robot and with a Franka Panda arm. Trained with at most nine cubes, it generalizes zero-shot to several times its training count, on the gantry to scenes of up to thirty cubes. Published methods that learn multi-object manipulation show count generalization up to six objects. Fine-tuning the skill, learning from the same local view without it, and an entity transformer, even when distilled from the pretrained skill, all collapse within their training counts under the same training budget. Code, videos and full results are available at https://hhi-robot-learning.github.io/wave-website/.

1 INTRODUCTION

Reinforcement learning (RL) learns a policy that maximizes the expected cumulative reward of an agent interacting with an environment (Sutton & Barto, 2018). Policies trained this way for robot manipulation solve the scene they were trained in and often degrade when that scene changes (Kirk et al., 2023). This work addresses a covariate shift (Shimodaira, 2000), a change of the state distribution under unchanged dynamics. In our case the number of objects in the scene changes, so a policy trained with a few objects is deployed with many. Such a shift also changes the dimension of the state, and appears for example when a robot that has learned to manipulate a few plates is asked to fill a dishwasher. This work asks what can be learned around a frozen pretrained skill to enable zero-shot generalization to object counts unseen during training.

Two families of prior work address this shift. Set and relational architectures let a policy act at any object count (Zaheer et al., 2017; Zambaldi et al., 2019). Robot policies built this way learn the manipulation behavior itself (Li et al., 2020; Haramati et al., 2024; Lanier et al., 2019; Li et al., 2022). That is, a learned policy issues the robot command at every control step, and no primitive,

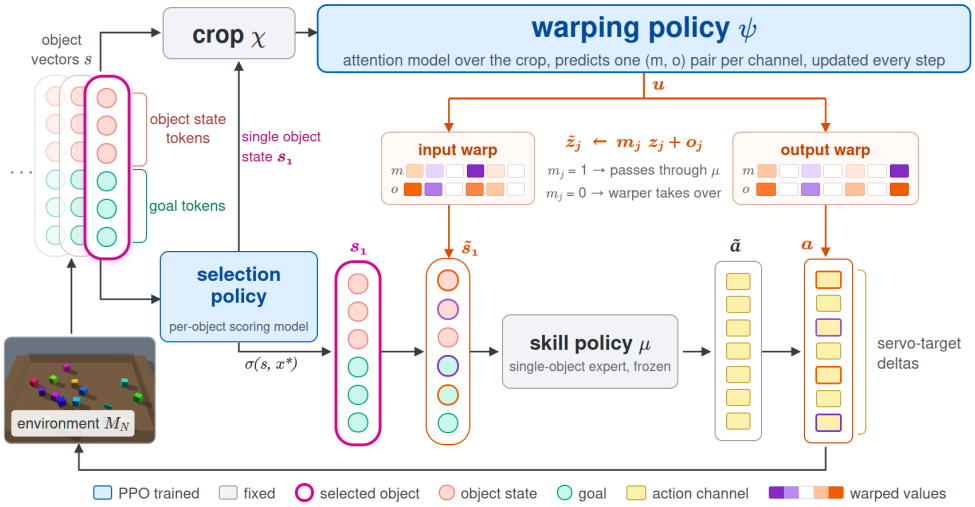


Figure 1: Overview of WAVE. The selection policy picks one object x^* that is not yet placed. The presentation σ shows the frozen skill μ only x^* and its goal. The warping policy ψ reads the crop χ around x^* and edits every input and action channel of μ with a multiplier and an offset. Once x^* is placed, the selection policy picks the next object. Only the two policies are trained.

planner or model supplies the motion. Such policies report whole-task success of at most 0.75 at six objects (Li et al., 2020) and at most 0.36 at eight (Park et al., 2025b; Li et al., 2025) on their respective benchmarks. Further, the theory of size extrapolation shows that networks which pool over every entity can fail to generalize to sets larger than those seen in training (Xu et al., 2021; Veličković et al., 2025; Haramati et al., 2024). Works that keep a trained skill frozen and learn a correction around it (Silver et al., 2018; Johannink et al., 2019; Yuan et al., 2025; Wagenmaker et al., 2025; Hirschowitz & Ramos, 2026; Zhang et al., 2023; Alqithami, 2026; Park & Lee, 2026) absorb dynamics shifts, sensor changes and task mismatch, but none edits both the input and the output interface of a frozen skill to generalize across the object count.

We propose WAVE (Warping Actions and Views of an Expert), which learns a warping policy around a frozen skill. Because the skill can already place one object, training only has to learn the corrections needed for many. Figure 1 gives an overview, and our contributions are as follows.

- We propose WAVE, which composes a frozen single-object skill with a learned warping policy. The policy applies an affine edit to every input and output channel of the skill and is trained with PPO. It uses the skill as a black box, so it applies to any goal-conditioned skill trained on the same robot and to any task that gives every object a goal pose. We show which shifts such a warp can absorb (Section 3.5).
- We train WAVE on six cube arrangement tasks, on a gantry robot and on a Franka Panda arm with a gripper and with a pusher, with at most nine objects and evaluate it zero-shot on up to thirty. On the gantry, this is two to five times the largest count to which published methods that learn the manipulation itself generalize.
- We compare WAVE with ten baselines spanning skill fine-tuning, distillation, subgoal control, scratch-trained entity transformers (Haramati et al., 2024), and a fixed-per-episode input edit (Zhang et al., 2023). They either fail beyond six cubes, even with a non-learned scripted selection of fulfillable goals, or learn their training counts but extrapolate worse. We also ablate the parts of WAVE's architecture.

2 Related Work

A large body of work builds policies that are defined for any object count by making the network permutation invariant over a set of object entities (Zaheer et al., 2017; Zambaldi et al., 2019). Entity-centric RL (ECRL) (Haramati et al., 2024) learns per-object latent particles from pixels and trains

an Entity Interaction Transformer with three objects that generalizes to six objects. EC-Diffuser (Qi et al., 2025) replaces RL by behavior cloning with an entity-centric diffusion Transformer. Li et al. (2020) stack blocks with an attention-based graph network trained by a curriculum that adds blocks, and their reported success declines beyond the counts the curriculum visited. Relational, curriculum and slot-based variants (Mambelli et al., 2022; Lanier et al., 2019; Li et al., 2022; Locatello et al., 2020; Zadaianchuk et al., 2021) and the successors of ECRL (Qi et al., 2025; Haramati et al., 2026) report whole-task success at up to six objects. Li et al. (2020) show up to nine blocks as a figure without values, Haramati et al. (2024) show a twelve-object sort qualitatively, and the stacking variant of that line is evaluated at two to four objects (Haramati et al., 2026). FIOC-WM (Feng et al., 2025) learns an object-centric world model from pixels and generalizes over object attributes and skill compositions; its count test, three to eight objects, covers pushing and toggling but no stacking or other precise placement. Janisch et al. (2020) reach thirty blocks in a symbolic block world without physics simulation. Slot models expose the count through their number of slots (Zimmermann et al., 2023; Krimmel et al., 2024). Theoretical work on size extrapolation shows that networks which pool over every entity can fail on larger sets (Yehudai et al., 2021; Veličković et al., 2025), and that they can generalize when the local structure they see does not change with the count (Yehudai et al., 2021; Ouellette et al., 2023). Large-scale RL learns stacking at a fixed small count (Lampe et al., 2024; Macaluso et al., 2026). On the offline cube benchmarks, which train one policy per count, the best published baseline completes the stacking goal in 0.23 of episodes with two cubes and in none with three or four (Park et al., 2025a). The hierarchical methods evaluated at eight cubes reach at most 0.36 averaged over five goals, from success on the rearrangement goals and almost none on the two stacking goals (Park et al., 2025b; Li et al., 2025).

In task and motion planning, count generalization is easy, because a discrete search over a known symbolic model scales to any number of objects and motion primitives supply the manipulation (Garrett et al., 2020; Chitnis et al., 2022). The same holds when a network selects scripted primitives (Zeng et al., 2020; Lin et al., 2022). Neither applies here, since we have no model and no primitives, so the learned manipulation itself must generalize to more objects. Methods that pass arguments to a frozen executor (Riedmiller et al., 2023) assume that the executor works for every argument.

Language and vision-language models can decompose a task into steps and call a skill for each step (Ahn et al., 2022; Liang et al., 2023). In our setting, such a planner would replace the selection policy, while each step would still be executed by the skill. This execution is the limiting part, since with a scripted order over fulfillable goals the unchanged skill completes no episode with three cubes (Table 1). A language-based planner could therefore be combined with the warp, which we leave for future work.

A single-object skill can also be reused without freezing it. Kickstarting distills a teacher policy into a new student while the student learns the task reward (Schmitt et al., 2018), and jump-start RL rolls a guide policy for the first part of every episode and hands the rest to an exploration policy, moving the switch point earlier as the exploration policy improves (Uchendu et al., 2023). Both leave the multi-object behavior to a network trained on the new task.

Another line of work keeps a trained policy or controller fixed and learns an edit of its interface. Most improve the policy on its own task and distribution, by an action residual (Silver et al., 2018; Johannink et al., 2019; Yuan et al., 2025; Ankile et al., 2025), steered diffusion noise (Wagenmaker et al., 2025), or inserted latents and gated activations (Yu et al., 2026; Miao et al., 2026). Others address a shift, such as changed dynamics (Hirschowitz & Ramos, 2026), heterogeneous or missing sensors (Alqithami, 2026), a task mismatch around a shared backbone (Park & Lee, 2026), a new behavior mode (Das et al., 2026), or out-of-distribution scenes (Liu et al., 2026), but never a change in the number of objects. Closest to our approach, Zhang et al. (2023) learn diagonal projections of the state and the action of a frozen pivoting skill with a policy gradient to transfer it to objects of other shapes and sizes. Their state projection depends on a learned descriptor of the manipulated object and is constant over an episode, while their action projection also depends on the current state. WAVE keeps the frozen skill and the per-channel edits, adds an offset on every channel, and conditions both edits at every step on the multi-object scene observation. Appendix I compares these methods with ours.

3 METHODOLOGY

3.1 Intuition. WAVE takes a skill μ that can move a single object to its goal and uses it, without changing its weights, to arrange many objects. A selection policy picks one object at a time, and μ is shown a scene in which this object is the only one. The other objects are still there, so a warping policy reads the neighborhood of the object and bends the inputs and actions of μ where needed, for example to steer the robot around a stack or to let go of a cube on top of another (Figure 1). Where nothing is in the way, μ acts unchanged.

Warping the skill is necessary because, with many objects, the behavior of μ itself has to change. Choosing the next object or sending μ to intermediate subgoals, as in hierarchical RL, leaves this behavior unchanged. For example, a subgoal can bring a cube above a stack, but it does not change how μ releases the cube there. A policy that adds a correction to the action of μ does change the behavior. However, μ still sees the unedited scene and moves along paths that collide with objects it is not shown, so the correction has to cancel part of μ's action at every step. The warp edits both what μ sees and what it outputs. Editing the input makes μ itself compute a corrected action, and editing the output produces behavior μ never learned, such as the release onto a stack.

WAVE generalizes to higher object counts for two reasons. First, μ can already place a single object, so it is a strong exploration prior that makes multi-object tasks learnable and training only learns the corrections needed with many objects. Second, nothing μ or the warping policy reads grows with the number of objects. Thirty cubes produce the same kind of input as nine, whereas a policy that reads all objects receives a new, out-of-distribution input with every added object.

3.2 Problem and composition. Let $M_1 = (\mathcal{S}_1, \mathcal{A}, P_1, R_1, \mathcal{G})$ be a goal-conditioned manipulation MDP with one robot and one movable object. A state $s_1 \in \mathcal{S}_1$ holds the joint positions and velocities of the robot and the pose and velocity of the object, an action $a \in \mathcal{A}$ is a command to the robot's actuators, $P_1(s_1, a)$ is the transition dynamics given by the physics, a goal $g \in \mathcal{G}$ is a target pose for the object, and $R_1(s_1, a, g)$ is a reward given for the step. Let $\mu : \mathcal{S}_1 \times \mathcal{G} \to \mathcal{A}$ be a deterministic policy trained on M_1. Let $M_N = (\mathcal{S}_N, \mathcal{A}, P_N, R_N, \mathcal{G}^N)$ be the task in which the same robot must bring N objects to the goal poses $(g_1, \ldots, g_N) \in \mathcal{G}^N$. Its state $s \in \mathcal{S}_N$ holds the robot and all N objects, the action space $\mathcal{A}$ is that of M_1, P_N is the physics of the full scene, in which the objects collide with, rest on and occlude each other, and R_N rewards the objects that are at their goals (Section 3.4). As illustrated in Figure 1, we compose μ into a policy π_N for M_N that acts on one selected object x^* at a time and computes

$$\pi_N(s) = W_{\text{out}}\big(\mu(W_{\text{in}}(\sigma(s, x^*); u)); u\big), \qquad u = \psi(\chi(s, x^*)), \tag{1}$$

with s the state of M_N, $\sigma(s, x^*) \in \mathcal{S}_1 \times \mathcal{G}$ the presentation of the selected object, $\chi(s, x^*)$ the crop, a local view of the scene with a fixed width, ψ the learned warping policy, and u the warp parameters it emits. W_{in} and W_{out} are the warping functions that apply u as per-channel affine edits of μ's input and of its action. The presentation $\sigma(s, x^*)$ is the state and goal μ would see in M_1 if x^* were the only object, with nothing about the other objects, expressed relative to the robot so that the presented scene stays inside the distribution μ was trained on. The warping policy runs at the same rate as μ. At every control step, ψ computes a new warp from the current crop before μ acts. The selection policy decides again whenever a cube is placed or a placed cube is knocked off its goal (Section 3.3). The warping policy reads only the crop, which describes every other object in the scene relative to the selected object and its goal. The selection policy scores every object that is not yet at its goal, each together with its surroundings, so a change anywhere in the scene enters its next decision. An attempt is the period in which the selected object stays the same.

WAVE makes three assumptions about μ. μ is a goal conditioned policy, it was trained on the same robot, so that its output is a command for the robot at hand and the output side needs no rewiring, and its input layout $\mathcal{S}_1 \times \mathcal{G}$ is known, so that the presentation can be rendered into it. No weights, gradients, activations or knowledge of what μ does are used.

3.3 Selection policy. The selection policy scores every object that is not at its goal. For each candidate x_i it reads the position of x_i relative to the gripper and to its goal, the robot state, and the neighborhood of x_i. A shared MLP maps these features to two scalars per object, a priority f_i and a feasibility $\hat{p}_i$ (the probability that x_i can be successfully placed). The score is then computed as

$$v_i = \beta \tanh f_i - \alpha \left(\tilde{d}_i + {}^{\ell_i}\!/_{\hat{p}_i} \right) + \theta_i, \qquad x^* = \arg\max_i \, v_i, \tag{2}$$

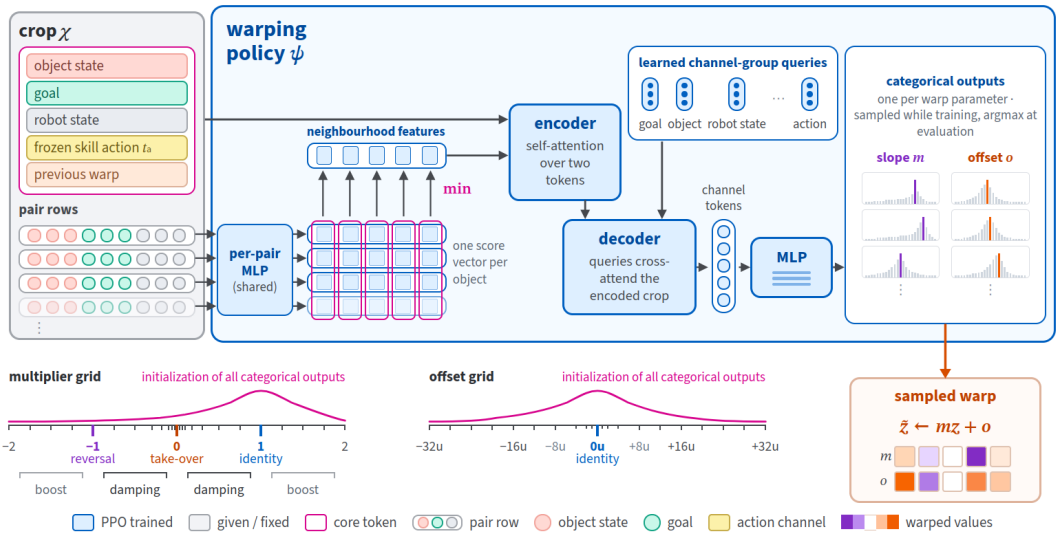


Figure 2: The warping policy ψ and its initialization. The lower axis shows the multiplier grid, dense between take over ($m = 0$) and identity ($m = 1$), with the initial mass on the identity bin.

where $\tilde{d}_i$ is the gripper to object distance, ℓ_i is an estimated length of one attempt at x_i (distance of x_i to its goal plus a small constant), and α a fixed travel price. Since $1/\hat{p}_i$ is the expected number of attempts x_i takes, the bracket is the expected travel that placing x_i costs. The priority head starts at 0 and the feasibility head starts at 1, so the untrained rule is shortest gripper-to-object-to-goal path first, from which the selection policy deviates only where another order yields a higher return. The priority is bounded by the constant β, so the learned priority can reorder candidates of similar expected travel but cannot overturn a travel difference larger than $2\beta/\alpha$. The feasibility head is trained on the realized outcomes of past attempts, so an object that has a low feasibility (e.g., because it is blocked by a neighbor) has a high expected travel and is less likely to be selected, but it is not removed from the pool. Order constraints are learned, not specified. For example, a cube can only be stacked once the cube below its slot is placed. The learned holding bonus θ_i is nonzero only for the current x^*, so that the selection policy keeps it until it is placed.

3.4 The warping policy. The crop $\chi(s, x^*)$ contains the position and goal of x^*, the robot state and the action of μ on the unwarped presentation $\sigma(s, x^*)$. In addition, we compute a set of geometric features for every other object that is not x^*. These features hold its position relative to x^*, to the goal of x^* and to the robot, and whether it is placed. A per-pair network maps each object feature vector to a fixed number of bounded scores, and an elementwise minimum over these scores reduces them to a single neighborhood encoding. We use the minimum because the closest obstacle is the one an attempt is most likely to hit. The minimum is also independent of the count, since a neighbor five centimeters away yields the same score in a scene of three objects and in a scene of thirty. Appendix J compares the minimum with a mean and with a learned attention over the other objects.

A channel is one scalar of μ's interface, a single coordinate of the state it reads or of the action it emits. The channels that carry one physical quantity, the presented goal position or the action for instance, form a channel group. The warping policy emits the warp parameters u of equation 1, one pair (m_j, o_j) per channel j. W_{in} and W_{out} apply that pair to the channel's value z_j as $\tilde{z}_j \leftarrow m_j z_j + o_j$. At $m_j = 1$ and $o_j = 0$ the channel passes through unchanged, at $m_j = 0$ its input is removed and o_j becomes its value, so the warping policy takes over the channel and commands it directly. As shown in Figure 2, ψ encodes the reduced pair scores and the features of x^*, its goal and the robot as the tokens of a small transformer, decodes one query per channel group and emits one categorical distribution per warp parameter over a fixed grid of values. The multiplier grid is dense between take over and identity, the offset grid is a signed ladder in multiples of a step size per robot (1.5 cm on the gantry), and one further bin per channel repeats the value applied at the previous step, so the warping policy can choose to keep its previous edit. Every distribution starts with most of its

mass on the identity value, so training starts from the unwarped skill and departs from it only where that yields a higher task reward.

The warping policy and the selection policy are trained jointly with PPO (Schulman et al., 2017) on the return of M_N. For each of the two policies $\pi_\phi \in \{\psi, \text{selection}\}$, PPO maximizes the clipped objective

$$L(\phi) = \mathbb{E}_t\Big[\min\big(r_t(\phi)\hat{A}_t,\ \text{clip}\big(r_t(\phi), 1{-}\epsilon, 1{+}\epsilon\big)\hat{A}_t\big) + c_H\,\mathcal{H}\big[\pi_\phi(\cdot \mid o_t)\big]\Big], \qquad r_t(\phi) = \frac{\pi_\phi(a_t \mid o_t)}{\pi_{\phi_{\text{old}}}(a_t \mid o_t)}, \tag{3}$$

where o_t is the policy's input, a_t its action, $\hat{A}_t$ the generalized advantage estimate (Schulman et al., 2016) of the return of M_N from a learned value function, ϵ the clip ratio and c_H an entropy coefficient. For ψ, o_t is the crop $\chi(s_t, x^*)$, a_t is the warp u_t and $\pi_\phi(u_t \mid o_t)$ is the product of its per-parameter categorical probabilities. For the selection policy, a_t is the selected object, drawn from a softmax over the scores v_i of equation 2 during training, and the objective counts only the steps at which it decides. The reward R_N (Appendix A) pays a small reward at every control step proportional to the fraction of objects currently placed, a bonus when all of them are, and a one time bonus the first time an object is placed. The reward depends only on which objects are at their goals. It contains no term for a route, a grasp or a release and is the same for every task in Section 4.

3.5 Absorbable shifts. A warp changes the coordinates of μ's input and of its action, which makes it a feedback transformation of the dynamics μ acts on (Brunovský, 1970; Jakubczyk & Respondek, 1980). Let M_1' be the single-object task at deployment, with μ's interface unchanged and only its dynamics P_1' altered. We call the shift from M_1 to M_1' *absorbed* when the warp turns every trajectory of M_1' into one of μ's own trajectories in M_1,

$$W_{\text{in}} P_1'(s_1, W_{\text{out}} a) = P_1(W_{\text{in}} s_1, a). \tag{4}$$

W_{in} is the input warp and W_{out} is the output warp. An attempt under an absorbed shift then runs like an episode of μ alone in M_1, only in warped coordinates. We call this a solo episode. Its success rate is therefore μ's own, and this holds without measuring anything about μ (Appendix H).

Which shifts a warp can absorb depends on what the warp reads. A warp that applies the same edit at every step already absorbs any recalibration of μ's sensors and actuators, since it inverts the recalibration channel by channel. On a robot whose dynamics do not change under such a shift, like the gantry, it also absorbs any translation of the scene parallel to the floor. A warp that varies with the state, as ψ does through the crop, can do more. It can shift the goal that μ pursues at every step, which produces detours, hovering, retreating and holding. Such a retarget changes only the goal, so while it is held the attempt is an ordinary episode of μ toward it. The warp can also take over a channel of μ and command it directly. Take-over (multiplier 0 on an output channel) inherits no guarantee from the frozen skill. Proposition 1 in Appendix H states which shifts such a warp can absorb. The learned warping policy is not limited to these edits and can learn any edit that raises the return, for example to compensate for small differences in the dimensions of the robot. More objects do not change what μ receives. σ shows μ only the selected object, and since σ is expressed relative to the robot, a larger arena only shifts positions. What changes with more objects is the crop, because the scene around x^* becomes denser. The warping policy reads this crop and turns the interactions with the other objects into a change of coordinates for μ. Through these coordinate changes, the warping policy can add behaviors to μ that it never learned, such as curved trajectories to avoid collisions, safe release of a cube on top of another, or nudging a neighbor out of the way. Per-channel affine maps are the minimal edit that changes single channels and contains both identity ($m = 1$, $o = 0$) and full takeover ($m = 0$, o the commanded value); since channels are physical quantities and ψ acts at every step, they suffice in our experiments. A transformation matrix in place of the per-channel edit would extend this to a robot with a different number of joints, or to learned latent states, which we leave to future work. Hyper-GoalNet (Zhou et al., 2025) instead generates the full weights of a policy from demonstrations; WAVE edits only the interface, a far smaller space that it learns from the task reward alone.

A warp cannot absorb shifts that are not observable from the crop, or shifts that it cannot invert. A shift is *unobservable* if two deployments that require different warps produce the same crop, so that a single fixed edit has to serve both. This would occur for example when two cubes of equal size differ in friction, which the crop does not carry. A shift is *not invertible* if canceling it would require

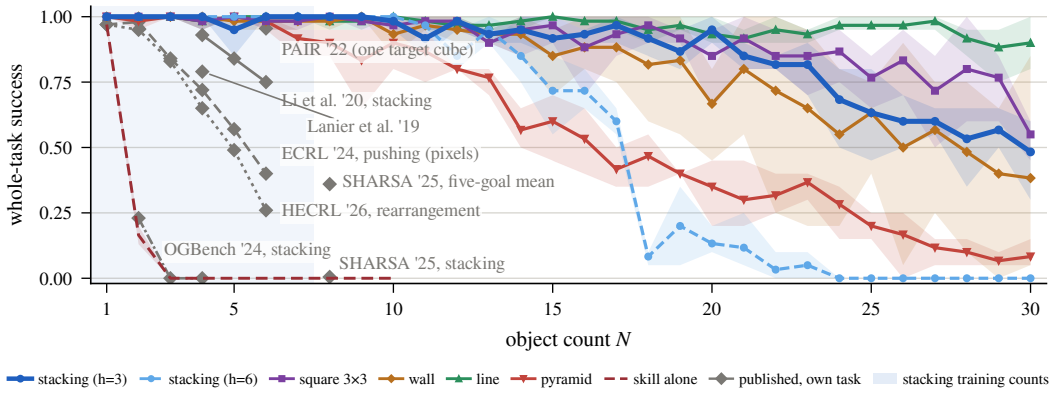


Figure 3: Whole task success against the number of cubes N for WAVE on the six gantry tasks (color), and for published methods on their own tasks (gray, Appendix E). The shaded band is the training range of cube stacking and the dashed red curve the skill alone baseline on cube stacking ($h = 3$). WAVE and skill alone are averaged over seeds with bands for the range. The gray series come from other tasks, simulators and embodiments and are shown for context only.

moving the state in a direction μ's action cannot reach, so that no multiplier or offset reproduces the needed correction. This would occur for example at a kinematic singularity, with the arm stretched straight, where no joint motion moves the gripper further out whatever gain the warp applies.

3.6 A conditional completion bound. Fix the object count N. Suppose that, conditioned on any history of the episode, the next attempt places its object with probability at least p_N and knocks at most λ_N placed objects off their goals in expectation, and never more than $\lambda_{\max}$. We call $\rho_N = p_N - \lambda_N$ the net progress per attempt. If $\rho_N > 0$, then for every factor $c > 1/\rho_N$

$$\Pr[\,\text{task not complete within } cN \text{ attempts}\,] \le \exp\left(-\frac{2(\rho_N c - 1)^2 N}{c(1+\lambda_{\max})^2}\right) \tag{5}$$

(Theorem 1 in Appendix H, from Azuma's inequality), and the expected number of attempts is at most N/ρ_N. The bound holds for a fixed N and only under the stated assumptions, which concern every possible history of an episode, so we do not claim a time budget that is uniform in N. We measure p_N and λ_N per attempt as averages over evaluation episodes (Table 10). On cube stacking, the probability p_N that an attempt places the selected cube stays between 0.75 and 0.84 at all counts, while the number λ_N of placed cubes it knocks off grows from 0.17 at three cubes to 0.64 at thirty, so the net progress ρ_N falls from $+0.64$ to $+0.11$. On the pyramid (policy trained on one to ten cubes) ρ_N falls from $+0.51$ to -0.03 and becomes negative between ten and twenty cubes, where whole-task success drops from 0.93 to 0.19. Since an attempt takes about 2.3 s, the limit of 20 s per cube allows roughly eight attempts per cube, which requires $1/\rho_N \le 8$, a net progress ρ_N of about 0.12 or more.

4 Experiments

We evaluate WAVE on six cube arrangement tasks, stacks of height $h = 3$ and $h = 6$, a line, a square, a pyramid and a wall (Figure 5). A gantry robot with a parallel-jaw gripper runs every task with one to thirty cubes (Section 4.1). A Franka Panda arm runs five of them with a parallel-jaw gripper or a sphere-tip pusher, up to fifteen, twenty or thirty cubes (Section 4.2). Each policy trains on mixed counts of one to N_{train}, with N_{train} between six and nine depending on the task and robot (Appendix A). An episode ends as a success as soon as every cube is within the position tolerance of its goal and on its intended support. This holds in training and in evaluation, so the last cube can still be in the gripper. We call the fraction of successful episodes *whole-task success*. We show the results for the gantry robot on stricter success criteria in J. For every object count evaluation, we use initial layouts which were not part of the training pool. Every evaluation uses a deterministic policy, 20 s per cube and three seeds. The robot, the skill, the tasks, all hyperparameters, and the full result tables are given in Appendices A and B.

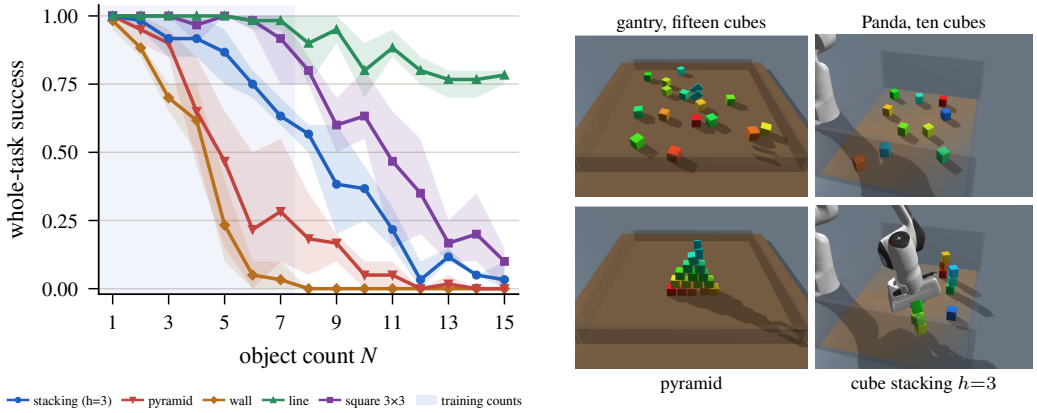


Figure 4: Left, whole-task success against the object count on the Panda arm, the first fifteen counts of every task. Right, a start scene (top) and a completed episode (bottom) on each robot, a fifteen-cube pyramid on the gantry and ten-cube stacking ($h = 3$) on the Panda. All tasks are rendered in Figures 5 and 6.

4.1 Count generalization on the gantry. Trained on one to seven cubes, WAVE (Figure 3) stays at or above 0.92 on cube stacking ($h = 3$) up to twenty cubes and falls to 0.48 at thirty. The line, square, pyramid and wall complete between 0.80 (pyramid) and 0.97 (line and square) at twice their training count and between 0.08 (pyramid) and 0.90 (line) at thirty (Table 2).

Published methods that learn the manipulation itself reach at most 0.75 at six objects (Li et al., 2020) and 0.36 at eight (Park et al., 2025b; Li et al., 2025), while every WAVE task is at or above 0.98 at six. WAVE stays above the 0.75 of Li et al. (2020) up to twenty-three cubes on cube stacking, the task closest to theirs, a nearly fourfold extension, and up to thirty on the line, a fivefold one. These comparisons cross tasks, simulators, embodiments and in places success criteria (Appendix E), since no controlled comparison exists at these counts.

Cube stacking with $h = 6$ tests placement precision. The training scenes hold at most seven cubes, one full stack and the base cube of a second, so no training episode places a cube onto the second stack. WAVE still builds the second stack up to five cubes high and completes 0.97 or more of the episodes up to eleven cubes. Success falls to 0.85 at twelve cubes, where the second stack needs its sixth cube, and is zero from twenty-four on, lost to toppled stacks rather than missed placements.

At high counts, episodes fail mostly by running out of time rather than by losing cubes (Table 5). On cube stacking ($h = 3$), on average 26.8 of 30 cubes are placed at the same time at the peak, but placed-to-unplaced transitions grow from 7.5 per episode at ten cubes to 126 at thirty. The policy replaces most of these cubes, which costs time and shrinks the net progress per attempt (Section 3.6). At thirty cubes, 0.49 of the episodes end at the time limit and 0.06 because a cube left the workspace. With 60 s instead of 20 s per cube, success at thirty cubes rises from 0.48 to 0.67 on stacking and from 0.08 to 0.27 on the pyramid..

Appendix J reports further generalization experiments with the unchanged skill. One pair of policies trained on five tasks at once completes all five. WAVE also handles larger cubes, boxes and scenes with objects of different sizes and shapes, and it still completes its tasks when disturbances set cubes to other positions during the episode. Success decreases beyond five cubes for cubes with 0.8 times the edge, and when a built structure of six or more cubes has to be taken apart without a cube falling.

4.2 Count generalization on the Panda arm. We transfer the same method and reward to a Franka Panda arm with 4 cm cubes in two embodiments, each with its own frozen skill (Appendix A). A parallel-jaw gripper handles cube stacking ($h = 3$), the pyramid and the wall. A sphere-tip pusher, whose skill pushes one cube to a planar position and yaw goal, handles the line and the square. Figure 4 shows the first fifteen counts and Table 3 the full range of each task.

With the pusher, WAVE completes 0.95 of the line episodes at nine cubes, its largest training count, 0.78 at fifteen and 0.62 at twenty, and failed episodes end only a cube or two short (14.2 of fifteen

Table 1: Whole task success of WAVE and seven of the ten baselines on the gantry, the mean over three seeds. The first three columns give how each method uses μ, which objects besides x^* it sees, and how it picks the next object. Every seed of every other baseline in this table is at 0.00 at six and at thirty cubes. Bold marks the highest mean per count. Seed ranges, every count and the other three baselines are in Table 4.

				line, N					cube stacking ($h = 3$), N				
	μ	other objects	selection	1	2	3	6	30	1	2	3	6	30
WAVE	frozen	crop	learned	**1.00**	**1.00**	**1.00**	**1.00**	**0.90**	**1.00**	**1.00**	**1.00**	**1.00**	**0.48**
input edit fixed per episode	frozen	crop	learned	**1.00**	0.93	0.98	0.75	0.25	**1.00**	0.98	**1.00**	0.90	0.00
skill alone	frozen	none	scripted	0.79	0.35	0.00	0.00	0.00	0.97	0.17	0.00	0.00	0.00
TD7 on the crop	–	crop	scripted	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
fine-tuning μ on the crop, $\omega = 1$	fine-tuned	crop	scripted	0.96	0.43	0.03	0.00	0.00	0.98	0.64	0.03	0.00	0.00
HRL subgoals	frozen	crop	learned	0.54	0.18	0.00	0.00	0.00	0.83	0.53	0.10	0.00	0.00
EIT (Haramati et al., 2024)	–	all	–	0.20	0.00	0.00	0.00	0.00	0.53	0.00	0.00	0.00	0.00
EIT distilled from μ	distilled	all	–	0.82	0.56	0.37	0.00	0.00	0.94	0.20	0.00	0.00	0.00

cubes placed on average on these episodes). The square is harder, 0.60 at nine and 0.10 at fifteen, since its interior slots leave the pusher no free approach side and late cubes are pushed in across placed ones until the episode runs out of time.

With the gripper, success already falls within the training range. Cube stacking completes 0.75 at six, level with the best published result (Li et al., 2020), and 0.37 at ten, while the pyramid and the wall complete 0.22 and 0.05 at six. Both failure modes of Section 4.1 appear early. At six cubes the wall ends 0.76 of its episodes by timeout, and on cube stacking the fraction of placed cubes that stay placed falls from 0.95 at six cubes to 0.66 at fifteen. This is likely due to the large body of the arm, which knocks neighbors off their goals when cubes are placed next to each other. Longer training, larger networks or a richer crop may close this gap and are left for future work.

We also deployed a cube stacking policy trained in simulation on a physical Panda arm, with cube poses tracked from AprilTag fiducials by four calibrated RGB cameras. It placed all five cubes of a two-tower task in 8 of 10 runs, and a policy fine-tuned for five-cube towers built a five-high tower (Appendix C).

4.3 Baselines. We compare WAVE on the line and on cube stacking ($h = 3$) with ten baselines, trained on the same gantry, task, reward and training counts, and with the same μ wherever one is used. Table 1 reports seven of them and Appendix D all ten. Apart from HRL subgoals and the fixed input edit, which keep WAVE's selection policy, the baselines that act on one object at a time use a scripted nearest-first rule that offers only objects whose goal is currently fulfillable, an easier task than the one WAVE learns from the reward. Nine of the ten baselines fail already within their training counts and complete at most 0.02 of the episodes with six cubes and none with more. For these, the comparison shows that they do not learn the multi-object task with our training budget, and it cannot tell how they would extrapolate if they did. Only the fixed input edit learns its training counts and thereby tests extrapolation.

TD7 on WAVE's crop without μ completes no episode at any count, and the Entity Interaction Transformer (EIT) of Haramati et al. (2024) learns only the one-cube task, 0.53 on cube stacking and 0.20 on the line. Distilling μ into the EIT raises its success on the line to 0.82 at one cube and 0.37 at three, but it still fails within its training counts. We did not isolate the cause of either failure.

Fine-tuning μ on its own input and JSRL (Uchendu et al., 2023) retain most of the one-cube skill (0.72 and 0.45 on the line, 0.88 and 0.87 on cube stacking) and improve on the skill alone at two cubes on cube stacking (0.42 and 0.45 against 0.17), but complete at most 0.13 of the episodes with three or more cubes. Fine-tuning on the crop without an anchor completes no episode at any count. Anchoring the actor to μ (Fujimoto & Gu, 2021) with $\omega = 1$ keeps the skill and improves on it at two cubes (0.64 against 0.17 on cube stacking) but drops to 0.03 at three. The anchor penalizes every deviation from μ equally, while a warp can remain at the identity on some channels and steps and take over others ($m_j = 0$).

HRL subgoals, which moves only the goal μ pursues, completes 0.53 of the two-cube stacking episodes and at most 0.10 from three cubes on. A subgoal changes where μ goes but not how it releases a cube onto a stack, the phase that needs the output side of the warp (Appendix G). Warping only μ's input channels, which also edits what μ sees of the robot and of the object, nearly matches the full warp, 0.81 against 0.82 mean success (Section 4.4).

As a further baseline we apply the interface of Zhang et al. (2023), whose state projection is constant over an episode, to our setting. We train WAVE with the input edit chosen at the first step of an episode and kept until its end and change nothing else, so the output edit is still chosen at every step, and the offsets, the crop and the learned selection policy, which are not part of their method, remain. This baseline learns its training counts (at least 0.83 on cube stacking and 0.75 on the line) but extrapolates worse than WAVE. On cube stacking it completes 0.60 of the episodes at ten cubes and 0.08 at twenty, compared to 0.98 and 0.95 for WAVE (Figure 7). On the line it completes 0.55 and 0.35, compared to 1.00 and 0.93.

4.4 Ablations. We ablate WAVE on cube stacking ($h = 3$), the square and the pyramid. First, we replace the selection rule of equation 2 with a random choice or a nearest-first choice over the objects whose goal is currently fulfillable, a filter the learned selection policy does not use. Second, we let the warp edit only the input or only the output channels of μ. Editing only the input is close to methods that remap observations (Alqithami, 2026; Liu et al., 2026), and editing only the output is close to residual and action distribution methods (Silver et al., 2018; Johannink et al., 2019; Hirschowitz & Ramos, 2026). Third, we let the warp emit only offsets or only multipliers. Random selection completes no episode with five or more cubes on cube stacking and the pyramid, or with nine or more on the square, while the fixed nearest-first rule matches the learned selection policy on stacking and the square and falls behind it on the pyramid. Warping only the input channels nearly matches the full warp, warping only the output channels collapses by ten cubes, and warps with only multipliers or only offsets fall in between, multipliers above offsets. Curves, analysis and the constants of equation 5 are in Appendix F.

To measure when in an episode each side of the warp is needed, we stop a running episode at every time step, copy the simulator, and continue the copies with either the input or the output warp switched off. We then measure how much longer the current phase of the manipulation (approach, grasp, carry, release or depart) takes compared to copies where the warp stays on. Switching off the output warp delays mainly the release, since the frozen skill never releases a cube on its own. Switching off the input warp slows down every phase, by 0.95–1.56 s, because it steers the skill throughout the episode. Appendix G gives the details and shows an example episode.

5 Conclusion and Limitations

We introduce WAVE, which extends a frozen single-object skill to multi-object arrangement tasks. It shows the skill one object at a time, learns a warping policy that edits every channel of the skill's interface, and learns a selection policy that finds from outcomes which object to attempt next. The skill is a black box, no learned component observes the object count, and all six tasks use the same reward and hyperparameters. Trained with at most nine cubes, WAVE generalizes on the gantry to two to five times the largest count reached by published methods that learn the manipulation itself, whereas fine-tuning the skill, a policy on the same local view and an entity transformer, even when distilled from the skill, collapse within their training counts under the same training budget. In the main experiments, one warping policy and one selection policy are trained per task.

We evaluated WAVE in simulation on three embodiments and demonstrated the real-world transfer of a multi-object stacking policy on a physical Panda arm (Appendix C). The Panda gripper tasks degrade within their training range (Section 4.2). The method applies only to tasks that provide a goal pose for every object. The main experiments use identical cubes, and an object class that the skill cannot manipulate alone would need its own skill. The policies do not anticipate objects that move on their own. The skill moves one object at a time, so tasks that need the coordinated motion of several objects, or in which objects affect each other without touching, are out of scope.

ACKNOWLEDGEMENTS

This work has received support from the German Federal Ministry of Research, Technology and Space (BMFTR) under the Robotics Institute Germany (RIG) and as grant BIFOLD (01IS18025A, 01IS180371I).

AI USE STATEMENT

In this work, we used generative AI tools for related work surveys, implementations, conduction of experiments and drafting parts of this paper. We have not used generative AI tools for summarization and reading of related papers, the ideation of our approach, the generation of plots and for the final version of this paper. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

WAVE is basic research on making manipulation policies generalize across object counts, and every result in this paper except one demonstration on a physical Panda arm is simulated. The risks it touches are the generic ones of more capable robot manipulation, displaced manual handling labor and the possibility of building manipulation into systems used for harmful ends, and none is specific to this method or amplified by it. On the positive side, reusing a trained skill instead of retraining from scratch lowers the compute cost of extending a robot's competence.

REPRODUCIBILITY STATEMENT

Appendix A specifies the robot, the simulation, the frozen skill and its training, the six tasks and their goal predicate, the reward with all its constants, every hyperparameter of the warping and selection policies and the evaluation protocol. Appendix B reports the full per count results with seed ranges, Appendix F describes how the constants of equation 5 are measured, and Appendix H states every assumption of Proposition 1 and Theorem 1 and gives the complete proof. Code, videos and full results are available at `https://hhi-robot-learning.github.io/wave-website/`, and the supplement contains the implementation and the code that reproduces our results.

REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on Robot Learning (CoRL)*, 2022.

Saad Alqithami. DC-Ada: Reward-only decentralized sensor adaptation for heterogeneous multi-robot teams. *IEEE Access*, 14:74241–74260, 2026. doi: 10.1109/ACCESS.2026.3687163.

Lars Ankile, Anthony Simeonov, Idan Shenfeld, Marcel Torne, and Pulkit Agrawal. From imitation to refinement – residual RL for precise assembly. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–8, 2025. doi: 10.1109/ICRA55743.2025.11127442.

Pavol Brunovský. A classification of linear controllable systems. *Kybernetika*, 6(3):173–188, 1970.

Hien Bui, Yufeiyang Gao, Haoran Yang, Eric Cui, Siddhant Mody, Brian Acosta, Thomas Stephen Felix, Bibit Bianchini, and Michael Posa. Push anything: Single- and multi-object pushing from first sight with contact-implicit MPC. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2026.

Rohan Chitnis, Tom Silver, Joshua B. Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Learning neuro-symbolic relational transition models for bilevel planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

Satyajeet Das, Darren Chiu, Shashank Hegde, and Gaurav S. Sukhatme. Steering multirobot behavior via closed-loop affine activation editing. *arXiv preprint arXiv:2606.11489*, 2026. arXiv:2606.11489.

Fan Feng, Phillip Lippe, and Sara Magliacane. Learning interactive world model for object-centric reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Scott Fujimoto, Wei-Di Chang, Edward J. Smith, Shixiang Shane Gu, Doina Precup, and David Meger. For SALE: State-action representation learning for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Alexandre Galashov, Siddhant M. Jayakumar, Leonard Hasenclever, Dhruva Tirumala, Jonathan Schwarz, Guillaume Desjardins, Wojciech M. Czarnecki, Yee Whye Teh, Razvan Pascanu, and Nicolas Heess. Information asymmetry in KL-regularized RL. In *International Conference on Learning Representations (ICLR)*, 2019.

Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. PDDLStream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning. In *International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 440–448, 2020.

Seyed Kamyar Seyed Ghasemipour, Satoshi Kataoka, Byron David, Daniel Freeman, Shixiang Shane Gu, and Igor Mordatch. Blocks assemble! learning to assemble with large-scale structured reinforcement learning. In *International Conference on Machine Learning (ICML)*, volume 162 of *Proceedings of Machine Learning Research*, pp. 7435–7469, 2022.

Raj Ghugare, Roger Creus Castanyer, Catherine Ji, Kathryn Wantlin, Jin Schofield, Karthik Narasimhan, and Benjamin Eysenbach. BuilderBench: The building blocks of intelligent agents. *arXiv preprint arXiv:2510.06288*, 2025.

Daniel Graves, Jun Jin, and Jun Luo. LISPR: An options framework for policy reuse with reinforcement learning. *arXiv preprint arXiv:2012.14942*, 2020. arXiv:2012.14942.

Dan Haramati, Tal Daniel, and Aviv Tamar. Entity-centric reinforcement learning for object manipulation from pixels. In *International Conference on Learning Representations (ICLR)*, 2024.

Dan Haramati, Carl Qi, Tal Daniel, Amy Zhang, Aviv Tamar, and George Konidaris. Hierarchical entity-centric reinforcement learning with factored subgoal diffusion. In *International Conference on Learning Representations (ICLR)*, 2026.

Ethan Hirschowitz and Fabio Ramos. Warp RL: Reshaping base policy distributions for dynamics adaptation. *arXiv preprint arXiv:2606.31043*, 2026. arXiv:2606.31043.

Bronisław Jakubczyk and Witold Respondek. On linearization of control systems. *Bulletin de l'Académie Polonaise des Sciences, Série des Sciences Mathématiques*, 28:517–522, 1980.

Jaromír Janisch, Tomáš Pevný, and Viliam Lisý. Symbolic relational deep reinforcement learning based on graph neural networks and autoregressive policy decomposition. *arXiv preprint arXiv:2009.12462*, 2020. arXiv:2009.12462.

Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6023–6029, 2019. doi: 10.1109/ICRA.2019.8794127.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. QT-Opt: Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning (CoRL)*, 2018.

Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of zero-shot generalisation in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 76:201–264, 2023.

Markus Krimmel, Jan Achterhold, and Joerg Stueckler. Attention normalization impacts cardinality generalization in slot attention. *Transactions on Machine Learning Research*, 2024, 2024. URL https://openreview.net/forum?id=llQXLfbGOq.

Thomas Lampe, Abbas Abdolmaleki, Sarah Bechtle, Sandy H. Huang, Jost Tobias Springenberg, Michael Bloesch, Oliver Groth, Roland Hafner, Tim Hertweck, Michael Neunert, Markus Wulfmeier, Jingwei Zhang, Francesco Nori, Nicolas Heess, and Martin Riedmiller. Mastering stacking of diverse shapes with large-scale iterative reinforcement learning on real robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7772–7779, 2024. doi: 10.1109/ICRA57147.2024.10610297.

John B. Lanier, Stephen McAleer, and Pierre Baldi. Curiosity-driven multi-criteria hindsight experience replay. *arXiv preprint arXiv:1906.03710*, 2019. arXiv:1906.03710.

Qiyang Li, Seohong Park, and Sergey Levine. Decoupled Q-chunking. *arXiv preprint arXiv:2512.10926*, 2025.

Richard Li, Allan Jabri, Trevor Darrell, and Pulkit Agrawal. Towards practical multi-object manipulation using relational reinforcement learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4051–4058, 2020. doi: 10.1109/ICRA40945.2020.9197468.

Yunfei Li, Tian Gao, Jiaqi Yang, Huazhe Xu, and Yi Wu. Phasic self-imitative reduction for sparse-reward goal-conditioned reinforcement learning. In *International Conference on Machine Learning (ICML)*, volume 162 of *Proceedings of Machine Learning Research*, pp. 12765–12781, 2022.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

Yixin Lin, Austin S. Wang, Eric Undersander, and Akshara Rai. Efficient and interpretable robot manipulation with graph neural networks. *IEEE Robotics and Automation Letters*, 7(2):2740–2747, 2022. doi: 10.1109/LRA.2022.3143518.

Shuo Liu, Ishneet Sukhvinder Singh, Yiqing Xu, Jiafei Duan, and Ranjay Krishna. VLS: Steering pretrained robot policies via vision-language models. *arXiv preprint arXiv:2602.03973*, 2026. arXiv:2602.03973.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Annabella Macaluso, Haochen Zhang, Ishaan Masilamony, Yingshan Chang, and Yonatan Bisk. Inductive generalization for robotic manipulation. *arXiv preprint arXiv:2606.20999*, 2026. arXiv:2606.20999.

Davide Mambelli, Frederik Träuble, Stefan Bauer, Bernhard Schölkopf, and Francesco Locatello. Compositional multi-object reinforcement learning with linear relation networks. *arXiv preprint arXiv:2201.13388*, 2022. arXiv:2201.13388.

Miranda Muqing Miao, Subin Kim, Brandon Yang, and Lyle Ungar. Contrastive conceptor activation steering (COAST): Unlocking vision-language-action models through hidden states. *arXiv preprint arXiv:2605.17144*, 2026. arXiv:2605.17144.

Simon Ouellette, Rolf Pfister, and Hansueli Jud. Counting and algorithmic generalization with transformers. *arXiv preprint arXiv:2310.08661*, 2023.

Bumgeun Park and Donghwan Lee. Adaptive policy backbone via shared network. In *International Conference on Machine Learning (ICML)*, 2026.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In *International Conference on Learning Representations (ICLR)*, 2025a.

Seohong Park, Kevin Frans, Deepinder Mann, Benjamin Eysenbach, Aviral Kumar, and Sergey Levine. Horizon reduction makes RL scalable. In *Advances in Neural Information Processing Systems*, volume 38, 2025b.

Carl Qi, Dan Haramati, Tal Daniel, Aviv Tamar, and Amy Zhang. EC-Diffuser: Multi-object manipulation via entity-centric behavior generation. In *International Conference on Learning Representations (ICLR)*, 2025.

Guannan Qu, Adam Wierman, and Na Li. Scalable reinforcement learning for multi-agent networked systems. *Operations Research*, 2022.

Balaraman Ravindran and Andrew G. Barto. SMDP homomorphisms: An algebraic approach to abstraction in semi-markov decision processes. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1011–1018, 2003. URL http://ijcai.org/Proceedings/03/Papers/145.pdf.

Martin Riedmiller, Andrea Gesmundo, Tim Hertweck, and Roland Hafner. Less is more – the dispatcher/executor principle for multi-task reinforcement learning. *arXiv preprint arXiv:2312.09120*, 2023. arXiv:2312.09120.

Luana Ruiz, Luiz F. O. Chamon, and Alejandro Ribeiro. Graphon neural networks and the transferability of graph neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1702–1712, 2020.

Simon Schmitt, Jonathan J. Hudson, Augustin Zídek, Simon Osindero, Carl Doersch, Wojciech M. Czarnecki, Joel Z. Leibo, Heinrich Küttler, Andrew Zisserman, Karen Simonyan, and S. M. Ali Eslami. Kickstarting deep reinforcement learning. *arXiv preprint arXiv:1803.03835*, 2018. arXiv:1803.03835.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations (ICLR)*, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, 2000.

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

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2 edition, 2018.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1–2):181–211, 1999. doi: 10.1016/S0004-3702(99)00052-1.

Matthew E. Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10:1633–1685, 2009.

Matthew E. Taylor, Peter Stone, and Yaxin Liu. Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research*, 8:2125–2167, 2007.

Dhruva Tirumala, Alexandre Galashov, Hyeonwoo Noh, Leonard Hasenclever, Razvan Pascanu, Jonathan Schwarz, Guillaume Desjardins, Wojciech Marian Czarnecki, Arun Ahuja, Yee Whye Teh, and Nicolas Heess. Behavior priors for efficient reinforcement learning. *Journal of Machine Learning Research*, 23(221):1–68, 2022. URL https://jmlr.org/papers/v23/20-1038.html.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

Marc Toussaint, Cornelius V Braun, Armand Jordana, Sayantan Auddy, Eckart Cobo-Briesewitz, Denis Shcherba, Tilman Burghoff, and Justin Carpentier. Combined constrained sampling and reinforcement learning for robotic manipulation. *arXiv preprint arXiv:2602.08557*, 2026.

Ikechukwu Uchendu, Ted Xiao, Yao Lu, Banghua Zhu, Mengyuan Yan, Joséphine Simon, Matthew Bennice, Chuyuan Fu, Cong Ma, Jiantao Jiao, Sergey Levine, and Karol Hausman. Jump-start reinforcement learning. In *International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 34556–34583, 2023.

Petar Veličković, Christos Perivolaropoulos, Federico Barbero, and Razvan Pascanu. Softmax is not enough (for sharp size generalisation). In *International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 61190–61211, 2025.

Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. *arXiv preprint arXiv:2506.15799*, 2025.

Keyulu Xu, Mozhi Zhang, Jingling Li, Simon S. Du, Ken ichi Kawarabayashi, and Stefanie Jegelka. How neural networks extrapolate: From feedforward to graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2021.

Gilad Yehudai, Ethan Fetaya, Eli Meirom, Gal Chechik, and Haggai Maron. From local structures to size generalization in graph neural networks. In *International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, 2021.

Dongjie Yu, Kun Lei, Zhennan Jiang, Jia Pan, and Huazhe Xu. Beyond action residuals: Real-world robot policy steering via bottleneck latent reinforcement learning. *arXiv preprint arXiv:2605.19919*, 2026. arXiv:2605.19919.

Xiu Yuan, Tongzhou Mu, Stone Tao, Yunhao Fang, Mengke Zhang, and Hao Su. Policy decorator: Model-agnostic online refinement for large policy model. In *International Conference on Learning Representations (ICLR)*, 2025. URL `https://openreview.net/forum?id=e5jGTEiJMT`.

Andrii Zadaianchuk, Maximilian Seitzer, and Georg Martius. Self-supervised visual reinforcement learning with object-centric representations. In *International Conference on Learning Representations (ICLR)*, 2021.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J. Smola. Deep sets. In *Advances in Neural Information Processing Systems*, volume 30, pp. 3391–3401, 2017.

Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, Murray Shanahan, Victoria Langston, Razvan Pascanu, Matthew Botvinick, Oriol Vinyals, and Peter Battaglia. Deep reinforcement learning with relational inductive biases. In *International Conference on Learning Representations (ICLR)*, 2019.

Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Ayzaan Wahid, Vikas Sindhwani, and Johnny Lee. Transporter networks: Rearranging the visual world for robotic manipulation. In *Conference on Robot Learning (CoRL)*, volume 155 of *Proceedings of Machine Learning Research*, pp. 726–747, 2020.

Xiang Zhang, Siddarth Jain, Baichuan Huang, Masayoshi Tomizuka, and Diego Romeres. Learning generalizable pivoting skills. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

Pei Zhou, Wanting Yao, Qian Luo, Xunzhe Zhou, and Yanchao Yang. Hyper-GoalNet: Goal-conditioned manipulation policy learning with hypernetworks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Roland S. Zimmermann, Sjoerd van Steenkiste, Mehdi S. M. Sajjadi, Thomas Kipf, and Klaus Greff. Sensitivity of slot-based object-centric models to their number of slots. *arXiv preprint arXiv:2305.18890*, 2023. arXiv:2305.18890.

Appendix

Table of Contents

A Experimental setup 16

The robots and the simulation, the frozen skill, the six tasks with the rule for when a cube counts as placed, the training hyperparameters, the reward and the evaluation protocol.

B Full results 19

Whole-task success at every object count for all gantry and Panda tasks, the curves and the table of all ten baselines, and how episodes end (success, time limit or a cube that left the workspace).

C Real-world Panda experiments 19

A cube stacking policy trained in simulation and run on a physical Panda arm with camera-based pose tracking, with ten consecutive rollouts and a tower of five cubes.

D Baselines 20

How each of the ten baselines is built and trained.

E Overview of published methods and results 24

The published multi-object results that Figure 3 shows in gray, with their tasks, embodiments and success criteria.

F Ablations and measured constants 25

Ablations of the selection rule, of the two sides of the warp and of its multipliers and offsets, and the measured quantities of the completion bound per attempt.

G Interventional estimation of warp necessity 29

The input or the output warp is switched off in copies of a running episode to measure in which phase of the manipulation each side is needed.

H Formal statements and proofs 29

The assumptions, Proposition 1 on when an attempt of the warped skill is an episode of the skill alone, and Theorem 1 on the number of attempts that complete a task, with proofs.

I Positioning of interface-editing methods 33

A comparison of WAVE with prior methods that edit the input or the output of a frozen policy.

J Generalization experiments 34

Eight further experiments: one pair of policies for five tasks, other object sizes and shapes, cubes that have to stay in place, disturbances, episodes that start from a built structure, stricter success criteria, and replacements of the minimum over the other objects.

A Experimental setup

Robot and simulation. The robot is a three axis gantry with a three joint wrist and a parallel-jaw gripper (8 joints, 7 actuators). It is commanded at 20 Hz with a 7-D action (three translations, three wrist rotations, one grip command). The scene is simulated in MuJoCo (Todorov et al., 2012) on

the GPU with 2 ms physics substeps and 2048 parallel environments. Objects are cubes of 16 cm edge length and 0.2 kg in a 2.7 m $\times$ 2.7 m workspace. The Panda experiments of Section 4.2 use a simulated 7-DoF Franka Panda arm commanded by joint velocities, with cubes of 4 cm edge length, in two embodiments, one with a parallel-jaw gripper and a binary grip channel (8-D action) and one with a rigid sphere-tip in place of the jaws (7-D action) that pushes cubes across the table and never lifts them.

The skill μ. μ moves one cube from any resting pose to any goal pose, including the goal orientation. We train it with TD7 (Fujimoto et al., 2023) in a single-object scene with constrained sampling and reinforcement learning (CSRL) (Toussaint et al., 2026). Its input is the 65-D state vector of joint and cube positions, joint and cube velocities, the servo errors of the 7 actuators, the positions of four cube corners, the goal-relative rotation and the corner to goal offsets, and its action is the 7-D command above. μ is frozen in every experiment except the fine-tuning baseline. On its own task it succeeds in 0.97 of 64 evaluation episodes at the checkpoint we freeze. Each Panda embodiment has its own skill, trained the same way in its own single-object scene. The gripper skill places one cube as the gantry skill does, and the pusher skill brings one cube to a position and yaw goal in the plane, where the cube's fourfold symmetry identifies yaws a quarter turn apart.

Tasks. The six tasks share one goal predicate and differ in the goal layout. The pyramid, square, line and the wall grow with N, and the stacks are repeated at their target height. Every task is trained and evaluated in the 2.7 m $\times$ 2.7 m workspace, except that the line is evaluated in a workspace widened to fit thirty cubes. A cube counts as placed when it is within the tolerance of its goal position and on its intended support (the floor or the cube beneath it). On the gantry, whose cubes are 16 cm, the tolerance is 7 cm in height and 8.5 cm in the plane for stacking and 7.65 cm in the plane for the line. On the Panda, whose cubes are 4 cm, it is 1.75 cm with the gripper and 1.25 cm with the pusher. Support is determined from the cube positions and not from contact forces. A cube counts as supported if its center is within a quarter of a cube edge of floor height, or if another cube lies one cube height below it, either directly underneath (within 0.6 of a cube edge horizontally) or as one of two cubes on opposite sides. A cube in an upper layer counts as placed only if the cubes it rests on in the goal structure are placed as well. Initial layouts are drawn from a pool of sampled resting configurations, and every count N is built as a scene of exactly N cubes. An episode lasts $20N$ seconds.

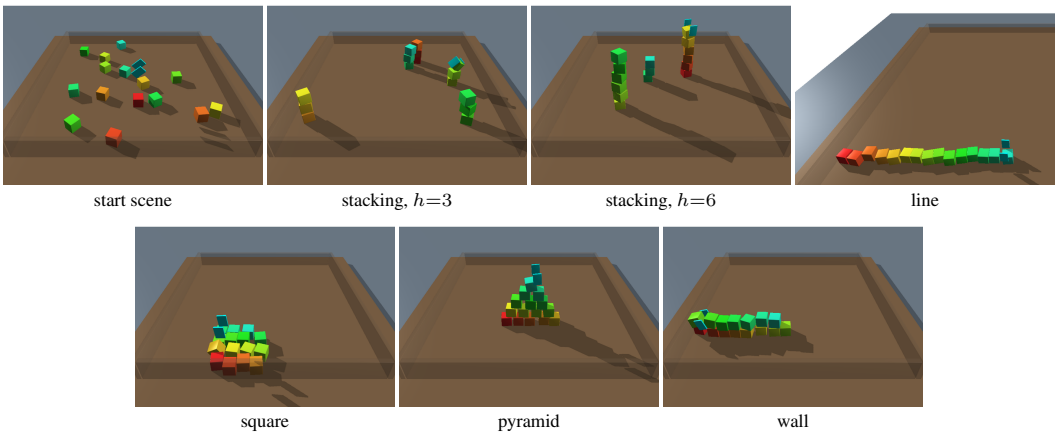


Figure 5: Renderings of the start scene and the final frames of all six gantry tasks.

Training. For every task we train one warping policy and one selection policy with PPO on scenes whose object count is drawn uniformly from 1 to N_{train}, with $N_{\text{train}} = 7$ for cube stacking with either height, 6 for the line and the pyramid, 9 for the square and 8 for the wall on the gantry, and 7 for cube stacking, 6 for the pyramid, 8 for the wall and 9 for the line and the square on the Panda. A gantry run uses 5.0×10^8 environment steps (2048 environments $\times$ 96 steps $\times$ 2543 iterations) on one A100 GPU within 10 h, and every Panda run uses 7.9×10^8 (4000 iterations) environment steps within 18 h. The warping policy uses a two layer transformer encoder with width 128 and four heads, followed by a two layer transformer decoder with one query per channel group that emits one categorical distribution per warp parameter. The warp edits 55 channels of μ with one multiplier

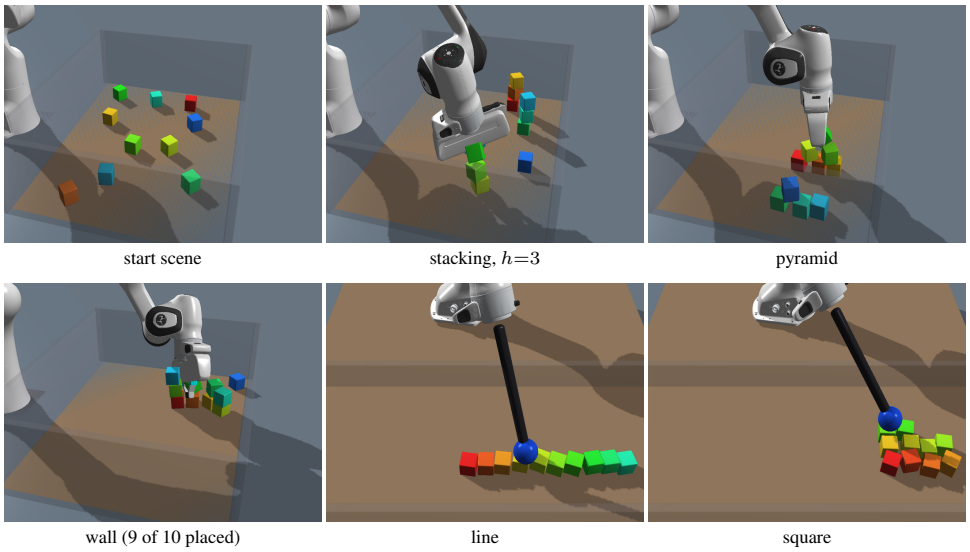


Figure 6: Renderings of the start scene and the final frames of all five Panda tasks.

and one offset each, i.e., 110 parameters. The 47 input channels are the presented goal position and orientation (3 + 3), the object position and orientation (3 + 3), the object linear and angular velocity (3 + 3), the robot joint positions and velocities (9 + 9), the object to goal offset (3) and the servo errors (8). The 8 output channels are the 7 motion commands and the grip command. The remaining inputs of μ, such as the cube corners, are computed from the warped channels. The channel counts fit the Panda gripper, the largest of our robots, and on the gantry the 4 surplus channels stay at the identity. Multipliers take values on the 25 point grid of Figure 2, and offsets take values on a 21 point grid of multiples of a 1.5 cm step, up to 32 steps. Every parameter distribution is initialized with 80% of its mass on the identity value. PPO uses a learning rate of 10^{-3} for the warping policy and 3×10^{-4} for the selection policy, both annealed by a cosine schedule to a tenth, a clip ratio of 0.2, entropy coefficients of 5×10^{-4} and 3×10^{-3}, GAE with $\lambda_{\mathrm{GAE}} = 0.95$, and four epochs over minibatches of 8192 samples. The selection policy scores are drawn through a softmax during training. We train three seeds for every evaluation.

Reward. The task reward is sparse and requires long successful manipulation trajectories before it pays a reward. Placed cubes can be knocked over again, so keeping an arrangement must be worth as much as extending it. Otherwise, a fast placement that knocks down other cubes would earn more return than a slower one that keeps them in place. Episode lengths grow with N, so the balance between these terms must not shift with the count. R_N is built from the goal completion predicate alone. Every control step pays a reward of $w(1-\gamma)/2$ times the fraction of currently placed cubes, completion pays $w\gamma$ once, and the first placement of each cube pays bw/N once, with $w = 4$ and $b = 0.65$. This per step reward makes holding a partial arrangement valuable, so a knockdown lowers the return without a dedicated penalty term. The completion bonus keeps the actual objective dominant in the return. The first-placement bonus makes the earliest success of every cube visible to the policy gradient, and its division by N fixes its total mass across counts. The discount is set per episode to $\gamma = 1 - 3/D$, with D the number of control steps in the episode, so the terminal bonus is discounted by e^{-3} and the ratio of per step reward to bonus is the same at every N. The same constants are used for every task in Section 4.

Evaluation. Initial layouts come from a held-out pool whose seed differs from the training pool. For every count, $N \in \{1, \ldots, 30\}$ on the gantry and the Panda pusher square, up to 20 on the pusher line and up to 15 on the Panda gripper tasks, we build scenes of exactly N cubes and run the first episode of each of 20 scenes with the deterministic (argmax) warping and selection policies and the training clock of 20 s per cube. Alongside whole-task success we record, per count, the mean peak number of simultaneously placed cubes, the retention (placed cubes at the end divided by the peak), the number of placed to unplaced transitions, the fraction of the episode elapsed at the last new placement and the fraction of episodes that end by timeout.

B FULL RESULTS

Table 2 reports the whole-task success of WAVE on the gantry per task and count and Table 3 on the Panda arm, as the mean and the range over seeds.

Table 2: Whole task success of WAVE on the gantry per task and object count, the first episode of each of 20 scenes per count and 20 s per cube, mean with the minimum and maximum over three seeds in parentheses. Cube stacking builds stacks of height h. The line is evaluated in a workspace widened to fit thirty cubes, and the wall is two courses high and grows longer with the count.

N	stacking, $h = 3$	stacking, $h = 6$	line	square	pyramid	wall
1	1.00 (1.00–1.00)	1.00 (1.00–1.00)	1.00 (1.00–1.00)	1.00 (1.00–1.00)	1.00 (1.00–1.00)	1.00 (1.00–1.00)
3	1.00 (1.00–1.00)	1.00 (1.00–1.00)	1.00 (1.00–1.00)	1.00 (1.00–1.00)	1.00 (1.00–1.00)	1.00 (1.00–1.00)
5	0.95 (0.85–1.00)	1.00 (1.00–1.00)	1.00 (1.00–1.00)	1.00 (1.00–1.00)	1.00 (1.00–1.00)	0.98 (0.95–1.00)
7	1.00 (1.00–1.00)	1.00 (1.00–1.00)	1.00 (1.00–1.00)	0.98 (0.95–1.00)	0.92 (0.85–0.95)	0.98 (0.95–1.00)
10	0.98 (0.95–1.00)	1.00 (1.00–1.00)	1.00 (1.00–1.00)	0.97 (0.95–1.00)	0.90 (0.80–0.95)	0.93 (0.85–1.00)
12	0.98 (0.95–1.00)	0.85 (0.80–0.95)	0.97 (0.95–1.00)	0.98 (0.95–1.00)	0.80 (0.80–0.80)	0.95 (0.90–1.00)
15	0.92 (0.85–1.00)	0.72 (0.55–0.85)	1.00 (1.00–1.00)	0.97 (0.90–1.00)	0.60 (0.45–0.70)	0.85 (0.70–0.95)
20	0.95 (0.90–1.00)	0.13 (0.10–0.20)	0.93 (0.90–1.00)	0.85 (0.65–0.95)	0.35 (0.20–0.45)	0.67 (0.45–0.80)
25	0.63 (0.45–0.80)	0.00 (0.00–0.00)	0.97 (0.95–1.00)	0.77 (0.65–0.85)	0.20 (0.15–0.25)	0.63 (0.40–0.75)
28	0.53 (0.35–0.65)	0.00 (0.00–0.00)	0.92 (0.85–0.95)	0.80 (0.60–1.00)	0.10 (0.05–0.15)	0.48 (0.10–0.75)
30	0.48 (0.30–0.60)	0.00 (0.00–0.00)	0.90 (0.80–1.00)	0.55 (0.45–0.60)	0.08 (0.05–0.15)	0.38 (0.05–0.85)

Table 3: Whole task success of WAVE on the Panda arm per task and object count, the same protocol as Table 2. Cube stacking ($h = 3$), the pyramid and the wall use the parallel-jaw gripper and the line and the square the sphere-tip pusher. A dash marks a count the robot was not evaluated on.

	gripper			pusher	
N	stacking, $h = 3$	pyramid	wall	line	square
1	1.00 (1.00–1.00)	1.00 (1.00–1.00)	0.98 (0.95–1.00)	1.00 (1.00–1.00)	1.00 (1.00–1.00)
3	0.92 (0.85–1.00)	0.90 (0.85–0.95)	0.70 (0.65–0.75)	1.00 (1.00–1.00)	1.00 (1.00–1.00)
5	0.87 (0.75–0.95)	0.47 (0.15–0.65)	0.23 (0.10–0.40)	1.00 (1.00–1.00)	1.00 (1.00–1.00)
7	0.63 (0.60–0.70)	0.28 (0.10–0.55)	0.03 (0.00–0.10)	0.98 (0.95–1.00)	0.92 (0.75–1.00)
9	0.38 (0.20–0.60)	0.17 (0.10–0.20)	0.00 (0.00–0.00)	0.95 (0.90–1.00)	0.60 (0.50–0.70)
12	0.03 (0.00–0.10)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.80 (0.75–0.85)	0.35 (0.20–0.55)
15	0.03 (0.00–0.10)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.78 (0.75–0.80)	0.10 (0.00–0.15)
20	–	–	–	0.62 (0.50–0.70)	0.00 (0.00–0.00)
25	–	–	–	–	0.00 (0.00–0.00)
30	–	–	–	–	0.00

C REAL-WORLD PANDA EXPERIMENTS

We deployed a cube stacking policy trained in simulation on a physical Panda arm. The policies are state-based, so we estimate the poses of the $4 \times 4 \times 4$ cm cubes from AprilTag fiducials tracked by four calibrated RGB cameras.

We evaluate the policy on ten consecutive rollouts of building a three-cube and a two-cube tower from five cubes (Table 6, Figure 9). The two base cubes start in their goal slots, so three cubes have to be picked and placed. The initial layout differs between runs. In 8 of the 10 runs all five cubes reached their goals (95% Wilson interval 49–94%), after 410 ± 248 policy steps (92 ± 54 s) on average. In 5 of the 10 runs the policy also released the last cube and both towers stood at the end. The gap between the two rates comes from the final release, which the policy never learned. In training and in the simulation evaluation, an episode ends as a success as soon as every cube is within tolerance and on its support, even if the last cube is still in the gripper. Appendix J reports in how many of these episodes every cube is still placed 2 s after a scripted release. On the robot the rollout continues, and without a remaining objective the policy keeps the cube near its goal (runs 6, 7 and 10) or lets go only after a long delay (29 s in run 2, about two minutes in run 3). Requiring the release for success during training would remove this failure mode, but we leave this for future

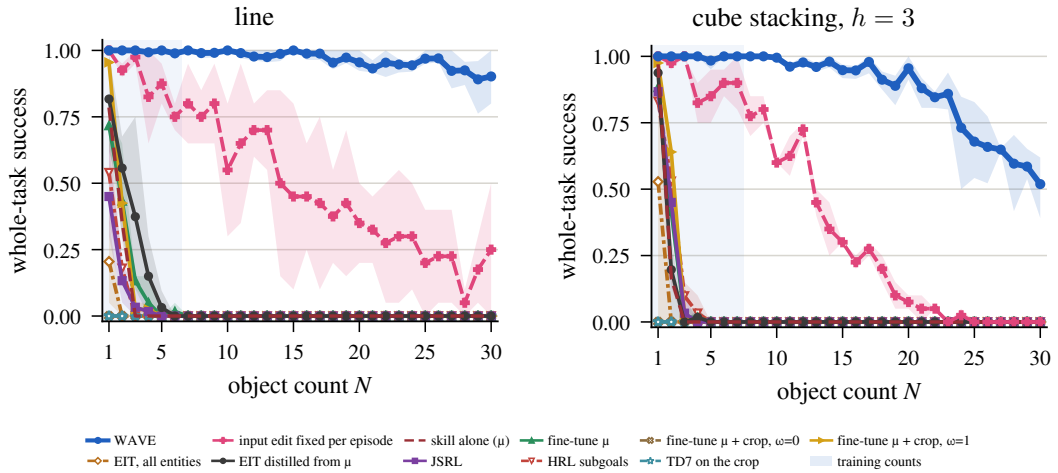


Figure 7: Whole task success for WAVE and all ten baselines of Section 4.3, left on the line and right on cube stacking. The curves are the mean over three seeds and the bands the seed range.

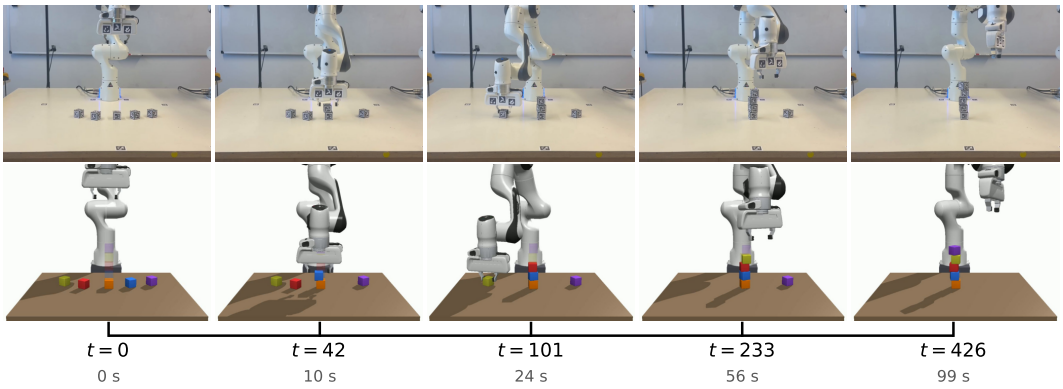


Figure 8: Five-high tower built on the real robot.

work. The two failed runs were stopped at a joint limit with four of five cubes placed (run 5) and at the time limit (run 8). Grasps often needed several attempts. Of the 78 close commands, 68% held a cube, and the policy recovered from failed grasps by trying again. Runs 4 and 9 needed the minimum of three grasps. Runs 6, 7 and 10 needed 9–11. The policy releases a cube a median of 13 mm above its goal slot, a behavior learned in simulation, where the jaws open slowly. Together with gripper stiction on the real system, this was the main reason cubes toppled.

We also deployed a policy fine-tuned for five-cube towers. It built a five-high tower on the real robot (Figure 8), but we did not evaluate this task systematically.

State estimation added a further source of error. We applied no temporal filtering, so tracking jitter, occluded fiducials and camera calibration errors entered the observation of the policy directly. The real gripper also has velocity limits and more stiction than the simulated one. A motion-capture system with temporal filtering and domain randomization of gripper friction and stiction would address both sources of error.

D BASELINES

Every baseline of Section 4.3 runs on the same gantry and task as WAVE, with the same reward unless noted, the same mixed training counts (one to six cubes on the line, one to seven on cube stacking) and 5×10^8 environment steps over 2048 parallel scenes, and every baseline that uses a

Table 4: Whole task success per object count for WAVE and the baselines of Section 4.3 (Figure 7), the mean over three seeds with the minimum and maximum over the seeds in parentheses below it. Every method trains on one to six cubes on the line and one to seven on cube stacking. Bold marks the highest mean per task and count.

N	1	2	3	4	5	6	10	30
line								
WAVE	**1.00** (1.00–1.00)	**1.00** (1.00–1.00)	**1.00** (1.00–1.00)	**0.98** (0.95–1.00)	**1.00** (1.00–1.00)	**1.00** (1.00–1.00)	**1.00** (1.00–1.00)	**0.90** (0.80–1.00)
input edit fixed per episode	**1.00** (1.00–1.00)	0.93 (0.90–0.95)	0.98 (0.95–1.00)	0.83 (0.65–1.00)	0.88 (0.80–0.95)	0.75 (0.70–0.80)	0.55 (0.30–0.80)	0.25 (0.00–0.50)
skill alone (μ)	0.79 (0.75–0.83)	0.35 (0.29–0.43)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)
fine-tuning μ	0.72 (0.65–0.75)	0.43 (0.30–0.65)	0.13 (0.10–0.15)	0.05 (0.00–0.15)	0.00 (0.00–0.00)	0.02 (0.00–0.05)	0.00 (0.00–0.00)	0.00 (0.00–0.00)
on the crop, $\omega = 0$	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)
on the crop, $\omega = 1$	0.96 (0.93–0.98)	0.43 (0.40–0.45)	0.03 (0.00–0.05)	0.03 (0.00–0.05)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)
EIT (Haramati et al., 2024)	0.20 (0.05–0.36)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)
EIT distilled from μ	0.82 (0.62–0.99)	0.56 (0.43–0.68)	0.37 (0.10–0.75)	0.15 (0.05–0.30)	0.03 (0.00–0.10)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)
JSRL (Uchendu et al., 2023)	0.45 (0.30–0.65)	0.13 (0.05–0.20)	0.03 (0.00–0.10)	0.02 (0.00–0.05)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)
HRL subgoals	0.54 (0.35–0.73)	0.18 (0.15–0.20)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)
TD7 on the crop	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)
cube stacking, $h = 3$								
WAVE	**1.00** (1.00–1.00)	**1.00** (1.00–1.00)	**1.00** (1.00–1.00)	**1.00** (1.00–1.00)	**0.95** (0.85–1.00)	**1.00** (1.00–1.00)	**0.98** (0.95–1.00)	**0.48** (0.30–0.60)
input edit fixed per episode	**1.00** (1.00–1.00)	0.98 (0.95–1.00)	**1.00** (1.00–1.00)	0.83 (0.75–0.90)	0.85 (0.75–0.95)	0.90 (0.85–0.95)	0.60 (0.60–0.60)	0.00 (0.00–0.00)
skill alone (μ)	0.97 (0.95–0.99)	0.17 (0.13–0.20)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)
fine-tuning μ	0.88 (0.85–0.95)	0.42 (0.35–0.50)	0.02 (0.00–0.05)	0.02 (0.00–0.05)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)
on the crop, $\omega = 0$	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)
on the crop, $\omega = 1$	0.98 (0.95–1.00)	0.64 (0.61–0.67)	0.03 (0.00–0.05)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)
EIT (Haramati et al., 2024)	0.53 (0.00–0.90)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)
EIT distilled from μ	0.94 (0.85–0.99)	0.20 (0.00–0.32)	0.00 (0.00–0.00)	0.02 (0.00–0.05)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)
JSRL (Uchendu et al., 2023)	0.87 (0.85–0.90)	0.45 (0.35–0.60)	0.03 (0.00–0.10)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)
HRL subgoals	0.83 (0.77–0.91)	0.53 (0.43–0.59)	0.10 (0.05–0.15)	0.03 (0.00–0.10)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)
TD7 on the crop	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)

skill uses the same checkpoint of μ. They are evaluated like WAVE from one to thirty cubes over three seeds.

We supply a scripted selection policy that chooses only objects with goals that are currently fulfillable. For example, in cube stacking only cubes in the bottom layer or on an already placed cube can be selected. The selection then chooses the fulfillable object nearest to the gripper. WAVE does not use a scripted selection policy, so the baselines train on an easier task. All baselines use this selection policy, except for the EIT baselines, HRL subgoals and the fixed input edit. The last two use the learned selection policy of WAVE.

Table 4 compares WAVE with the baselines of Section 4.3 on the line and on cube stacking ($h = 3$), with the mean and the range over all seeds. Table 1 in Section 4.3 shows seven of these baselines.

Table 5: How the episodes of WAVE on the gantry end, per task and object count, with whole-task success, at the time limit, or because a cube left the workspace. The policies and the protocol are those of Table 2 but evaluated in a separate run with unpinned stochasticities, so the success rows agree with that table within its sampling error. For each cause of episode end we give the mean, the minimum and maximum over three seeds. No episode ended with a simulation failure.

N	ends with	stacking, $h = 3$	stacking, $h = 6$	line	square	pyramid	wall
5	success	0.98 (0.97–1.00)	0.99 (0.98–1.00)	1.00 (1.00–1.00)	0.99 (0.97–1.00)	0.99 (0.97–1.00)	0.99 (0.98–1.00)
	time limit	0.01 (0.00–0.02)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.01 (0.00–0.03)	0.01 (0.00–0.02)
	cube lost	0.01 (0.00–0.02)	0.01 (0.00–0.02)	0.00 (0.00–0.00)	0.01 (0.00–0.03)	0.00 (0.00–0.00)	0.00 (0.00–0.00)
10	success	0.94 (0.86–1.00)	1.00 (1.00–1.00)	0.99 (0.98–1.00)	0.98 (0.97–1.00)	0.91 (0.88–0.97)	0.97 (0.95–1.00)
	time limit	0.05 (0.00–0.12)	0.00 (0.00–0.00)	0.01 (0.00–0.02)	0.01 (0.00–0.02)	0.09 (0.03–0.12)	0.02 (0.00–0.05)
	cube lost	0.01 (0.00–0.02)	0.00 (0.00–0.00)	0.01 (0.00–0.02)	0.02 (0.00–0.03)	0.00 (0.00–0.00)	0.01 (0.00–0.02)
15	success	0.90 (0.84–0.92)	0.86 (0.84–0.88)	0.97 (0.94–1.00)	0.95 (0.92–0.98)	0.61 (0.56–0.66)	0.92 (0.83–0.97)
	time limit	0.09 (0.06–0.14)	0.10 (0.08–0.12)	0.02 (0.00–0.03)	0.05 (0.02–0.08)	0.36 (0.33–0.41)	0.07 (0.00–0.17)
	cube lost	0.01 (0.00–0.02)	0.04 (0.03–0.05)	0.01 (0.00–0.03)	0.00 (0.00–0.00)	0.03 (0.02–0.03)	0.02 (0.00–0.03)
20	success	0.81 (0.75–0.86)	0.13 (0.08–0.20)	0.96 (0.92–0.98)	0.88 (0.80–0.92)	0.43 (0.30–0.50)	0.77 (0.56–0.89)
	time limit	0.14 (0.09–0.22)	0.74 (0.69–0.77)	0.03 (0.02–0.05)	0.09 (0.05–0.16)	0.53 (0.45–0.67)	0.20 (0.06–0.41)
	cube lost	0.06 (0.03–0.09)	0.13 (0.11–0.16)	0.01 (0.00–0.03)	0.03 (0.02–0.05)	0.04 (0.03–0.05)	0.03 (0.02–0.05)
25	success	0.66 (0.59–0.77)	0.00 (0.00–0.00)	0.93 (0.92–0.94)	0.77 (0.70–0.81)	0.18 (0.11–0.22)	0.60 (0.12–0.97)
	time limit	0.26 (0.19–0.33)	0.85 (0.83–0.88)	0.04 (0.02–0.06)	0.20 (0.12–0.30)	0.80 (0.77–0.88)	0.40 (0.03–0.88)
	cube lost	0.09 (0.05–0.14)	0.15 (0.12–0.17)	0.03 (0.02–0.05)	0.04 (0.00–0.06)	0.02 (0.02–0.03)	0.01 (0.00–0.02)
30	success	0.45 (0.39–0.48)	0.00 (0.00–0.00)	0.92 (0.88–0.95)	0.66 (0.66–0.67)	0.06 (0.00–0.16)	0.40 (0.08–0.69)
	time limit	0.49 (0.44–0.58)	0.76 (0.72–0.81)	0.05 (0.02–0.09)	0.28 (0.25–0.30)	0.90 (0.81–0.95)	0.56 (0.25–0.89)
	cube lost	0.06 (0.03–0.08)	0.24 (0.19–0.28)	0.03 (0.03–0.03)	0.06 (0.05–0.08)	0.04 (0.03–0.05)	0.04 (0.02–0.06)

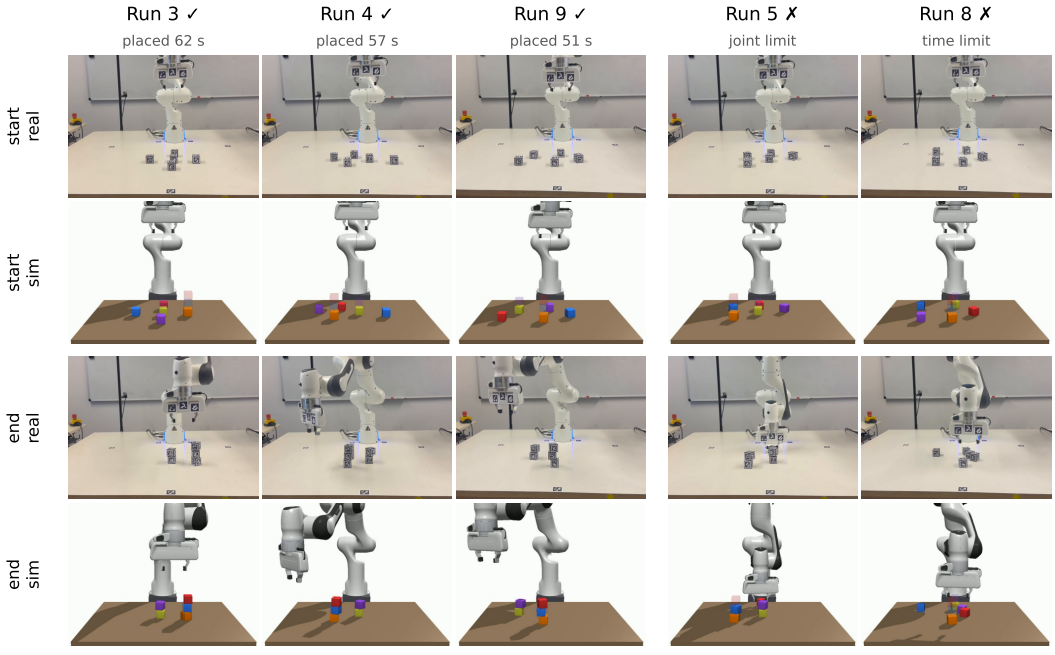


Figure 9: Selected real rollouts of building a three-cube and a two-cube tower (columns) with the same rollouts replayed in simulation from the logged joint states, gripper widths and tracked cube poses. The rows show the start and end on the real robot and in the replay. Runs 3, 4 and 9 succeeded. Run 5 stopped at a joint limit and run 8 at the time limit.

Plain fine-tuning of μ, fine-tuning on the crop without the anchor ($\omega = 0$) and JSRL are reported only here, in Table 4 and Figure 7.

Table 6: Ten consecutive real-robot rollouts of the task of building a three-cube and a two-cube tower from five cubes (two base cubes start in their slots, three have to be moved). *Placed* means all five cubes are within tolerance of their goals. *Released* additionally requires that the last cube was let go and the structure stands at the end of the run. Steps are policy actions at 5 Hz, with the time in seconds in parentheses and means over the successful runs. *Grasps* counts grasps that held a cube, *closes* counts gripper close commands and *cubes* counts cubes within tolerance at the end.

	success		steps to (time [s])					
run	placed	released	placed	released	grasps	closes	cubes	stop
1	✓	✓	373 (82)	472 (102)	5	5	5/5	
2	✓	✓	235 (53)	402 (88)	4	5	5/5	
3	✓	✓	276 (62)	873 (184)	5	5	5/5	
4	✓	✓	254 (57)	259 (58)	3	4	5/5	
5	–	–	–	–	2	8	4/5	joint limit
6	✓	–	300 (66)	–	9	11	5/5	
7	✓	–	844 (194)	–	9	16	5/5	
8	–	–	–	–	2	6	2/5	time limit
9	✓	✓	234 (51)	252 (55)	3	3	5/5	
10	✓	–	764 (175)	–	11	15	4/5	
rate	8/10	5/10						
95% CI	49–94%	24–76%						
mean			410 ± 248	452 ± 254	5.3	7.8	4.5/5	

Skill alone (μ) runs the frozen skill with no training on the selected object. *Fine-tuning* μ keeps the same selection but continues to train μ's own actor and critic on the multi-object reward. During the first 10% of training only the critic is updated and the actor remains equal to μ, so that the critic can adapt to the multi-object reward before the actor changes. *EIT* is the Entity Interaction Transformer of Haramati et al. (2024) with every object and goal as a token, trained with TD3 and hindsight relabeling, with the same hyperparameters as reported in their paper. *EIT distilled from* μ first imitates the actions of μ, whose rollouts also fill its replay buffer. It is then trained like *EIT*, with an added loss that keeps the actor close to μ. In *JSRL* (Uchendu et al., 2023) μ is used as a guide policy that acts the first 90% of every training episode and a TD7 exploration policy initialized from μ's weights acts the rest of the episode. Control is handed over to the exploration policy in stages, each time the success of the combined policy stops improving, until the exploration policy alone acts on the robot at the end of training. *HRL subgoals* is a hierarchy with the frozen μ as its low level, which keeps WAVE's selection policy and its crop χ, and in place of the warp a TD7 manager that reads the crop and proposes goals for μ to pursue. *TD7 on the crop* trains a TD7 policy from random initialization on the same crop χ as WAVE, with the scripted selection and without μ. *Input edit fixed per episode* is WAVE with one modification: the input side of the warp is chosen at the first step of an episode and kept until the episode ends, as the state projection of Zhang et al. (2023) is, while the output side is still chosen at every step. Offsets, crop, selection policy and training budget are those of WAVE.

Fine-tuning μ *on the crop* gives the fine-tuned μ the view of the other objects that plain fine-tuning lacks. The crop χ is the same as for the warping policy (Section 3.4). This crop is passed to the first layer of μ's actor as additional input, through new weights initialized at zero, so that the actor outputs exactly $\mu(\sigma)$ at the start of training. The critic is extended in the same way and reads (σ, χ, a). For the first 10% of training only the critic is trained and the actor stays fixed, so that the critic has learned the multi-object reward before it updates the actor. The actor then minimizes the TD7 loss with an anchor to the frozen skill, as in TD3+BC (Fujimoto & Gu, 2021),

$$\mathcal{L}(\theta) = -\frac{Q(\sigma, \chi, \pi_\theta(\sigma, \chi))}{\overline{|Q|}} + \omega \left\| \pi_\theta(\sigma, \chi) - \mu(\sigma) \right\|^2,$$

where $\overline{|Q|}$ is the mean absolute value of Q over the batch, which makes ω independent of the scale of the reward. With $\omega = 0$ the actor is trained on the reward alone. With $\omega = 1$, staying close to the action of μ is weighted about as much as improving the normalized value. This baseline keeps the scripted selection, and like μ inside WAVE it chooses an action at every control step.

E Overview of published methods and results

Table 7: Training conditions and best reported results of the published methods cited in Section 2 that learn the manipulation itself, in a physics simulator or on a real robot, with a goal pose for every object. All values are taken from their reports. The gray series of Figure 3 are the rows that report whole-task success per count.

Method	*Trains on*	*Task*	*Counts (train → test)*	*Best reported*
Li et al. (2020)	MuJoCo Fetch, privileged state, attention graph network	stacking every block	curriculum 1–6 → 9	0.93 at 4, 0.84 at 5, 0.75 at 6, separate policies per count
ECRL (Haramati et al., 2024)	IsaacGym, two RGB views to latent particles, end effector position deltas on a pusher	planar pushing, 2-D goals	3 → 1–6	0.84 at 3, 0.40 at 6, and a 12 object sort is shown qualitatively only
HECRL (Haramati et al., 2026)	MuJoCo adapted from OGBench, state and image variants	rearrangement, stacking	3 → 1–6 (stacking 2–4)	0.83 at 3 and 0.26 at 6 on rearrangement from state, 0.64 and 0.10 from images
EC-Diffuser (Qi et al., 2025)	IsaacGym, two RGB views to latent particles, behavior cloning	pushing cubes and T blocks	3 → 6	0.24 at 6, 0.83 at 3 by pushing cubes
SMORL (Zadaianchuk et al., 2021)	MuJoCo, RGB to object slots	pushing and rearranging pucks	per count, 1–2 from pixels and 2–4 from state	— (transfer only downward, 2 to 1)
PAIR (Li et al., 2022)	PyBullet, privileged state, discretized actions	six-cube stacking	6 → 6	0.955 at 6, where success counts one target cube among six
Lanier et al. (2019)	MuJoCo Fetch, privileged state, binary or incremental reward	stacking	per count, 2–4	0.98 at 3 and 0.79 at 4 with the incremental reward, 0.95 and 0.00 at 4 with the binary one, all whole-task success
LRN (Mambelli et al., 2022)	CausalWorld TriFinger, privileged state	pushing and lifting one cube among distractors	3 cubes → 1–10 cubes	— (distractors, not goal objects)
OGBench (Park et al., 2025a)	offline datasets, state, UR5e, one policy per count	five goals per count, pick and place, cycles and one stack from two cubes on	per count, 1–4	0.68, 0.40, 0.04 and 0.00 at 1 to 4 over the five goals; on the stack goal 0.23 at 2 and 0.00 at 3 and 4, best of six baselines
SHARSA (Park et al., 2025b)	OGBench cube octuple, state, up to 10^9 transitions, hierarchical	five goals on eight cubes, three rearrangements and two stacks	8 → 8	0.36 at 8 over the five goals with 10^8 transitions and 0.21 with 10^9; 0.00 and 0.01 on the two stacking goals with 10^8, 0.00 on both with 10^9
DQC (Li et al., 2025)	same benchmark and data as SHARSA, flat action chunking	as SHARSA	8 → 8	0.34 at 8 over the goal types, no subtask breakdown
Lampe et al. (2024)	ten real Sawyer arms, three workspace cameras with force and torque sensing	stacking diverse shapes	3 in scene, a two object stack, fixed	0.96 at the fixed count

Tables 7 and 8 list, for every count generalization method cited in Section 2 and every gray series of Figure 3, what it trains on, the counts it trains and tests with, and the best result it reports at its highest evaluated count. A dash marks a quantity the paper does not report in a comparable form.

Every result at a high object count uses privileged state. The only pixel-based method, ECRL, pushes on a plane without gripper actuation, and its successor HECRL, which grasps and stacks, completes 0.10 at six objects from images. Several of the best numbers hold under a criterion less strict than whole-task success, namely one target cube of six for PAIR and the fraction of goals filled for IL-GNN. Methods that learn the manipulation itself, in a physics simulator or on a real robot and with a goal pose for every object, publish whole-task success at up to eight objects. The offline methods on cube octuple reach that count with 0.21 to 0.36 over the five goals and at most 0.01 on the stacking

Table 8: The highest object counts we know of for methods that relax at least one condition of Table 7, with the same columns. All values are taken from their reports.

Method	*Trains on*	*Task*	*Counts (train → test)*	*Best reported*
Janisch et al. (2020)	symbolic block world, relational graph, no physics simulation	goal-conditioned reconfiguration	5 → 2–30	0.78 at 20 and 0.54 at 30
IL-GNN (Lin et al., 2022)	PyBullet and a real Franka, privileged object graph, imitation over a predefined pick-and-place primitive	stacking, rearrangement	3–4 → 9 with data	0.75 averaged over counts up to 40, no per count value above 9, manipulation supplied by the primitive
Blocks Assemble! (Ghasemipour et al., 2022)	MuJoCo, privileged state, virtual grippers that move blocks without grasping	magnetic block blueprints	2–16 → 12–16 held out	1.00 at 6, 0.99 at 12, 0.91 at 16, success up to a rigid motion of the structure
Transporter (Zeng et al., 2020)	PyBullet, RGB-D, scripted two pose pick and place	palletizing 18 boxes into a zone	18 → 18	0.98 at 18 with 1000 demonstrations, scored as the fraction of box volume inside the zone, no per box pose
QT-Opt (Kalashnikov et al., 2018)	real arms, RGB, closed-loop grasping	clearing a bin	4–10 → 28	0.76 over the first 30 grasps of a 28 object bin and 0.96 with replacement, no goal pose
Push Anything (Bui et al., 2026)	real arm, contact implicit MPC, controller not learned	pushing to per-object goal poses	1–4	0.98 at 2 and 0.79 at 4 within 2 cm and 0.1 rad

goals. Every higher count we know of relaxes one of these conditions (Table 8). Blocks Assemble! moves blocks with virtual grippers and scores a blueprint up to a rigid motion. Transporter and IL-GNN execute a scripted primitive between network-chosen poses and score partial credit. Janisch et al. (2020) act in a symbolic block world without physics simulation. Bin clearing policies such as QT-Opt remove objects that have no goal pose. Push Anything reaches four objects with a model-based controller. Li et al. (2020) test up to nine blocks, split over several towers, and show the result as a figure without values, and BuilderBench (Ghugare et al., 2025) reports that its evaluated agents solve two of its twenty-seven hard block structures and that no tabula rasa RL baseline reaches non-zero success.

F Ablations and measured constants

We ablate WAVE one element at a time on three of the gantry tasks: cube stacking ($h = 3$), whose stacks are independent local structures, the square, whose goals lie flat on the table and constrain each other only through space, and the pyramid, whose upper goals cannot be filled before the cubes beneath them are in place. Every ablation keeps the full method and the training budget of Section 4 and changes exactly one part, either the selection rule, the side of μ's interface the warp may edit, or the component of the warp the policy may use. Each ablation is trained from scratch with three seeds, with one to seven cubes on cube stacking, one to nine on the square and one to six on the pyramid.

Table 9 shows the results. The WAVE rows repeat Table 2 for the same task. We summarize each ablation by the mean of its success over the counts.

Table 9: Whole task success per object count for the ablations, on cube stacking ($h = 3$), the square and the pyramid, the mean over three seeds with the minimum and maximum over the seeds in parentheses below it. MEAN is the mean over the counts beyond the training range, up to thirty cubes. WAVE is the three seed column of Table 2 for the same task. Bold marks the highest mean per task and count.

N	3	5	7	10	15	20	25	30	MEAN
cube stacking, $h = 3$, trained on one to seven									
WAVE	**1.00** (1.00–1.00)	0.95 (0.85–1.00)	**1.00** (1.00–1.00)	0.98 (0.95–1.00)	0.92 (0.85–1.00)	**0.95** (0.90–1.00)	0.63 (0.45–0.80)	0.48 (0.30–0.60)	0.82 (0.78–0.86)
nearest-first selection	**1.00** (1.00–1.00)	**1.00** (1.00–1.00)	0.97 (0.95–1.00)	0.97 (0.95–1.00)	**0.95** (0.95–0.95)	0.93 (0.85–1.00)	**0.75** (0.70–0.80)	**0.60** (0.45–0.75)	**0.85** (0.83–0.88)
random selection	0.30 (0.20–0.40)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)
input channels only	**1.00** (1.00–1.00)	**1.00** (1.00–1.00)	**1.00** (1.00–1.00)	**1.00** (1.00–1.00)	0.93 (0.90–0.95)	0.85 (0.85–0.85)	0.70 (0.65–0.75)	0.42 (0.20–0.65)	0.81 (0.77–0.85)
output channels only	**1.00** (1.00–1.00)	0.97 (0.95–1.00)	0.82 (0.80–0.85)	0.55 (0.55–0.55)	0.23 (0.20–0.25)	0.12 (0.10–0.15)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.22 (0.19–0.26)
multipliers only	0.97 (0.95–1.00)	0.97 (0.95–1.00)	**1.00** (1.00–1.00)	0.90 (0.90–0.90)	0.82 (0.75–0.90)	0.78 (0.65–0.90)	0.47 (0.40–0.55)	0.25 (0.15–0.35)	0.65 (0.58–0.71)
offsets only	0.97 (0.95–1.00)	**1.00** (1.00–1.00)	0.93 (0.90–0.95)	0.88 (0.85–0.90)	0.62 (0.55–0.70)	0.60 (0.55–0.65)	0.23 (0.15–0.30)	0.10 (0.05–0.15)	0.52 (0.50–0.54)
square, trained on one to nine									
WAVE	**1.00** (1.00–1.00)	**1.00** (1.00–1.00)	0.98 (0.95–1.00)	**0.97** (0.95–1.00)	0.97 (0.90–1.00)	0.85 (0.65–0.95)	0.77 (0.65–0.85)	0.55 (0.45–0.60)	0.87 (0.77–0.92)
nearest-first selection	**1.00** (1.00–1.00)	**1.00** (1.00–1.00)	**1.00** (1.00–1.00)	**0.97** (0.95–1.00)	**1.00** (1.00–1.00)	**0.93** (0.90–0.95)	**0.93** (0.85–1.00)	**0.95** (0.90–1.00)	**0.97** (0.94–0.99)
random selection	0.80 (0.80–0.80)	0.30 (0.25–0.35)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)
input channels only	**1.00** (1.00–1.00)	**1.00** (1.00–1.00)	0.97 (0.95–1.00)	**0.97** (0.95–1.00)	0.88 (0.85–0.90)	0.75 (0.65–0.85)	0.60 (0.45–0.75)	0.60 (0.60–0.60)	0.76 (0.71–0.80)
output channels only	**1.00** (1.00–1.00)	0.82 (0.65–1.00)	0.68 (0.35–1.00)	0.33 (0.00–0.65)	0.20 (0.00–0.40)	0.05 (0.00–0.10)	0.03 (0.00–0.05)	0.00 (0.00–0.00)	0.14 (0.01–0.28)
multipliers only	**1.00** (1.00–1.00)	**1.00** (1.00–1.00)	0.95 (0.95–0.95)	0.95 (0.90–1.00)	0.95 (0.95–0.95)	0.75 (0.70–0.80)	0.78 (0.65–0.90)	0.60 (0.55–0.65)	0.80 (0.77–0.82)
offsets only	**1.00** (1.00–1.00)	0.97 (0.95–1.00)	0.93 (0.90–0.95)	0.78 (0.70–0.85)	0.72 (0.55–0.90)	0.45 (0.30–0.60)	0.20 (0.10–0.30)	0.17 (0.10–0.25)	0.44 (0.34–0.54)
pyramid, trained on one to six									
WAVE	**1.00** (1.00–1.00)	**1.00** (1.00–1.00)	0.92 (0.85–0.95)	**0.90** (0.80–0.95)	0.60 (0.45–0.70)	0.35 (0.20–0.45)	0.20 (0.15–0.25)	**0.08** (0.05–0.15)	0.47 (0.46–0.49)
nearest-first selection	**1.00** (1.00–1.00)	0.93 (0.90–0.95)	0.50 (0.10–0.90)	0.47 (0.10–0.85)	0.25 (0.00–0.50)	0.15 (0.00–0.30)	0.03 (0.00–0.05)	0.00 (0.00–0.00)	0.19 (0.01–0.37)
random selection	0.93 (0.90–0.95)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)
input channels only	0.97 (0.95–1.00)	0.55 (0.30–0.80)	0.35 (0.15–0.55)	0.17 (0.00–0.35)	0.05 (0.00–0.10)	0.03 (0.00–0.05)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.07 (0.01–0.14)
output channels only	0.95 (0.95–0.95)	0.75 (0.75–0.75)	0.57 (0.50–0.65)	0.42 (0.40–0.45)	0.20 (0.15–0.25)	0.05 (0.00–0.10)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.15 (0.11–0.18)
multipliers only	0.97 (0.95–1.00)	**1.00** (1.00–1.00)	**0.95** (0.90–1.00)	0.88 (0.85–0.90)	**0.72** (0.60–0.85)	**0.53** (0.40–0.65)	**0.33** (0.30–0.35)	0.07 (0.05–0.10)	**0.54** (0.52–0.57)
offsets only	0.80 (0.65–0.95)	0.38 (0.00–0.75)	0.38 (0.00–0.75)	0.15 (0.00–0.30)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.00 (0.00–0.00)	0.05 (0.00–0.11)

F.1 The selection rule. Both the random and the nearest-first rule consider only objects whose goal is currently fulfillable, like the scripted selection of the baselines (Appendix D). The learned selection policy of equation 2 has no such filter. Random selection draws uniformly from these objects. Nearest-first takes the one whose position is closest to the end effector in the plane, an analytic rule that needs no learning and no reward. Figure 10 shows the three tasks.

Random selection has the largest effect in the ablations. Its mean success is 0.00 on all three tasks, and no seed completes an episode from five cubes on cube stacking, nine on the square and five on the pyramid.

Nearest-first is close to the learned selection policy. On cube stacking the two lie within each other's seed range (mean 0.85 against 0.82). On the square nearest-first is ahead (0.97 against 0.87) and still reaches 0.95 at thirty cubes, where WAVE reaches 0.55. On the pyramid, whose goals depend

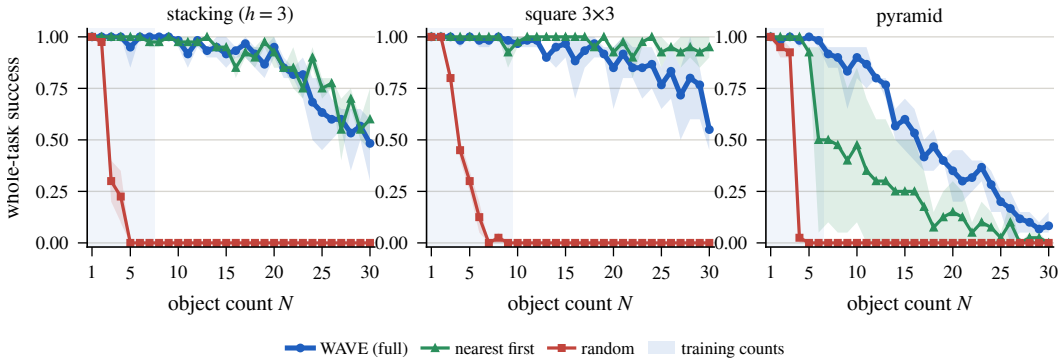


Figure 10: The selection rule, on cube stacking ($h = 3$), the square and the pyramid. Random selection draws uniformly from the objects whose goal is currently fulfillable. Nearest-first takes the fulfillable object closest to the end effector in the plane. The learned selection policy uses no such filter. Shading is the range over the ablation's three seeds, WAVE is its three seed curve, and the tinted band marks the task's training counts.

on each other, WAVE is ahead (0.47 against 0.19), and the mean success of the single nearest-first seeds ranges from 0.01 to 0.37. Nearest-first only considers objects whose goal is fulfillable, while the learned policy has to find a workable order from the reward alone. It comes close to the scripted rule and is more reliable on the pyramid.

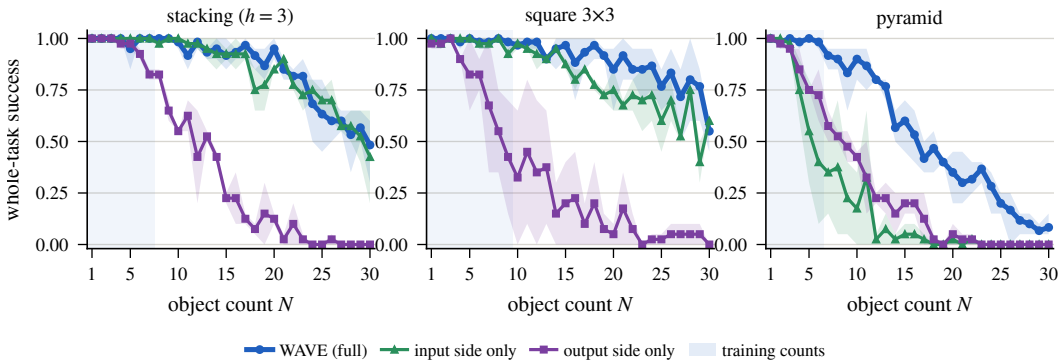


Figure 11: The side of μ's interface the warp may edit. Input channels only edits what μ observes and executes its action unedited (94 parameters). Output channels only edits what μ commands and shows it the scene as the presentation built it (16). Bands as in Figure 10.

F.2 The warp side. The warp emits an affine edit of all 55 channels of μ's interface, 110 parameters in total (Appendix A lists the channels). The input-only ablation restricts it to the 47 channels μ reads, so μ is shown an edited scene but its action is executed unedited (94 parameters). The output-only ablation restricts it to the 8 channels μ commands, so μ sees the scene as the presentation built it and only the command stream is edited (16 parameters). Figure 11 shows the three tasks.

On cube stacking, the input-only warp stays within the seed range of the full warp at nearly every count (mean 0.81 against 0.82). The output-only warp drops from seven cubes (0.82), halves by ten (0.55) and reaches a mean of 0.22. The square shows the same order (0.76 input only, 0.14 output only, 0.87 full warp). On these two tasks, editing the input of the skill is close to sufficient. An output side warp can slow, stop or replace the action of μ, but it cannot change the scene μ is shown, so μ still moves toward whatever stands in its way.

On the pyramid, the input-only warp reaches 0.07, below the output-only warp at 0.15, and both are far below the full warp at 0.47. Here cubes rest on each other, so the release itself has to change. A

cube has to be set down on its support and let go without disturbing it, which is an edit of the output. Neither side is sufficient alone on this task, and only the full warp works on all three tasks.

In Appendix G, switching off either side of the full warp slows WAVE down, even on cube stacking. When both sides are trained together, the learned warp uses both. When only one side is trained, that side alone can often solve the task.

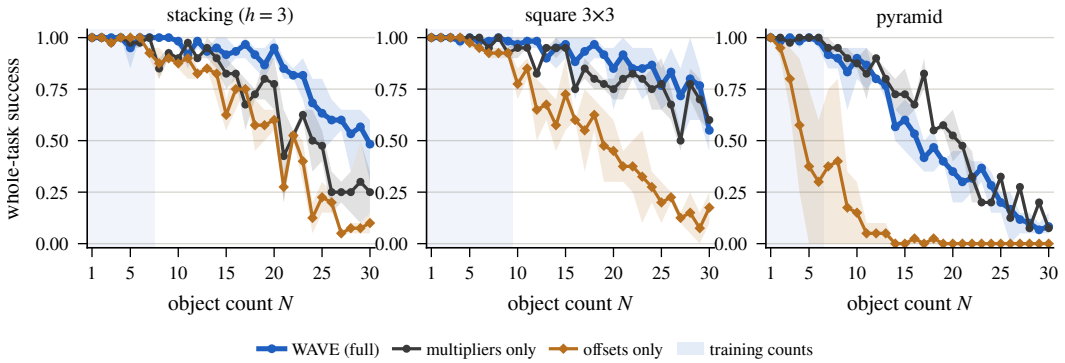


Figure 12: The component of the affine warp the policy may emit. Multipliers only fixes every offset at zero and offsets only fixes every multiplier at one, 55 parameters each against the full 110. Bands as in Figure 10.

F.3 The warp component. The affine warp $z \mapsto m \odot z + o$ has 55 multipliers and 55 offsets. The multipliers-only ablation fixes $o = 0$, and the offsets-only ablation fixes $m = 1$ (Figure 12).

Multipliers only reaches a mean success of 0.65, 0.80 and 0.54 on cube stacking, the square and the pyramid, against 0.82, 0.87 and 0.47 for the full warp. Offsets only reaches 0.52, 0.44 and 0.05. On the pyramid it already fails within the training counts (0.38 at five cubes).

A multiplier can scale or damp a channel, and at $m = 0$ take it over. An offset can shift a channel to any value, but it cannot make μ act more slowly or weakly. On the square and the pyramid the multipliers alone carry the extension. Cube stacking needs both halves.

F.4 Measured progress per attempt. We measure the constants of Theorem 1 per attempt on evaluation episodes of the trained policies. An attempt is the period in which one object stays selected (Section 3.2). p is the fraction of attempts that end with the selected cube placed, λ the mean number of previously placed cubes that an attempt knocks off their goals, and $\lambda_{\max}$ the largest such number. Table 10 reports averages over the evaluation episodes. Since Theorem 1 requires bounds that hold after every history, the table serves as a diagnostic and does not verify the assumptions of the theorem. In at most 0.14% of the attempts, the nearest neighbor of the selected cube is closer than any neighbor seen at the training counts.

Table 10: Averages per attempt on cube stacking ($h = 3$, trained on one to seven cubes) and on the pyramid (trained on one to ten): probability p that the attempt places its cube, number λ of placed cubes it knocks off their goals, net progress $\rho = p - \lambda$, largest number $\lambda_{\max}$ knocked off by a single attempt, and whole-task success of the same episodes. ρ is computed from the unrounded values. The values are averages over episodes and do not verify the assumptions of Theorem 1.

cube stacking, $h = 3$

N	p	λ	ρ	$\lambda_{\max}$	success
3	0.81	0.17	+0.64	2	1.00
7	0.84	0.32	+0.52	4	0.98
10	0.79	0.41	+0.37	7	0.91
20	0.75	0.55	+0.20	6	0.73
30	0.76	0.64	+0.11	9	0.41

pyramid

N	p	λ	ρ	$\lambda_{\max}$	success
3	0.76	0.24	+0.51	2	0.98
7	0.71	0.31	+0.39	4	1.00
10	0.67	0.43	+0.25	6	0.93
20	0.63	0.64	−0.01	10	0.19
30	0.61	0.64	−0.03	12	0.01

G INTERVENTIONAL ESTIMATION OF WARP NECESSITY

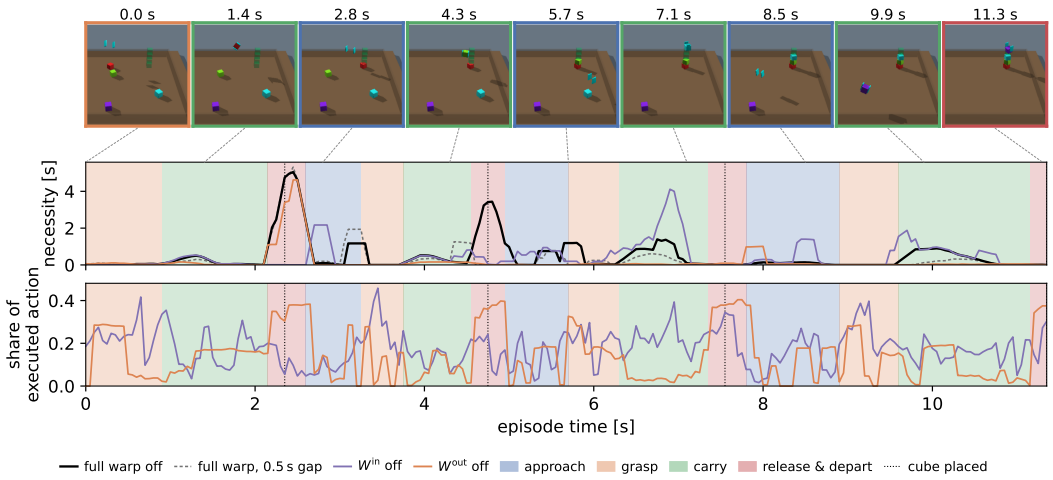


Figure 13: One cube stacking episode with $h = 6$, four cubes in one stack. Top, scene views framed in the color of the manipulation phase. Middle, necessity, the extra time the current phase takes when the full warp (black), only W_{in} (purple) or only W_{out} (orange) is held at the identity, or the full warp pauses for 0.5 s (dashed gray). Bottom, contribution, the share of the executed action that each side accounts for.

To measure when each side of the warp is needed, we split each episode into the phases approach, grasp, carry, release and depart, read from the simulator state. At every step we copy the simulator and continue the copies with one of four changes, six copies for each. Either the full warp, only W_{in} or only W_{out} is held at the identity, or the full warp pauses for 0.5 s. The *necessity* of a change is the extra time the current phase then takes, compared with six unchanged copies. A copy that does not finish the phase within 6 s, or knocks over a placed cube, counts as 6 s.

We also measure the *contribution* of each side, how much it changes the executed action. From the same state we compute the skill action a_μ, the action a_{in} under the input warp only, the action a_{out} under the output warp only, and the executed action a under both. The two-player Shapley value splits $a - a_\mu$ exactly between the sides,

$$\Delta_{\text{in}} = \tfrac{1}{2}\big[(a_{\text{in}} - a_\mu) + (a - a_{\text{out}})\big], \qquad \Delta_{\text{out}} = \tfrac{1}{2}\big[(a_{\text{out}} - a_\mu) + (a - a_{\text{in}})\big],$$

and the contribution of a side is $\|\Delta\|$ relative to the diagonal $2\sqrt{d_a}$ of the action range $[-1, 1]^{d_a}$.

Over 32 held-out episodes with four cubes, a phase takes about a second. Switching W_{out} off costs 1.05 s in release and depart, but only 0.27–0.34 s in the other phases. Switching W_{in} off costs 0.95–1.56 s in every phase.

Figure 13 shows one episode. W_{in} accounts for 20–40% of the executed action throughout, but its necessity stays low during approach and carry, where μ's own feedback absorbs a short gap. At the releases the necessity rises, and it comes almost entirely from W_{out}. μ is a grasp and hold policy and never lets go, so W_{out} overrides its action to release the cube (Section 4.4). The output side thus adds a missing behavior at a few critical moments, and the input side steers μ continuously.

H FORMAL STATEMENTS AND PROOFS

This appendix answers two questions. First, when does an attempt of the warped skill behave like an episode of the skill alone, so that it inherits the skill's success (Lemma 1 and Proposition 1)? Second, given how often an attempt places its object and how many placed objects it knocks off their goals, how many attempts complete a task of N objects (Theorem 1 and Corollary 1)? Neither result predicts how often an attempt of the learned policy succeeds, which we measure in Table 10.

Setting. We recall Section 3.2. The skill μ is trained in the single-object MDP $M_1 = (\mathcal{S}_1, \mathcal{A}, P_1, R_1, \mathcal{G})$. We let P_1 act on a (state, goal) pair by carrying the state and leaving the goal fixed. A *solo episode* is an episode of μ alone in M_1, and $p_1(s_1, g)$ is the probability that a solo episode started at (s_1, g) brings the object to g. $\mathcal{D}_1$ is the set of pairs that the training of μ visited, on which p_1 is high. In the N-object MDP M_N, the presentation $\sigma(s, x^*) \in \mathcal{S}_1 \times \mathcal{G}$ is the scene state s with every object but the selected x^* deleted, together with the goal of x^*. The warp edits every channel z of the presentation and of μ's action as $z \leftarrow mz + o$. An *attempt* is the period in which the selected object stays the same.

Assumption 1 (the skill)**.** μ is deterministic and has no memory, so its action depends only on the pair it is shown.

Assumption 2 (an undisturbed attempt)**.** During the attempt under consideration no other object touches x^* or the gripper, so the robot and x^* move as in M_1: if the command a is executed in the scene state s, the next presentation is distributed as $P_1(\sigma(s, x^*), a)$.

When a warped attempt is an episode of the skill. It is not enough that every frame shown to μ is a valid single-object state. A warp that shows the object 2 cm below its true position shows plausible frames at every step, yet no solo episode produces that sequence, and μ closes the gripper above the surface. The whole sequence shown to μ must be one that μ could have produced on its own.

Definition 1 (solo-equivalent edit)**.** An input edit W_{in} of (state, goal) pairs and an output edit W_{out} of actions are *solo-equivalent* if editing the next state equals stepping from the edited state with the unedited action, $W_{\text{in}} P_1(s_1, W_{\text{out}} a) = P_1(W_{\text{in}} s_1, a)$ for every pair s_1 and action a. This is equation 4 with $P_1' = P_1$.

Lemma 1 (inheritance)**.** *Let Assumptions 1 and 2 hold for an attempt that starts in the scene state s, and let the warp present $W_{\text{in}}\sigma(s_t, x^*)$ to μ and execute W_{out} of μ's action at every step t of the attempt, with one solo-equivalent pair $(W_{\text{in}}, W_{\text{out}})$. Then the pairs shown to μ form a solo episode started at $W_{\text{in}}\sigma(s, x^*)$. In particular the presented object reaches the presented goal with probability $p_1(W_{\text{in}}\sigma(s, x^*))$, whatever the number N of objects in the scene.*

Proof. Write $y_t = W_{\text{in}}\sigma(s_t, x^*)$ for the pair shown at step t. The skill emits $a_t = \mu(y_t)$ and the robot executes $W_{\text{out}} a_t$. By Assumption 2 the next presentation is distributed as $P_1(\sigma(s_t, x^*), W_{\text{out}} a_t)$, so y_{t+1} is distributed as $W_{\text{in}} P_1(\sigma(s_t, x^*), W_{\text{out}} a_t) = P_1(y_t, a_t)$ by Definition 1. This is the transition of a solo episode in which μ acts on y_t, and by Assumption 1 nothing else enters μ's action. □

Proposition 1 (what a per-channel affine edit preserves, bounds and replaces)**.** *Let Assumptions 1 and 2 hold for an attempt, and let the warp keep one edit $z \leftarrow mz + o$ per channel over the attempt.*

- *(a)* Exact edits. *Each of the following is solo-equivalent, so Lemma 1 applies.*
  - *(a1)* Retargeting. *An offset on the presented goal channels and the identity on all others replaces the goal g by $\tilde{g} = g + o$. The attempt is a solo episode toward $\tilde{g}$ and reaches $\tilde{g}$ with probability $p_1(s_1, \tilde{g})$, where s_1 is the presented state at the start of the attempt.*
  - *(a2)* Translation. *Let the robot be one whose dynamics do not change under a translation parallel to the floor, such as the gantry inside its travel limits. Equal offsets on the presented positions of the goal, the object and the gripper translate the presented scene. The object reaches its true goal exactly when the presented object reaches the presented goal.*
  - *(a3)* Recalibration. *Let the deployment differ from M_1 by a per-channel affine recalibration C_{in} of the sensors and C_{out} of the actuators, $P_1'(s_1, a) = C_{\text{in}}^{-1} P_1(C_{\text{in}} s_1, C_{\text{out}} a)$. The constant warp $W_{\text{in}} = C_{\text{in}}$, $W_{\text{out}} = C_{\text{out}}^{-1}$ satisfies equation 4, and the conclusion of Lemma 1 holds with P_1' in place of P_1 in Assumption 2.*
- *(b)* A bound under an assumption. *Assume that constant biases b_{in} on μ's input and b_{out} on its output with norms at most δ lower the solo success by at most $L_{\text{in}}\|b_{\text{in}}\| + L_{\text{out}}\|b_{\text{out}}\|$. Then an attempt under an exact edit of (a) with additional constant offsets o_{in} and o_{out} of norms at most δ reaches the presented goal with probability at least $p_1(W_{\text{in}}\sigma(s, x^*)) - L_{\text{in}}\|o_{\text{in}}\| - L_{\text{out}}\|o_{\text{out}}\|$. This is an error bound under the stated assumption and does not follow from equation 4.*

(c) Take-over. *At $m = 0$ on an output channel the executed value is o, whatever μ outputs. Take-over is control expressivity only and inherits no guarantee from the frozen skill: equation 4 does not hold, and whether the attempt succeeds is learned from the task reward and measured (Table 9, Appendix G).*

For any other edit, a multiplier outside $\{0, 1\}$ that is not a recalibration or a bias larger than δ, we make no statement.

Proof. (a1) P_1 neither reads nor changes the goal, so replacing the goal commutes with P_1, with W_{out} the identity. (a2) By the assumed invariance, translating the next state equals stepping from the translated state with the same action, and the translation moves the object and its goal by the same amount. (a3) Inserting the two maps gives $C_{\text{in}} P_1'(s_1, C_{\text{out}}^{-1} a) = C_{\text{in}} C_{\text{in}}^{-1} P_1(C_{\text{in}} s_1, C_{\text{out}} C_{\text{out}}^{-1} a) = P_1(C_{\text{in}} s_1, a)$, and the proof of Lemma 1 goes through with P_1' as the dynamics of the attempt. (b) By (a) the attempt is a solo episode with the two offsets injected as constant biases, and the assumption bounds the loss of success. (c) With $m = 0$ the action of μ does not enter the executed value on that channel, so the left side of equation 4 does not depend on a there while the right side does. □

Table 11: The edits of Proposition 1, the behavior each one produces, and what follows for the attempt.

edit	channels	behavior	what follows
identity	$m = 1$, $o = 0$ everywhere	μ unchanged	a solo episode
retargeting (a1)	offset on the goal	detour, hover, retreat, hold	a solo episode to the new goal
translation (a2)	equal offsets on all positions	the same task elsewhere in the arena	μ's solo success (gantry)
recalibration (a3)	constant m, o per channel	other sensor or actuator scaling	μ's solo success
small constant bias (b)	offset below δ	slightly shifted grasp or release	a bound, under an assumption
take-over (c)	$m = 0$ on an output channel	release onto a stack, nudging	no statement
any other edit			no statement

Remark 1 (re-decided retargets). If the retarget changes during the attempt, the skill starts a new solo episode toward the new goal from the current state, because it has no memory. At every step μ then acts on a state and a goal of M_1, which keeps its input of the kind it was trained on. The success probability of Lemma 1 refers to a retarget that is held until it is reached.

Remark 2 (the proposition and the learned policy). Proposition 1 describes edits that are held over an attempt. The learned warping policy ψ chooses a new edit at every step and is not restricted to the exact edits, so the proposition characterizes the class of edits and makes no statement about ψ. It motivates the per-channel affine form, which is the smallest class that contains the identity (at which training starts), the exact edits of (a) and take-over. Other classes may work as well or better. Some shifts are outside the class. A rotation of the presented frame about the vertical requires rotating the action back, which mixes channels, and a change of the size, shape or dynamics of the object changes P_1 itself, so that no edit of μ's interface maps it to M_1.

Where the object count can enter. The learned maps never receive N. The warping policy reads the other objects through per-pair scores reduced by an elementwise minimum, so its input at a fixed local geometry does not change when objects are added elsewhere, and the selection policy reads a neighborhood of radius 0.48 m. The count can then affect the two policies only through the neighborhoods that occur in a scene of N objects. Whether these remain within what training covered is an empirical question. Table 10 addresses it per count through the success and the knockdowns of an attempt.

How many attempts complete the task. Fix the count N and the policy π_N. Let $\tau_0 = 0 < \tau_1 < \tau_2 < \dots$ be the control steps at which an attempt ends, because its object is placed or the selection policy picks another object (Section 3.3), so attempt t spans the steps from τ_{t-1} to τ_t. An object is *placed* in a scene state when it is within tolerance of its goal on its intended support. Let $\Phi_t \in \{0, \dots, N\}$ be the number of placed objects at step τ_t, let $\mathcal{F}_t$ be the history of the episode up to that step, and let $T_N = \min\{t : \Phi_t = N\}$ be the number of attempts after which the task is complete. Φ_t rises by one when an attempt places its object and falls when an attempt knocks placed objects off their goals.

Assumption 3 (progress of an attempt)**.** There are constants p_N, λ_N and $\lambda_{\max}$ such that after every history $\mathcal{F}_t$ with $\Phi_t < N$ the next attempt ends, places its object with probability at least p_N, places no other object, unplaces at most λ_N placed objects in expectation, and never unplaces more than $\lambda_{\max}$.

The bounds are conditional on the history. They must hold after every history that can occur, and an average over whole episodes is not enough, since frequent successes early in an episode could hide later attempts with a smaller expected gain.

Theorem 1 (conditional completion bound for a fixed N)**.** *Let Assumption 3 hold with net progress $\rho_N = p_N - \lambda_N > 0$. Then for every $c > 1/\rho_N$ for which cN is an integer,*

$$\Pr[\, T_N > cN \,] \;\le\; \exp\!\left(-\frac{2(\rho_N c - 1)^2 N}{c(1+\lambda_{\max})^2}\right),$$

which is equation 5.

The number of placed objects is a random walk that gains at least ρ_N per attempt in expectation, so filling N places takes about N/ρ_N attempts, and a budget of cN attempts with $c > 1/\rho_N$ fails only if the walk falls far behind its expectation.

Proof. Step 1, one attempt gains ρ_N objects in expectation. Fix t with $\Phi_t < N$. By Assumption 3, $\mathbb{E}[\Phi_{t+1} - \Phi_t \mid \mathcal{F}_t] \ge p_N - \lambda_N = \rho_N$ and $\Phi_{t+1} - \Phi_t \in [-\lambda_{\max}, 1]$.

Step 2, remove the expected gain. Write $t \wedge T_N = \min(t, T_N)$ and $M_t = \Phi_{t\wedge T_N} - \rho_N(t \wedge T_N)$. Before T_N the increment of M_t is $\Phi_{t+1} - \Phi_t - \rho_N$, which is non-negative in expectation by Step 1, and from T_N on it is zero. So M_t is a submartingale, $\mathbb{E}[M_{t+1} \mid \mathcal{F}_t] \ge M_t$, with increments in $[-\lambda_{\max} - \rho_N, 1 - \rho_N]$, an interval of length $1 + \lambda_{\max}$. Azuma's inequality then gives $\Pr[M_n - M_0 \le -a] \le \exp(-2a^2/(n(1+\lambda_{\max})^2))$ for every $a > 0$ and every number of attempts n.

Step 3, failure is such a fall. If $T_N > cN$, then $\Phi_{cN} < N$ and $\Phi_0 \ge 0$, hence $M_{cN} - M_0 = \Phi_{cN} - \Phi_0 - \rho_N cN < -(\rho_N c - 1)N$, which is negative because $c > 1/\rho_N$. Step 2 with $n = cN$ and $a = (\rho_N c - 1)N$ gives the bound. □

Corollary 1 (expected number of attempts)**.** *Under the hypotheses of Theorem 1, $\mathbb{E}[T_N] \le N/\rho_N$. The policy needs at most $1/\rho_N$ attempts per object in expectation.*

Proof. M_t is a submartingale with $M_0 = \Phi_0 \ge 0$, so $\mathbb{E}[\Phi_{t\wedge T_N}] - \rho_N \mathbb{E}[t \wedge T_N] \ge 0$ for every t. Since $\Phi_{t\wedge T_N} \le N$, this gives $\mathbb{E}[t \wedge T_N] \le N/\rho_N$, and letting t grow gives the claim by monotone convergence. □

Remark 3 (scope). The hypotheses are assumptions on the closed loop of π_N at this N, and the theorem does not establish them. Table 10 reports averages over evaluation episodes, which are a diagnostic and do not verify bounds that must hold after every history. The theorem gives no lower bound on ρ_N that is uniform in N, and the measured averages decrease with N. A budget of attempts that is linear in N with one constant for all N would need such a bound, which we do not have.

Remark 4 (how to read the constants). The tail bound is loose, because it charges every attempt with the largest knockdown $\lambda_{\max}$. With the averages measured at seven cubes on cube stacking ($\rho_N = 0.52$, $\lambda_{\max} = 4$) and eight attempts per cube it gives 0.50, while 0.02 of the episodes fail. The expected number of attempts per object, $1/\rho_N$ (Corollary 1), is more informative. The theorem counts attempts, whereas an episode is limited in time. An attempt lasts 2.2 to 2.5 s on average at every count, so the limit of 20 s per cube allows about eight attempts per cube, which requires a net progress of about 0.12 or more.

Relation to value-based bounds. Haramati et al. (2024) bound the error of a scene value that attends over every entity, and this error grows linearly in N. Theorem 1 bounds a different quantity, the number of attempts, and contains no approximated scene value. It depends on N through the constants p_N and λ_N, which we measure but do not bound. The idea that a learned quantity defined on a local neighborhood can transfer across problem sizes has been used before (Ruiz et al., 2020; Qu et al., 2022; Yehudai et al., 2021). In WAVE it motivates the design and is tested empirically.

I POSITIONING OF INTERFACE-EDITING METHODS

Table 12: Interface-editing methods by where they edit a frozen policy, in what form, what the edit is a function of, and which shift it is meant to absorb. Proposition 1 states for which edits of WAVE the behavior of the skill is preserved.

Method	*Edit site*	*Edit form*	*The edit is a function of*	*Shift the edit absorbs*
Residual RL (Silver et al., 2018; Johannink et al., 2019; Yuan et al., 2025)	action	additive (bounded in Yuan et al.)	the current state	same task, refined
Warp RL (Hirschowitz & Ramos, 2026)	action distribution	monotone spline	the current state	dynamics
CLAE (Das et al., 2026)	internal activations	affine	the current state	behavior mode
ZPRL (Yu et al., 2026)	inserted latent	latent perturbation	the context, once per action chunk	same task, refined
DC-Ada (Alqithami, 2026)	observation	residual network found by reward-only random search	the current observation, with parameters fixed per robot	sensor heterogeneity
APB (Park & Lee, 2026)	before and after a backbone	linear	nothing; fixed per task	task (meta-transfer)
Pivoting (Zhang et al., 2023)	observation and action	diagonal scaling	observation: a descriptor of the manipulated object, constant over an episode. Action: that descriptor and the current state	shape and size of the manipulated object
Inter-task mappings (Taylor et al., 2007)	state and action	hand-specified map	nothing; fixed per task pair	task pair, including more agents; the value function is trained further
This work	observation and action	per-channel discrete affine	the other objects around the selected one, at every step	*scene composition (object count)*

Prior work edits the observation of a frozen policy (Alqithami, 2026; Park & Lee, 2026; Zhang et al., 2023), both its input and its output (Zhang et al., 2023; Park & Lee, 2026; Taylor et al., 2007), or its internal activations with an affine edit chosen by an RL policy (Das et al., 2026). These edits adapt a policy to new sensors, a new task, a new object or a new behavior mode (Table 12), and none of them changes the number of objects the fixed policy has to handle. The inter-task mappings of Taylor et al. (2007) transfer between tasks with different numbers of agents, from three against two to five against four players, but with a hand-specified mapping and further training in the target task. WAVE is closest to Zhang et al. (2023), from which it takes the frozen skill, per-channel edits of its input and its action that are trained with a policy gradient, and an action edit that depends on the current state. It differs in four points. (i) Both edits are conditioned on the objects around the selected one, whereas the pivoting setting contains a single object and the edits depend on a descriptor of that object. (ii) The input edit changes during an episode. When we fix it per episode, as their state projection is, the policy still learns its training counts, but the mean success beyond the training range drops from 0.82 to 0.24 on cube stacking and from 0.96 to 0.43 on the line (Section 4.3). (iii) Every channel has an offset in addition to the multiplier, which raises the mean success on cube stacking from 0.65 to 0.82 and makes no difference on the square and the pyramid (Table 9). (iv) The selection policy is learned, which improves on a scripted nearest-first rule on the pyramid (0.47 against 0.19) but not on the other two tasks. The ablations of Zhang et al. (2023) rank the two sides of the interface in the same order as ours. Their pivoting skill transfers in 30 of 40 trials with both projections, in 27 without the action projection, in 19 without the state projection, and in none when the diagonal structure is replaced by free-form neural maps.

The classical transfer literature contains related framings that edit no interface. MDP homomorphisms and inter task transfer (Ravindran & Barto, 2003; Taylor & Stone, 2009) relate two tasks through structural correspondences, options and learned initiation sets (Sutton et al., 1999; Graves et al., 2020) decide when a fixed sub policy may run, and prior policies shape the exploration of a

new task (Uchendu et al., 2023; Schmitt et al., 2018; Galashov et al., 2019; Tirumala et al., 2022). None of them edits what a skill reads or emits, so Table 12 does not list them.

J GENERALIZATION EXPERIMENTS

Section 4 varies the number of cubes and keeps everything else fixed. The eight experiments of this section each change one of these fixed parts. Every experiment has one paragraph, which states in this order the question, how the experiment differs from Section 4, the result, and what the result shows. Unless a paragraph says otherwise, the setting is that of Section 4.1: the gantry, the frozen skill μ trained on a single cube with an edge of 16 cm, whole-task success in the first episode of each scene with deterministic policies, and a time limit of 20 s per object. Only the warping policy and the selection policy are trained. Where a paragraph says that the training is *continued*, the two policies of the first paragraph are trained for a further 4.0×10^8 environment steps on scenes with 1 to 10 objects and evaluated in 64 scenes per count.

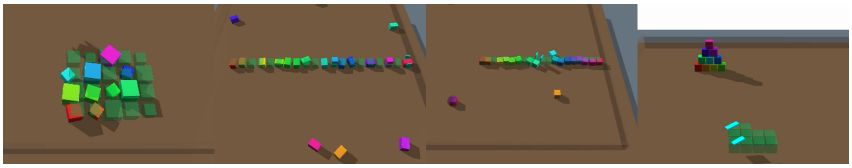


Figure 14: From left to right: cubes of three sizes (square), boxes (line), a line episode in which a disturbance has set a cube to another position (red marker), and a built pyramid at the start of an episode whose goal is the square.

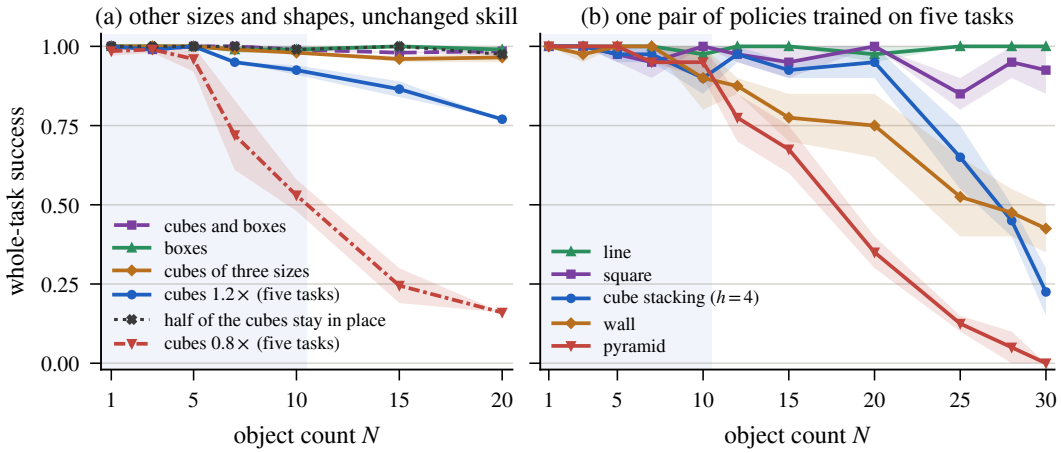


Figure 15: Whole-task success against the object count N. The blue band marks the training counts. (a) Continued training with objects of other sizes and shapes and with cubes that have to stay in place. (b) One warping policy and one selection policy trained on five tasks at once, 20 scenes per count.

One pair of policies for five tasks. Section 4 trains a separate warping policy and selection policy for every task, and we ask whether one pair can learn several tasks. We train one warping policy and one selection policy on the line, the square, the wall, the pyramid and cube stacking ($h = 4$) at once. Every training scene shows one of the five tasks with 1 to 10 cubes, and the training takes 1.57×10^9 environment steps, compared to 5.0×10^8 for one task in Section 4. The inputs of both policies are unchanged, so they can tell the tasks apart only by the goal poses. This pair completes at least 0.90 of the episodes of every task with up to ten cubes. Beyond the training counts it completes 1.00 of the line episodes and 0.93 of the square episodes at thirty cubes and 0.95 of the stacking episodes at twenty, while the wall and the pyramid fall to 0.75 and 0.35 at twenty (Figure 15b, Table 13). Training one pair of policies per task is therefore not a requirement of WAVE. The numbers do not

show whether training on five tasks is better or worse than training on one, because the training counts, the stack height and the workspace, which is larger for the square, the wall and the pyramid, differ from those of Table 2.

Table 13: Whole-task success in five of the generalization experiments. The tasks of the training are in parentheses in the row labels. Disturbances: policies of Table 2 evaluated with a time limit of 40 s per cube. All values are the mean of three seeds.

objects of other sizes and shapes	$N = 5$	$N = 10$	$N = 15$	$N = 20$
cubes with 1.2 times the edge (five tasks)	1.00	0.93	0.87	0.77
cubes with 0.8 times the edge (five tasks)	0.96	0.53	0.25	0.16
boxes of $16 \times 24 \times 16$ cm (line, square)	1.00	0.99	1.00	0.99
cubes of 0.8, 1.0 and 1.2 times the edge in one scene (line, square)	1.00	0.98	0.96	0.97
cubes and boxes in one scene (line, square)	1.00	0.99	0.98	0.99
cubes that have to stay in place	$N = 5$	$N = 10$	$N = 15$	$N = 20$
half of the cubes have their start pose as the goal (line, square)	1.00	0.99	1.00	0.98
starting from a built structure	$N = 3$	$N = 6$	$N = 10$	$N = 15$
pyramid to square	0.98	0.87	0.59	0.30
stacks of height three to line	0.98	0.95	0.90	0.90
pyramid to square, failure when a cube falls	0.92	0.42	0.10	0.00
stacks of height three to line, failure when a cube falls	0.82	0.64	0.56	0.40

	$N = 10$			$N = 20$		
one disturbance every	never	20 s	10 s	never	20 s	10 s
cube stacking ($h = 3$)	0.95	0.99	0.98	0.91	0.93	0.90
line	0.99	0.98	0.96	0.98	0.97	0.88
square	0.99	0.97	0.97	0.95	0.96	0.90
pyramid	0.94	0.99	0.98	0.59	0.69	0.52
one pair of policies for five tasks	$N = 5$	$N = 10$	$N = 15$	$N = 20$	$N = 25$	$N = 30$
line	1.00	0.98	1.00	0.98	1.00	1.00
square	0.98	1.00	0.95	1.00	0.85	0.93
cube stacking ($h = 4$)	0.98	0.90	0.93	0.95	0.65	0.23
wall	1.00	0.90	0.78	0.75	0.53	0.43
pyramid	1.00	0.95	0.68	0.35	0.13	0.00

Cubes of another size. The skill was trained on a cube with an edge of 16 cm, and neither the skill nor the two policies receive the dimensions of an object, so we ask whether WAVE can use this skill for cubes of another size. The training is continued on the five tasks with every cube scaled to 1.2 times the edge (19.2 cm) or to 0.8 times the edge (12.8 cm). The distance between neighboring goals scales with the cubes, while the gripper and the position tolerance keep their size. With the larger cubes, whole-task success over the five tasks is 0.77 / 0.77 at twenty cubes, twice the largest training count (Figure 15a). The line and the square stay at 0.94 or above there, and the wall (0.67 / 0.72) and the pyramid (0.48 / 0.36) account for the loss. With the smaller cubes, success is 0.92 / 1.00 at five cubes, 0.48 / 0.58 at ten and 0.16 / 0.16 at twenty. The warp thus adapts the skill to a larger cube but not to the smaller one. Because the decrease with the smaller cubes begins within the training counts, it is not a failure to generalize over the count. The failed episodes end at the time limit, with about 100 placed-to-unplaced transitions per episode at twenty cubes. We attribute this to the unchanged gripper, which has to work between goals that lie closer together, but we did not isolate the cause.

Boxes, and objects that differ within a scene. We ask whether the skill for one cube is also sufficient for an object of another shape and for scenes whose objects are not all the same. The training is continued on the line and the square with three sets of objects: boxes of $16 \times 24 \times 16$ cm, cubes with 0.8, 1.0 and 1.2 times the edge in one scene, and cubes and boxes in one scene. All three sets reach a whole-task success of at least 0.95 at twenty objects, twice the largest training count (Figure 15a, Table 13). A skill trained on one cube is therefore sufficient for boxes that are 1.5 times as long as the cube and for scenes with objects of different sizes and shapes. The scenes with three sizes also contain the small cubes of the previous paragraph, so the skill can place a small cube, and

the failure there concerns scenes that consist only of small cubes. Two limits remain. On the line and the square every object lies on the floor, so these results do not cover stacking such objects. Objects of other classes, for example cylinders, were not tested, and a class that the skill cannot manipulate alone would need its own skill.

Cubes that have to stay in place. In Section 4 every cube of a scene has a goal in the structure. We ask whether WAVE can build a structure between cubes that are not part of it. On the line and the square, half of the N cubes have their own start pose as the goal, and they start next to the goal positions of the structure. The training is continued on such scenes. Whole-task success is defined as in Section 4, so it requires each of these cubes to be within the tolerance of its start pose when the episode ends, but it does not forbid contact with them during the episode. Success is 1.00 / 0.98 at ten cubes and 0.98 / 0.97 at twenty (Table 13). The policies thus complete the structure while the other cubes end at their start poses, also at twice the largest training count. They do not avoid every contact: at twenty cubes a placed cube is knocked off its goal and placed again about 33 times per episode, counted over both kinds of cubes.

Disturbances during the episode. In Section 4 only the robot changes the scene. We ask whether the policies also complete a task when the scene is changed from outside while they act. We evaluate the policies of Table 2 without retraining. Every 20 s or every 10 s, a disturbance sets one cube instantly to a free position on the floor. This cube is at rest, is not in the gripper and carries no other cube, and it is drawn with equal probability from the placed cubes or from the cubes not yet placed. Since every disturbance adds work, all episodes of this experiment, also those without disturbances, have a time limit of 40 s per cube. Whole-task success is at most 0.10 lower than without disturbances (Table 13), also on the line with twenty cubes and one disturbance every 10 s (0.88 against 0.98), where about 45 disturbances occur per episode. The policies handle a disturbance because the warp is computed from the current crop at every control step and the selection policy decides again whenever a cube is placed or knocked off its goal (Section 3.2), so a cube set to another position is treated like any cube that is not at its goal. The cost is time: with the 20 s per cube of Section 4, success on the line with twenty cubes is 0.98 without disturbances, 0.84 with one every 20 s and 0.47 with one every 10 s. We did not test objects that move continuously on their own.

Starting from a built structure. In Section 4 no episode starts from a built structure. We ask whether WAVE can turn one structure into another, which requires taking cubes out of a structure. All N cubes start as a built pyramid with the square as the goal, or as stacks of height three with the line as the goal, and the training is continued on each of these two tasks. The policies reach the goal structure, in 0.84 / 0.89 of the episodes from a pyramid of six cubes and in 0.88 / 0.92 from stacks with ten cubes (Table 13), but they do so by knocking the start structure down and collecting the cubes from the floor. To test whether a structure can be taken apart cube by cube, we train for a further 8.0×10^8 steps with one added condition: an episode ends as a failure when a cube that the gripper has not yet held drops more than 4 cm below its resting height in the start structure. Under this condition a pyramid is turned into a square in 0.86 / 0.98 of the episodes with three cubes, 0.34 / 0.50 with six and 0.02 / 0.17 with ten, and stacks into a line in 0.77 / 0.34 with ten cubes. WAVE thus reaches a new structure from a built one, but it learns to take a structure apart without a falling cube reliably only for three cubes. The skill was trained to pick up a cube that stands free, which likely explains this limit, but we did not test this explanation.

Success that lasts, and success after a release. In Section 4 an episode ends as a success at the first step at which every cube is placed, so cubes can still be in motion and the last cube can still be in the gripper. We ask whether the finished structure lasts. We evaluate the policies of Table 2 without retraining under two stricter criteria (three seeds, 64 episodes per seed and count). Under the first, the policies keep acting after the first success, and an episode counts as a success only when all cubes have stayed placed for 2 s without interruption. Its time limit is 50 s per cube, because a structure that does not last has to be completed again. Under the second, with the time limit of Section 4, a scripted routine opens and lifts the gripper as soon as all cubes are placed and slower than 0.05 m/s, and we check 2 s later whether every cube is still placed. On cube stacking ($h = 3$), success is held for 2 s in 0.99 of the episodes at seven cubes, 0.92 at twenty and 0.59 at thirty, compared to 0.99, 0.94 and 0.65 for the first success within the same time limit (Table 14). After the release, all cubes are placed in 0.93, 0.73 and 0.38 of the episodes at these counts, which is 0.95, 0.90 and 0.85 of

the episodes that reach whole-task success. Although the policies were trained for neither criterion, the whole-task success of Section 4 thus mostly corresponds to structures that last. The exceptions are stacks of height six, which remain after the release in 0.82 of the successful episodes at seven cubes and in 0.48 at fifteen, and the wall, where success is held for 2 s in 0.82 of the episodes at five cubes compared to 0.98 for the first success. The release loses about 0.1 already with a single cube, so part of the loss after the release is caused by the scripted routine and not by the structure.

Table 14: The policies of Table 2 under two stricter success criteria, mean of three seeds with 64 episodes each. Top: whole-task success / fraction of episodes in which every cube is still placed 2 s after a scripted release, with a time limit of 20 s per cube (40 s for stacks of height six). Bottom: first success / success held for 2 s without interruption, both with a time limit of 50 s per cube. Stacks of height six were not evaluated at 25 and 30 cubes.

whole-task success / placed after release	$N = 1$	$N = 7$	$N = 15$	$N = 20$	$N = 30$
cube stacking ($h = 3$)	1.00 / 0.89	0.97 / 0.93	0.90 / 0.85	0.81 / 0.73	0.45 / 0.38
cube stacking ($h = 6$)	0.99 / 0.90	0.99 / 0.81	0.86 / 0.41	0.13 / 0.02	0.00 / 0.00
line	1.00 / 0.89	1.00 / 0.88	0.97 / 0.88	0.96 / 0.86	0.92 / 0.83
square	0.99 / 0.83	0.98 / 0.84	0.95 / 0.79	0.88 / 0.66	0.66 / 0.52
pyramid	0.99 / 0.87	0.91 / 0.78	0.61 / 0.53	0.43 / 0.41	0.06 / 0.05
wall	1.00 / 0.88	0.98 / 0.82	0.92 / 0.79	0.77 / 0.66	0.40 / 0.36

first success / held for 2 s	$N = 5$	$N = 10$	$N = 15$	$N = 20$	$N = 25$	$N = 30$
cube stacking ($h = 3$)	0.98 / 0.98	0.99 / 0.99	0.93 / 0.93	0.94 / 0.92	0.82 / 0.76	0.65 / 0.59
cube stacking ($h = 6$)	0.99 / 0.99	0.97 / 0.97	0.88 / 0.68	0.17 / 0.05	–	–
line	1.00 / 0.98	1.00 / 0.98	0.98 / 0.98	1.00 / 0.99	0.98 / 0.98	0.97 / 0.97
square	1.00 / 0.97	0.99 / 0.99	0.98 / 0.95	0.98 / 0.96	0.91 / 0.86	0.92 / 0.89
pyramid	0.98 / 0.90	0.93 / 0.87	0.78 / 0.72	0.69 / 0.60	0.44 / 0.35	0.19 / 0.15
wall	0.98 / 0.82	0.98 / 0.82	0.97 / 0.83	0.91 / 0.77	0.87 / 0.69	0.61 / 0.53

Minimum, mean or attention over the other objects. The warping policy reads the other objects through per-pair scores, which an elementwise minimum reduces to one encoding, and Section 3.4 motivates the minimum by its independence of the count. We ask whether the count generalization of Section 4 depends on the minimum. We retrain WAVE on cube stacking ($h = 3$) and on the pyramid as in Section 4, and replace only the minimum. The first replacement is the mean of the scores over the other objects. The second is an attention-weighted mean, in which a second network maps each pair to one logit per score, a softmax over the other objects turns the logits into weights, and the scores are averaged with these weights. On cube stacking, the mean success beyond the training counts is 0.55 with the mean and 0.64 with the attention, compared to 0.82 with the minimum (Table 15). On the pyramid it is 0.47 with the mean, 0.66 with the attention and 0.47 with the minimum. The minimum is thus the best of the three on cube stacking but not on the pyramid, so count generalization does not depend on the minimum, and we do not claim that the minimum is the best choice for every task. All three variants score each of the other objects separately and give the warping policy an input whose size does not depend on N. We did not compare against self-attention among the other objects or against inputs whose size grows with N.

Table 15: Whole-task success when the elementwise minimum over the other objects is replaced by a mean or by an attention-weighted mean. The minimum rows are the policies of Table 2, all rows use three seeds. MEAN is the mean over the counts beyond the training counts.

	$N = 7$	$N = 10$	$N = 15$	$N = 20$	$N = 25$	$N = 30$	MEAN
cube stacking ($h = 3$), trained on one to seven cubes							
minimum (WAVE)	1.00	0.98	0.92	0.95	0.63	0.48	0.82
mean	0.95	0.88	0.80	0.35	0.40	0.05	0.55
attention	1.00	0.90	0.90	0.78	0.30	0.08	0.64
pyramid, trained on one to six cubes							
minimum (WAVE)	0.92	0.90	0.60	0.35	0.20	0.08	0.47
mean	1.00	0.88	0.55	0.38	0.13	0.05	0.47
attention	1.00	0.93	0.83	0.63	0.43	0.43	0.66