FAULT-AWARE POLICY REUSE VIA SKILL LIBRARIES FOR ADAPTIVE ROBOT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances in embodied agent adaptation have largely focused on changes in the task or environmental dynamics. However, a robot's own dynamics can also change mid-deployment due to a hardware failure. Consequently, policies trained under nominal dynamics can degrade significantly, even when the task/environment remains unchanged. Existing fault-tolerant approaches either require fault-specific engineering or training a single monolithic policy to generalize across diverse changes in robot dynamics. Policy reuse offers a more targeted alternative, but existing methods are not explicitly organized around the robot's fault state for subsequent adaptation. We propose *Skill Library*, a framework that organizes reusable policies by fault condition. We first pretrain a collection of lightweight, fault-indexed policies offline. During deployment, when the detected fault is represented in the library, the matched policy is retrieved for direct reuse; when the fault is unseen, stored policies serve as initialization for online adaptation. Our fault-aware *Skill Library* focuses on changes in the robot's dynamics with policies indexed by the affected joint and fault severity. We evaluate our framework across four manipulation environments on a 7-DOF Franka Panda in MuJoCo. Our results demonstrate that our framework consistently improves adaptation sample efficiency over both training from scratch and fine-tuning from the nominal policy, reducing the recovery interaction budget and improving final success rates. Moreover, experiments on unseen compound faults show that a library of single-joint policies alone can provide useful initialization for complex, multi-joint failures, extending the fault adaptation beyond the conditions explicitly represented in the offline library.

1 INTRODUCTION

Recent advances in robot learning have been centered around enabling robots to learn model-free control policies directly from sensory input while being able to efficiently adapt to new conditions. Robotic systems, particularly manipulators, are typically exposed to a range of failure modes during real-world operations (Milecki & Nowak, 2023). For example, a joint actuator may fail completely or lose part of its range of motion due to wear or overheating. When that happens, a policy trained under nominal dynamics, namely the *healthy policy*, can drop from near-perfect performance to near-zero within the same task. Developing data-efficient, adaptive learning is crucial for the deployment of robots in diverse real-world environments, such as household (Li & Ding, 2023) and industry (Nievas et al., 2024) settings. Adaptation in embodied agents can arise from different sources of change, most notably changes in the *task* the robot must solve, or in the robot's own *body*. Existing work has largely focused on task-level adaptation, including policy reuse across task variations (Guo et al., 2025) and generalist foundation-model policies capable of solving diverse tasks (Goyal et al., 2024; Parisi et al., 2022; Nair et al., 2022; Shridhar et al., 2022; Jang et al., 2022). In contrast, adapting to changes in a robot's physical capabilities, such as a faulty actuator or restricted joint motion, has received comparatively less attention (Schoepp, 2024; Pham et al., 2024).

When physical repair is infeasible or costly, e.g., a robot on the Moon cannot be serviced, the robot must instead compensate for the fault through changes in its control policy. Prior approaches to fault adaptation follow several strategies. Classical fault-tolerant control detects a fault and reconfigures the controller accordingly, often using hardware redundancy or analytical models (Milecki

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

& Nowak, 2023). While effective, such approaches typically require fault-specific engineering and knowledge of the robot dynamics. Another strategy is to learn a single policy that is robust to a distribution of faults before deployment, using domain or dynamics randomization (Schoepp, 2024; Peng et al., 2018) or learned context representations that enable online adaptation (Kumar et al., 2021; Rakelly et al., 2019; Clavera et al., 2019). Rather than specializing to individual conditions, these methods ask one policy to generalize across a potentially broad fault space, which may limit specialization to specific fault states and leave performance dependent on how well deployment-time faults are represented by the training distribution.

A different direction is policy reuse, which is particularly appealing for fault adaptation and echoes how humans and animals preserve task goals while reorganizing existing motor skills after injury (Hu et al., 2025; Todorov & Jordan, 2002). In this family of methods, a collection of policies is constructed offline, and an appropriate one is selected and reused/adapted when a fault is detected (Cully et al., 2015; Kaushik et al., 2020; Fernández & Veloso, 2006; Li et al., 2019; Guo et al., 2025). This formulation is well suited to settings where the task remains fixed but the robot must adapt to the fault. However, existing approaches do not directly combine fault-aware policy retrieval with subsequent policy optimization. Repertoire-based methods such as IT&E and APROL (Cully et al., 2015; Kaushik et al., 2020) primarily adapt by selecting among predefined behaviors rather than updating a retrieved policy through RL. In contrast, SRSA (Guo et al., 2025) retrieves a policy and fine-tunes it with RL, but its retrieval space is defined by variation across assembly tasks under a fixed robot embodiment. Fault adaptation presents a different setting: the task remains unchanged while the robot's dynamics vary with the physical fault. This makes the robot's fault state a natural basis for organizing reusable policies, allowing policies to be retrieved according to the specific physical condition of the robot and subsequently adapted when needed.

Building on this perspective, we ask: how can a finite set of fault-specific policies be organized and reused to support efficient adaptation across *seen* and *unseen* faults? To this end, we propose *Skill Library* (Figure 2), in which policies are pre-trained offline for a discrete set of single-joint fault conditions and indexed by the robot's fault state, specifically the affected joint and fault severity. At deployment, an upstream fault-diagnosis module identifies the active fault and uses this fault descriptor to access the library. For a fault represented in the library, the corresponding fault-specific policy can be retrieved and either deployed directly or further adapted with RL. When the encountered fault is unseen, the library instead provides prior policies that can serve as starting points for adaptation, avoiding the need to learn a new policy entirely from scratch. The library therefore provides targeted fault-specific experience without requiring exhaustive coverage of the fault space, which would be impractical given continuous variations in fault severity and the combinatorial growth of multi-joint fault configurations.

We evaluate our method across four manipulation tasks on a Franka Panda 7-DOF arm in MuJoCo (Todorov et al., 2012), selected to span qualitatively different fault-adaptation regimes. Our experiments examine whether initialization from a fault-matched library policy improves adaptation efficiency and final task performance relative to healthy-policy transfer and training from scratch. Across the evaluated tasks, fault-matched initialization consistently improves recovery for critical faults, with gains of up to 91.70 percentage points in success rate at the same interaction budget. We further find that policies trained only on single-joint faults can provide useful initializations for previously unseen compound faults, showing that the library can support adaptation beyond the fault conditions explicitly represented during offline training.

Our contributions are:

- We introduce *Skill Library*, a framework for sample-efficient robot adaptation, in which policies are trained offline for representative fault conditions, indexed by fault state, and retrieved for direct reuse or RL-based fine-tuning after a fault.
- We conduct a systematic study of joint-fault criticality across four manipulation tasks, characterizing how the effect of a fault depends on both the affected joint and the task, and identifying fault conditions for which policy adaptation is particularly challenging.
- We demonstrate the effectiveness of the proposed library across seen single-joint faults and unseen compound faults, comparing fault-specific initialization against healthy-policy transfer and training from scratch.

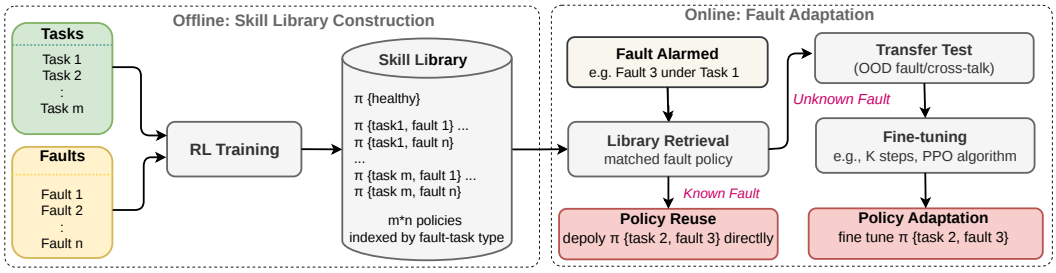

Figure 1: Fault-aware skill library pipeline. *Left (offline)*: one policy is trained per fault-task condition and stored in the library. *Right (online)*: the matching library policy is retrieved and either deployed directly (known fault) or fine-tuned before deployment (unseen fault).

2 Related Work

Fault-tolerant robot adaptation. Classical approaches use hardware redundancy or analytical fault detection and isolation (Milecki & Nowak, 2023), which requires new reconfiguration logic for each fault type and each robot. RL-based approaches avoid this by training a single policy to be robust across fault conditions, using domain randomization over fault conditions (Schoepp, 2024), dynamics randomization for sim-to-real transfer (Peng et al., 2018), or a POMDP formulation trained end-to-end on a Franka arm (Pham et al., 2024). RMA (Kumar et al., 2021) and PEARL (Rakelly et al., 2019) adapt to dynamics changes via learned context encoders trained on privileged information, while Clavera et al. (2019) meta-learn a dynamics model that adapts online within a handful of gradient steps instead of retraining or retrieving a policy. All of these rely on one policy or model that must generalize, or adapt quickly, across the fault space. Repertoire-based methods offer intelligent trial-and-error, pre-computing a behavior–performance map offline and selecting a compensatory behavior online via Bayesian optimization (Cully et al., 2015); APROL (Kaushik et al., 2020) extends this to multiple repertoires. Our approach differs from both families by explicitly constructing and retrieving from a fault-indexed policy library, rather than learning a single robust policy, and by fine-tuning the retrieved policy via RL rather than selecting among fixed behaviors, requiring no privileged information at deployment beyond fault identity.

Policy reuse and skill libraries. Policy reuse (Fernández & Veloso, 2006) maintains a library of past policies and uses it to bias learning on new tasks. Reuse of context-aware policies (Li et al., 2019) extends this with the learned context to select the most relevant source policy at each step, the closest prior work to our retrieval setting. More recently, REGENT (Sridhar et al., 2025) and STRAP (Memmel et al., 2025) retrieve relevant context or sub-trajectories from prior experience to act in new environments in-context, without fine-tuning.SRSA (Guo et al., 2025), the most recent skill library and reuse method, applies retrieval and fine-tuning to robotic assembly under task variation. It maintains a library $\Pi = \{\pi_1, \ldots, \pi_K\}$ and trains a transfer success predictor from geometry, dynamics, and action features; the best-predicted policy is then retrieved and fine-tuned on the target task. Skill libraries have been studied mainly for task-level adaptation, whereas our method leverages this paradigm for fault variation, where the robot's own dynamics change rather than the task.

3 Preliminaries

We formulate the robot learning task as an episodic Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma, \mu_0)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and continuous action spaces, $P(s_{t+1} \mid s_t, a_t)$ is the transition dynamics, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\gamma \in [0, 1)$ is the discount factor, and μ_0 is the initial-state distribution. At each timestep t, the agent observes $s_t \in \mathcal{S}$, samples an action $a_t \sim \pi_\theta(\cdot \mid s_t)$ from a policy π_θ parameterized by θ, receives reward $r_t = r(s_t, a_t)$, and transitions according to $s_{t+1} \sim P(\cdot \mid s_t, a_t)$. A trajectory is denoted by $\tau = (s_0, a_0, s_1, a_1, \ldots, s_H)$, where H denotes the finite episode horizon. The goal of RL is to learn a policy that maximizes the expected discounted return $J(\pi_\theta) = \mathbb{E}_{\tau \sim (\mu_0, \pi_\theta, P)}[\sum_{t=0}^{H-1} \gamma^t r_t]$.

We optimize π_θ with Proximal Policy Optimization (PPO) (Schulman et al., 2017), an on-policy actor–critic method. Each iteration collects a batch of transitions with the current policy $\pi_{\theta_{\text{old}}}$. It then runs several epochs of minibatch gradient ascent on the clipped surrogate objective

$$\mathcal{L}^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t\Big[\min\big(\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t,\ \text{clip}\big(\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}, 1-\epsilon, 1+\epsilon\big)\hat{A}_t\big)\Big], \tag{1}$$

where $\hat{\mathbb{E}}_t$ denotes the empirical average over the batch, $\hat{A}_t$ is an estimate of the advantage function, and ϵ is the clipping range. PPO has been widely used as a stable algorithm for robot learning (Ma et al., 2024; Yang et al., 2022). This formulation assumes that the transition dynamics P remains unchanged during training and deployment; we illustrate how a fault violates this assumption. A robot fault can alter the effective transition dynamics, causing a policy trained under nominal dynamics become suboptimal.

4 Fault-Aware Skill Library

4.1 Problem Formulation

A fault alters the transition dynamics P, so a policy π_θ trained to maximize $J(\pi_\theta)$ under the nominal (healthy) dynamics becomes suboptimal once a fault occurs. Retraining from scratch recovers performance but requires substantial interaction; our goal is instead to recover performance with as little additional interaction as possible by reusing previously trained policies.

We simulate joint faults by varying severity and location. For the 7-DOF manipulator, location is the affected joint $j \in \{0, \ldots, 6\}$; severity s is half (range and stiffness reduced to 50%) or full (joint frozen at its nominal center), giving 14 single-joint fault conditions plus one healthy condition. Writing $z = (j, s)$ for a fault descriptor ($z = \emptyset$ for healthy) and indexing tasks by $k = 1, \ldots, K$, we denote the policy trained for task k under fault z by $\pi_\theta^{k,z}$.

We formulate the problem as a library built over prior tasks and fault descriptors, each associated with a stored policy $\pi_\theta^{k,z}$. When a new fault z_{new} is detected on task k, we classify it as *seen* if it matches a descriptor already covered during offline training for that task (Section 4.3), and *unseen* otherwise (e.g. a fault combination, or a fault transferred from a different task); our framework handles the two cases differently, as described next in Section 4.2.

4.2 Skill Library

Our fault-aware skill library operates in two phases, illustrated in Figure 2.

4.3 Phase 1: Offline Library Acquisition

We consider a set of tasks $\mathcal{K} = \{1, 2, 3, 4\}$ and the set of fault descriptors $\mathcal{Z} = \{(j, s) : j \in \{0, \ldots, 6\},\ s \in \{\text{half}, \text{full}\}\} \cup \{\emptyset\}$ from Section 4.1; enumerating its 15 elements gives $z \in \{1, \ldots, 15\}$, with $z = 1$ denoting $\emptyset$ (Panel A of Figure 2). For every task-fault pair $(k, z) \in \mathcal{K} \times \mathcal{Z}$, we train a specialist policy $\pi_\theta^{k,z}$ from scratch by maximizing Equation 1, using 10 random seeds; hyperparameters are summarized in Table 4 (Appendix section). Each trained policy is stored in the library indexed by its task-fault pair,

$$\mathcal{L} = \big\{(k, z) \mapsto \pi_\theta^{k,z} : k \in \mathcal{K},\ z \in \mathcal{Z}\big\},$$

yielding $|\mathcal{K}| \times |\mathcal{Z}| = 4 \times 15 = 60$ policies in total.

4.4 Phase 2: Online Query to the Library

Our goal is to study how effectively a robot can recover from a fault, not how faults are detected, so we isolate the recovery problem from detection by adopting *oracle fault-matched retrieval*: the fault identity z_{new} on task k is assumed known once a fault alarm is raised (Figure 2, part B). The library $\mathcal{L}$ is searched for a policy indexed by (k, z_{new}). If z_{new} exactly matches a descriptor covered during offline training for task k (the *seen* setting), the corresponding policy $\pi_\theta^{k,z_{\text{new}}}$ is deployed directly

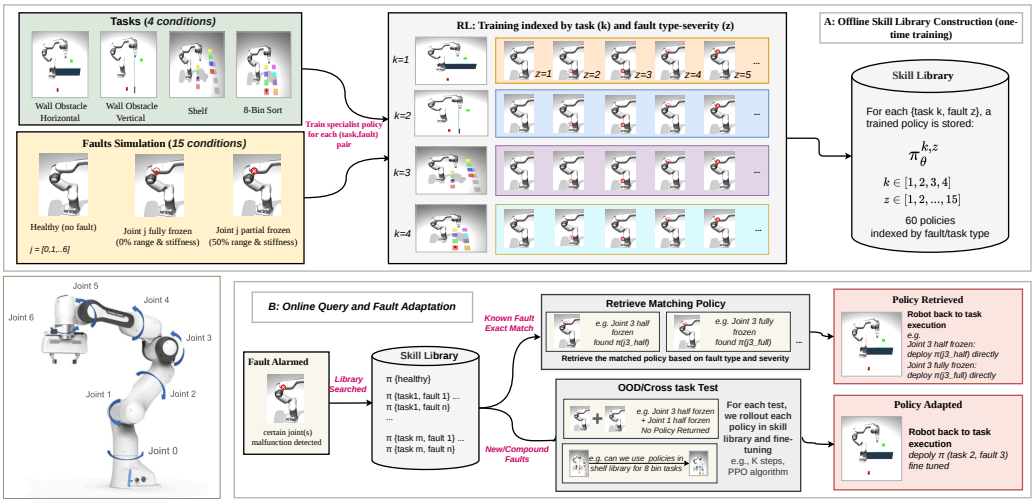

Figure 2: Proposed fault-aware skill library for robot manipulation, across four pick-and-place tasks and subject to joint/actuator faults. *Phase 1 (offline)*: trains and stores a specialist policy per task and fault descriptor. *Phase 2 (online)*: retrieves or adapts the best-matching policy upon a fault alarm.

(*Policy Retrieved*). Otherwise, e.g., an out-of-distribution, compound fault, or no exact match in the task library, every policy in $\mathcal{L}$ instead serves as a candidate initialization (*OOD/Cross-Task Test*): starting from each candidate's parameters θ, we continue optimizing Equation 1 under the current fault dynamics for 1 million steps, continuing from the candidate's original training point rather than restarting its schedule from step zero. The candidate with the highest return $J(\pi_\theta)$ after fine-tuning is deployed (*Policy Adapted*).

5 Experimental Setup

We evaluate our framework on four pick-and-place manipulation tasks, 8-Bin Sort, Shelf Placement, Vertical Wall Pick-Place, and Horizontal Wall Pick-Place, using a Franka Panda 7-DOF manipulator in the MuJoCo Menagerie model (DeepMind, 2022). All manipulation environments share the same observation and action interfaces. The 17-dimensional observation includes the end-effector position, object position, target position, robot joint positions, and a binary grasp indicator. Actions are 3-D Cartesian end-effector displacements bounded to ± 0.05 m per step and are mapped to joint commands using damped least-squares inverse kinematics with damping coefficient $\lambda = 10^{-4}$. We consider four manipulation environments with different workspace geometries and kinematic requirements. Together with the evaluated fault conditions, they are used to probe qualitatively different regimes of fault adaptation.

8-Bin Sort (E1). Eight bins are arranged in a 2×4 grid at $x \in \{0.25, 0.50\}$ m and $y \in \{-0.30, -0.10, +0.10, +0.30\}$ m. At the beginning of each episode, a source bin and a different target bin are selected randomly. The robot must pick the object from the source bin and place it in the target bin, with success defined as placing the object within 0.08 m of the target. The wide workspace requires substantial variation in arm configuration and is used to study the *broad fault-sensitivity* regime, where faults across multiple joints may affect task performance.

Shelf Placement (E2). The robot picks an object from one of four source positions at $x = 0.28$ m and places it at one of four elevated target positions at $x = 0.52$ m, with $y \in \{-0.30, -0.10, +0.10, +0.30\}$ m and a shelf height of 0.45 m. Success is defined as placing the object within 0.05 m of the target. The more constrained workspace is used to study the *selective fault-sensitivity* regime, where the effect of a fault can depend strongly on the role of the affected joint in the task.

Wall Pick-Place–Horizontal (E3). The robot performs a fixed pick-and-place task in the presence of a rectangular horizontal obstacle. The obstacle requires the end effector to follow a curved trajectory between the pick and place locations. Success is defined as placing the object within 0.05 m of the target.

Wall Pick-Place–Vertical (E3). We additionally evaluate a vertical orientation of the wall obstacle. The manipulation objective remains unchanged, but the altered obstacle geometry imposes different kinematic requirements on the robot.

For the comparison, we report four evaluation metrics averaged over 10 seeds: area under the learning curve (AUC), Success rate at the end of the training budget (Final SR), success rate at 200k fine-tuning steps (SR@20%), and steps to reach 80% success rate (PDE). Together, these metrics ensure assessment of policy performance and sample efficiency.

6 RESULTS

6.1 PERSPECTIVE 1: SAME FAULT ACROSS DIFFERENT ENVIRONMENTS

Table 1 answers the question: *which joints are critical, and does this depend on the task?* We deploy the healthy policy on each faulted environment without fine-tuning and measure the success rate drop. Two joints emerge as *universally critical*: Joint 1 (shoulder pitch, J1) and Joint 3 (elbow, J3), with drops of 70–99% depending on the environment. J3 full freeze causes complete task failure (0% SR) on both Wall variants, where navigating around the obstacle requires elbow articulation. J1 is catastrophic on 8-Bin (98%) and Shelf (86%), but less critical on the vertical wall due to different kinematic demands of that task geometry.

Fault criticality is *task-dependent*: J5 causes a 42% drop on 8-Bin (wide lateral workspace stresses the wrist) but negligible impact on Shelf (constrained row layout allows compensation). J0, J2, J4, and J6 are consistently non-critical across all tasks. This cross-task pattern has practical value: a robot can immediately prioritize library retrieval for J1 and J3 faults, while continuing with the healthy policy for other joints.

Table 1: Healthy policy success rate drop (%) under full joint failure. Mean over 10 seeds, 50 episodes each. **Bold**: catastrophic drop ($> 50\%$).

Environment	Baseline SR	Joint 0	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
Wall-H (E3)	99.5%	35.7	**69.7**	−0.1	**99.5**	−0.1	0.5	−0.2
Wall-V (E3)	98.8%	9.7	0.8	−0.7	**98.8**	−0.4	−0.6	0.6
8-Bin (E1)	99.2%	2.8	**98.2**	2.2	**81.6**	0.0	**42.2**	0.4
Shelf (E2)	99.2%	0.4	**86.2**	1.8	**89.6**	1.4	1.6	0.2

6.2 PERSPECTIVE 2: LIBRARY VS. SCRATCH WITHIN THE SAME ENVIRONMENT

Table 2 compares initialization strategies within each environment: same-fault library retrieval, healthy policy transfer, and scratch training. This perspective answers: *does the library help, and by how much?* For critical faults, the same-fault library source substantially outperforms the healthy source in early-phase efficiency. The most striking result is Shelf J1_full: the library source reaches 99.4% SR within 200k steps (SR@20%), while the healthy source achieves only 7.7%—a $\sim$189$\times$ reduction in steps needed to reach 80% SR (PDE).

On 8-Bin J1_full, the library source achieves AUC = 0.702 vs. 0.285 for healthy (SR@20% = 71% vs. 17%). For J5_full on 8-Bin, library and healthy sources perform comparably (AUC 0.930 vs. 0.954), confirming the library provides no advantage when the fault is non-critical, so the healthy policy already provides a good initialization (E2 selective regime). For J3_full, both sources struggle: Wall achieves AUC = 0.000 from any source, and Shelf J3_full yields 0% SR@20% regardless of initialization. These cases represent the E3/E4 boundary where no single-fault library policy can recover meaningful performance within the fine-tuning budget.

Table 2: Transfer fine-tuning results: same-fault library source vs. healthy source across all environments (AUC, PDE, SR@20%, Final SR). **Bold**: library outperforms healthy source.

			Same-fault library				Healthy source			
Env	**Fault**	**Scratch**	**AUC**	**PDE**	**SR@20%**	**Final SR**	**AUC**	**PDE**	**SR@20%**	**Final SR**
Wall-H (E3)	J3_full	<5%	0.000	>2000k	0.0%	0.0%	0.000	>2000k	0.0%	0.0%
Wall-H (E3)	J1_full	<5%	0.298	1403k	29.6%	29.9%	**0.974**	**30k**	**98.1%**	**99.2%**
8-Bin (E1)	J1_full	69.5%	**0.702**	**1070k**	**71.0%**	**72.9%**	0.285	>2000k	16.8%	36.7%
8-Bin (E1)	J3_full	21.5%	0.227	>2000k	23.5%	24.3%	0.244	>2000k	19.7%	26.5%
8-Bin (E1)	J5_full	89.8%	0.930	**11k**	98.8%	89.4%	**0.954**	39k	**95.8%**	**99.1%**
Shelf (E2)	J1_full	88.4%	**0.609**	**9k**	**99.4%**	21.4%	0.102	1701k	7.7%	**4.0%**
Shelf (E2)	J3_full	1.1%	0.000	>2000k	0.0%	0.0%	0.000	>2000k	0.0%	0.0%

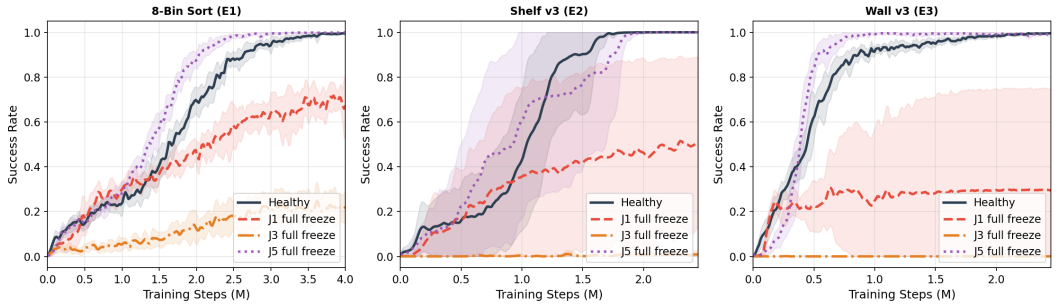

Figure 3: Policy learning under full joint freeze across three environments (mean ± std, 10 seeds).

6.3 Perspective 3: Fault Severity Spectrum Within One Environment

Figure 3 shows how fault severity affects learning dynamics. J1_full and J3_full converge significantly slower than healthy, with high seed-to-seed variance reflecting PPO instability under hard constraints. J5_full converges similarly to healthy, visually confirming its non-critical status. The learning curves also reveal why the best-checkpoint callback is essential: healthy policies peak near 100% around 3–4M steps then collapse in some seeds, motivating our library construction strategy.

Figure 4 shows the full-fault-only transfer matrices (AUC) for all four environments. The severity contrast is striking: for J1_full, the same-fault source achieves AUC = 0.70 on 8-Bin while all other sources, including healthy, score below 0.30. This single bright cell in an otherwise dark column is the visual signature of a critical full-severity fault, where fault-matched retrieval is essential. For J5_full, the column is uniformly green on 8-Bin: any source transfers well, confirming that severity alone does not determine library necessity; task geometry matters, too. The same pattern of isolated bright cells appears on Wall-H and Wall-V, where J1_full and J3_full are similarly critical (Section 6), underscoring that fault-matched retrieval is essential across tasks, not just within a single environment.

6.4 Perspective 4: Cross-Task Transfer

A skill library built for one task may generalize to a different task on the same robot. We test this by using the 8-Bin Sort library to fine-tune on Shelf faults and vice versa.

Table 3 shows the results. The 8-Bin library transfers strongly to Shelf: for J1_full, the best library source (j1_half) reaches 97.5% final SR vs. 12.7% for the healthy source, with PDE of 518k vs. 1845k steps. For J5_full and the half-severity faults, the gains are even larger in sample efficiency—J3_half reaches 100% final SR in 48k steps vs. 1627k for healthy, a 34× speedup. J3_full remains infeasible from all sources, consistent with within-task results.

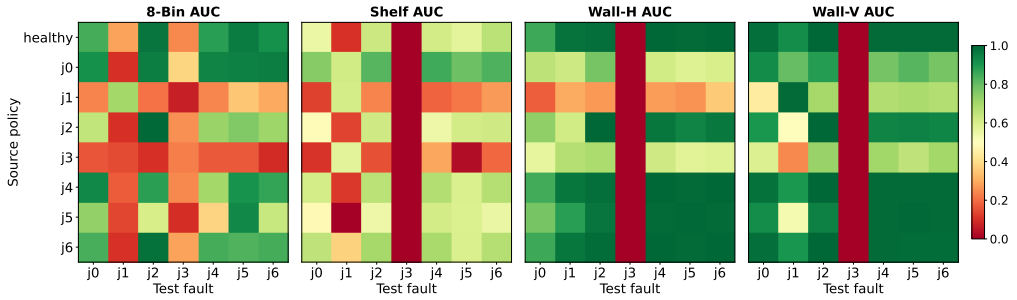

Figure 4: AUC transfer matrices for all four environments (full-fault conditions only): 8-Bin Sort, Shelf, Wall horizontal, Wall vertical.

Table 3: Cross-task transfer: 8-Bin library applied to Shelf faults and vice versa (AUC, PDE, SR@20%, Final SR). **Bold**: library outperforms healthy.

			Best library source				Healthy source			
Direction	**Fault**	**Src**	**AUC**	**PDE**	**SR@20%**	**Final SR**	**AUC**	**PDE**	**SR@20%**	**Final SR**
8-Bin→Shelf	J1_full	j1_half	**0.628**	**518k**	**13.6±15.3%**	**97.5±6.7%**	0.089	1845k	0.7±1.7%	12.7±25.4%
	J3_full	—	0.000	>2000k	0.0%	0.0%	0.000	>2000k	0.0%	0.0%
	J5_full	j1_half	**0.961**	**54k**	**100.0±0.0%**	**100.0±0.0%**	0.207	1161k	0.4±0.8%	31.6±35.2%
	J1_half	j1_half	**0.942**	**65k**	**99.4±0.9%**	**100.0±0.0%**	0.331	1109k	25.9±29.9%	29.7±34.8%
	J3_half	j3_half	**0.967**	**48k**	**100.0±0.0%**	**100.0±0.0%**	0.123	1627k	2.4±6.6%	32.8±31.9%
Shelf→8-Bin	J1_full	j0_half	0.167	>2000k	5.6±3.8%	29.6±3.6%	0.167	>2000k	5.6±3.8%	29.6±3.6%
	J3_full	j6_half	0.037	>2000k	3.3±1.5%	5.2±2.8%	0.037	>2000k	3.3±1.5%	5.2±2.8%
	J5_full	j0_full	0.162	>2000k	11.5±3.7%	27.5±5.7%	0.162	>2000k	11.5±3.7%	27.5±5.7%
	J1_half	healthy	0.374	>2000k	18.3±4.9%	61.5±7.7%	0.374	>2000k	18.3±4.9%	61.5±7.7%
	J3_half	healthy	0.091	>2000k	6.6±1.8%	12.9±3.5%	0.091	>2000k	6.6±1.8%	12.9±3.5%

The Shelf library provides no gain on 8-Bin: best-lib and healthy source are identical across all metrics. The shelf task has a narrow, constrained workspace; the policies it produces lack the joint coverage needed for the wider 8-Bin workspace. Library richnes, coverage of diverse joint configurations, determines cross-task transferability.

6.5 Perspective 5: Compound Faults as OOD Generalization

We probe the outer boundary of the library by testing all 15 single-fault source policies on 9 compound fault conditions (simultaneous freezing of two joints) and 3 triple fault conditions (simultaneous freezing of three joints) not seen during library training, across all four environments. Figure 12 (Appendix E) summarizes the double fault results.

The results reveal a clear severity-driven pattern across all environments: *half+half* compound faults remain largely learnable (best SR up to 100% across all tasks), while *full+full* faults define a near-infeasible regime. For J1_full+J3_full, the best library source achieves only 29% SR on 8-Bin and 0% on Wall and Shelf, compared to 0% for the healthy policy in all cases. The library still provides meaningful benefit over healthy transfer in moderate cases (J1_full+J5_half: 99% vs. 13% on Wall horizontal).

Triple faults push the boundary further. Table 7 shows results for three triple combinations on 8-Bin Sort. When all three critical joints are frozen at full severity (J1_full+J3_full+J5_full), the best library source achieves only 18.6% SR vs. 0.5% for healthy, the library still helps, but barely. Adding a non-critical joint (J1_full+J3_full+J0_full) drops performance to 0% from all sources, marking complete infeasibility. Reducing one joint to half severity (J1_full+J3_half+J5_full) raises the ceiling to 27.9%, suggesting severity is the dominant factor even in multi-joint faults. Table 7 reports triple-fault results (Appendix E).

7 DISCUSSION

The benefit of fault-matched retrieval depends on fault severity and task geometry. Library initialization helps most when a fault substantially changes the required control strategy. On Shelf under full-severity J1, the fault-matched source reaches 99.4% success within 200k steps versus 7.7% for the healthy source (13× at equal budget, 189× fewer steps to threshold); on 8-Bin the same fault gives a smaller but still substantial 2.5× AUC and 4× SR@20% improvement. For full-severity J5 on both tasks, the advantage largely disappears, since the healthy policy is already a good starting point – retrieval helps most exactly where the fault creates a large mismatch with the healthy policy.

Full-severity compound faults expose limitations in exact-match retrieval. Half-severity compound faults remain manageable (100% success across all environments), but full-severity J1+J3 compound faults are not: the best policy reaches only 29% on 8-Bin and 0% on Wall and Shelf, even from the most relevant source. This reflects a structural limit of exact matching on $z = (j, s)$: compound faults, intermediate severities, and other conditions outside the 14 trained combinations have no library entry. Treating fault space as structured rather than discrete, e.g. a continuous severity/joint-relationship similarity metric, as in GP-based repertoire methods such as IT&E (Cully et al., 2015) and APROL (Kaushik et al., 2020), could let retrieval identify policies for nearby, untrained fault conditions; augmenting the library with compound and intermediate-severity entries and testing whether the fault-matched advantage survives is a direct next step.

Library diversity determines cross-task transferability. 8-Bin transfers effectively to Shelf, but not the reverse. 8-Bin's broad lateral manipulation exposes policies to a wider range of joint configurations, giving useful initializations even under Shelf's more constrained geometry; Shelf's localized motions provide less coverage in the other direction. Fault-adaptation libraries built from tasks that induce diverse robot configurations may therefore generalize across tasks better than ones built from geometrically constrained tasks alone.

Limitations and future directions. All experiments are conducted in simulation, so transfer to physical hardware remains untested. The library covers only single-joint faults at two severity levels, limiting direct retrieval to the 14 trained fault conditions and motivating the structured fault-space extension discussed above. In addition, our retrieval procedure assumes oracle knowledge of fault identity and severity. Future work may extend the Skill Library framework to more complex tasks and fault scenarios, develop similarity-based policy retrieval for unseen fault conditions, and validate the proposed approach on physical robots.

8 CONCLUSION

We introduced Skill Library, a fault-aware policy reuse framework for efficient robot adaptation under changes in the robot's own dynamics. Rather than relying solely on a healthy policy or retraining from scratch after a failure, the framework stores fault-specific policies learned offline and reuses them as starting points for deployment-time adaptation. Across four manipulation tasks, fault-matched initialization consistently improved adaptation efficiency for critical joint faults, with the largest gains occurring when the healthy policy was poorly suited to the faulted dynamics. We further showed that policies trained only on single-joint faults can provide useful initialization for previously unseen compound faults, although severe multi-joint failures remain challenging. The results also highlight that transferability depends on both fault severity and the diversity of behaviors represented in the library. Overall, these findings support fault-aware policy reuse as a practical direction for reducing the interaction required for post-failure adaptation. Future work should extend the library to structured similarity-based retrieval, broader fault conditions, and validation on physical robots.

REPRODUCIBILITY STATEMENT

Our code for reproducing the results of this study is available at `https://anonymous.4open.science/r/Fault_Adaptation-6E78`.

AI USE DISCLOSURE

In this work, we used generative AI tools (Claude, Anthropic) to assist with editing for readability, formatting references, and editing software code for the simulation environments and training pipelines. We have not used generative AI tools to generate synthetic datasets, develop theoretical models, formulate mathematical claims, prove mathematical claims, propose or refine hypotheses, or interpret results; these tasks were performed entirely by the authors. Design of research methodology and experiments, and implementation of methods were performed by the authors with AI-assisted code editing for implementation details. All AI-assisted code was verified and tested for correctness by the authors through empirical evaluation on the simulation environments. AI-assisted prose was reviewed and revised by the authors for scientific accuracy. We take responsibility for the final content of this work, including all text, claims, and experimental artifacts.

REFERENCES

Ignasi Clavera, Anusha Nagabandi, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt: Meta-learning for model-based control. In *Proc. ICLR*, 2019.

Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. *Nature*, 521:503–507, 2015. doi: 10.1038/nature14422.

DeepMind. MuJoCo menagerie: A collection of high-quality models for MuJoCo. `https://github.com/google-deepmind/mujoco_menagerie`, 2022.

Fernando Fernández and Manuela Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *Proc. AAMAS*, pp. 720–727, 2006.

Ankit Goyal, Valts Blukis, Jie Xu, Yijie Guo, Yu-Wei Chao, and Dieter Fox. RVT-2: Learning precise manipulation from few demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

Yijie Guo, Bingjie Tang, Iretiayo Akinola, and Dieter Fox. SRSA: Skill retrieval and adaptation for robotic assembly tasks. In *Proc. ICLR*, 2025.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proc. ICML*, 2018.

Yuhang Hu, Boyuan Chen, and Hod Lipson. Egocentric visual self-modeling for autonomous robot dynamics prediction and adaptation. *npj Robotics*, 3(1):14, 2025.

Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. BC-Z: Zero-shot task generalization with robotic imitation learning. In *Proceedings of the 5th Conference on Robot Learning (CoRL)*, volume 164 of *Proceedings of Machine Learning Research*, pp. 991–1002. PMLR, 2022.

Rituraj Kaushik, Pierre Desreumaux, and Jean-Baptiste Mouret. Adaptive prior selection for repertoire-based online adaptation in robotics. *Frontiers in Robotics and AI*, 6:151, 2020. doi: 10.3389/frobt.2019.00151.

Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. RMA: Rapid motor adaptation for legged robots. *arXiv:2107.04034*, 2021.

Haizhen Li and Xilun Ding. Adaptive and intelligent robot task planning for home service: A review. *Engineering Applications of Artificial Intelligence*, 117:105618, 2023. ISSN 0952-1976. doi: https://doi.org/10.1016/j.engappai.2022.105618.

Siyuan Li, Fangda Gu, Guangxiang Zhu, and Chongjie Zhang. Context-aware policy reuse. In *Proc. AAMAS*, 2019.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. In *International Conference on Learning Representations*, 2024.

Marius Memmel, Jacob Berg, Bingqing Chen, Abhishek Gupta, and Jonathan Francis. STRAP: Robot sub-trajectory retrieval for augmented policy learning. In *Proc. ICLR*, 2025.

Andrzej Milecki and Patryk Nowak. Review of fault-tolerant control methods for robotic systems. *Robotics and Autonomous Systems*, 163, 2023.

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. In *6th Annual Conference on Robot Learning*, 2022. URL https://openreview.net/forum?id=tGbpgz6yOrI.

Nuria Nievas, Adela Pagès-Bernaus, Francesc Bonada, Lluís Echeverria, and Xavier Domingo. Reinforcement learning for autonomous process control in industry 4.0: Advantages and challenges. *Applied Artificial Intelligence*, 38(1):2383101, 2024. doi: 10.1080/08839514.2024.2383101.

Simone Parisi, Aravind Rajeswaran, Senthil Purushwalkam, and Abhinav Gupta. The unsurprising effectiveness of pre-trained vision models for control. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, volume 162 of *Proceedings of Machine Learning Research*, pp. 17359–17371. PMLR, 2022.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *Proc. IEEE ICRA*, pp. 3803–3810, 2018.

Tan-Hanh Pham, Godwyll Aikins, Tri Truong, and Kim-Doang Nguyen. Adaptive compensation for robotic joint failures using partially observable reinforcement learning. *Algorithms*, 17(10):436, 2024.

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Devin Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *Proc. ICML*, 2019.

Sheila Schoepp. Enhancing robotic fault tolerance through reinforcement learning. Master's thesis, University of Alberta, 2024.

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

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. CLIPort: What and where pathways for robotic manipulation. In *Proceedings of the 5th Conference on Robot Learning (CoRL)*, volume 164 of *Proceedings of Machine Learning Research*, pp. 894–906. PMLR, 2022.

Kaustubh Sridhar, Souradeep Dutta, Dinesh Jayaraman, and Insup Lee. REGENT: A retrieval-augmented generalist agent that can act in-context in new environments. In *Proc. ICLR*, 2025.

Emanuel Todorov and Michael I. Jordan. Optimal feedback control as a theory of motor coordination. *Nature Neuroscience*, 5(11):1226–1235, 2002.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *Proc. IEEE/RSJ IROS*, pp. 5026–5033, 2012.

Ruihan Yang, Minghao Zhang, Nicklas Hansen, Huazhe Xu, and Xiaolong Wang. Learning vision-guided quadrupedal locomotion end-to-end with cross-modal transformers. In *International Conference on Learning Representations*, 2022.

A Hyperparameter Tuning

Table 4 shows the hyperparameters we used for training the PPO algorithm over each task. All environments share $n_{\text{epochs}} = 10$, $\gamma = 0.99$, $\lambda_{\text{GAE}} = 0.95$. 8-Bin Sort policies are saved at the best checkpoint rather than the final model, because the wide workspace and high task diversity cause PPO to collapse after peaking (Section 3); Shelf and Wall converge stably within their training budgets and use final models.

Table 4: PPO hyperparameters per environment.

Environment	lr	n_steps	batch_size	Library policy
8-Bin Sort	2×10^{-4}	4096	256	best checkpoint
Shelf	3×10^{-4}	2048	256	final model
Wall	3×10^{-4}	4096	512	final model

B Complete Transfer Matrices: Half-Severity Faults

Half-severity faults transfer broadly from many sources—the columns are mostly warm across all sources, confirming that mild faults do not require fault-matched initialization.

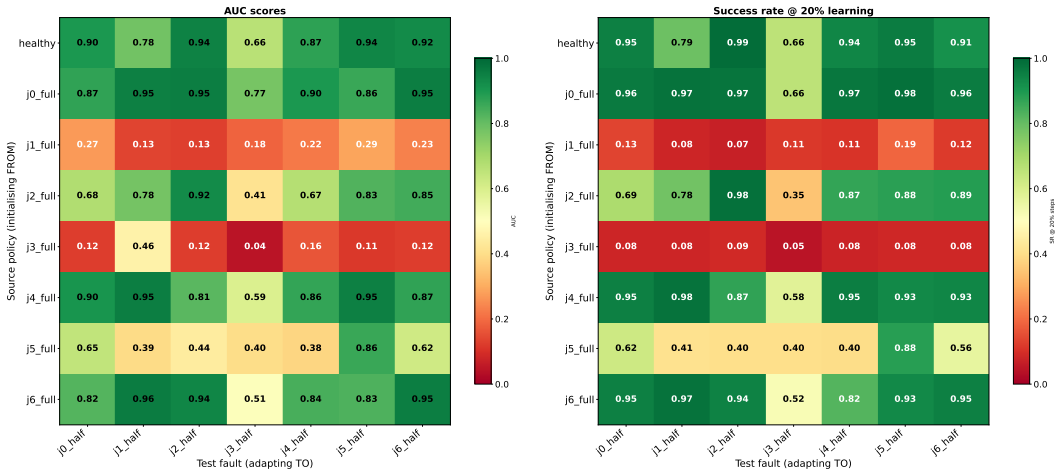

Figure 5: Transfer matrix for 8-Bin Sort (half-fault conditions only). **Left**: AUC scores. **Right**: Success rate @ 20% learning. Half-severity faults transfer well from most sources.

C Complete Transfer Matrices: Full-Severity Faults

Full-severity faults show sparse structure across all four environments: same-fault source stands out clearly while most other sources score near zero for critical joints. For Wall Pick-Place (Figures 9 and 10), the J3_full column is uniformly dark (AUC = 0.000, SR@20% = 0%) for all sources in both variants, confirming complete adaptation infeasibility for this joint-task combination. The horizontal variant also shows J1_full as critical (dark column), while the vertical variant shows J1_full as recoverable from most sources.

D Cross-Algorithm Study

To investigate whether the PPO training instability we observed is algorithm-specific, we trained SAC (Haarnoja et al., 2018) policies on the same 8-Bin Sort task under the same four conditions

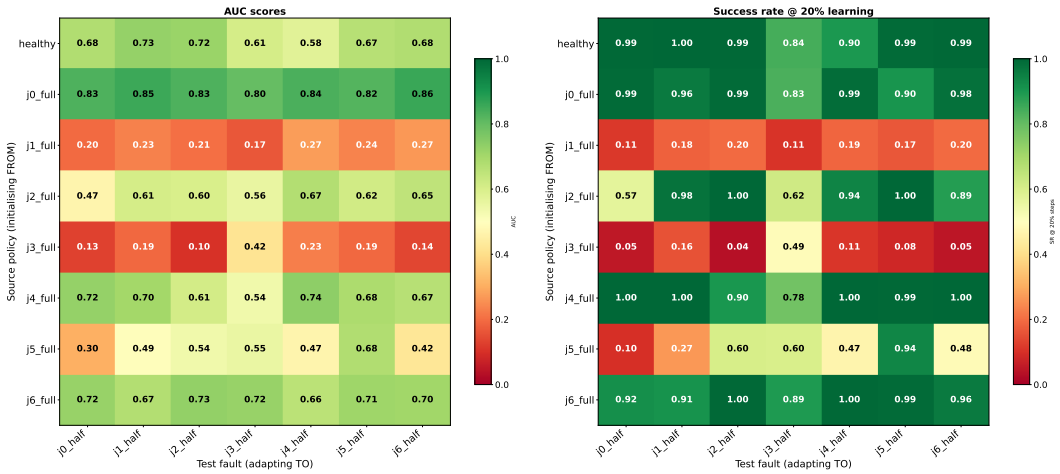

Figure 6: Transfer matrix for Shelf (half-fault conditions only). **Left**: AUC scores. **Right**: Success rate @ 20% learning.

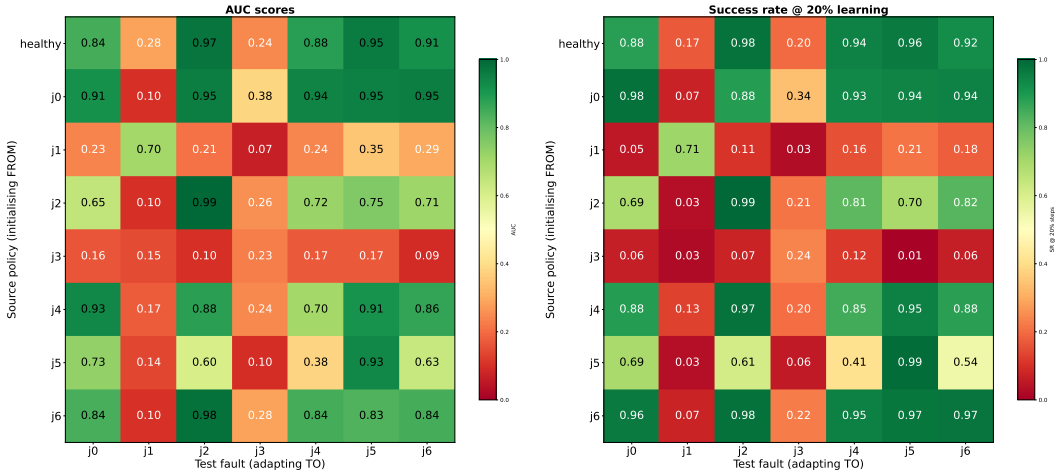

Figure 7: Transfer matrix for 8-Bin Sort (full-fault conditions only). **Left**: AUC scores. **Right**: Success rate @ 20% learning.

(healthy, J1_full, J3_full, J5_full) for up to 1M steps. SAC maintains an experience replay buffer and performs off-policy updates, which in principle should prevent the catastrophic forgetting observed in PPO.

Figure 11 shows learning curves for SAC vs. PPO over the first 650k steps. SAC achieves higher success rates than PPO at the same step count for all conditions (e.g. 28.7% vs. 19.0% for healthy at 600k steps), indicating better early sample efficiency. However, SAC also exhibits instability at longer horizons: peak SR values (healthy: 93%, J5: 97%) are substantially higher than final SR values (healthy: 28%, J5: 27%), suggesting that SAC also suffers from policy degradation, though through a different mechanism than PPO.

These results suggest that training instability is a general challenge for deep RL in long-horizon manipulation, independent of the algorithm choice. The best-checkpoint callback we use for library construction addresses this for both algorithms. A systematic study comparing PPO, SAC, and other algorithms for fault-aware library construction is left for future work.

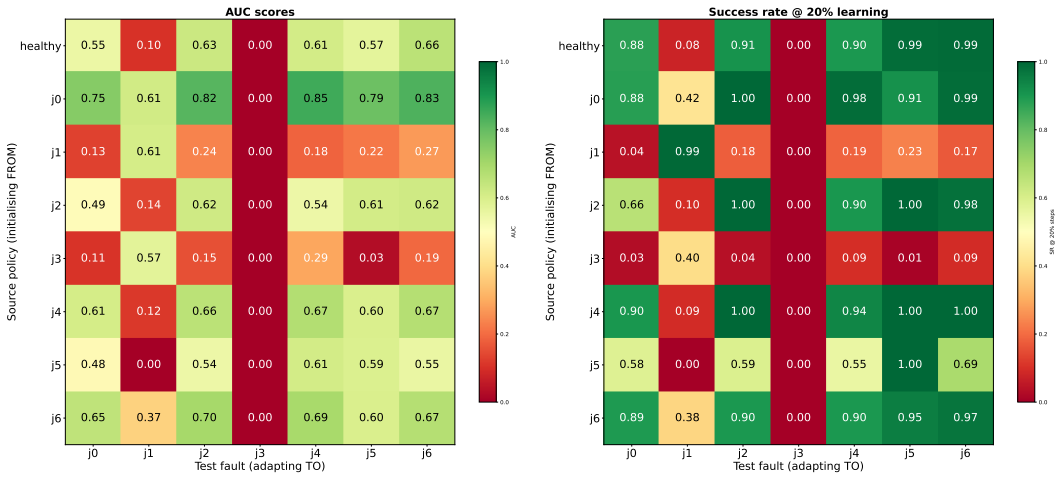

Figure 8: Transfer matrix for Shelf (full-fault conditions only). **Left**: AUC scores. **Right**: Success rate @ 20% learning.

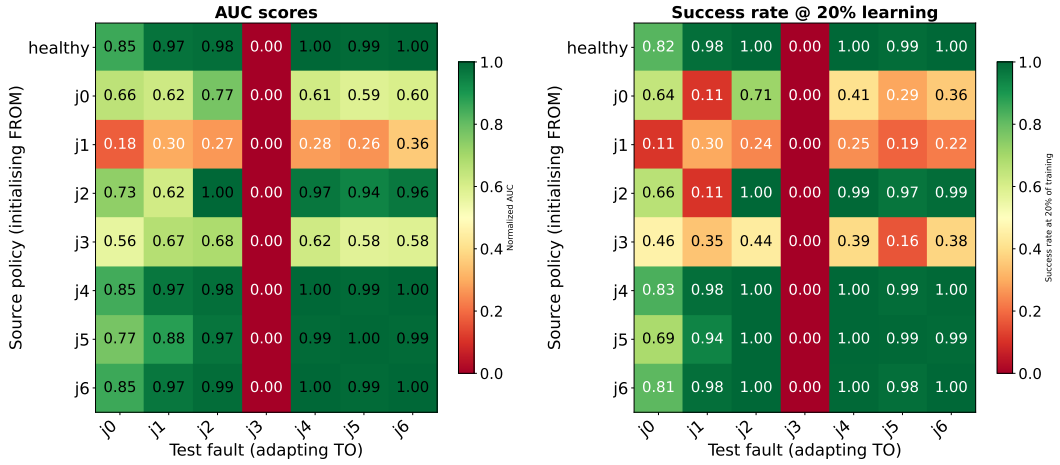

Figure 9: Transfer matrix for Wall Pick-Place, horizontal variant (full-fault conditions only). **Left**: AUC. **Right**: SR@20%. J3_full column is uniformly 0 from all sources.

Table 5: SAC vs. PPO comparison on 8-Bin Sort at 600k steps. Peak SR is the maximum achieved during training.

Condition	**SR @ 600k steps**		**Peak SR**	
	SAC	**PPO**	**SAC**	**PPO**
Healthy	28.7%±5.1%	19.0%±4.0%	93.0%±8.7%	100%±0%
J1_full	26.5%±6.2%	22.5%±6.9%	75.5%±6.6%	∼95%
J3_full	17.6%±5.8%	5.3%±2.5%	40.3%±6.8%	∼40%
J5_full	25.5%±7.5%	19.3%±3.2%	87.1%±10.7%	100%±0%

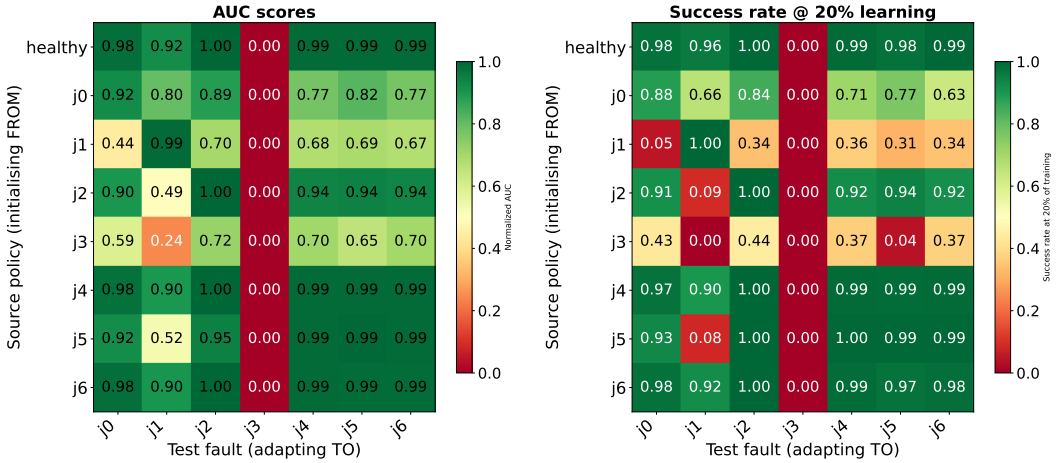

Figure 10: Transfer matrix for Wall Pick-Place, vertical variant (full-fault conditions only). **Left**: AUC. **Right**: SR@20%. J3_full is again infeasible; J1_full is more recoverable in the vertical geometry.

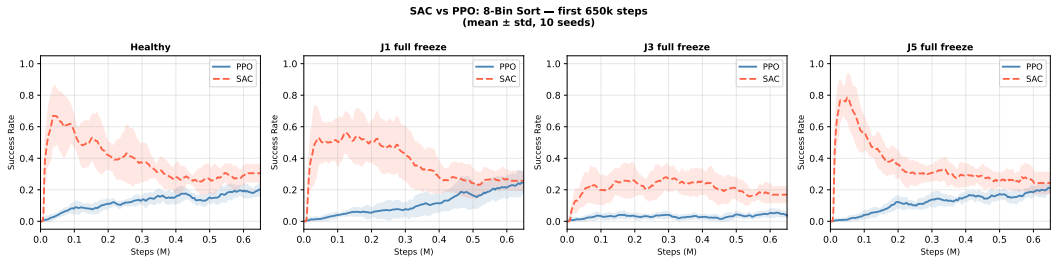

Figure 11: SAC vs. PPO learning curves on 8-Bin Sort (first 650k steps, mean ± std, 10 seeds). SAC (red dashed) shows better early sample efficiency than PPO (blue solid) across all fault conditions, but both algorithms exhibit instability at longer training horizons.

E Compound Fault Transfer Results

Table 6: Double compound fault transfer on 8-Bin Sort (E4): best single-fault library source after 1M fine-tuning steps vs. healthy source. Mean over 10 seeds.

Compound Fault	Best Source	Best SR	Healthy SR	Gain
J1_full + J3_full	j1_half	29.1%	0.0%	+29.1%
J1_full + J5_full	j1_half	59.8%	1.4%	+58.4%
J3_full + J5_full	j1_half	26.3%	12.3%	+14.0%
J1_half + J3_half	j3_half	100.0%	70.3%	+29.7%
J1_half + J5_half	j1_half	99.4%	98.3%	+1.1%
J3_half + J5_half	j0_half	99.8%	88.7%	+11.1%
J1_full + J3_half	j3_half	66.0%	4.8%	+61.2%
J1_half + J3_full	j3_half	44.9%	29.4%	+15.5%
J1_full + J5_half	j1_full	65.2%	2.8%	+62.4%

Table 7: Triple fault transfer on 8-Bin Sort: best single-fault library source after 1M fine-tuning steps vs. healthy source. Mean over 10 seeds. Triple faults push the adaptation boundary further than double faults.

Triple Fault	Best Source	Best SR	Healthy SR	Gain
J1_full + J3_full + J5_full	j1_full	18.6%	0.5%	+18.1%
J1_full + J3_half + J5_full	j3_half	27.9%	3.1%	+24.8%
J1_full + J3_full + J0_full	healthy	0.0%	0.0%	0.0%

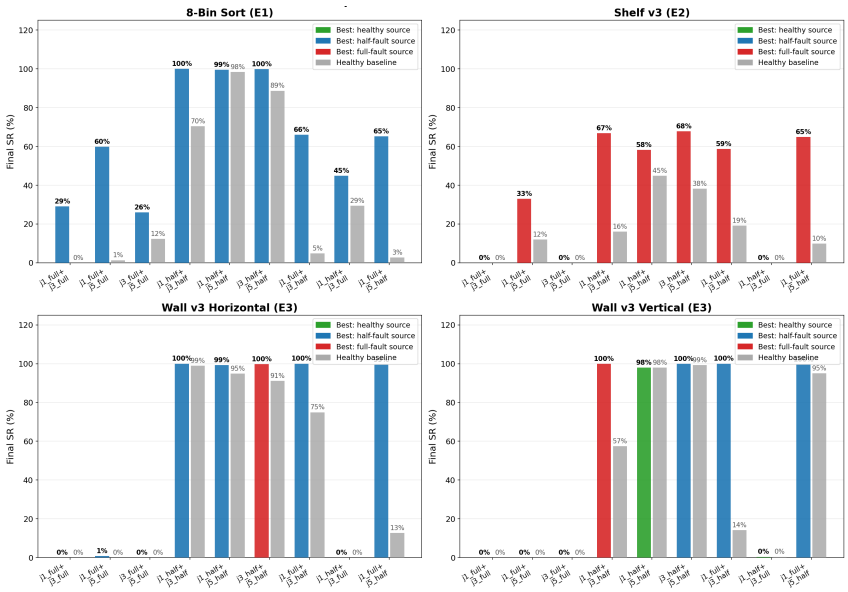

Figure 12: Compound fault transfer across all environments. Blue bars: best library source SR. Grey bars: healthy source SR. Half+half faults remain learnable; full+full faults are largely infeasible from any single-fault source.