MOSAIC: MULTI-TASK ORCHESTRATION VIA SPARSE ATOMIC INTERPOLATED CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-task robot learning is not limited only by policy capacity, but also by the structure through which tasks share control. Dense sharing couples incompatible task dynamics in a common pathway, while expert isolation reduces transfer and scales poorly. We introduce MOSAIC, a sparse basis-sharing framework for scalable multi-task manipulation. MOSAIC learns a shared low-coherence dictionary of atomic control bases and infers, for each context, a sparse set of active bases with continuous coefficients through a deterministic expected spike-and-slab bottleneck. The resulting active basis span allows related task phases to reuse control factors while reducing update coupling among incompatible dynamics. To avoid mode averaging in continuous mixtures, MOSAIC aligns dictionary bases with temporal options and uses continuous skill coordinates to modulate low-level control. Across Meta-World MT-10 and MT-50, MOSAIC achieves strong multi-task performance, improves hard contact-rich tasks, and preserves single-pass closed-loop execution efficiency. Mechanism-level analyses, including ablations, sparse-routing diagnostics, update-coupling measurements, and basis interventions, support that the gains arise from structured selective sharing rather than increased capacity or task-specific parameter isolation. All code, dataset, and reproducibility scripts will be released.

1 INTRODUCTION

Scaling robot learning to many manipulation tasks requires policies that can reuse control structure without entangling incompatible dynamics. Dense multi-task policies route all tasks through the same representation and control pathway, which can amplify destructive interference when tasks involve different contact modes, object interactions, or temporal phases (Espeholt et al.; Cho et al.; Hu et al., 2024; He et al., 2025). Isolated experts avoid such interference, but sacrifice transfer and scale poorly with the number of tasks. The key challenge is therefore not whether to share, but how to structure sharing: related tasks should reuse common control factors, while incompatible dynamics should interact only weakly.

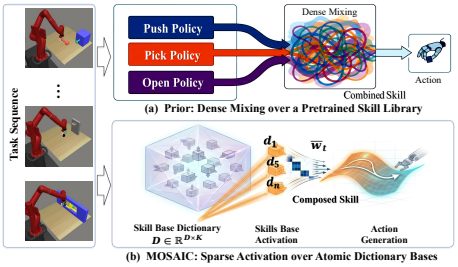


Figure 1: **Entangled sharing vs. selective sharing.** **(a)** Dense sharing mixes monolithic skills, causing entangled activations and interference. **(b)** MOSAIC selectively activates sparse task-aligned atomic bases with weights $\overline{\mathbf{w}}_t$.

Existing methods address this problem only partially. Gradient surgery and loss balancing (Yu et al., 2020; Chen et al.) modify task updates after conflicts arise, but leave the shared control pathway unchanged. Modular and mixture-of-experts policies (Shen et al., 2025; Yang et al., 2025) introduce conditional computation, but typically route to experts or branches rather than composing reusable control factors. Discrete skill abstractions (Li et al., 2025; van den Oord et al., 2018) improve interpretability, but may lose the continuous variation required for precise manipulation. Generative policies, including diffusion-based skill policies (Wang et al., 2024; Liang et al., 2024; Gupta et al., 2025), can model multimodal behavior, but iterative inference can increase closed-loop latency. These limitations motivate a sharing mechanism that is selective, compositional, continuous where needed, temporally coherent, and efficient at test time.

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.

We propose **M**ulti-task **O**rchestration via **S**parse **A**tomic **I**nterpolated **C**ontrol (MOSAIC), a structured sharing framework for multi-task robot control. MOSAIC is based on a simple principle: tasks should share reusable control factors through the basis directions relevant to the current context, rather than through a single dense pathway or isolated task-specific experts. To this end, MOSAIC maintains a global dictionary of atomic control bases and infers, at each timestep, a sparse coefficient vector over this dictionary. The resulting active basis span allows related task phases to reuse overlapping bases, while encouraging incompatible dynamics to induce low-overlap supports and weakly coupled dictionary updates.

MOSAIC realizes this principle with three mechanisms. First, an expected spike-and-slab bottleneck separates basis relevance from continuous coefficient modulation while preserving a differentiable single-pass policy path. Second, a near-orthogonal low-coherence dictionary regularizer discourages basis collapse and redundant control directions. Third, a basis-aligned option hierarchy separates discrete interaction-mode selection from continuous within-mode modulation, reducing mode averaging in continuous mixtures. Together, these components turn dense multi-task sharing into sparse, compositional, and temporally coherent basis sharing. We evaluate MOSAIC on Meta-World (Yu et al., 2021) MT-10 and MT-50 against monolithic (Haarnoja et al., 2018), optimization-level (Yu et al., 2020; Chen et al., 2018), modular (Huang et al., 2025; Cho et al., 2024), and generative baselines (Wang et al., 2024). MOSAIC achieves strong multi-task performance, especially on contact-rich and precision-heavy tasks. Beyond aggregate success rates, we analyze component ablations, sparse-routing patterns, dictionary-update coupling, basis interventions, and closed-loop latency. These results support the interpretation that MOSAIC's gains arise from structured selective sharing rather than task-specific parameter isolation or hierarchy alone.

Contributions. Our contributions are as follows.

1. We formulate *sparse basis sharing* as a structural alternative to dense parameter sharing, expert isolation, and post-hoc gradient correction.
2. We provide a first-order update-coupling view showing how sparse, low-overlap coefficients reduce shared-dictionary interference while preserving transfer through reused bases.
3. We instantiate this principle with an expected spike-and-slab bottleneck, a near-orthogonal low-coherence skill dictionary, and basis-aligned temporal options.
4. We evaluate MOSAIC on Meta-World MT-10 and MT-50 with baseline comparisons, ablations, routing diagnostics, basis interventions, and closed-loop efficiency measurements.

2 Related Work

MOSAIC is related to multi-task reinforcement learning, modular policy, skill abstraction, and hierarchical control. Unlike prior sharing schemes based on dense representations, expert routing, latent skills, temporal options, or low-dimensional control, MOSAIC composes tasks from shared atomic basis directions with sparse continuous coefficients and executes them through aligned options.

Multi-task RL and mitigation of optimization-level interference. Negative transfer is a central challenge in multi-task RL (Espeholt et al.; Yu et al., 2021), especially when heterogeneous tasks are trained through a densely shared policy. Gradient-level methods such as PCGrad (Yu et al., 2020) and GradNorm (Chen et al., 2018) reduce the update conflict by projecting or reweighting task gradients, but do not change the representation through which tasks share control. Large-scale multi-task robot learning methods (Haarnoja et al., 2018; Yu et al., 2021; Zhang et al., 2026) demonstrate the promise and difficulty of shared robotic policies, but dense sharing can still entangle incompatible contact modes and temporal phases. MOSAIC addresses this issue at the sharing-structure level: all tasks access a common dictionary, but each context activates only sparse support over reusable bases.

Modular policies, sparse MoE, and routing architectures. A closely related line of work uses modular networks, mixture-of-experts (MOE) (Huang et al., 2025), or routing policies (He et al., 2024; Zhang et al., 2026) to reduce interference by activating only parts of a model. Sparse MoE (Shen et al., 2025; Yang et al., 2025) and modular policy (Thapa & Li) route observations or tasks to experts, modules, or transformer blocks (Vaswani et al., 2017), and have shown that conditional computation can improve multi-task learning. MOSAIC differs in the unit of sharing. Rather than routing to separate expert networks, it routes over a shared dictionary of atomic control bases. The policy

therefore comprises reusable basis directions with continuous coefficients, allowing related tasks to share subcomponents without assigning each task to a fixed expert. This distinction is important: generic expert routing can reduce interference, but it does not provide a basis-level mechanism for fine-grained reuse across task phases.

Sparse and compositional skill representations. Skill abstraction methods (Pertsch et al.; Rana et al.) seek reusable primitives for long-horizon control. Classical movement primitives such as DMPs (Peters & Schaal, 2007) and ProMPs (Rothfuss et al., 2022) provide an interpretable structure but are often limited in scaling to various robot tasks (Paraschos et al.; Saveriano et al., 2023). Latent or semantic skill models (He & Ciocarlie, 2022; Puang et al., 2025) learn continuous task representations, while sparse latent skill (Gupta et al., 2025) or composable skill methods (Kim et al., 2021; Hausman et al., 2018) aim to represent tasks through combinations of reusable components. Discrete quantization methods (van den Oord et al., 2018; Li et al., 2025) improve interpretability but can lose continuous variation needed for precise manipulation. MOSAIC follows a hybrid design: sparse spike-and-slab (Ray et al., 2020; Sheikh & Lücke, 2016) routing exposes which bases are active, while continuous coefficients retain within-support adaptation. Unlike methods that learn only a task embedding or skill code (van den Oord et al., 2018; Li et al., 2025), MOSAIC couples the sparse skill coordinate to both dictionary regularization and low-level policy modulation.

Hierarchical control and option graphs. Hierarchical RL and option-based methods (Cho et al., 2024) decompose behavior into temporally extended modes. Graph-structured options and option discovery methods (Kim et al., 2021; Hao et al., 2026) further model transitions among reusable skills. These methods are complementary to MOSAIC but do not fully specify how shared control factors should be organized across many tasks. MOSAIC aligns temporal options with dictionary bases: sparse routing selects the active basis support, and the basis-aligned option provides temporally persistent execution. Our Hierarchy-only with FiLM (Perez et al., 2017) control directly tests this distinction: temporal options and FiLM modulation alone are insufficient without sparse dictionary routing.

Generative and diffusion-based robot policies. Generative robot policies, including diffusion-style skill policies (Liang et al., 2024; Wang et al., 2024), can model multimodal action distributions and complex manipulation behaviors. However, iterative denoising or sampling can introduce additional closed-loop latency. MOSAIC targets a complementary regime: it represents multimodal control through sparse basis selection and discrete option choice while preserving a deterministic single-pass policy path.

Structured low-dimensional control and basis regularization. Robotic manipulation often exhibits low-dimensional regularities such as motor synergies, null-space structure, and locally smooth control variations (Ficuciello et al., 2018; Colome & Torras, 2018; Cai et al., 2020; Tosatto et al., 2020). Such observations motivate compact control coordinates and locally linear approximations (He & Ciocarlie, 2022; Puang et al., 2025). MOSAIC uses this perspective operationally rather than as a strong geometric claim: sparse coefficients induce active linear control subspaces over a learned dictionary. Orthogonality regularization discourages basis collapse and cross-basis leakage, while basis-usage diagnostics test whether the resulting supports are sparse, task-structured, and temporally varying.

3 METHODOLOGY

MOSAIC implements selective sharing through a sparse-route over a shared skill base dictionary. It does not assign task-specific parameters. Instead, all tasks access the same dictionary, while each context activates a sparse support over dictionary bases. Related task phases can therefore reuse bases, whereas incompatible dynamics are encouraged to induce low-overlap active supports and low-overlap dictionary updates. For clarity, we describe the main architectural components in the paper and defer the full loss formulation and optimization algorithm to Appendix.A and B.1.

3.1 ENCODING AND TEMPORAL CONTEXT

Perception and language encoding. We encode visual and proprioceptive inputs using ϕ_I and ϕ_e and fuse them into

$$\boldsymbol{z}_t \;=\; f_{\text{enc}}(\phi_I(I_t), \phi_e(e_t)) \in \mathbb{R}^D, \tag{1}$$

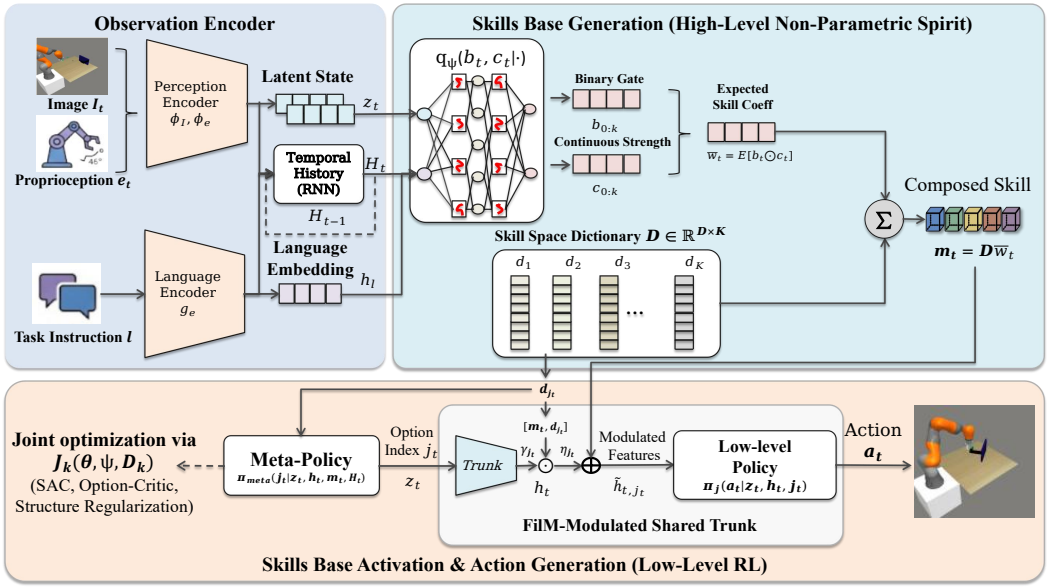


Figure 2: The Architecture of MOSAIC.

where $\boldsymbol{z}_t$ is the fused latent at timestep t. I_t is the raw image observation, e_t is the low-dimensional proprioceptive vector, and $f_{\text{enc}}(\cdot)$ is a fusion backbone that aggregates features into a D-dimensional representation.

The natural language instruction l is processed by an encoder g_θ to yield:

$$\boldsymbol{h}_\ell \;=\; g_\theta(l) \in \mathbb{R}^H. \tag{2}$$

Here, $\boldsymbol{h}_\ell$ is the semantic instruction embedding projected into $\mathbb{R}^H$, which provides static task context for the hierarchical control loop. Language conditioning improves generality, but MOSAIC's core mechanism is independent of the task specification format. In our main experiments, tasks can be specified by fixed task identifiers or natural-language instructions.

Temporal history. To capture short-horizon interaction context, MOSAIC maintains a recurrent history state $\boldsymbol{H}_t$ using an RNN history encoder. At each timestep, the RNN receives the current observation and task embedding:

$$\boldsymbol{H}_t \;=\; \text{RNN}(\boldsymbol{H}_{t-1}, [\boldsymbol{z}_t, \boldsymbol{h}_\ell]) \in \mathbb{R}^{D_H}, \tag{3}$$

where $\boldsymbol{H}_{t-1}$ is the hidden state from the previous timestep, and $[\boldsymbol{z}_t, \boldsymbol{h}_\ell]$ denotes the concatenation of the current latent state and task embedding. The routing context is then defined as $\mathcal{X}_t = (\boldsymbol{z}_t, \boldsymbol{h}_\ell, \boldsymbol{H}_t)$. The history state provides phase information for temporally extended manipulation, such as approach, contact, insertion, and release. This allows the sparse routing module to select different active basis supports not only across tasks, but also across different phases within the same task.

3.2 Atomic Skill Bases and the Global Dictionary

MOSAIC does not isolate tasks by assigning task-specific parameters. Instead, all tasks access a shared dictionary of atomic skill bases, while each context activates a sparse support over this dictionary. Related task phases can therefore reuse common bases, whereas incompatible dynamics are encouraged to induce low-overlap active supports and low-overlap updates on the shared dictionary.

We parameterize the atomic skill space using a learnable global dictionary: $\boldsymbol{D} = [\boldsymbol{d}_1, \ldots, \boldsymbol{d}_K] \in \mathbb{R}^{D\times K}$, $\boldsymbol{d}_k \in \mathbb{R}^D$, where each column $\boldsymbol{d}_k$ is a learnable basis vector representing a reusable control factor. The dictionary $\boldsymbol{D}$ defines a shared coordinate system for composing task- and phase-dependent behaviors. To keep the bases distinct and encourage broad coverage of reusable dynamics, we impose a near-orthogonality regularizer:

$$\mathcal{L}_{\text{ortho}}(\boldsymbol{D}) = \|\boldsymbol{D}^\top \boldsymbol{D} - \boldsymbol{I}\|_F^2, \tag{4}$$

where $\|\cdot\|_F$ is the Frobenius norm and $\boldsymbol{I}$ is the identity matrix. We also ℓ_2-normalize each column $\boldsymbol{d}_k$ to remove scale ambiguity and improve optimization stability. Sparsity determines which bases are used for a given context, while near-orthogonality reduces basis collapse and cross-basis leakage.

3.3 TASK-FAMILY SKILL SUBSPACE VIA STRUCTURED INFERENCE

Variational Spike-and-Slab Inference. To infer context-dependent sparse basis usage, we adopt a factorized *spike-and-slab* variational posterior. This formulation decouples binary basis activation from continuous coefficient modulation. The inference network q_ψ maps the context $\mathcal{X}_t = (\boldsymbol{z}_t, \boldsymbol{h}_\ell, \boldsymbol{H}_t)$ to $\boldsymbol{b}_t$ and $\boldsymbol{c}_t$, as shown in Eq. equation 5.

$$q_\psi(\boldsymbol{b}_t, \boldsymbol{c}_t \mid \mathcal{X}_t) = \prod_{k=1}^{K} \mathrm{Bernoulli}\big(b_{t,k} \mid \pi_k(\mathcal{X}_t)\big) \times \mathcal{N}\big(c_{t,k} \mid \mu_k(\mathcal{X}_t), \sigma_k^2(\mathcal{X}_t)\big) \tag{5}$$

where π_k, μ_k, and σ_k are output heads of the inference network. The Bernoulli term acts as the *spike* that selects whether basis k is active, while the Gaussian term acts as the *slab* that provides the continuous modulation strength. Thus, each basis can be inactive when $b_{t,k} = 0$, and when selected, its contribution is governed by the coefficient $c_{t,k}$.

Differentiable Bottleneck via Expected Activation. Directly sampling discrete activations in hierarchical RL can introduce high-variance credit assignment. We therefore use the *expected activation* vector $\bar{\boldsymbol{w}}_t$ as a deterministic bottleneck for the actor and critic paths, while keeping stochasticity within the variational regularizer $\mathcal{L}_{\mathrm{KL}}$:

$$\bar{\boldsymbol{w}}_t \triangleq \mathbb{E}_{q_\psi}[\boldsymbol{b}_t \odot \boldsymbol{c}_t] = \boldsymbol{\pi}(\mathcal{X}_t) \odot \boldsymbol{\mu}(\mathcal{X}_t). \tag{6}$$

Crucially, MOSAIC does not sample $\boldsymbol{b}_t$ in the control path. The policy receives the deterministic expected coefficient $\bar{\boldsymbol{w}}_t$, which allows gradients from the policy loss to flow analytically to the inference network ψ and the dictionary $\boldsymbol{D}$. The composed skill embedding is obtained by projecting the coefficient vector through the shared dictionary $\boldsymbol{m}_t = \boldsymbol{D}\bar{\boldsymbol{w}}_t$. Here, $\boldsymbol{m}_t$ is a context-dependent skill coordinate that summarizes which dictionary bases are used and with what continuous strengths.

Task-conditioned active subspaces. While the control policy operates on the continuous vector $\bar{\boldsymbol{w}}_t$ to maintain differentiability, we analyze the learned sparse structure by defining the active basis indices $\mathcal{K}_t$ via thresholding:

$$\mathcal{K}_t \triangleq \{k \mid |\bar{w}_{t,k}| > \varepsilon\}, \qquad |\mathcal{K}_t| \ll K, \tag{7}$$

where $\bar{w}_{t,k}$ is the k-th element of $\bar{\boldsymbol{w}}_t$. The threshold ε is used only for analysis. During execution, $\bar{\boldsymbol{w}}_t$ itself is not truncated; the soft tails are preserved for stable differentiable control. The index set $\mathcal{K}_t$ determines the active control subspace $\mathcal{U}_t \triangleq \mathrm{span}\{\boldsymbol{d}_k \mid k \in \mathcal{K}_t\} \subset \mathbb{R}^D$. We interpret $\mathcal{U}_t$ as the active linear control subspace induced by the learned sparse routing pattern, rather than as an assumed ground-truth task manifold. A central inductive bias of MOSAIC is that related task phases can share overlapping active supports, while incompatible dynamics are encouraged to use low-overlap supports over the same shared dictionary.

This mechanism reduces destructive coupling without task-specific parameter isolation. To see this, consider two task losses that locally depend on composed skill embeddings $\boldsymbol{m}_i = \boldsymbol{D}\boldsymbol{w}_i$ and $\boldsymbol{m}_j = \boldsymbol{D}\boldsymbol{w}_j$. The dictionary update induced by task i takes the form $\nabla_{\boldsymbol{D}}\mathcal{L}_i = \nabla_{\boldsymbol{m}_i}\mathcal{L}_i\, \boldsymbol{w}_i^\top$. Therefore, the first-order coupling between two dictionary updates is

$$\langle \nabla_{\boldsymbol{D}}\mathcal{L}_i, \nabla_{\boldsymbol{D}}\mathcal{L}_j \rangle_F = \big\langle \nabla_{\boldsymbol{m}_i}\mathcal{L}_i, \nabla_{\boldsymbol{m}_j}\mathcal{L}_j \big\rangle \cdot \langle \boldsymbol{w}_i, \boldsymbol{w}_j \rangle\,. \tag{8}$$

Thus, even though $\boldsymbol{D}$ is shared, sparse coefficients with low support overlap reduce dictionary-update coupling between incompatible tasks, while still allowing related tasks to reuse common bases.

Structured regularization objective. To encourage sparse and non-collapsed routing, we impose a variational information bottleneck on the inference network q_ψ. We minimize a KL-divergence penalty between the variational posterior parameterized by ψ and factorized priors:

$$\mathcal{L}_{\mathrm{KL}}(t) = \mathrm{KL}(q_\psi(\boldsymbol{b}_t, \boldsymbol{c}_t \mid \cdot) \,\|\, p(\boldsymbol{b}_t)p(\boldsymbol{c}_t))\,. \tag{9}$$

Specifically, the continuous prior $p(\boldsymbol{c}_t)$ is an isotropic Gaussian $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ that standardizes feature magnitudes. The binary prior $p(\boldsymbol{b}_t)$ is modeled as a product of Bernoulli distributions with a fixed target activation rate $\rho \ll 0.5$, encouraging sparse basis usage. To further sharpen basis selection, we explicitly penalize the ℓ_1-norm of the expected activations, $\|\bar{\boldsymbol{w}}_t\|_1$, which acts as a Lasso-style regularizer encouraging the agent to compose behavior from a small number of reusable bases. This formulation should be interpreted as structured regularization for selective sharing, rather than strict Bayesian inference over task distributions.

3.4 Hierarchical Control via Subspace Navigation

Sparse routing selects active control factors, but purely continuous mixtures may average over incompatible interaction modes. MOSAIC therefore decouples discrete mode selection from continuous modulation. The composed skill coordinate $\boldsymbol{m}_t$ conditions execution, while a discrete hierarchy selects the dominant basis-aligned option for the current interaction regime.

Basis-aligned option set. We define a set of temporal abstractions, $\Omega = \{\omega_k\}_{k=1}^{K}$, which stands in one-to-one alignment with the dictionary columns. Each option ω_k comprises a specialized low-level policy π_k and a termination condition β_k, associating each atomic basis $\boldsymbol{d}_k$ with a learnable control mode. This alignment trades maximal expressivity for interpretability and optimization stability.

Meta-policy for mode selection. At decision boundaries, a meta-policy chooses an option index based on the current state, task context, history, and composed skill embedding:

$$j_t \sim \pi_{\text{meta}}(j \mid \boldsymbol{z}_t, \boldsymbol{h}_\ell, \boldsymbol{m}_t, \boldsymbol{H}_t), \qquad j \in \{1, \dots, K\}. \tag{10}$$

This choice selects a dominant basis direction $\boldsymbol{d}_{j_t}$ and its associated option ω_{j_t}. The continuous embedding $\boldsymbol{m}_t$ then provides within-mode modulation for execution. Using a discrete option index promotes temporal persistence and reduces rapid oscillation between incompatible modes, which is difficult to enforce with continuous modulation alone.

Semi-Markovian persistence. Following the option framework, execution persists temporally. Let $\tau_{t+1} \sim \text{Bernoulli}(\beta_{j_t}(\boldsymbol{z}_{t+1}))$ be the termination signal. The option transition follows a semi-Markov decision process (SMDP):

$$j_{t+1} = \begin{cases} j_t, & \text{if } \tau_{t+1} = 0, \\ j \sim \pi_{\text{meta}}(\cdot \mid \boldsymbol{z}_{t+1}, \dots), & \text{if } \tau_{t+1} = 1. \end{cases} \tag{11}$$

This temporal abstraction stabilizes exploration and provides well-defined Bellman targets for off-policy learning.

3.5 Action Generation via Contextual Modulation

Feature-wise affine modulation. MOSAIC instantiates option-specific control using Feature-wise Linear Modulation (FiLM) (Perez et al., 2017) on a shared backbone. For the active option ω_{j_t}, we predict affine modulation parameters from the continuous skill coordinate $\boldsymbol{m}_t$ and the selected basis $\boldsymbol{d}_{j_t}$:

$$\boldsymbol{h}_t = f_{\text{trunk}}(\boldsymbol{z}_t) \in \mathbb{R}^C, \tag{12}$$

$$(\boldsymbol{\gamma}_{j_t}, \boldsymbol{\eta}_{j_t}) = \xi([\boldsymbol{m}_t, \boldsymbol{d}_{j_t}]) \in \mathbb{R}^C \times \mathbb{R}^C, \tag{13}$$

$$\tilde{\boldsymbol{h}}_{t,j_t} = \boldsymbol{\gamma}_{j_t} \odot \boldsymbol{h}_t + \boldsymbol{\eta}_{j_t}. \tag{14}$$

Here, ξ is a lightweight hyper-network that produces the FiLM parameters. Conditioning on both $\boldsymbol{m}_t$ and $\boldsymbol{d}_{j_t}$ lets the modulator separate global task and phase context from the chosen primitive, so the same basis can realize different variants under different contexts.

Policy and termination heads. We feed the modulated feature $\tilde{\boldsymbol{h}}_{t,j_t}$ to an option-specific Gaussian policy and a termination head:

$$a_t \sim \pi_{j_t}(a \mid \tilde{\boldsymbol{h}}_{t,j_t}), \qquad \beta_{j_t}(\boldsymbol{z}_t) = \sigma(g_\beta(\tilde{\boldsymbol{h}}_{t,j_t})). \tag{15}$$

In this design, j_t selects the dominant control mode through the dictionary-aligned option, and $\boldsymbol{m}_t$ provides continuous coordinates that refine execution within the selected mode.

4 Experiments

Our experiments are designed to test the central claim of MOSAIC: multi-task robot control can be improved through *selective basis sharing*. Unlike task-specific expert isolation (Cheng et al., 2025; Huang et al., 2025), MOSAIC lets all tasks access the same skill dictionary, while each context activates a sparse support over dictionary bases. This structure should allow related tasks to reuse control factors while reducing destructive coupling among incompatible dynamics. **Benchmarks.**

We evaluate on Meta-World (Yu et al., 2021). MT-10 tests controlled multi-task learning across ten manipulation tasks, while MT-50 evaluates scalability under a larger and more heterogeneous task set. See Appendix.B.2 for a description of the baseline environment. **Baselines.** We compare against four groups of methods: monolithic sharing methods (MT-SAC (Yu et al., 2021) and SAC (Haarnoja et al., 2018)), optimization-level interference mitigation methods (PCGrad (Yu et al., 2020) and GradNorm (Chen et al., 2018) with SAC), modular or routed policies (MENTOR (Huang et al., 2025) and SMT (Cho et al., 2024)), and generative policies (SDP (Wang et al., 2024)) when applicable. **Protocol.** Unless stated otherwise, all methods use the same observation modality, training budget, evaluation frequency, and random seeds. We report success rates averaged over the final 15% of training checkpoints (See Appendix B.3.1 for definitions), and additionally analyze support overlap, dictionary-update overlap, component ablations, basis interventions, and closed-loop latency. See the Appendix B.5 for parameter settings.

We organize the evaluation around five questions: 1. Does MOSAIC improve per-task success rates on Meta-World (Yu et al., 2021) MT-10 and MT-50? 2. Can standard gradient-level (Yu et al., 2020) remedies match MOSAIC? 3. Does MOSAIC induce low-overlap active supports and low-overlap dictionary-update footprints? 4. What components are required—sparsity, orthogonality, hierarchy, and recurrent history? 5. Does MOSAIC preserve single-pass closed-loop efficiency, and do learned bases have functional control roles?

4.1 Main Results on MT-10: Learning Dynamics and Task-level Reliability

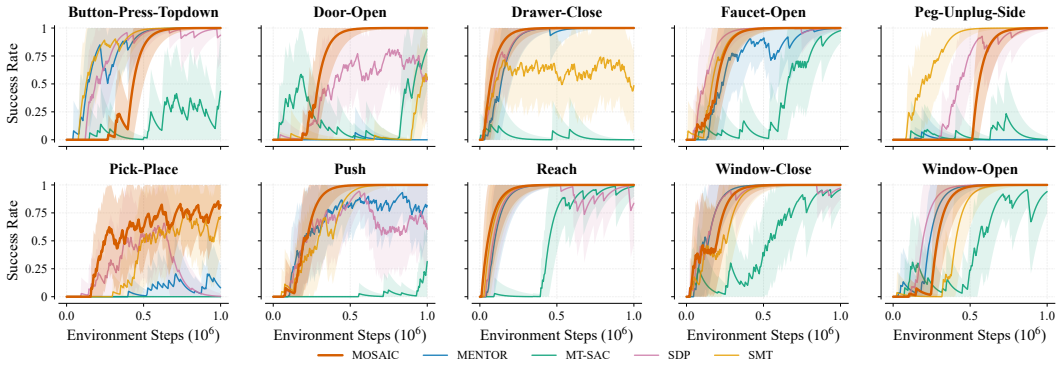


Figure 3: **MT-10 learning dynamics.** Each panel shows task success over training. MOSAIC exhibits more consistent convergence across tasks, while baselines often show uneven performance and persistent failures on hard tasks. For more experimental results, please refer to Appendix C.

We first evaluate MT-10 learning dynamics under the shared multi-task training protocol. Fig. 3 shows per-task success curves for the baselines and MOSAIC. MOSAIC converges more consistently across the task suite, while several baselines solve only a subset of tasks or show persistent tail-task failures, especially on contact-rich or long-horizon tasks such as Pick-Place, Door-Open, and Push.

Table 1: Compact aggregate results on Meta-World. † is an optimization-based diagnostic baseline rather than a structured policy-learning method.

Method	MT-10 ↑	MT-50 ↑
PCGrad-SAC† (Yu et al., 2020)	52.0±11.3	53.6±14.4
GradNorm-SAC† (Chen et al., 2018)	48.2±10.9	37.1±11.6
MT-SAC (Yu et al., 2021)	49.5±12.7	26.2±13.7
MENTOR (Huang et al., 2025)	69.1±5.4	67.9±25.2
SDP (Wang et al., 2024)	83.0±9.5	71.0±16.2
SMT (Cho et al., 2024)	84.8±10.1	71.3±23.5
MOSAIC	**98.3±2.3**	**90.4±12.8**

Tab.1 summarizes aggregate results on MT-10 and MT-50. PCGrad and GradNorm are included as optimization-level diagnostic baselines: they modify the training dynamics of a densely shared SAC policy but do not change the underlying sharing structure. MOSAIC achieves 98.3% on MT-10 and 90.4% on MT-50, outperforming PCGrad-SAC by 46.3 and 36.8 points, respectively. This suggests that post-hoc gradient correction alone is insufficient; the structure through which tasks share control factors is critical. The larger variance of several baselines on MT-50 indicates unstable convergence under heterogeneous task mixtures, whereas MOSAIC maintains higher mean performance with comparatively smaller variance. More experiments are provided in Appendix C.5.

4.2 Mechanism Ablations

We ablate the components associated with selective basis sharing: sparse support selection, dictionary direction separation, basis-aligned temporal execution, and recurrent phase inference. We also include a Hierarchy-only+FiLM control, which keeps temporal options and FiLM (Perez et al., 2017) but removes sparse dictionary routing, to test whether hierarchy alone explains the gains.

Table 2: **Ablation and mechanism-control results on MT-10.** Metric definitions are provided in Appendix.B.3.1. Gray numbers denote drops relative to full MOSAIC; red numbers denote increased interference gap. See Appendix C.1 for the hyperparameter analysis.

Metric	Full	w/o Sparsity	w/o Ortho	w/o Hierarchy	Hierarchy-only+FiLM	w/o RNN
Final Acc.	99.10%	87.4% (-11.7)	92.8% (-6.3)	76.2% (-22.9)	52.6% (-46.5)	90.3% (-8.8)
Norm. AUC	0.88	0.74 (-0.14)	0.79 (-0.09)	0.51 (-0.37)	0.56 (-0.31)	0.76 (-0.12)
Hard Task	95.70%	72.1% (-23.6)	77.5% (-18.2)	12.4% (-83.3)	24.0% (-71.7)	76.9% (-18.8)
Interf. Gap ↓	7.50%	16.9% (+9.4)	10.2% (+2.7)	22.1% (+14.6)	19.5% (+12.0)	13.1% (+5.6)

Tab.2 supports the proposed mechanism. Removing sparsity turns selective routing into denser basis mixing, increasing interference gap from 7.5% to 16.9% and reducing hard-task success from 95.7% to 72.1%. Removing orthogonality weakens basis separation and reduces both sample efficiency and hard-task success. Removing hierarchy causes severe hard-task collapse, consistent with mode averaging when continuous skill coordinates are used without discrete temporal execution. The Hierarchy-only+FiLM control remains far below full MOSAIC (52.6% vs. 99.1%), despite retaining options and FiLM. This shows that hierarchy alone is insufficient; the gain comes from coupling basis-aligned temporal execution with sparse dictionary routing. Removing the RNN also degrades performance, indicating that phase-aware context is important for selecting appropriate basis supports.

4.3 Closed-Loop Efficiency

We evaluate closed-loop control efficiency under a unified Meta-World protocol on a single RTX 3090 GPU. We report both frequency and latency metrics. Specifically, full-loop Hz measures end-to-end control frequency, including observation processing, policy inference, and environment stepping. We also report average loop latency and p95 latency to characterize temporal stability. As shown in Tab.3, MOSAIC achieves 531.2 Hz, comparable to MT-SAC (559.9 Hz) and SAC (566.5 Hz), while significantly outperforming SDP-based methods. Compared to SDP, MOSAIC reduces average loop latency from 4.631 ms to 1.883 ms, a 59.3% reduction. These results indicate that MOSAIC preserves high closed-loop efficiency without incurring the overhead of iterative inference. All reported frequencies are simulation-based and should be interpreted as relative comparisons rather than real-robot control rates.

Table 3: Closed-loop efficiency on Meta-World. MOSAIC retains near-SAC frequency without iterative inference.

Method	Full-Loop Hz ↑	Policy ms ↓	Loop ms ↓	p95 ms ↓
SDP (Wang et al., 2024)	215.9	3.117	4.631	6.049
Light-SDP (Wang et al., 2024)	292.7	1.924	3.416	4.507
MT-SAC (Yu et al., 2021)	559.9	0.272	1.787	2.463
SAC (Haarnoja et al., 2018)	566.5	0.262	1.766	2.421
MOSAIC	**531.2**	**0.397**	**1.883**	**2.450**

4.4 Basis and Option Usage Diagnostics

We analyze whether the shared dictionary is used in a task-structured, sparse, and temporally varying manner. Since MOSAIC does not assign task-specific parameters, selective sharing should appear as context-dependent basis usage rather than as dense mixing or a fixed one-task-one-basis assignment. Tab.4 indicates sparse and temporally varying basis usage. Basis-usage clusters align with task identity on MT-10 (NMI $= 0.762$), and phase-aware clustering is especially helpful on MT-50 ($0.388 \rightarrow 0.549$),

Table 4: Basis usage diagnostics. Phase-aware clustering preserves temporal segments within each rollout.

Metric	MT-10	MT-50
NMI(task, cluster) ↑	0.762	0.388
NMI(task, phase-aware) ↑	0.780	0.549
Dominant-basis switches ↑	10.9±2.0	20.2±3.2
Active bases / step ↓	1.10±0.05	1.74±0.54

suggesting that temporal phase information becomes more important as task diversity increases. Meanwhile, routing remains sparse (1.10 and 1.74 active bases per step on MT-10 and MT-50), and frequent dominant-basis switches show that MOSAIC does not collapse to dense mixing or a static one-task-one-basis pattern.

Fig. 4 visualizes task-level basis usage on MT-10. The task–basis heatmap shows that each task activates only a subset of the dictionary, while the task–task overlap matrix reveals partial support sharing rather than full dense coupling. This pattern supports the intended selective-sharing behavior: MOSAIC reuses common bases across tasks while maintaining task-dependent routing structure. To complement the aggregate and task-level diagnostics, Fig. 5 visualizes a representative successful peg-unplug-side rollout. The heatmap shows that dominant bases change across temporal segments, while the active support remains sparse within each segment. This pattern is consistent with Tab. 4 and Fig. 4, showing that MOSAIC uses phase-dependent routing rather than a fixed basis per task.

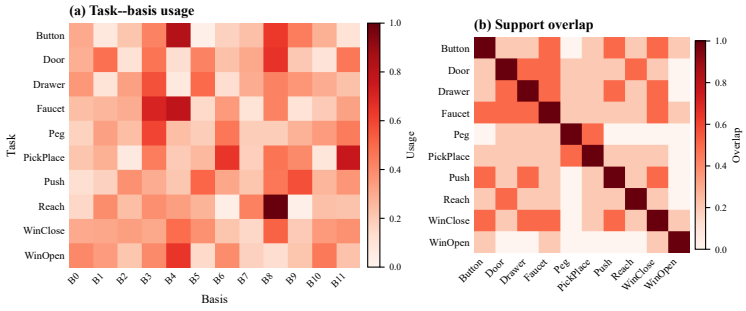


Figure 4: **Basis sharing on MT-10.** MOSAIC reuses partially overlapping basis subsets while preserving task-dependent routing.

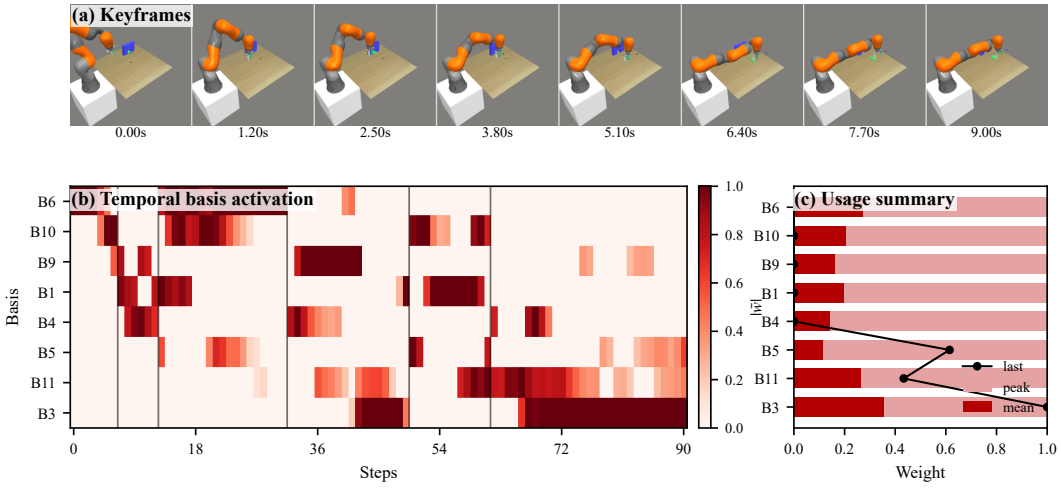


Figure 5: **Temporal basis activation.** (a) Keyframes show the progression of a successful Peg-Unplug-Side rollout. (b) Basis activations over time reveal sparse, phase-dependent routing. (c) Usage summary aggregates peak, mean, and terminal activations. The policy activates different bases across temporal segments while maintaining a small active support at each step, illustrating context-dependent composition rather than dense mixing or a static one-task-one-basis assignment.

5 Conclusion

We introduced MOSAIC, a structured multi-task reinforcement learning framework that replaces dense sharing with sparse basis sharing over a shared dictionary of atomic control bases. By inferring context-dependent sparse coefficients and executing them through basis-aligned temporal options, MOSAIC enables task- and phase-dependent sharing while reducing update coupling among incompatible dynamics. On Meta-World MT-10 and MT-50, MOSAIC achieves strong success rates (98.3% and 90.4%), preserves single-pass closed-loop efficiency, and exhibits sparse, task-structured routing. These results support sparse basis sharing as an effective mechanism for mitigating multi-task interference. See the Appendix.F for more information.

AI USE STATEMENT

AI tools were used solely for language polishing and improving the clarity of the manuscript. They were not used for data analysis, result generation, or scientific decision-making.

ETHICS STATEMENT

This study does not involve human participants, personal data, or animal experiments, and therefore does not require ethical approval.

REPRODUCIBILITY STATEMENT

The code used in this study will be made publicly available to facilitate reproducibility.

REFERENCES

Caixia Cai, Ying Siu Liang, Nikhil Somani, and Wu Yan. Inferring the Geometric Nullspace of Robot Skills from Human Demonstrations. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7668–7675, May 2020. doi: 10.1109/ICRA40945.2020.9197174.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International conference on machine learning*, pp. 794–803. PMLR, 2018.

Baiye Cheng, Tianhai Liang, Suning Huang, Maanping Shao, Feihong Zhang, Botian Xu, Zhengrong Xue, and Huazhe Xu. MoE-DP: An MoE-Enhanced Diffusion Policy for Robust Long-Horizon Robotic Manipulation with Skill Decomposition and Failure Recovery, November 2025.

Myungsik Cho, Jongeui Park, Suyoung Lee, and Youngchul Sung. Hard Tasks First: Multi-Task Reinforcement Learning Through Task Scheduling.

Myungsik Cho, Jongeui Park, Suyoung Lee, and Youngchul Sung. Hard tasks first: multi-task reinforcement learning through task scheduling. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 8556–8577, 2024.

Adria Colome and Carme Torras. Dimensionality Reduction for Dynamic Movement Primitives and Application to Bimanual Manipulation of Clothes. *IEEE Transactions on Robotics*, 34(3):602–615, June 2018. ISSN 1552-3098, 1941-0468. doi: 10.1109/TRO.2018.2808924.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures.

Fanny Ficuciello, Pietro Falco, and Sylvain Calinon. A Brief Survey on the Role of Dimensionality Reduction in Manipulation Learning and Control. *IEEE Robotics and Automation Letters*, 3(3): 2608–2615, July 2018. ISSN 2377-3766, 2377-3774. doi: 10.1109/LRA.2018.2818933.

Vedant Gupta, Haotian Fu, Calvin Luo, Yiding Jiang, and George Konidaris. Learning Parameterized Skills from Demonstrations, October 2025.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor, August 2018.

Ce Hao, Xuanran Zhai, Yaohua Liu, and Harold Soh. Abstracting Robot Manipulation Skills via Mixture-of-Experts Diffusion Policies, 2026.

Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. LEARNING AN EMBEDDING SPACE FOR TRANSFERABLE ROBOT SKILLS. 2018.

Jinmin He, Kai Li, Yifan Zang, Haobo Fu, Qiang Fu, Junliang Xing, and Jian Cheng. Not All Tasks Are Equally Difficult: Multi-Task Deep Reinforcement Learning with Dynamic Depth Routing. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(11):12376–12384, March 2024. ISSN 2374-3468, 2159-5399. doi: 10.1609/aaai.v38i11.29129.

Jinmin He, Kai Li, Yifan Zang, Haobo Fu, Qiang Fu, Junliang Xing, and Jian Cheng. Goal-Oriented Skill Abstraction for Offline Multi-Task Reinforcement Learning, July 2025.

Zhanpeng He and Matei Ciocarlie. Discovering Synergies for Robot Manipulation with Multi-Task Reinforcement Learning, June 2022.

Shengchao Hu, Ziqing Fan, Li Shen, Ya Zhang, Yanfeng Wang, and Dacheng Tao. HarmoDT: Harmony Multi-Task Decision Transformer for Offline Reinforcement Learning, May 2024.

S Huang, Z Zhang, T Liang, Y Xu, Z Kou, C Lu, et al. Mentor: mixture-of-experts network with task-oriented perturbation for visual reinforcement learning. 10.48550. *arXiv preprint arXiv.2410.14972*, 2025.

Jaekyeom Kim, Seohong Park, and Gunhee Kim. Unsupervised Skill Discovery with Bottleneck Option Learning, June 2021.

Hao Li, Qi Lv, Rui Shao, Xiang Deng, Yinchuan Li, Jianye Hao, and Liqiang Nie. STAR: Learning Diverse Robot Skill Abstractions through Rotation-Augmented Vector Quantization, June 2025.

Zhixuan Liang, Yao Mu, Hengbo Ma, Masayoshi Tomizuka, Mingyu Ding, and Ping Luo. SkillDiffuser: Interpretable Hierarchical Planning via Skill Abstractions in Diffusion-Based Task Execution. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16467–16476, Seattle, WA, USA, June 2024. IEEE. ISBN 979-8-3503-5300-6. doi: 10.1109/CVPR52733.2024.01558.

Alexandros Paraschos, Christian Daniel, Jan R Peters, and Gerhard Neumann. Probabilistic Movement Primitives.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual Reasoning with a General Conditioning Layer, December 2017.

Karl Pertsch, Youngwoon Lee, and Joseph J Lim. Accelerating Reinforcement Learning with Learned Skill Priors.

Jan Peters and Stefan Schaal. Policy learning for motor skills. In *International Conference on Neural Information Processing*, pp. 233–242. Springer, 2007.

En Yen Puang, Federico Ceola, Giulia Pasquale, and Lorenzo Natale. PCHands: PCA-based Hand Pose Synergy Representation on Manipulators with N-DoF. In *2025 IEEE-RAS 24th International Conference on Humanoid Robots (Humanoids)*, pp. 475–482, September 2025. doi: 10.1109/Humanoids65713.2025.11203193.

Krishan Rana, Ming Xu, Brendan Tidd, and Michael Milford. Residual Skill Policies: Learning an Adaptable Skill-based Action Space for Reinforcement Learning for Robotics.

Kolyan Ray, Botond Szabo, Gabriel Clara, et al. Spike and slab variational bayes for high dimensional logistic regression. In *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, 2020.

Jonas Rothfuss, Dennis Lee, Ignasi Clavera, Tamim Asfour, and Pieter Abbeel. ProMP: Proximal Meta-Policy Search, February 2022.

Matteo Saveriano, Fares J Abu-Dakka, Aljaž Kramberger, and Luka Peternel. Dynamic movement primitives in robotics: A tutorial survey. *The International Journal of Robotics Research*, 42(13): 1133–1184, November 2023. ISSN 0278-3649, 1741-3176. doi: 10.1177/02783649231201196.

Abdul-Saboor Sheikh and Jörg Lücke. Select-and-sample for spike-and-slab sparse coding. *Advances in Neural Information Processing Systems*, 29, 2016.

Weijie Shen, Yitian Liu, Yuhao Wu, Zhixuan Liang, Sijia Gu, Dehui Wang, Tian Nian, Lei Xu, Yusen Qin, Jiangmiao Pang, Xinping Guan, Xiaokang Yang, and Yao Mu. Expertise need not monopolize: Action-Specialized Mixture of Experts for Vision-Language-Action Learning, October 2025.

Jeevan Thapa and Rui Li. Bayesian Adaptation of Network Depth and Width for Continual Learning.

Samuele Tosatto, Jonas Stadtmueller, and Jan Peters. Dimensionality Reduction of Movement Primitives in Parameter Space, February 2020.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural Discrete Representation Learning, May 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Yixiao Wang, Yifei Zhang, Mingxiao Huo, Ran Tian, Xiang Zhang, Yichen Xie, Chenfeng Xu, Pengliang Ji, Wei Zhan, Mingyu Ding, and Masayoshi Tomizuka. Sparse Diffusion Policy: A Sparse, Reusable, and Flexible Policy for Robot Learning, October 2024.

Jiange Yang, Haoyi Zhu, Yating Wang, Gangshan Wu, Tong He, and Limin Wang. Tra-MoE: Learning Trajectory Prediction Model from Multiple Domains for Adaptive Policy Conditioning, April 2025.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2019.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in neural information processing systems*, 33: 5824–5836, 2020.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Avnish Narayan, Hayden Shively, Adithya Bellathur, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning, June 2021.

Likui Zhang, Tao Tang, Zhihao Zhan, Xiuwei Chen, Zisheng Chen, Jianhua Han, Jiangtong Zhu, Pei Xu, Hang Xu, Hefeng Wu, Liang Lin, and Xiaodan Liang. AtomicVLA: Unlocking the Potential of Atomic Skill Learning in Robots, March 2026.

APPENDIX

A TERMINATION-AWARE MAXIMUM-ENTROPY OBJECTIVES

This appendix provides the complete termination-aware maximum-entropy objectives used for the discrete meta-policy and option-value learning, consistent with the semi-MDP persistence in Eq. equation 11. We use $\alpha > 0$ for low-level SAC temperature and $\alpha_\Omega > 0$ for meta-level temperature.

A.1 OPTION-VALUE CRITIC WITH TERMINATION BRANCHING

Option-value function. We define an option-value critic $Q_\Omega(\boldsymbol{z}, j)$ over latent observation features $\boldsymbol{z}$ and option index $j \in \{1, \dots, K\}$. We also define the meta soft value

$$V_\Omega(\boldsymbol{z}) \triangleq \alpha_\Omega \log \sum_{j'=1}^{K} \exp\Big(\tfrac{1}{\alpha_\Omega} Q_\Omega(\boldsymbol{z}, j')\Big). \tag{16}$$

Termination-conditioned continuation value. When executing option j_t, the next option equals j_t if not terminated; otherwise a new option is selected by the meta-policy. This yields the termination-aware continuation value

$$U_\Omega(\boldsymbol{z}_{t+1}, j_t) \triangleq (1 - \beta_{j_t}(\boldsymbol{z}_{t+1}))\, Q_\Omega(\boldsymbol{z}_{t+1}, j_t) + \beta_{j_t}(\boldsymbol{z}_{t+1})\, V_\Omega(\boldsymbol{z}_{t+1}). \tag{17}$$

Eq. equation 17 is the key ingredient that resolves the ambiguity of "j_{t+1}" in bootstrapping and matches the semi-MDP semantics in Eq. equation 11.

Option-value Bellman target. We use the soft Bellman target

$$y_t^{\Omega} \;=\; r_t + \gamma(1-\mathrm{done}_{t+1})\, U_{\Omega}(\boldsymbol{z}_{t+1}, j_t), \tag{18}$$

where $\mathrm{done}_{t+1} \in \{0,1\}$ indicates episode termination.

Critic loss. The option-value critic is trained by regression:

$$\mathcal{L}^{\Omega}_{\mathrm{critic}} = \mathbb{E}\big[\big(Q_{\Omega}(\boldsymbol{z}_t, j_t) - y_t^{\Omega}\big)^2\big]. \tag{19}$$

In practice, Q_{Ω} uses a target network $\bar{Q}_{\Omega}$ for bootstrapping and Polyak averaging.

A.2 Discrete Maximum-Entropy Meta-Policy

Meta-policy actor objective. Given the critic Q_{Ω}, we optimize a discrete maximum-entropy policy over options. Let $\boldsymbol{x}_t \triangleq (\boldsymbol{z}_t, \boldsymbol{h}_{\ell}, \boldsymbol{m}_t, \boldsymbol{H}_t)$ denote the meta-policy input. We use the standard soft actor-critic style objective for discrete actions:

$$\mathcal{L}^{\Omega}_{\mathrm{actor}} = \mathbb{E}\Big[\sum_{j=1}^{K} \pi_{\mathrm{meta}}(j \mid \boldsymbol{x}_t)\Big(\alpha_{\Omega} \log \pi_{\mathrm{meta}}(j \mid \boldsymbol{x}_t) - Q_{\Omega}(\boldsymbol{z}_t, j)\Big)\Big]. \tag{20}$$

This objective minimizes KL divergence to the Boltzmann distribution induced by Q_{Ω} and avoids any separate hand-written entropy term in the return (no double-counting).

A.3 Termination Loss

Termination advantage. Define the termination advantage

$$A^{\mathrm{term}}(\boldsymbol{z}_t, j) \;\triangleq\; Q_{\Omega}(\boldsymbol{z}_t, j) - V_{\Omega}(\boldsymbol{z}_t). \tag{21}$$

Termination objective. We learn $\beta_j(\boldsymbol{z})$ to terminate when switching is beneficial, i.e., when $A^{\mathrm{term}}(\boldsymbol{z}_t, j)$ is low. An implementable termination loss that matches the option-critic gradient direction is:

$$\mathcal{L}_{\mathrm{term}} = \mathbb{E}\Big[\beta_{j_t}(\boldsymbol{z}_t)\; A^{\mathrm{term}}(\boldsymbol{z}_t, j_t)\Big], \tag{22}$$

where gradients flow through $\beta_{j_t}(\boldsymbol{z}_t)$ (the critic is treated as a baseline for this update). Minimizing Eq. equation 22 increases termination probability when A^{term} is negative (switching preferred), and decreases it when persisting is advantageous.

A.4 Low-Level SAC (Per-Option) with FiLM-Conditioned Features

For completeness, we summarize the low-level maximum-entropy objective. Let $\tilde{\boldsymbol{h}}_{t,j_t}$ be the FiLM-modulated feature in Eq. equation 14. We train per-option low-level critics $Q_j(\tilde{\boldsymbol{h}}, a)$ and policies $\pi_j(a \mid \tilde{\boldsymbol{h}})$ with standard SAC. The actor loss for the executed option j_t is

$$\mathcal{L}^{\mathrm{low}}_{\mathrm{sac}} = \mathbb{E}\big[\alpha \log \pi_{j_t}(a_t \mid \tilde{\boldsymbol{h}}_{t,j_t}) - Q_{j_t}(\tilde{\boldsymbol{h}}_{t,j_t}, a_t)\big], \tag{23}$$

and the critic loss regresses to the soft target using the next option determined by Eq. equation 11. (We omit the standard target-network details for brevity.)

B Implementation details.

B.1 Training Objective and Optimization

Composite structure regularization. MOSAIC is trained end-to-end. We introduce a composite structural regularization term $\mathcal{L}_{\mathrm{struct}}$ to enforce the geometric inductive biases (sparsity, orthogonality, and valid posterior):

$$\mathcal{L}_{\mathrm{struct}}(t) = \beta_{\mathrm{KL}}\, \mathcal{L}_{\mathrm{KL}}(t) + \lambda_{\mathrm{ortho}}\, \mathcal{L}_{\mathrm{ortho}}(\boldsymbol{D}) + \lambda_{\mathrm{sp}}\, \|\bar{\boldsymbol{w}}_t\|_1. \tag{24}$$

Here, the ℓ_1 penalty explicitly encourages the agent to compose behaviors from a minimal set of atomic primitives, complementing the distributional KL constraint. Specifically, $\mathcal{L}_{\mathrm{KL}}$ prevents posterior collapse by anchoring to the prior; $\mathcal{L}_{\mathrm{ortho}}$ prevents the dictionary from degenerating into similar bases. While this introduces multiple coefficients, we find in practice that they act orthogonally and are robust to hyperparameter settings.

Hierarchical Soft Actor-Critic. We optimize the control policy using a hierarchical maximum-entropy objective. This comprises: (i) Low-level SAC: Standard continuous SAC updates for option policies $\{\pi_k\}$ and their critics; (ii) Meta-level Discrete SAC: A discrete soft actor-critic objective for the meta-policy π_{meta} and option-value function Q_Ω; and (iii) Termination Gradient: An option-termination loss derived from the Option-Critic theorem. Crucially, MOSAIC does not require a specialized optimization algorithm; it introduces a structured parameterization that renders complex multi-task control tractable via standard RL.

The total loss aggregates the hierarchical control errors (via Hierarchical SAC) and this structure penalty:

$$\mathcal{L}_{\mathrm{total}} = \underbrace{\mathcal{L}_{\mathrm{SAC}}^{\mathrm{low}} + \mathcal{L}_{\mathrm{disc}}^{\mathrm{meta}} + \mathcal{L}_{\mathrm{term}}}_{\text{Hierarchical Control}} + \lambda_{\mathrm{struct}} \sum_{t \in \mathcal{B}} \mathcal{L}_{\mathrm{struct}}(t). \tag{25}$$

Gradients are backpropagated jointly through the encoders θ, inference network ψ, and dictionary $\boldsymbol{D}$, aligning the representation with control requirements. We optimize Eq. 25 with a Hierarchical Soft Actor-Critic (H-SAC) objective adapted to semi-Markov decision processes (SMDPs). To support temporally extended options, we use an SMDP-aware replay buffer that stores transition-specific termination flags τ_{t+1} for correct bootstrapping, as shown in Sec.3.4

SMDP-aware replay mechanism. To support correct off-policy updates for the semi-Markov hierarchy, the replay buffer stores augmented transitions: $(\boldsymbol{o}_t, \boldsymbol{z}_t, \boldsymbol{H}_t, \boldsymbol{h}_\ell, \boldsymbol{m}_t, j_t, a_t, r_t, \mathrm{done}, \tau_{t+1})$, where τ_{t+1} is the realized boolean termination flag. This allows the critic target computation to correctly branch between bootstrapping (if $\tau_{t+1} = 0$) and meta-policy resampling (if $\tau_{t+1} = 1$), ensuring consistent Bellman estimates.

B.2 Benchmark Descriptions

This work adopts two established robotic manipulation benchmarks from Meta-World Yu et al. (2021) to evaluate the generalization and sample efficiency of meta-reinforcement learning algorithms in multi-task settings.

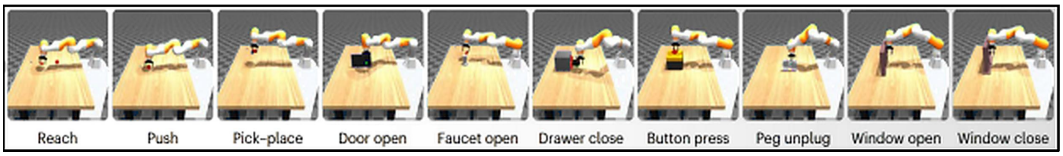
Meta-World MT-10. To study embodiment generalization and real-world relevance, we adopt the community extension of Meta-World Yu et al. (2019). It contains ten canonical manipulation tasks, including `Reach`, `Push`, `PickPlace`, `DoorOpen`, and `WindowClose`.

Meta-World MT-50. The **MT-50** benchmark comprises 50 distinct manipulation tasks designed around the Sawyer robot environment. Tasks span diverse categories, including object interaction (e.g., `Push`, `PickPlace`, `Hammer`), articulated mechanism operation (e.g., `DoorOpen`, `FaucetOpen`, `WindowClose`), and precision-focused behaviors (e.g., `ButtonPress`, `PegUnplug`, `DialTurn`). This diversity stresses an algorithm across varied state–action spaces, reward structures, and physics constraints, making MT-50 a stringent testbed for multi-task and meta-RL.

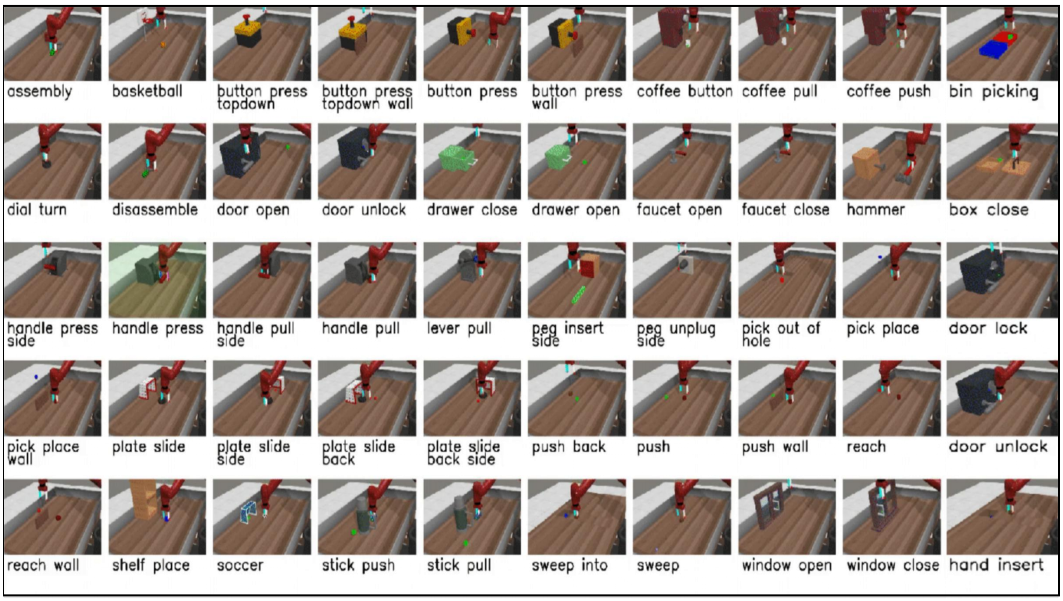
Both benchmarks provide structured, reproducible environments with sparse rewards, proprioceptive and object-state observations, and scripted success detectors. They are widely recognized in the meta-RL community for assessing task complexity, generalization breadth, and adaptation speed.

B.3 Training and Evaluation Protocol

We train MOSAIC with Soft Actor-Critic (SAC) (Haarnoja et al., 2018) under the same training budget and evaluation protocol as all baselines. During training, we periodically evaluate the current policy at a set of checkpoints and compute success rates by averaging over all tasks in the suite and N_{eval} evaluation episodes per task.



(a) Meta-World MT-10



(b) Meta-World MT-50

Figure 6: Visualization of Meta-World manipulation benchmarks. (a) Representative tasks from MT-10. (b) A subset of tasks from MT-50 demonstrating diverse manipulation challenges. Both benchmarks feature robotic manipulators interacting with task-specific objects in simulated physics environments.

B.3.1 METRICS

Evaluation. During training, we evaluate the policy at N checkpoints $\{step_i\}_{i=1}^{N}$, where $step_i$ is the environment step and the total training budget is $S_{\max}$ steps. At each checkpoint we compute a suite-level success rate $SR_i \in [0,1]$ (defined below). Let $\mathcal{I}_{15} = \{\, i \mid step_i \geq 0.85\, S_{\max} \,\}$ denote the index set of checkpoints within the last 15% of training.

Task-averaged success at a checkpoint. Let $\widehat{SR}_t(step_i) \in [0,1]$ be the success rate of task $t \in \{1,\dots,T\}$ at checkpoint $step_i$, computed as the fraction of successes over N_{eval} evaluation episodes. In our experiments, $AllSteps{=}10^6$ for MT-10 and $AllSteps{=}10^7$ for MT-50. We evaluate every $10{,}000$ environment steps during training. We define the suite-level success as the mean across tasks:

$$SR_i \;=\; \frac{1}{T}\sum_{t=1}^{T} \widehat{SR}_t(step_i). \tag{26}$$

Primary metrics. We report two complementary suite-level metrics:

$$\textbf{TrainAvg} \;=\; \frac{1}{N}\sum_{i=1}^{N} SR_i, \qquad \textbf{Final-15\%} \;=\; \frac{1}{|\mathcal{I}_{15}|}\sum_{i\in\mathcal{I}_{15}} SR_i. \tag{27}$$

Unless otherwise stated, **Final-15%** is used as the primary metric for final performance, while **TrainAvg** summarizes learning efficiency over the full training horizon (i.e., a discrete AUC of the suite-success curve).

Hard-task success (tail performance). Let $\mathcal{H} \subseteq \{1,\dots,T\}$ denote a fixed set of hard tasks (defined below). We report tail success on hard tasks as

$$\text{HardSuccess} \;=\; \frac{1}{|\mathcal{H}|}\sum_{t\in\mathcal{H}} \left(\frac{1}{|\mathcal{I}_{15}|}\sum_{i\in\mathcal{I}_{15}} \widehat{SR}_t(step_i) \right). \tag{28}$$

Interference Gap (negative-transfer severity). To quantify destructive interference, we compare each task's tail success under multi-task (MT) training against a single-task (ST) reference trained with the same budget. For seed s and task t, we define the tail success as

$$\text{TailSR}^{\text{MT}}_{s,t} \;=\; \frac{1}{|\mathcal{I}_{15}|}\sum_{i\in\mathcal{I}_{15}} \widehat{SR}^{\text{MT}}_{s,t}(step_i), \qquad \text{TailSR}^{\text{ST}}_{s,t} \;=\; \frac{1}{|\mathcal{I}_{15}|}\sum_{i\in\mathcal{I}_{15}} \widehat{SR}^{\text{ST}}_{s,t}(step_i). \tag{29}$$

Instead of counting only whether a task is interfered, we measure the excess negative-transfer gap beyond a tolerance δ:

$$G_{s,t} \;=\; \max\Big(0,\; \text{TailSR}^{\text{ST}}_{s,t} - \text{TailSR}^{\text{MT}}_{s,t} - \delta\Big). \tag{30}$$

We report the average interference gap over all seeds and tasks:

$$\text{IG} \;=\; \frac{1}{ST}\sum_{s=1}^{S}\sum_{t=1}^{T} G_{s,t}. \tag{31}$$

We use $\delta = 0.05$ absolute success rate in all experiments. Lower IG indicates less severe negative transfer relative to the corresponding single-task references.

Hard-task set. When single-task references are unavailable (e.g., for our MT-50 summary table), we construct $\mathcal{H}$ based on baseline difficulty. For each task t, let

$$B_t = \max\{\text{MENTOR}_t, \text{MT-SAC}_t, \text{SDP}_t, \text{SMT}_t\}$$

be the strongest tail performance (i.e., **Final-15%** computed per task) among the non-MOSAIC baselines reported in the same table. We include task t in $\mathcal{H}$ if $B_t < 70\%$, yielding $|\mathcal{H}| = 15$ hard tasks listed in Table 5.

All metrics are reported as mean$\pm$std over 5 random seeds under the same evaluation protocol.

Table 5: Hard-task set $\mathcal{H}$ for Meta-World MT-50, selected by $B_k < 70\%$ where B_k is the strongest non-MOSAIC baseline success on task k .

Hard tasks				
CloseBox	CloseDoor	CloseWindow	GetCoffee	InsertHand
Pick&Place	PressButtonTop	PressButtonTopWall	PullHandleSide	Push
PushWithWall	RetrievePlate	SlidePlate	SlidePlateSide	UnplugPeg

B.4 HYPERPARAMETERS

Table 6 summarizes the key method hyperparameters introduced in our formulation, including the dictionary size K, the temporal context settings (D_H, T_{ctx}), the structural regularization coefficients in Eq. equation 25 and Eq. equation 24, and the SAC optimization hyperparameters. Unless otherwise noted (e.g., in ablations), we hold these hyperparameters fixed across MT-10 and MT-50. **Notes.** ϵ is used only for analysis to define the active basis set and does not affect inference. T_{ctx} is the BPTT unroll length used to train the history model.

Table 6: Hyperparameters for MOSAIC.

Hyperparameter	**Symbol / Key**	**Value**
Architecture (MOSAIC)		
Dictionary size	K	12 (MT-10) / 60 (MT-50)
VAE hidden dim / layers	–	400 / 3
Latent feature dim	D	10 / 100
Actor hidden dim / layers	–	256 / 4
Critic hidden dim / layers	–	256 / 4
Policy log-std bounds	–	$[-20, 2]$
Temporal context		
History state	$\boldsymbol{H}_t$	GRU
History dim	D_H	256
Unroll length (BPTT)	T_{ctx}	10
Active-set threshold	ε	0.05
Structure regularization (Eq.equation 25–equation 24)		
KL weight	β_{KL}	10^{-3}
Orthogonality weight	λ_{ortho}	10^{-2}
Sparsity (ℓ_1) weight	λ_{sp}	10^{-3}
Overall structure weight	λ_{struct}	1.0
RL optimization (SAC)		
Discount	γ	0.99
Target smoothing	τ	0.005
Init temperature	α_0	1.0
Batch size	–	512
Replay capacity	–	10^7
Optimizer	–	Adam, lr $3 \cdot 10^{-4}$
Init random steps	–	1500
Eval frequency	–	every 10,000 steps
Eval episodes	N_{eval}	10

B.5 HARDWARE AND SOFTWARE

For training MOSAIC and running the experiments, we had access to two workstations with 8 $\times$ GeForce RTX 3090, 1 $\times$ L40 and 2 $\times$ GeForce RTX 4090. It took on average 18.7 hours on these machines to run 5 seeds for each one of the baselines and our method. We estimate that the total compute cost for the project was approximately 1205 GPU-hours, including experimentation and testing.

C MORE EXPERIMENTS.

C.1 HYPERPARAMETER ANALYSIS

A key design choice in MOSAIC is the atomic skill dictionary size K. If K is too small, the dictionary cannot span the behavior space (underfitting); if K is too large, it may introduce redundancy or optimization issues. We therefore evaluate the *asymptotic* success rate on Meta-World MT-10 and MT-50 over a wide range $K \in [2, 128]$, keeping all other hyperparameters fixed. As shown in Fig. 7, the curves indicate that K mainly controls representational capacity, and MOSAIC is not sensitive to precise tuning. On **MT-10** (left), small dictionaries underfit ($\sim 45\%$ at $K = 2$, $\sim 70\%$

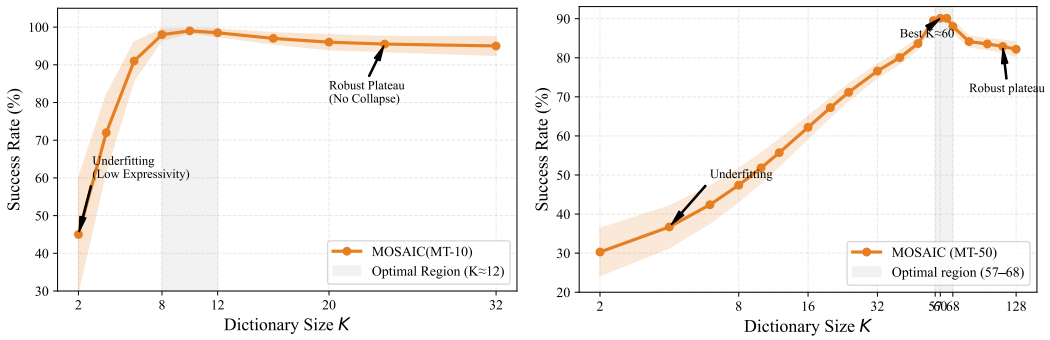


Figure 7: **Parameter Sensitivity: Dictionary Size K on MT-10 and MT-50.** The shaded region is the standard deviation across 5 seeds.

at $K = 4$, $\sim 90\%$ at $K = 6$). Performance peaks at $\sim$ 98–99% for $K \in [8, 12]$ with reduced variance, suggesting that MT-10 is well covered by a compact set of shared primitives. For $K \geq 12$, performance saturates without degradation, consistent with the sparsity bottleneck $\mathcal{L}_{\text{sp}}$ and the orthogonality regularizer $\mathcal{L}_{\text{ortho}}$ mitigating harmful over-parameterization.

On **MT-50** (right), capacity demands are higher and improvements are more gradual: success rises from $\sim 30\%$ at $K = 2$ to $\sim 89\%$ near $K \approx 56$, and peaks around $\sim 90\%$ at $K \approx 60$. The interval $K \in [57, 68]$ is near-optimal with relatively low variability. For larger dictionaries ($K = 80$–128), success decreases mildly to $\sim$ 82–85% but remains stable, indicating a broad high-performance regime rather than collapse.

Overall, the optimum shifts from $K \in [8, 12]$ (MT-10) to $K \approx 60$ (MT-50), while both benchmarks exhibit a wide overcomplete regime with stable performance. In practice, extra bases tend to remain unused rather than inducing overfitting, reinforcing that MOSAIC supports a robust operating range for K.

C.2 VISUAL ANALYSIS

In this section, we provide a qualitative evaluation of **MOSAIC** on the Meta-World MT-10 and MT-50 benchmarks. These visualizations complement the quantitative results in the main paper and offer additional insight into the learned strategies and their effectiveness. For each task, we use rollouts from the final policy checkpoint and render a filmstrip of eight representative frames. The sequence starts from initialization at $t = 0$, includes intermediate frames with timestamps to show progress and interaction, and ends at the goal state. The final frame is highlighted with a colored border to indicate successful completion.

C.3 MT-50 CROSS-TASK BASIS SHARING ANALYSIS

We further visualize cross-task basis usage on MT-50 to examine whether selective basis sharing persists at larger task scale. For each rollout, we compute the mean absolute expected basis coefficient $\frac{1}{T}\sum_{t=1}^{T}|\bar{\boldsymbol{w}}_t|$ and average it over rollouts of the same task. Fig. 8 shows the resulting task–basis usage matrix. The activation pattern is sparse and heterogeneous: tasks activate different subsets of bases, while several bases are reused across multiple tasks. This suggests that MOSAIC does not collapse to a dense shared representation or to a strict one-task-one-basis assignment.

To quantify support sharing between tasks, we also compute a task–task overlap matrix from the top active bases of each task. As shown in Fig. 9, the diagonal entries are high by definition, whereas off-diagonal entries are generally sparse and structured. Some task pairs exhibit moderate overlap, consistent with reusable control factors across related manipulation behaviors, while many pairs maintain low overlap. These results complement the aggregate statistics in Table 4 and support the interpretation that MOSAIC scales selective basis sharing to MT-50.

Compared with MT-10, MT-50 exhibits more diffuse support because the task set is larger and more heterogeneous. Nevertheless, basis usage remains far from uniform: the learned dictionary is reused selectively rather than activated densely for all tasks.

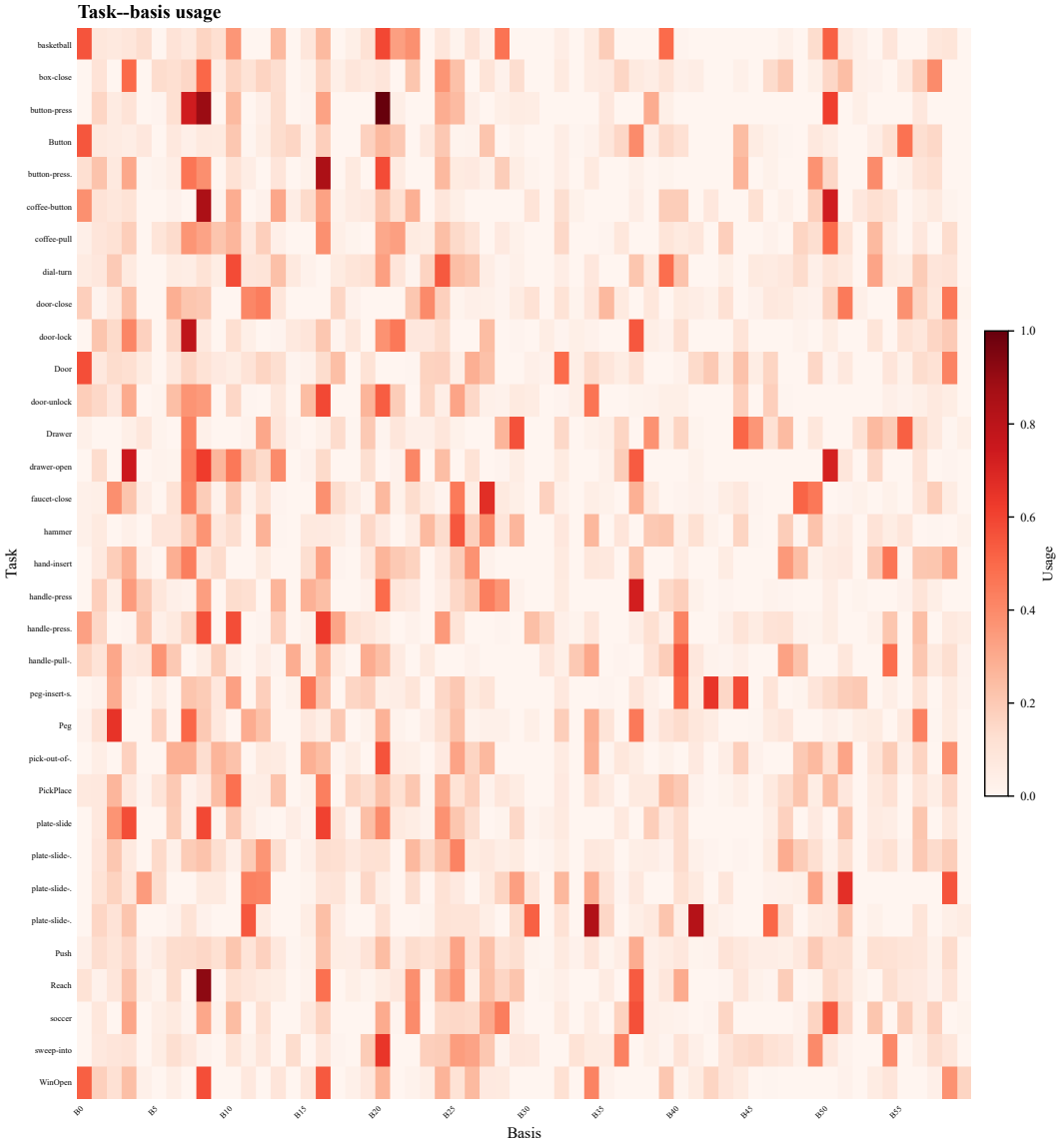


Figure 8: **Task–basis usage on MT-50.** Each row corresponds to a task and each column to a learned basis. Values denote normalized mean absolute basis usage averaged over rollouts. The sparse and heterogeneous activation pattern indicates that MT-50 tasks use different subsets of the shared dictionary while still reusing some bases across tasks. Task names are abbreviated for readability.

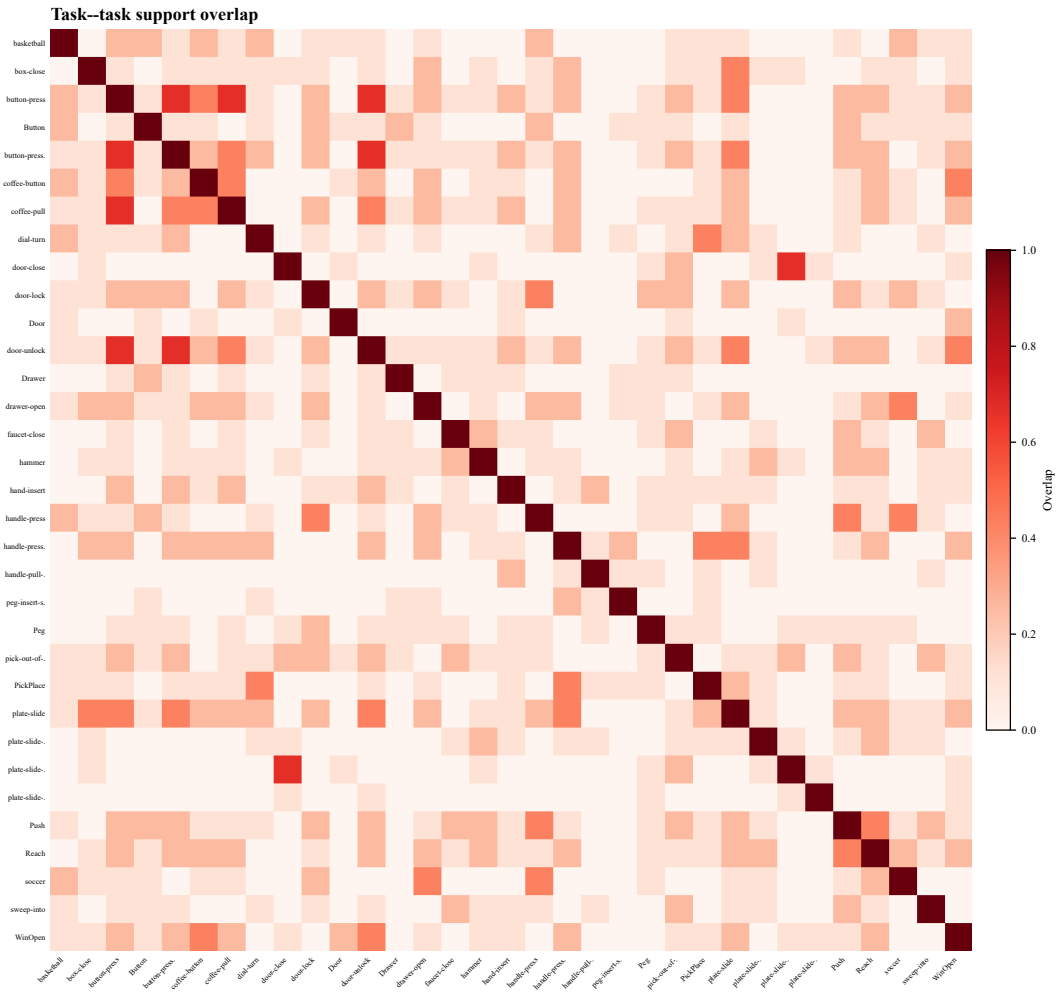


Figure 9: **Task–task support overlap on MT-50.** Overlap is computed from the top active bases of each task. The matrix shows structured partial sharing: related tasks can share basis support, while many task pairs maintain low overlap. This pattern is consistent with selective basis reuse rather than dense coupling across all tasks.

C.4 More analysis on MT-10

The Sec.4 reports training curves for qualitative comparison of learning dynamics. For completeness, Tab.7 provides the corresponding final per-task success rates, averaged over the final 15% training checkpoints.

Table 7: **Per-task success rate on Meta-World MT-10.** Results are reported as mean ± mean band width over the final training checkpoints.

Task	MENTOR	MT-SAC	SDP	SMT	MOSAIC
Button-Press-Topdown	98.7±3.2	30.1±29.5	98.6±4.0	**100.0±0.0**	**100.0±0.0**
Door-Open	0.4±1.3	70.3±28.6	65.6±35.6	32.3±27.1	**100.0±0.0**
Drawer-Close	**100.0±0.0**	0.3±1.2	**100.0±0.0**	61.0±36.8	**100.0±0.0**
Faucet-Open	97.6±6.5	97.6±6.3	99.2±2.7	**100.0±0.0**	**100.0±0.0**
Peg-Unplug-Side	0.0±0.1	1.5±3.6	**100.0±0.0**	**100.0±0.0**	**100.0±0.0**
Pick-Place	11.3±17.2	0.0±0.1	1.6±5.1	54.5±36.7	**82.8±22.8**
Push	84.8±20.8	18.1±23.2	76.9±27.7	**100.0±0.0**	**100.0±0.0**
Reach	**100.0±0.0**	97.5±6.8	92.0±10.9	**100.0±0.0**	**100.0±0.0**
Window-Close	97.9±4.9	94.0±9.7	96.0±8.7	**100.0±0.0**	**100.0±0.0**
Window-Open	**100.0±0.0**	85.6±17.5	**100.0±0.0**	**100.0±0.0**	**100.0±0.0**
Average	69.1±5.4	49.5±12.7	83.0±9.5	84.8±10.1	**98.3±2.3**

Although the main paper reports success-rate curves for the primary comparison, the reward curves provide a complementary view of learning and training stability across methods. As shown in the Fig. 10 shows the training reward curves for each Meta-World MT-10 task.

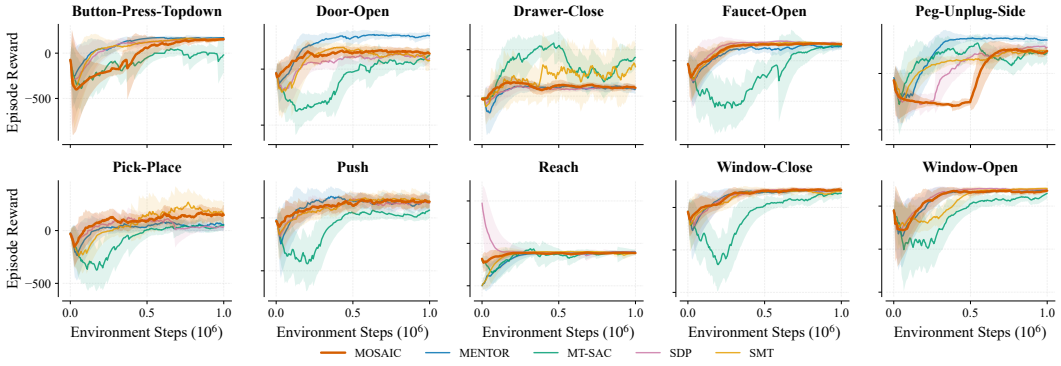


Figure 10: Per-task training reward curves on Meta-World MT-10. The figure provides a qualitative view of learning dynamics and training stability for MOSAIC and baseline methods.

On most tasks, MOSAIC achieves more stable reward improvement with significantly reduced oscillation compared to the baseline methods. This advantage is particularly pronounced in long-horizon tasks involving delayed rewards, such as Pick-Place and Window-Open, where MOSAIC consistently attains higher and more monotonic reward growth. In contrast, several baseline exhibit marked training instability or remain trapped in low-reward regimes for prolonged periods, even when eventual task success is achieved. We emphasize that reward curves are presented here solely as supplementary evidence: absolute reward values are task-specific and do not necessarily correlate directly with objective measures of task completion quality.

C.5 More analysis on MT-50

C.5.1 Per-Task Analysis

This section reports the full training curves for all 50 tasks in Meta-World MT-50. Figs. 11 and 12 show per-task success rates and episode returns over 10^7 environment steps. We compare **MOSAIC** to strong multi-task baselines: MENTOR, MT-SAC, SDP, and SMT.

Overall, MOSAIC learns faster and reaches higher final performance on most tasks. The gains are clearest on precision-heavy manipulation, including *Pick-and-Place*, *Insert-Peg*, and *Hammer*, where MOSAIC surpasses diffusion policies (SDP) and other modular methods (SMET, MENTOR). MOSAIC also remains reliable across heterogeneous mechanics, from simple kinematics (*Reach*) to articulated interactions (*Open Door*, *Turn Faucet*). This supports our claim that a structured shared skill subspace can capture task-specific geometry while reducing negative transfer. Finally, on high-difficulty tasks where baselines often show high variance or collapse (e.g., *Assembly*, *Disassemble Nut*), MOSAIC stays stable throughout training, consistent with the regularization effect of the geometric skill dictionary.

Tab.8 provides the granular per-task success rates for MOSAIC compared to the strongest baseline (SDP) and the monolithic baseline (MT-SAC). MOSAIC achieves consistently high performance across diverse task families, particularly excelling in contact-rich manipulation tasks (e.g., `insert-peg-side`, `pick-place-wall`) where baselines often fail.

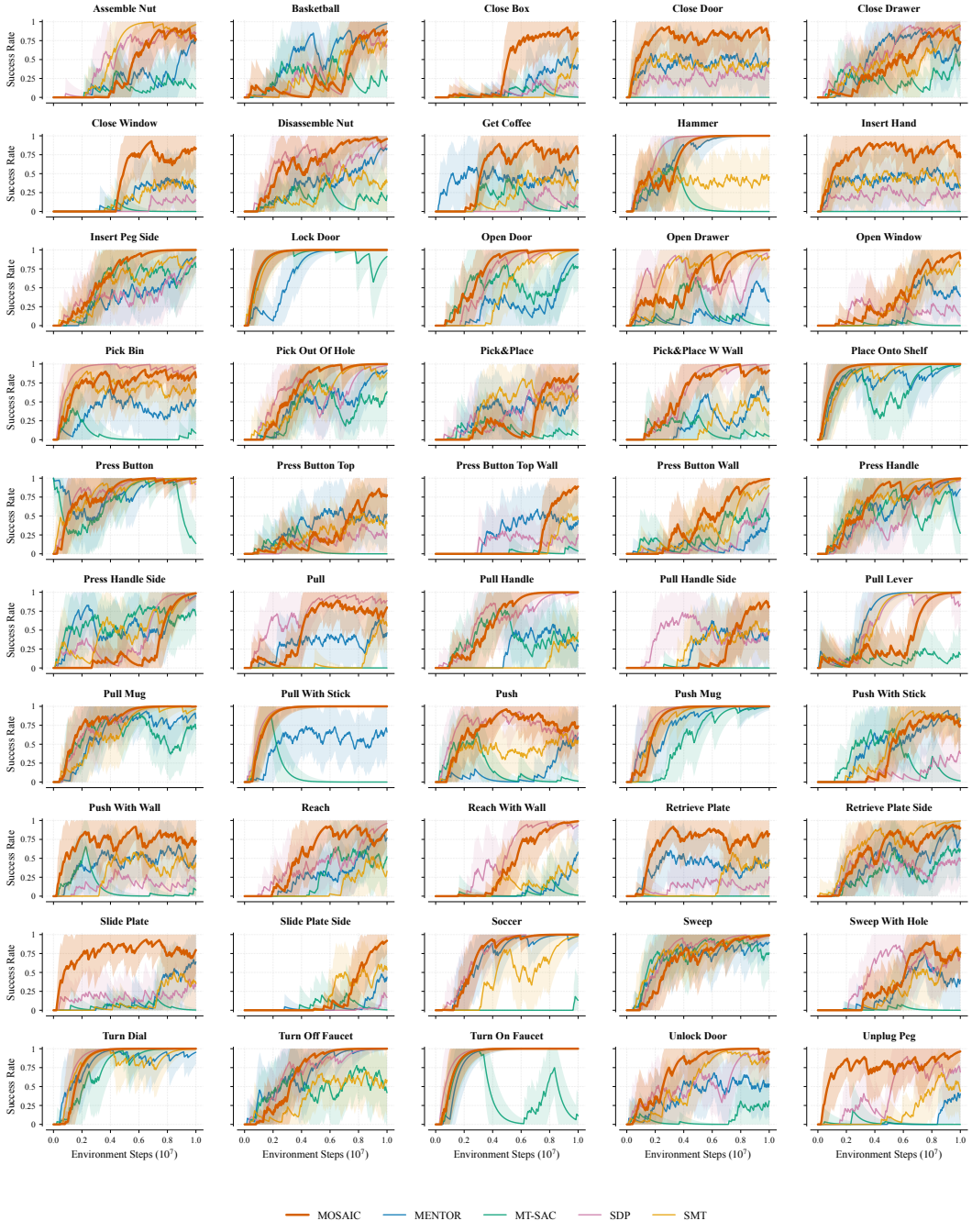


Figure 11: Per-task success rates on Meta-World MT-50. The curves show the mean over 5 seeds, and shaded regions indicate the standard error. Across tasks, **MOSAIC** (orange) typically learns faster and reaches a higher final success rate than the baselines. It also solves a number of tasks where other multi-task methods make little progress.

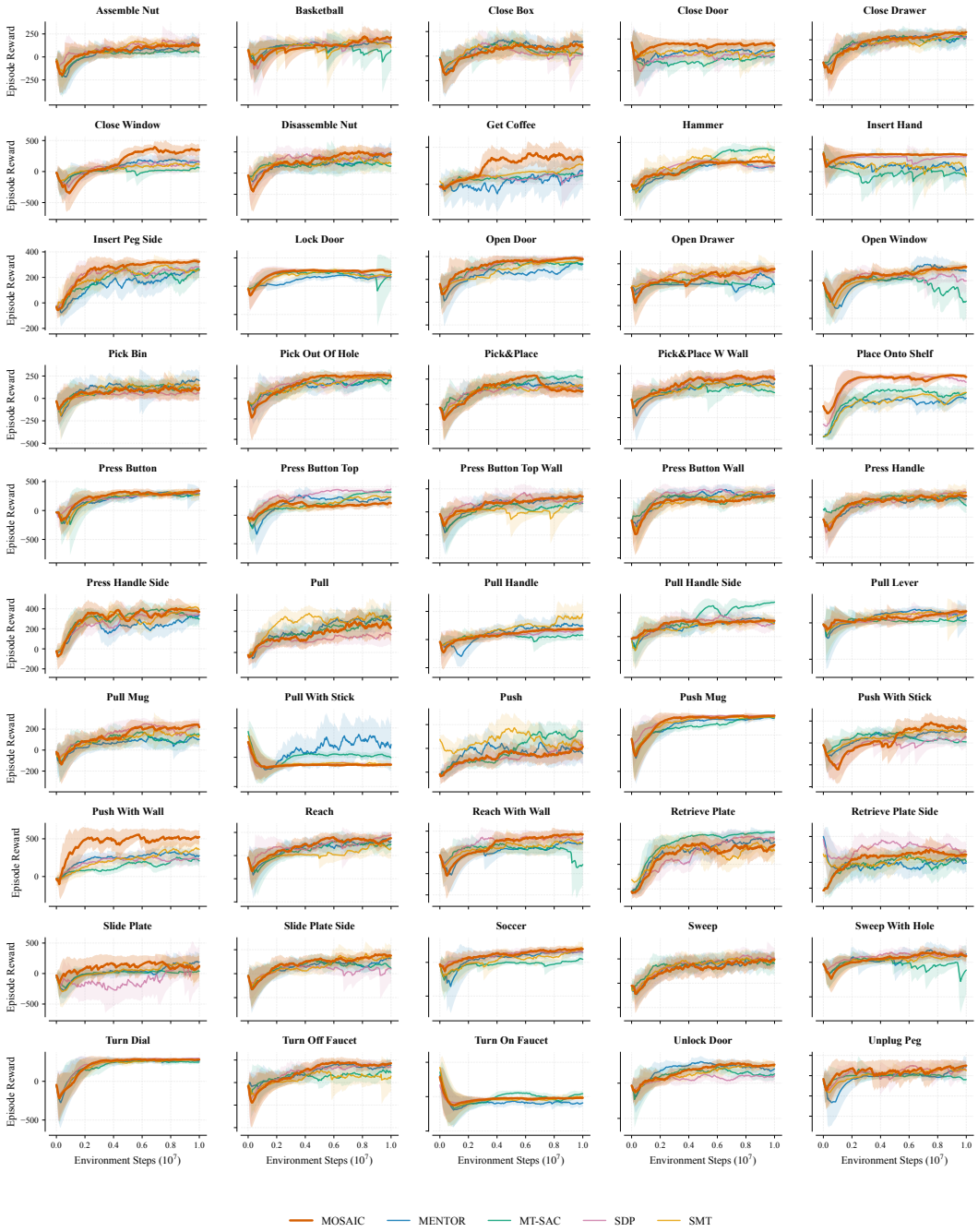


Figure 12: Per-task episode rewards on Meta-World MT-50. The curves report the mean over 5 seeds, with shaded bands showing the standard error. MOSAIC (orange) generally improves returns earlier and finishes higher than the baselines, with the largest gains on contact-rich tasks such as *Pick-Place*, *Push*, and *Door Open*.

Table 8: Final Success Rates on MT-50. The mean success rate over the final 15% of training steps (mean ± std).

Task	MENTOR	MT-SAC	SDP	SMT	MOSAIC
Assemble Nut	65.6 ± 34.2	7.4 ± 12.4	77.6 ± 27.6	91.7 ± 14.4	82.2 ± 23.1
Basketball	98.6 ± 3.2	23.9 ± 27.4	80.8 ± 24.1	68.5 ± 32.4	82.3 ± 24.0
Close Box	50.0 ± 37.6	1.1 ± 3.5	13.6 ± 18.6	54.6 ± 37.3	81.9 ± 23.7
Close Door	48.6 ± 39.3	0.0 ± 0.1	23.8 ± 28.4	49.3 ± 38.9	79.9 ± 25.3
Close Drawer	67.0 ± 34.4	38.8 ± 36.9	91.0 ± 15.2	84.5 ± 20.7	72.5 ± 29.8
Close Window	40.5 ± 34.7	0.0 ± 0.1	18.7 ± 24.4	48.9 ± 38.5	80.4 ± 25.8
Disassemble Nut	71.8 ± 31.1	19.5 ± 24.2	90.5 ± 14.8	42.0 ± 37.2	93.5 ± 11.7
Get Coffee	48.1 ± 37.5	4.3 ± 8.1	19.0 ± 24.6	41.0 ± 37.4	84.1 ± 21.9
Hammer	100.0 ± 0.0	0.0 ± 0.1	100.0 ± 0.0	42.0 ± 37.6	100.0 ± 0.0
Insert Hand	41.5 ± 37.4	0.0 ± 0.1	17.4 ± 22.7	42.1 ± 37.2	79.0 ± 25.4
Insert Peg Side	80.2 ± 25.0	80.6 ± 24.4	84.6 ± 21.0	90.0 ± 15.3	100.0 ± 0.0
Lock Door	100.0 ± 0.0	79.6 ± 18.8	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
Open Door	95.5 ± 7.4	71.5 ± 32.5	100.0 ± 0.0	99.4 ± 1.7	100.0 ± 0.0
Open Drawer	41.1 ± 36.8	0.1 ± 0.4	97.7 ± 6.0	95.1 ± 9.1	100.0 ± 0.0
Open Window	42.8 ± 36.8	0.0 ± 0.2	23.0 ± 27.8	70.5 ± 32.4	88.5 ± 18.6
Pick Bin	40.3 ± 37.0	4.9 ± 10.1	95.1 ± 9.8	69.3 ± 32.4	76.8 ± 27.0
Pick Out Of Hole	93.5 ± 11.2	54.9 ± 38.0	94.2 ± 11.3	87.8 ± 18.5	100.0 ± 0.0
Pick&Place	41.1 ± 38.1	6.2 ± 11.7	61.5 ± 36.3	60.3 ± 34.7	76.4 ± 28.2
Pick&Place W Wall	67.9 ± 32.1	6.3 ± 11.7	99.5 ± 1.9	53.8 ± 38.2	94.9 ± 10.2
Place Onto Shelf	97.9 ± 5.1	96.7 ± 7.3	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
Press Button	100.0 ± 0.0	24.1 ± 18.8	96.1 ± 8.9	99.2 ± 2.1	99.6 ± 1.5
Press Button Top	47.0 ± 38.5	0.0 ± 0.1	18.9 ± 24.1	43.8 ± 37.5	88.5 ± 16.9
Press Button Top Wall	46.1 ± 39.3	9.1 ± 15.1	21.6 ± 26.2	43.2 ± 38.7	78.5 ± 26.6
Press Button Wall	45.7 ± 38.4	46.2 ± 39.0	72.1 ± 30.7	75.8 ± 28.6	98.3 ± 4.3
Press Handle	83.7 ± 21.7	42.3 ± 27.4	98.3 ± 4.2	94.3 ± 11.1	98.5 ± 3.8
Press Handle Side	97.0 ± 7.2	82.5 ± 22.7	91.1 ± 15.1	99.3 ± 1.8	99.0 ± 2.8
Pull	44.4 ± 36.9	0.0 ± 0.1	93.2 ± 12.8	42.2 ± 38.5	83.2 ± 21.8
Pull Handle	45.9 ± 38.4	40.4 ± 34.5	94.5 ± 10.8	48.1 ± 36.4	99.2 ± 2.6
Pull Handle Side	46.1 ± 35.9	0.0 ± 0.1	51.7 ± 37.4	44.3 ± 37.1	84.3 ± 20.3
Pull Lever	100.0 ± 0.0	18.3 ± 24.9	93.3 ± 11.5	100.0 ± 0.0	100.0 ± 0.0
Pull Mug	87.9 ± 17.8	80.5 ± 24.1	100.0 ± 0.0	95.7 ± 8.7	99.4 ± 2.0
Pull With Stick	59.1 ± 35.8	0.4 ± 0.9	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
Push	46.4 ± 39.2	0.3 ± 0.9	68.1 ± 35.1	53.0 ± 37.2	74.6 ± 29.1
Push Mug	98.8 ± 3.6	98.4 ± 5.1	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
Push With Stick	83.3 ± 22.3	1.6 ± 4.0	29.0 ± 32.0	80.2 ± 25.2	82.7 ± 22.4
Push With Wall	45.6 ± 39.2	2.6 ± 4.6	24.7 ± 29.1	47.5 ± 38.4	85.3 ± 20.7
Reach	78.6 ± 26.1	42.4 ± 37.0	91.7 ± 12.9	54.1 ± 37.6	86.6 ± 19.6
Reach With Wall	56.5 ± 36.1	4.6 ± 7.9	96.3 ± 7.6	43.8 ± 37.4	95.1 ± 9.7
Retrieve Plate	45.8 ± 38.5	0.0 ± 0.1	22.2 ± 26.7	43.9 ± 39.8	82.4 ± 22.8
Retrieve Plate Side	78.1 ± 26.3	48.9 ± 37.8	39.3 ± 36.8	96.2 ± 8.3	96.3 ± 8.1
Slide Plate	39.8 ± 35.2	0.0 ± 0.1	18.7 ± 23.7	46.8 ± 37.7	77.0 ± 28.5
Slide Plate Side	42.1 ± 38.4	4.0 ± 8.9	13.7 ± 18.7	46.0 ± 38.5	81.6 ± 22.6
Soccer	99.8 ± 0.7	5.2 ± 8.5	98.4 ± 3.4	99.5 ± 1.7	100.0 ± 0.0
Sweep	97.4 ± 6.4	83.1 ± 21.9	97.9 ± 5.7	98.4 ± 3.8	93.7 ± 11.8
Sweep With Hole	45.4 ± 35.9	0.7 ± 2.4	69.4 ± 33.3	78.4 ± 26.2	81.0 ± 24.3
Turn Dial	94.7 ± 10.5	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
Turn Off Faucet	100.0 ± 0.0	50.3 ± 35.4	100.0 ± 0.0	60.4 ± 36.6	100.0 ± 0.0
Turn On Faucet	100.0 ± 0.0	5.4 ± 8.6	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
Unlock Door	61.6 ± 36.4	21.4 ± 26.2	92.8 ± 13.5	88.6 ± 16.8	99.3 ± 2.1
Unplug Peg	45.7 ± 37.4	0.0 ± 0.1	68.5 ± 32.8	52.5 ± 36.3	82.1 ± 23.5
Average	67.9 ± 25.2	26.2 ± 13.7	71.0 ± 16.2	71.3 ± 23.5	90.4 ± 12.8

D VISUAL ANALYSIS

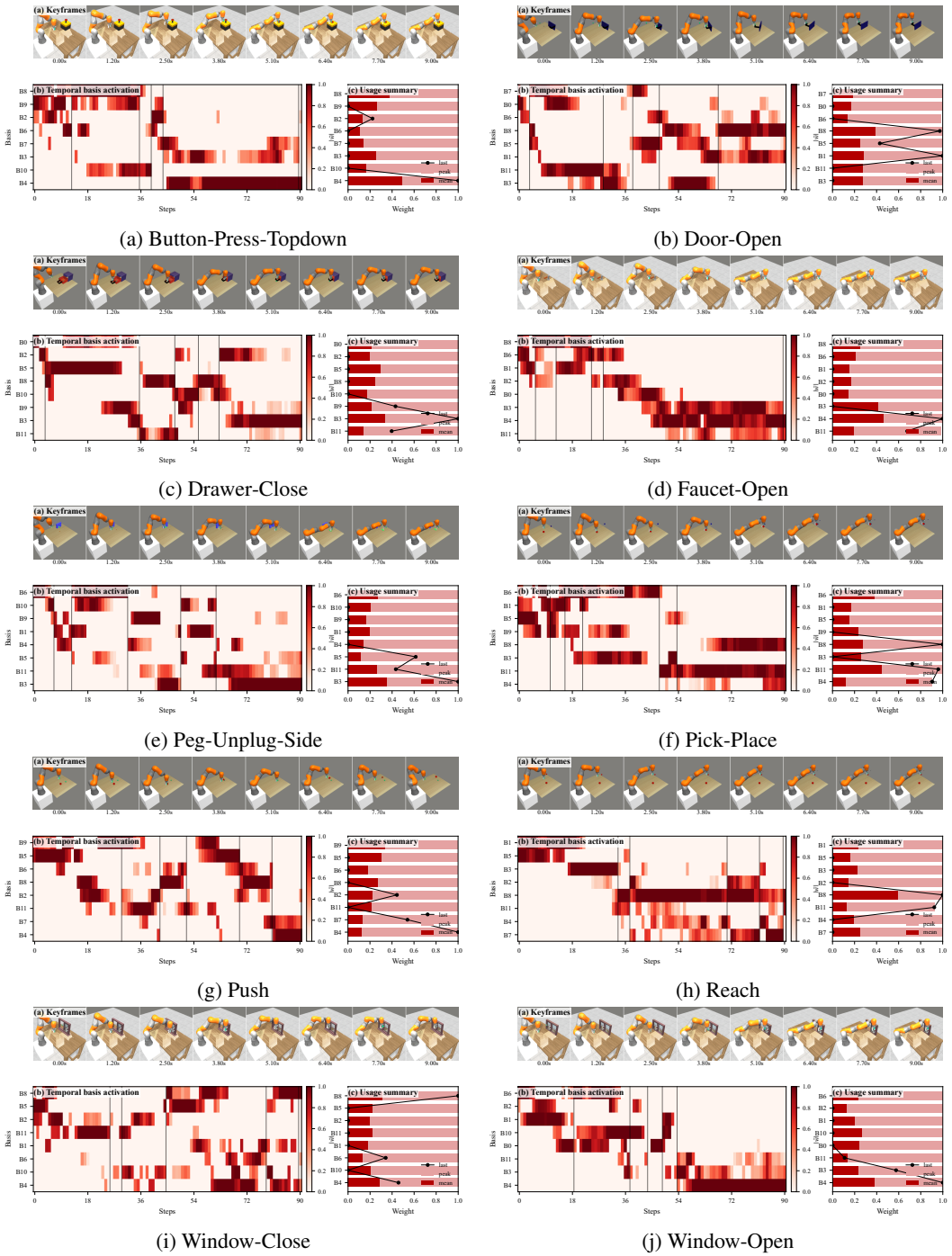
We reports aggregate basis-usage diagnostics on MT-10 and MT-50. For completeness, we provide per-task rollout-level visualizations in this appendix. Each panel contains sampled keyframes, temporal basis activation, and basis-usage summaries. These visualizations show that MOSAIC adapts its active basis support across task phases rather than using a fixed basis throughout execution.

D.1 PER-TASK TEMPORAL BASIS ACTIVATION ON MT-10

The main text reports aggregate basis-usage diagnostics on MT-10. For completeness, we provide per-task rollout-level visualizations in this appendix. Each panel contains sampled keyframes, temporal basis activation over the rollout, and a summary of basis-usage statistics. These visualizations show that MOSAIC changes its active basis support across interaction phases, rather than assigning each task to a fixed basis or using a dense basis mixture throughout execution.

D.2 PER-TASK TEMPORAL BASIS ACTIVATION ON MT-50

The main text reports aggregate basis-usage diagnostics on MT-50. For completeness, we provide per-task rollout-level visualizations in this appendix. Each panel contains sampled keyframes, temporal basis activation over the rollout, and a summary of basis-usage statistics. These visualizations show that MOSAIC changes its active basis support across interaction phases, rather than assigning each task to a fixed basis or using a dense basis mixture throughout execution.



(a) Button-Press-Topdown (b) Door-Open

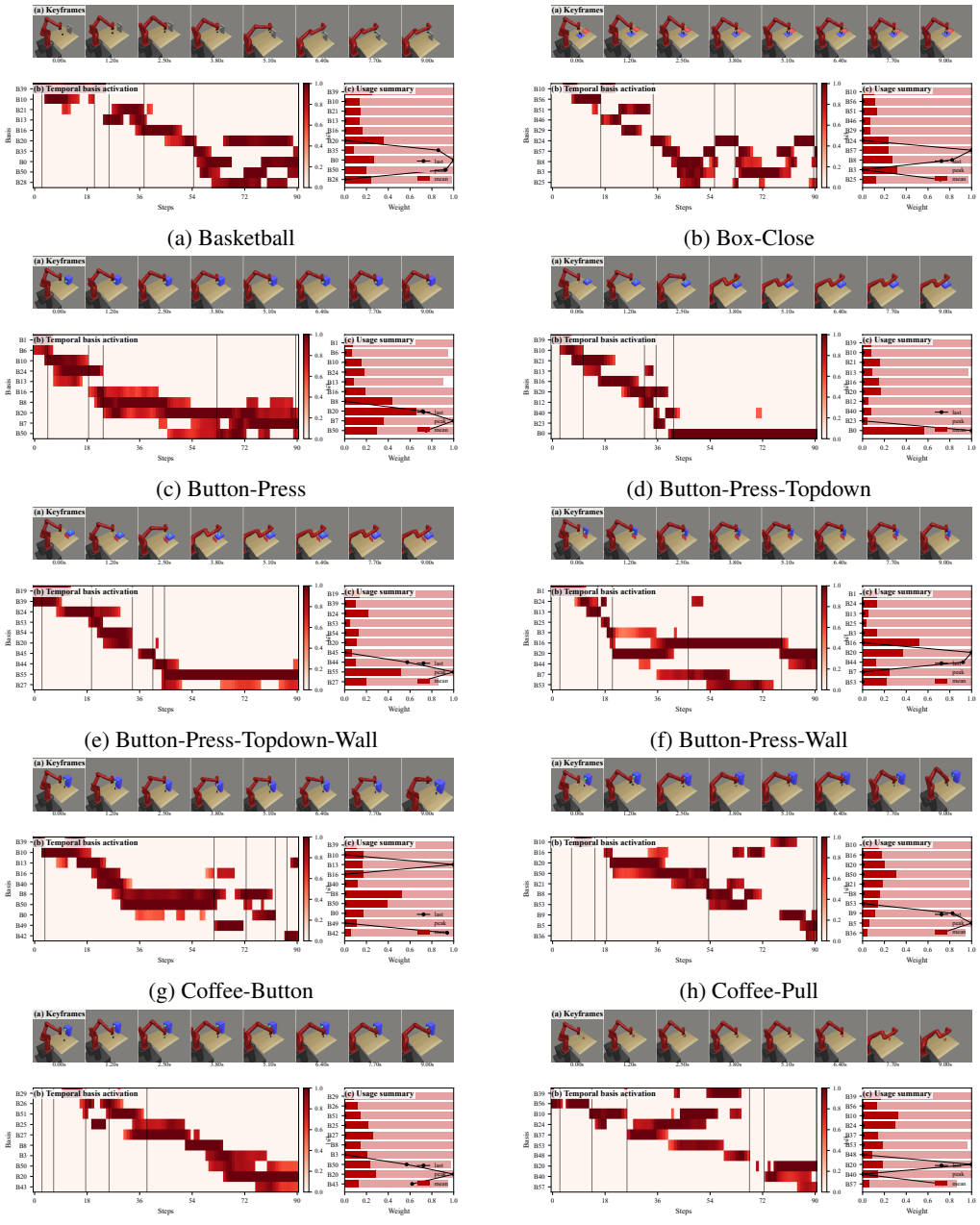
(c) Drawer-Close (d) Faucet-Open

(e) Peg-Unplug-Side (f) Pick-Place

(g) Push (h) Reach

(i) Window-Close (j) Window-Open

Figure 13: **Per-task temporal basis activation on MT-10.** Each panel shows rollout keyframes, temporal basis activation, and basis-usage summary for one task. MOSAIC changes its active basis support across interaction phases rather than relying on a fixed basis throughout execution.



(a) Basketball (b) Box-Close

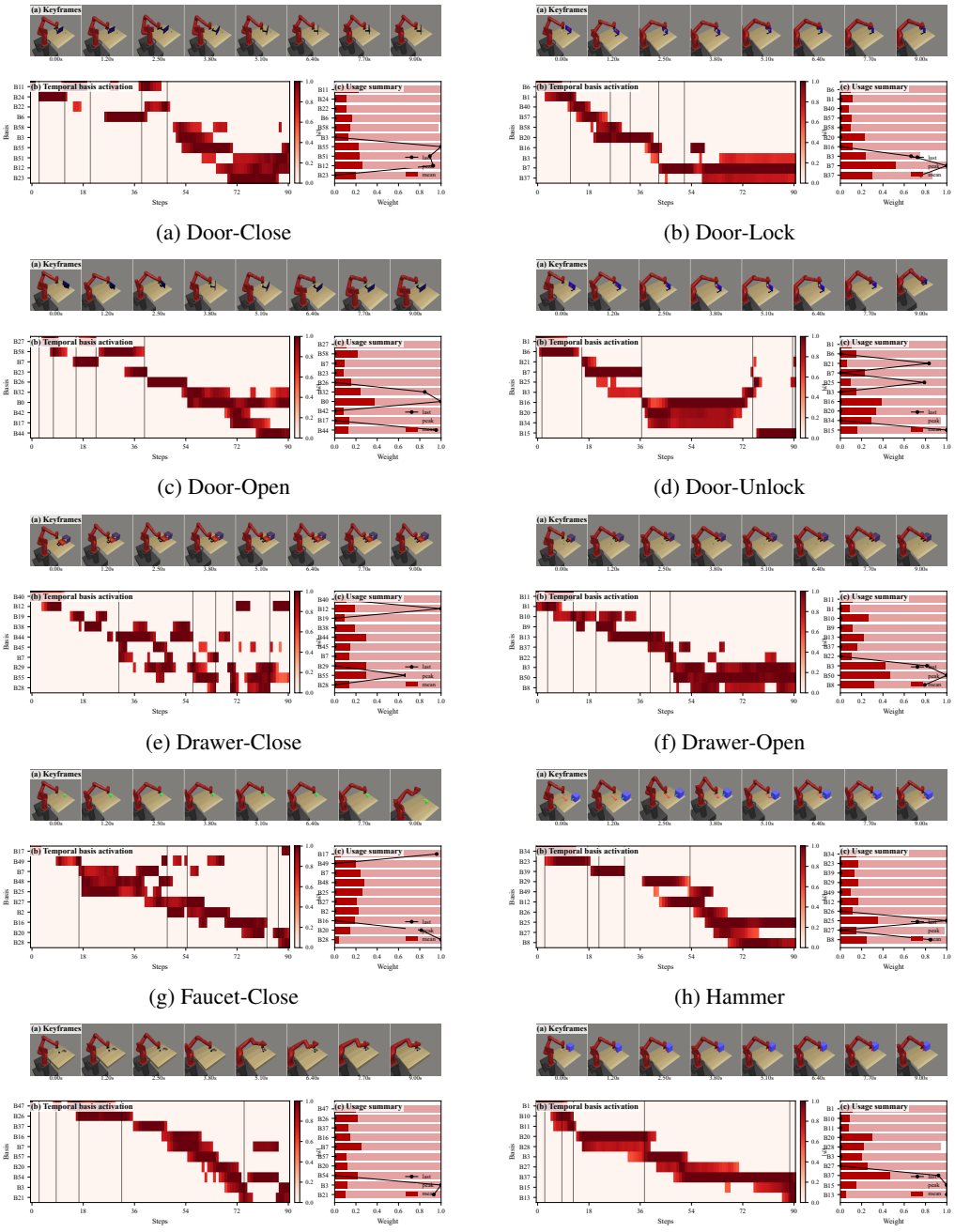
(c) Button-Press (d) Button-Press-Topdown

(e) Button-Press-Topdown-Wall (f) Button-Press-Wall

(g) Coffee-Button (h) Coffee-Pull

(i) Coffee-Push (j) Dial-Turn

Figure 14: **Per-task temporal basis activation on MT-50.** Each panel shows rollout keyframes, temporal basis activation, and basis-usage summaries for one task.



(a) Door-Close (b) Door-Lock

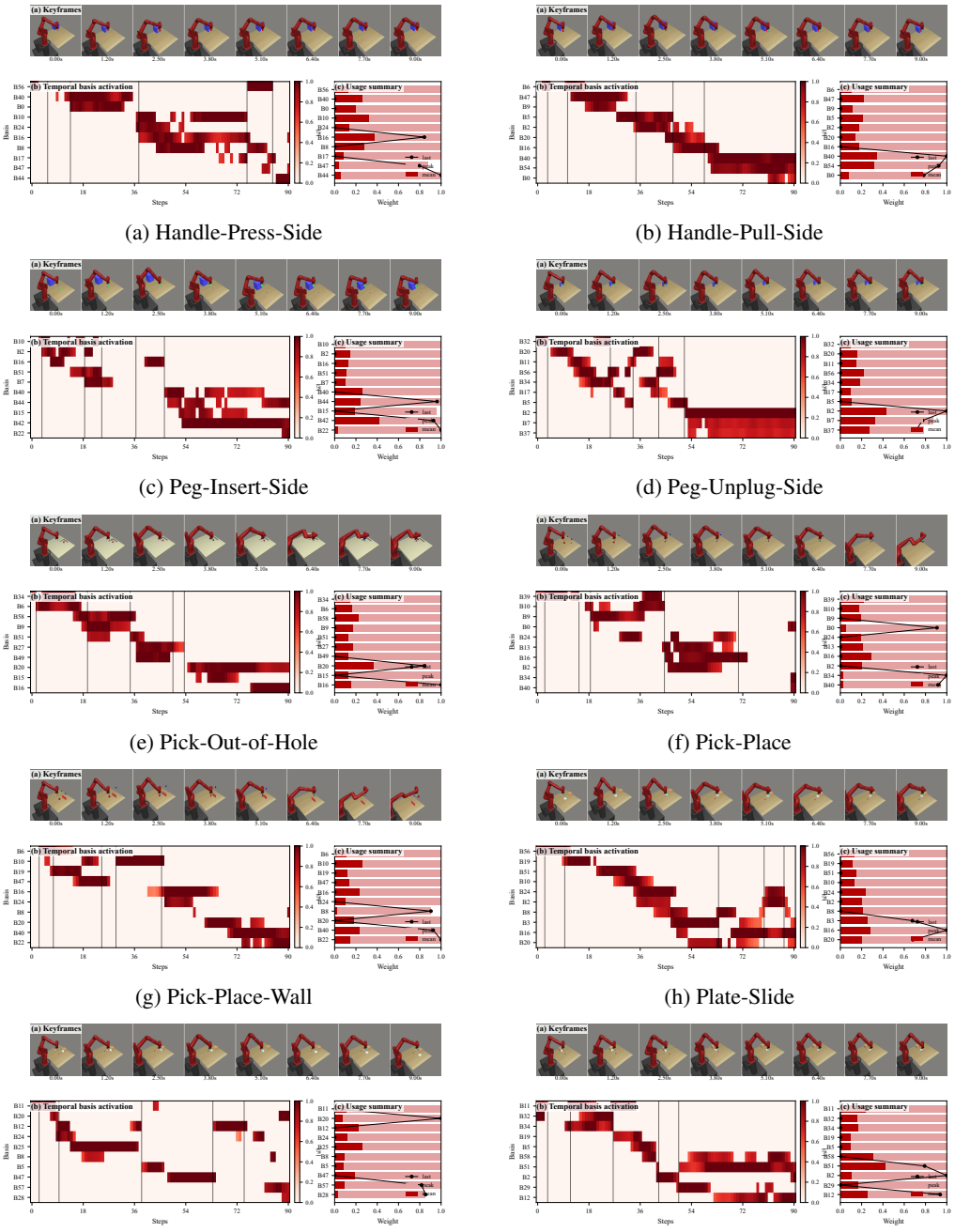
(c) Door-Open (d) Door-Unlock

(e) Drawer-Close (f) Drawer-Open

(g) Faucet-Close (h) Hammer

(i) Hand-Insert (j) Handle-Press

Figure 15: **Per-task temporal basis activation on MT-50.**



(a) Handle-Press-Side (b) Handle-Pull-Side

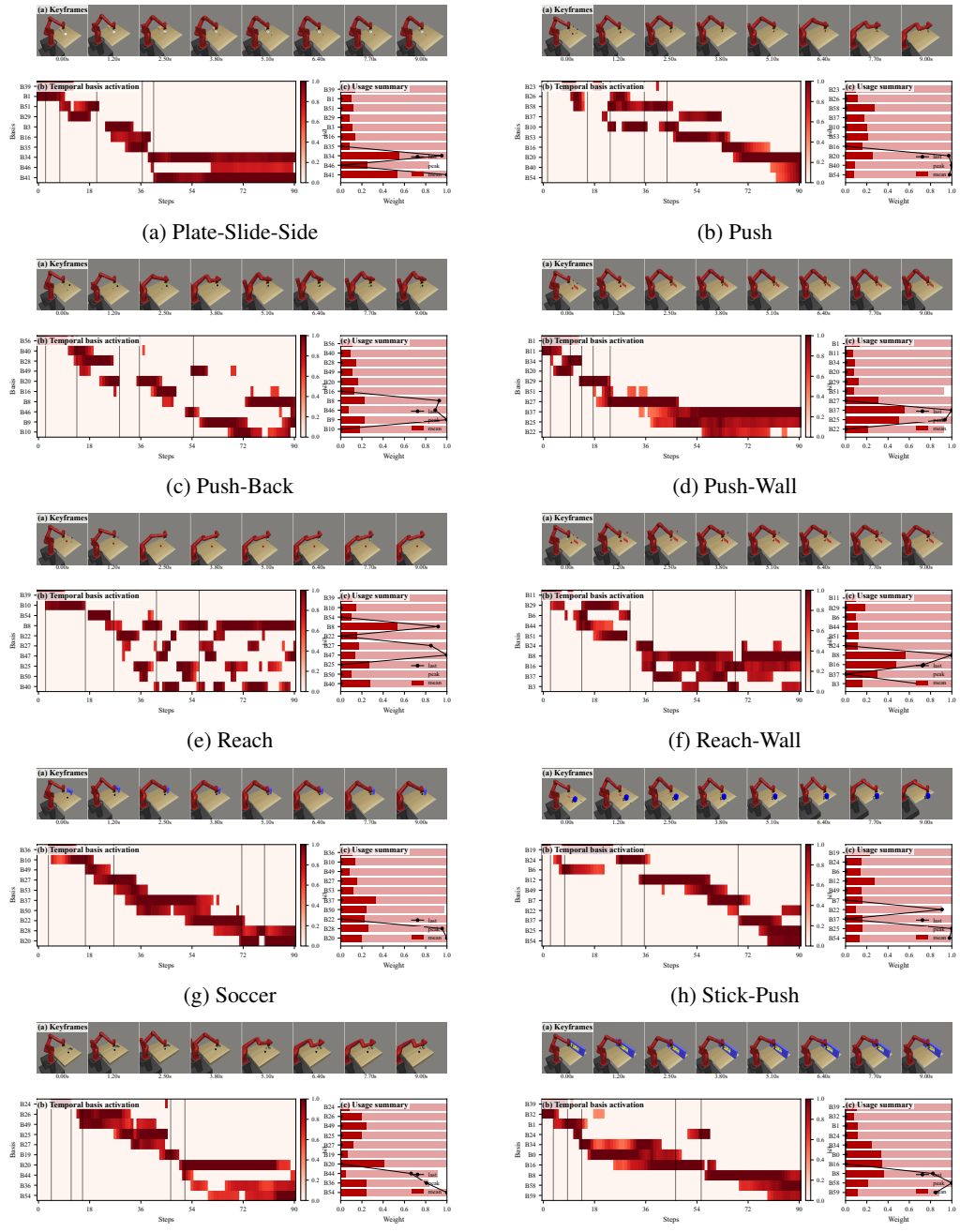
(c) Peg-Insert-Side (d) Peg-Unplug-Side

(e) Pick-Out-of-Hole (f) Pick-Place

(g) Pick-Place-Wall (h) Plate-Slide

(i) Plate-Slide-Back (j) Plate-Slide-Back-Side

Figure 16: **Per-task temporal basis activation on MT-50.**



(a) Plate-Slide-Side (b) Push

(c) Push-Back (d) Push-Wall

(e) Reach (f) Reach-Wall

(g) Soccer (h) Stick-Push

(i) Sweep-Into (j) Window-Open

Figure 17: **Per-task temporal basis activation on MT-50, part 4.** Across tasks, MOSAIC exhibits temporally varying basis supports, indicating that the learned dictionary is reused across task phases rather than assigned as fixed task-specific experts.

E ALGORITHMIC PROCEDURE

Algorithm 1 MOSAIC

Input: dictionary $\boldsymbol{D} = [\boldsymbol{d}_1, \ldots, \boldsymbol{d}_K]$; encoders ϕ_I, ϕ_e, g_θ; trunk f_{trunk}; inference network q_ϕ; meta-policy π_{meta}; option policies $\{\pi_k\}_{k=1}^K$; terminations $\{\beta_k\}_{k=1}^K$; option-value critic Q_Ω; per-option critics $\{Q_k\}_{k=1}^K$; replay buffer $\mathcal{B}$; temperatures α, α_Ω; discount γ; structure weights $(\beta_{\text{KL}}, \lambda_{\text{ortho}}, \lambda_{\text{sp}}, \lambda_{\text{struct}})$.
Output: learned parameters for $\boldsymbol{D}$, q_ϕ, π_{meta}, $\{\pi_k, \beta_k\}_{k=1}^K$, and all critics.
Initialize all networks and target critics; set recurrent state $\boldsymbol{H}_0 = \boldsymbol{0}$; set $\text{term}_0 = 1$.
repeat
  Observe $\boldsymbol{o}_t = (I_t, e_t)$, reward r_t, done flag done_t, and instruction l (if provided).
  Encode $\boldsymbol{z}_t \leftarrow f(\phi_I(I_t), \phi_e(e_t))$; encode $\boldsymbol{h}_\ell \leftarrow g_\theta(l)$ (if available).
  Update history $\boldsymbol{H}_t \leftarrow \text{RNN}(\boldsymbol{H}_{t-1}, [\boldsymbol{z}_t, \boldsymbol{h}_\ell])$.
  Infer composition parameters with $q_\phi(\cdot \mid \boldsymbol{z}_t, \boldsymbol{h}_\ell, \boldsymbol{H}_t)$ and compute expected weights $\bar{\boldsymbol{w}}_t \leftarrow \sigma(\boldsymbol{u}(\cdot)) \odot \boldsymbol{\mu}(\cdot)$.
  Compose skill embedding $\boldsymbol{m}_t \leftarrow \boldsymbol{D}\,\bar{\boldsymbol{w}}_t$.
  if $t = 0$ **or** $\text{term}_t = 1$ **then**
    Sample option index $j_t \sim \pi_{\text{meta}}(\cdot \mid \boldsymbol{z}_t, \boldsymbol{h}_\ell, \boldsymbol{m}_t, \boldsymbol{H}_t)$.
  end if
  Compute trunk feature $\boldsymbol{h}_t \leftarrow f_{\text{trunk}}(\boldsymbol{z}_t)$.
  Compute FiLM parameters $(\boldsymbol{\gamma}_{j_t}, \boldsymbol{\eta}_{j_t}) \leftarrow \psi([\boldsymbol{m}_t, \boldsymbol{d}_{j_t}])$ and modulate $\tilde{\boldsymbol{h}}_{t,j_t} \leftarrow \boldsymbol{\gamma}_{j_t} \odot \boldsymbol{h}_t + \boldsymbol{\eta}_{j_t}$.
  Sample and execute action $a_t \sim \pi_{j_t}(\cdot \mid \tilde{\boldsymbol{h}}_{t,j_t})$; observe next $\boldsymbol{o}_{t+1}$, reward r_t, and done flag done_{t+1}.
  Encode $\boldsymbol{z}_{t+1}$ and update $\boldsymbol{H}_{t+1}$; infer $\bar{\boldsymbol{w}}_{t+1}$ and $\boldsymbol{m}_{t+1}$ analogously.
  Sample termination $\text{term}_{t+1} \sim \text{Bernoulli}(\beta_{j_t}(\boldsymbol{z}_{t+1}))$.
  Store transition $(\boldsymbol{o}_t, \boldsymbol{z}_t, \boldsymbol{H}_t, \boldsymbol{h}_\ell, \boldsymbol{m}_t, j_t, a_t, r_t, \boldsymbol{o}_{t+1}, \text{done}_{t+1}, \text{term}_{t+1})$ into $\mathcal{B}$.
  if time to update (every N environment steps) **then**
    for $g = 1$ **to** G **do**
      Sample a minibatch from $\mathcal{B}$.
      Compute structure penalty $\mathcal{L}_{\text{struct}}(t) = \beta_{\text{KL}}\mathcal{L}_{\text{KL}}(t) + \lambda_{\text{ortho}}\mathcal{L}_{\text{ortho}}(\boldsymbol{D}) + \lambda_{\text{sp}}\|\bar{\boldsymbol{w}}_t\|_1$.
      Update per-option critics $\{Q_k\}$ and policies $\{\pi_k\}$ with SAC using $\tilde{\boldsymbol{h}}_{t,j_t}$ and temperature α.
      Compute meta soft value $V_\Omega(\boldsymbol{z}) \leftarrow \alpha_\Omega \log \sum_{j'} \exp(Q_\Omega(\boldsymbol{z}, j')/\alpha_\Omega)$.
      Compute continuation $U_\Omega(\boldsymbol{z}_{t+1}, j_t) \leftarrow (1-\beta_{j_t}(\boldsymbol{z}_{t+1}))Q_\Omega(\boldsymbol{z}_{t+1}, j_t) + \beta_{j_t}(\boldsymbol{z}_{t+1})V_\Omega(\boldsymbol{z}_{t+1})$.
      Compute target $y_t^\Omega \leftarrow r_t + \gamma(1 - \text{done}_{t+1})U_\Omega(\boldsymbol{z}_{t+1}, j_t)$.
      Update Q_Ω by minimizing $\mathbb{E}[(Q_\Omega(\boldsymbol{z}_t, j_t) - y_t^\Omega)^2]$.
      Update π_{meta} by minimizing $\mathbb{E}[\sum_j \pi_{\text{meta}}(j \mid \boldsymbol{x}_t)(\alpha_\Omega \log \pi_{\text{meta}}(j \mid \boldsymbol{x}_t) - Q_\Omega(\boldsymbol{z}_t, j))]$, where $\boldsymbol{x}_t = (\boldsymbol{z}_t, \boldsymbol{h}_\ell, \boldsymbol{m}_t, \boldsymbol{H}_t)$.
      Update terminations $\{\beta_k\}$ by minimizing $\mathbb{E}[\beta_{j_t}(\boldsymbol{z}_t)(Q_\Omega(\boldsymbol{z}_t, j_t) - V_\Omega(\boldsymbol{z}_t))]$.
      Update q_ϕ and $\boldsymbol{D}$ using gradients from RL losses plus $\lambda_{\text{struct}} \sum_{t \in \text{batch}} \mathcal{L}_{\text{struct}}(t)$.
      Update all target networks by Polyak averaging.
    end for
  end if
  if $\text{done}_{t+1} = 1$ **then**
    Reset environment; set $\boldsymbol{H}_{t+1} \leftarrow \boldsymbol{0}$; set $\text{term}_{t+1} \leftarrow 1$.
  end if
  $t \leftarrow t + 1$.
until training budget is exhausted

Implementation note (deterministic gates). Although Eq. equation 5 defines Bernoulli gates, in our off-policy implementation we use the deterministic expectation in Eq. equation 6 to avoid high-variance gradients from discrete sampling; the KL term in Eq. equation 9 is treated as a structured sparsity regularizer on the gate logits and Gaussian strengths.

Table 9: Summary of notation for MOSAIC. Here, $\boldsymbol{H}_t$ denotes the temporal history (recurrent hidden state) and $\boldsymbol{c}_t$ denotes the continuous skill strength.

Symbol	Definition	Explanation / Usage
1. Environment & Inputs		
t	Timestep	Discrete time index.
$\boldsymbol{s}_t$	Environment state	Standard RL state.
r_t	Reward	Scalar reward at time t.
$\boldsymbol{o}_t$	Observation	Agent input $\boldsymbol{o}_t = (I_t, e_t)$ (visual + proprioception).
I_t	Image	Visual observation (single frame or stack) at time t.
e_t	Proprioception	Low-dimensional robot state vector at time t.
l	Task instruction	Natural language instruction describing the task.
a_t	Action	Action executed at time t (output of the low-level policy).
2. Encoding & Latent States		
ϕ_I, ϕ_e	Visual / proprio encoders	Feature extractors for I_t and e_t.
g_θ	Language encoder	Instruction encoder for l.
$\boldsymbol{z}_t$	Latent observation	Fused backbone feature $\boldsymbol{z}_t = f(\phi_I(I_t), \phi_e(e_t)) \in \mathbb{R}^D$.
$\boldsymbol{h}_\ell$	Language embedding	$\boldsymbol{h}_\ell = g_\theta(l) \in \mathbb{R}^H$.
$\boldsymbol{H}_t$	Temporal history state	Recurrent hidden state $\boldsymbol{H}_t \in \mathbb{R}^{D_H}$ aggregating past context.
3. Skill Bases & Dictionary		
K	skill bases	Number of atomic skill bases (dictionary size).
$\boldsymbol{D}$	Skill dictionary	$\boldsymbol{D} = [\boldsymbol{d}_1, \ldots, \boldsymbol{d}_K] \in \mathbb{R}^{D \times K}$.
$\boldsymbol{d}_k$	Atomic skill embedding	The k-th dictionary column, $\boldsymbol{d}_k \in \mathbb{R}^D$.
4. Skill Inference & Composition		
q_ψ	Inference network	Infers skill composition parameters from latent features (and instruction if used).
$\boldsymbol{b}_t$	Binary gate	Sparse gate $\boldsymbol{b}_t \in \{0,1\}^K$ indicating active bases.
$\boldsymbol{c}_t$	Continuous strength	Strength vector $\boldsymbol{c}_t \in \mathbb{R}^K$ controlling activation magnitudes.
$\bar{\boldsymbol{w}}_t$	Expected skill weights	Typically used for composition, e.g., $\bar{\boldsymbol{w}}_t \approx \mathbb{E}[\boldsymbol{b}_t \odot \boldsymbol{c}_t]$.
$\mathcal{K}_t$	Active basis indices	Set of indices $\mathcal{K}_t = \{k \mid \bar{w}_{t,k} > \varepsilon\}$ corresponding to active skills.
$\mathcal{U}_t$	Active skill subspace	Linear subspace $\mathcal{U}_t = \mathrm{span}(\{\boldsymbol{d}_k\}_{k \in \mathcal{K}_t})$ specialized to context.
$\boldsymbol{m}_t$	Composed skill embedding	Global skill embedding $\boldsymbol{m}_t = \boldsymbol{D}\,\bar{\boldsymbol{w}}_t = \sum_{k=1}^K \bar{w}_{t,k} \boldsymbol{d}_k \in \mathbb{R}^D$.
5. Hierarchical Control & Execution		
π_{meta}	Meta-policy	Discrete policy selecting an option, conditioned on $(\boldsymbol{z}_t, \boldsymbol{h}_\ell, \boldsymbol{m}_t, \boldsymbol{H}_t)$ as specified.
j_t	Selected option index	Discrete index $j_t \in \{1, \ldots, K\}$ chosen by π_{meta}.
$\boldsymbol{d}_{j_t}$	Active skill basis	Selected dictionary column corresponding to option j_t.
FiLM	Feature modulation	Uses $[\boldsymbol{m}_t, \boldsymbol{d}_{j_t}]$ to produce modulation $(\boldsymbol{\gamma}_{j_t}, \boldsymbol{\eta}_{j_t})$ for the shared trunk feature.
π_{j_t}	Low-level option policy	Continuous controller for option j_t producing a_t from modulated features.

F LIMITATIONS AND FUTURE WORK

While MOSAIC demonstrates strong performance and efficiency, it entails specific assumptions and limitations that open avenues for future research:

Static Dictionary Capacity: The current framework relies on a pre-defined dictionary size K. Although our ablation studies (see Section 4.2) indicate robustness to K within a reasonable range, the model does not yet support open-ended lifelong learning. As the agent encounters novel, orthogonal dynamics that cannot be spanned by the existing bases, the performance may saturate. Future work could incorporate non-parametric Bayesian methods (e.g., Indian Buffet Process) to dynamically expand $\boldsymbol{D}$ online.

Locally Linear Dynamics Assumption: MOSAIC is grounded in the hypothesis that non-linear manifolds can be approximated by switching between locally linear subspaces. This works well for rigid-body manipulation and kinematic transport. However, for tasks involving highly deformable objects (e.g., cloth folding, fluid pouring) or chaotic dynamics, this approximation might be insufficient. Such domains may require non-linear basis functions or neural operators to capture high-dimensional state evolutions.

Sim-to-Real Gap: Our experiments are conducted on the Meta-World benchmark in simulation. While the structured subspaces provide a form of regularization that typically aids transfer, we have not yet validated the system on physical hardware. Deploying the learned $>$500 Hz policies to the real world will likely require domain randomization or adaptation modules to handle sensor noise, friction mismatches, and visual domain shifts.