SPARSE ACTIVATION ATTRIBUTES FOR FRAME CURATION AND CURRICULUM SCHEDULING IN IMITATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Training vision-language-action (VLA) models requires large-scale robot demonstration datasets, making data curation critical for both data efficiency and policy performance. Existing approaches typically select entire episodes or coarse temporal blocks and sample the retained data from a fixed distribution, leaving substantial frame-level redundancy and overlooking how the value of data with different attributes evolves throughout training. We introduce **Sparse Activation Attributes for Frame Curation and Curriculum Scheduling** (**SAFari**), a fine-grained framework that uses sparse activation attributes to jointly select training frames and schedule their presentation. SAFari first employs a sparse autoencoder to extract intermediate activation features and characterizes them by their generality and specificity. It then uses the strengths of these two attributes to assess individual frames and independently constructs general and specific frame streams. Finally, a step-wise sigmoid curriculum dynamically shifts the sampling emphasis from general to specific frames as training progresses. In this way, SAFari not only prioritizes informative frames at a fine-grained level, but also presents data with different attributes at appropriate stages of training through curriculum scheduling. Experiments across multiple VLA models in both simulation and real-robot settings demonstrate consistent performance gains. Moreover, SAFari maintains strong performance with only half the training steps, substantially improving training efficiency.

1 INTRODUCTION

While many advances in vision-language-action (VLA) policies have come from scaling model architectures, the composition of the training data remains a critical driver of policy performance. The success of large vision and language models has accordingly motivated growing interest in data attribution and selection guided by a model (Albalak et al., 2024; Coleman et al., 2019), as well as automatic deduplication and data curation (Lee et al., 2022; Tirumala et al., 2023; Albalak et al., 2024), all grounded in the recognition that training samples do not contribute equally to a model. This issue becomes increasingly important as robot learning datasets grow in size and diversity (O'Neill et al., 2024; Khazatsky et al., 2024): demonstrations vary in quality, task relevance, and information density, while the number of stored frames can grow much faster than the amount of distinct supervision they provide.

Recent robot data curation methods rank and select demonstrations using preference and representation signals, mutual information objectives, online policy experience, datamodels, or influence functions (Kuhar et al., 2023; Hejna et al., 2025; Chen et al., 2025; Dass et al., 2026; Agia et al., 2025). In particular, DataMIL (Dass et al., 2026) uses datamodels to select episodes from large imitation learning datasets, whereas CUPID (Agia et al., 2025) estimates the influence of individual demonstrations on closed loop policy performance. Although effective at identifying valuable demonstrations, these approaches typically operate on entire episodes (Dass et al., 2026; Agia et al., 2025) or coarse temporal blocks (Yu et al., 2026; Zhang et al., 2025), leaving substantial redundancy because adjacent frames are often nearly identical and coarse blocks contain both informative transitions and long periods of little change. In fact, regardless of the overall quality of a demonstration, it may contain both valuable frames and redundant ones. Moreover, existing curation methods mainly

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.

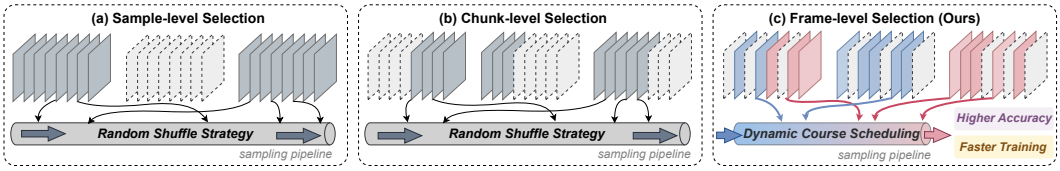

Figure 1: Comparison of data selection and sampling strategies. Methods that select samples or chunks retain coarse units and train with a fixed shuffled stream, whereas SAFari performs selection of individual frames and dynamically shifts the training emphasis from general to specific frames.

focus on selecting high quality data to improve policy performance, with less attention to reducing training cost or accelerating training. Even after offline curation, standard training pipelines usually shuffle the retained data and sample from a fixed distribution throughout optimization. Such a fixed data stream treats the value of each example as constant, ignoring that frames showing broadly reusable manipulation patterns and rare frames specific to tasks or scenes may be most useful at different stages of training. These limitations motivate jointly considering *which* individual frames are worth training on and *when* they should be presented.

To identify truly valuable frames from robot demonstrations, we focus on two important attributes, generality and specificity, which have been widely considered in data selection (Swann et al., 2026). Generality reflects reusable manipulation patterns shared across tasks, while specificity captures valuable information associated with particular tasks, scenes, or transient events. Inspired by the interpretability of sparse autoencoders (SAEs) for understanding model representations, we use SAEs to analyze these two attributes in VLA representations. In particular, the activation patterns of sparse features can reveal whether they capture broadly shared manipulation patterns or information specific to particular tasks and scenes. Such structure is difficult to observe directly in the original dense representations. Meanwhile, curriculum learning studies suggest that training examples can have different utility at different stages of learning and that presenting them in a meaningful order can improve optimization and generalization (Bengio et al., 2009; Kumar et al., 2010; Platanios et al., 2019). Motivated by these two perspectives, we first identify valuable general and specific frames and then dynamically adjust their sampling proportions throughout training using curriculum learning, as illustrated in Figure 1.

Motivated by these observations, we introduce sparse activation attributes for frame curation and curriculum scheduling (SAFari). First, **sparse feature attribute factorization** characterizes each SAE feature using episode coverage and sustained activation duration. A lightweight classifier trained on manually labeled features then estimates the strength of its generality and specificity attributes. Second, **attribute-guided frame scoring and selection** computes generality and specificity scores for each frame and constructs two separate data streams based on these scores. It retains high-scoring frames while filtering out temporally redundant neighbors and preserving boundary states. Finally, **sigmoid-controlled dual-stream curriculum** dynamically samples from the two curated streams and uses a sigmoid schedule to control their sampling weights: general frames receive greater weight early to stabilize learning, whereas specific frames are progressively emphasized to expand task and scene coverage at later stages. Frame scoring and selection therefore determines which frames deserve computation, while curriculum scheduling determines when each retained stream is most useful. Because SAFari changes only the data pipeline, the policy architecture, action targets, optimizer, and loss remain unchanged.

Our main contributions are as follows:

- we jointly consider high-value frame selection and training-time scheduling to improve policy performance while accelerating training. We derive interpretable generality and specificity attributes for sparse activation features from episode coverage and sustained activation duration, enabling frame-level scoring.
- We use these attributes to select informative frames while suppressing temporal redundancy and preserving episode boundaries. A sigmoid curriculum then shifts sampling from general to specific frames without changing the policy architecture or objective.
- SAFari consistently improves performance across different policy architectures, including Octo-Small, π_0, and OpenVLA, as well as across four real-robot tasks. On LIBERO-90,

it raises Octo-Small success from 55.6% with full data to 78.2%, and still reaches 66.2% with half the optimization steps.

2 Related Work

2.1 Data Curation for Robot Imitation Learning

Robot data curation methods differ primarily in how they estimate data value and at what temporal granularity they make selection decisions. Existing approaches rank demonstrations using preference or representation signals (Kuhar et al., 2023), mutual information objectives (Hejna et al., 2025), or experience collected from intermediate policies (Chen et al., 2025). DataMIL uses datamodels to estimate the contribution of episodes in large multitask datasets (Dass et al., 2026), while CUPID uses influence functions to connect individual demonstrations to closed loop policy performance (Agia et al., 2025). These methods improve over treating all demonstrations uniformly, but their episode based decisions retain both informative and redundant phases of every selected demonstration. More recent methods increase temporal resolution by scoring coarse blocks using relative task progress or representations learned without labels (Yu et al., 2026; Zhang et al., 2025). However, coarse blocks can still contain many nearly identical frames. Moreover, existing methods mainly focus on selecting high quality data to improve policy performance, with less attention to reducing training cost. SAFari instead selects data at the frame level to remove local redundancy while preserving valuable information. It further organizes general and specific frames into separate streams and dynamically schedules them during training, improving both policy performance and training efficiency.

2.2 Sparse Autoencoders for Representation Analysis

Sparse autoencoders (SAEs) reconstruct dense hidden states through an overcomplete sparse dictionary and decompose superposed representations into sparse and more interpretable features (Huben et al., 2024; Gao et al., 2025). They have been widely studied as tools for understanding internal representations and identifying interpretable features in language models (Huben et al., 2024; Gao et al., 2025). Recent work has begun to extend SAE analysis to VLA policies. SAE-VLA studies sparse features in VLA representations and investigates their relation to policy behavior (Swann et al., 2026). Event-Grounded SAE further connects sparse features to behaviorally meaningful manipulation events and evaluates their effects through closed loop interventions (Jin et al., 2026). These studies demonstrate the potential of SAEs for understanding internal representations of VLA policies. Rather than focusing on feature interpretation or intervention, we characterize their generality and specificity to assess frame value, enabling fine grained frame selection and curriculum scheduling.

3 Methodology

To address frame level temporal redundancy and the limitations of a fixed training stream, we propose SAFari, which combines frame curation guided by sparse activation attributes with curriculum scheduling. SAFari first learns interpretable sparse features from intermediate VLA representations and characterizes their generality and specificity. These attributes are aggregated into two frame scores that guide independent frame selection while suppressing local temporal redundancy. Finally, the two curated streams are dynamically mixed using a sigmoid schedule. An overview of the framework is shown in Figure 2.

3.1 Sparse Feature Attribute Factorization

Let the demonstration dataset contain N episodes, $\mathcal{D} = \{e_i\}_{i=1}^{N}$, where $e_i = \{f_{i,t}\}_{t=0}^{T_i-1}$ and $f_{i,t}$ denotes the t-th frame of episode e_i. To obtain interpretable features for analysis of individual frames, we train a sparse autoencoder (SAE) on intermediate activations extracted from a VLA policy. Specifically, we feed every frame $f_{i,t} \in e_i$ into a frozen Octo model[1] and cache its inter-

[1]We use the publicly released Octo-Small checkpoint, `rail-berkeley/octo-small`, pretrained on the Open X-Embodiment (OXE) mixture (O'Neill et al., 2024).

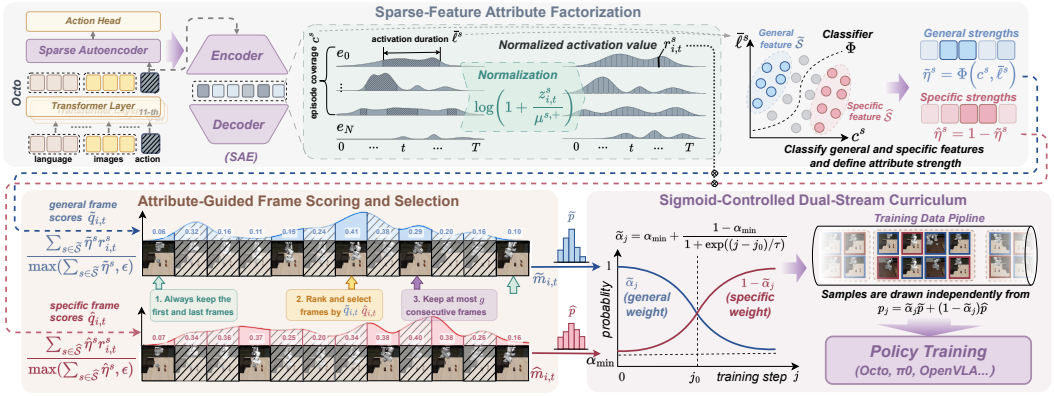

Figure 2: Overview of SAFari. Intermediate VLA activations are encoded by an SAE, and episode coverage and activation duration are used to estimate the generality and specificity strengths of each sparse feature. The corresponding feature activations are aggregated into two frame scores, which guide independent frame selection while preserving boundary states and prioritizing temporal spacing before filling each stream's budget. A sigmoid schedule then dynamically mixes the two curated streams for policy training, shifting emphasis from general to specific frames over time.

mediate representation from the residual stream $h_{i,t}$. These cached representations are then used to train an SAE based on the BatchTopK architecture (Bussmann et al., 2024) with the AuxK auxiliary loss (Gao et al., 2025). BatchTopK controls the average number of active features, while AuxK alleviates the problem of inactive latent features (Gao et al., 2025). The trained SAE learns a sparse feature dictionary indexed by $\mathcal{S}$ and encodes each cached frame representation as

$$z_{i,t} = \mathrm{Encode}_{\mathrm{SAE}}(h_{i,t}) \in \mathbb{R}^{|\mathcal{S}|}, \qquad I^s_{i,t} = \mathbb{1}[z^s_{i,t} > u], \tag{1}$$

where $z^s_{i,t}$ denotes the activation of sparse feature s on frame $f_{i,t}$, and $I^s_{i,t}$ is its thresholded activity indicator. We fix an activation threshold $u > 0$ and count only responses exceeding it toward episode coverage and sustained activation duration, treating weaker positive responses as inactive to prevent noise from inflating these statistics. The continuous activations $z^s_{i,t}$ are retained for frame scoring. Further architectural and training details are provided in Appendix A.1. We summarize each sparse feature s using two activation statistics: **episode coverage** c^s and **sustained activation duration** $\bar{\ell}^s$. Episode coverage measures how broadly a feature activates across episodes. Sustained activation duration measures the average length of a continuous activation run relative to the corresponding episode length. Short bursts aligned with events are commonly associated with reusable manipulation events, whereas long activations are more likely to encode persistent task, scene, or episode context. We manually label a small subset of SAE features as **general** or **specific** based on their strongest activating frames and temporal patterns.Features labeled general respond consistently to semantically coherent events, such as grasping, placing, task progress, or the appearance of a relevant object, across diverse tasks and scenes. By contrast, specific features are concentrated in particular tasks, visual scenes, or episode contexts. Using the two activation statistics of the labeled features, we train a lightweight binary classifier Φ and apply it to the full SAE dictionary. For feature s, the classifier produces attribute strengths:

$$\widetilde{\eta}^s = \Phi\big(c^s, \bar{\ell}^s\big), \qquad \widehat{\eta}^s = 1 - \widetilde{\eta}^s, \tag{2}$$

where $\widetilde{\eta}^s$ and $\widehat{\eta}^s$ denote the predicted generality and specificity strengths of feature s, respectively. We define the corresponding feature sets as $\widetilde{\mathcal{S}} = s \in \mathcal{S} : \widetilde{\eta}^s > \delta$ and $\widehat{\mathcal{S}} = s \in \mathcal{S} : \widehat{\eta}^s > \delta$, where δ is the decision threshold above which a feature is considered general or specific. A larger value indicates a stronger association with the corresponding attribute. We retain these continuous strengths for subsequent frame scoring. Formal definitions and classifier details are provided in Appendix A.2. The distinction between general and specific features is not absolute. Some features combine details specific to a scene with structure relevant to a task, while semantically general features may be rare in the dataset. We therefore use the classifier to estimate how strongly each feature tends toward either attribute, rather than to define its exact semantics.

3.2 Attribute-Guided Frame Scoring and Selection

Having identified the generality and specificity of each sparse feature, we use these attributes to score individual frames. Because SAE features can differ substantially in activation scale, their raw values are not directly comparable. We therefore normalize each feature by its mean positive activation and apply a logarithmic transformation to limit the influence of unusually large values:

$$\mu^{s,+} = \max(\mathbb{E}[z^s \mid z^s > 0], \epsilon), \qquad r^s_{i,t} = \log\left(1 + \frac{z^s_{i,t}}{\mu^{s,+}}\right). \tag{3}$$

An inactive feature has $z^s_{i,t} = 0$ and thus contributes zero, while positive activations become comparable across features. We obtain the general and specific frame scores by aggregating the normalized activations with their corresponding attribute strengths:

$$\widetilde{q}_{i,t} = \frac{\sum_{s\in\widetilde{\mathcal{S}}} \widetilde{\eta}^s r^s_{i,t}}{\max(\sum_{s\in\widetilde{\mathcal{S}}} \widetilde{\eta}^s, \epsilon)}, \qquad \widehat{q}_{i,t} = \frac{\sum_{s\in\widehat{\mathcal{S}}} \widehat{\eta}^s r^s_{i,t}}{\max(\sum_{s\in\widehat{\mathcal{S}}} \widehat{\eta}^s, \epsilon)}. \tag{4}$$

Because the two scores are computed independently, a frame may show both reusable manipulation structure and details specific to a task or episode. The small ϵ is used only for numerical stability.

For each attribute, we rank frames globally and retain a budget of $\rho \sum_i T_i$ frames to form two separate streams. We preserve the first and last frames of each episode as initial and completion states. Candidates are then selected in descending score order only if they are at least g time steps from previously retained frames in the same episode. If the target budget is not reached, the remaining slots are filled with the highest scoring unselected frames. This produces the general and specific masks $\widetilde{m}i, t$ and $\widehat{m}i, t$, which may overlap because the two attributes are complementary. Here, ρ denotes the retention rate of each stream, while the total unique retention rate is computed over the union of both masks (Appendix B.4).

3.3 Sigmoid-Controlled Dual-Stream Curriculum

After the frame scoring and selection stage produces the two complementary masks, the curriculum treats the retained general and specific frames as two sampling distributions, denoted by $\widetilde{p}$ and $\widehat{p}$, and dynamically controls their mixture during training. At training step j, the probability of drawing from the general stream is

$$\widetilde{\alpha}_j = \alpha_{\min} + \frac{1 - \alpha_{\min}}{1 + \exp((j - j_0)/\tau)}, \tag{5}$$

where j_0 is the transition midpoint, τ controls the transition width, and $\alpha_{\min}$ is the asymptotic minimum probability of sampling the general stream. The sampling distribution for the batch at step j is

$$p_j = \widetilde{\alpha}_j \widetilde{p} + (1 - \widetilde{\alpha}_j)\widehat{p}. \tag{6}$$

Each sample is independently drawn from p_j, giving an expected general fraction of $\widetilde{\alpha}_j$ per batch. We emphasize the general stream early because frequently recurring manipulation patterns provide broad and stable supervision. As optimization progresses, the probability of selecting the specific stream increases, exposing the policy to rare tasks, scenes, and events localized to an episode that expand distributional coverage. This curriculum dynamically adjusts the expected mixture of general and specific frames without changing the task labels or model architecture.

3.4 Optimization Objective

Let π_θ denote the policy and $p_j(x)$ the training data distribution determined by the curriculum at step j. We retain the original objective for action prediction:

$$\min_\theta \ \mathbb{E}_{x\sim p_j(x)} \left[\mathcal{L}_{\text{action}}(\pi_\theta; x)\right]. \tag{7}$$

SAFari changes only the selection and scheduling of training frames, leaving the policy architecture, action targets, optimizer, and loss unchanged. Since generality and specificity primarily reflect properties of the demonstration data rather than a specific policy architecture, we use lightweight Octo to analyze these attributes and select frames for other policy models. Experiments show that these selections transfer effectively across architectures, allowing the curated data to be reused without additional frame scoring or inference overhead.

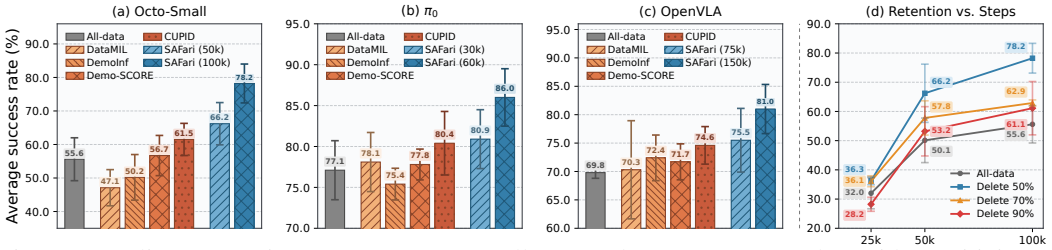

Figure 3: Policy comparisons across Octo-Small, π_0, and OpenVLA, together with sensitivity to frame retention and training budget. SAFari retains 50% of frames per stream unless otherwise stated. We report the mean and standard deviation over three training seeds and two evaluation seeds per trained model, with 50 rollouts per task and evaluation seed.

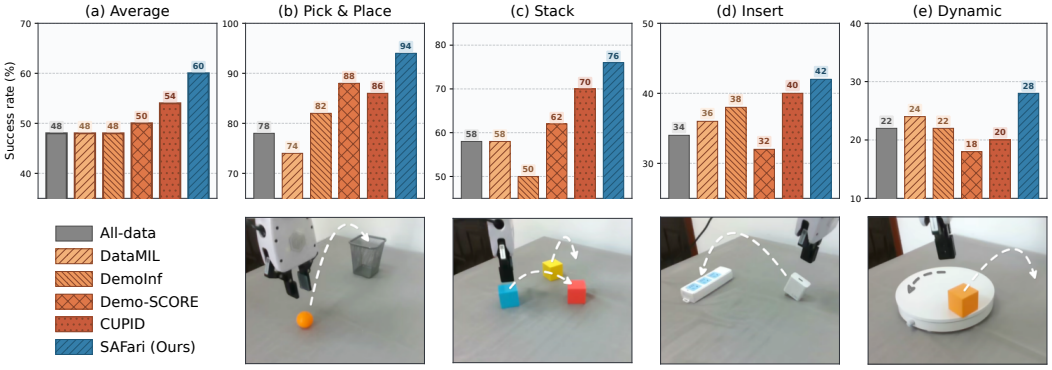

Figure 4: Real robot success rates on four xArm6 (Munoz et al., 2021) tasks, Because real robot evaluation is costly and time consuming, we report results from one training seed and one evaluation seed, with 50 trials per task.

4 Experiments

In this section, we evaluate SAFari in both simulated and real world settings. We study its effectiveness across multiple policy models, frame retention ratios, curriculum design, and training efficiency. We also visualize the learned activation attributes and their resulting frame selections. More experiment details are provided in Appendix.

4.1 Experimental Setup

Datasets. We conduct simulation experiments on LIBERO-90 (Liu et al., 2023), which contains 4,500 demonstrations across 90 manipulation tasks and 669,043 valid training frames after preprocessing. We additionally collect a real world dataset on an xArm6 (Munoz et al., 2021) robot using SpaceMouse teleoperation (Zhou et al., 2026) at 30 Hz. Visual observations are recorded by an Intel RealSense L515 (Han et al., 2024) wrist camera and a D435i (Wang et al., 2022) third person camera. The dataset covers four tasks involving picking and placing objects, stacking, insertion, and dynamic manipulation, with 60 demonstrations per task and 240 demonstrations in total. To test scene specificity, we collect the demonstrations for each of the first three tasks (pick and place, stacking, and insertion) in two distinct scene configurations, with 30 demonstrations per configuration. Representative sequences from both configurations are shown in Appendix B.2.

Models. Our primary policy is Octo-Small (Team et al., 2024), initialized from the publicly released checkpoint pretrained on the Open X-Embodiment mixture (O'Neill et al., 2024). We further evaluate π_0 (Black et al., 2024) and OpenVLA (Kim et al., 2024) to assess SAFari's effectiveness across multiple policy models.

Baselines. We compare SAFari with four data curation baselines: CUPID (Agia et al., 2025), DataMIL (Dass et al., 2026), DemoInf (Hejna et al., 2025), and Demo-SCORE (Chen et al., 2025). For the controlled Octo-Small comparison, we reproduce each baseline's core scoring strategy, retain its highest ranked 50% of episodes, and train all methods for 100,000 steps under the same optimiza-

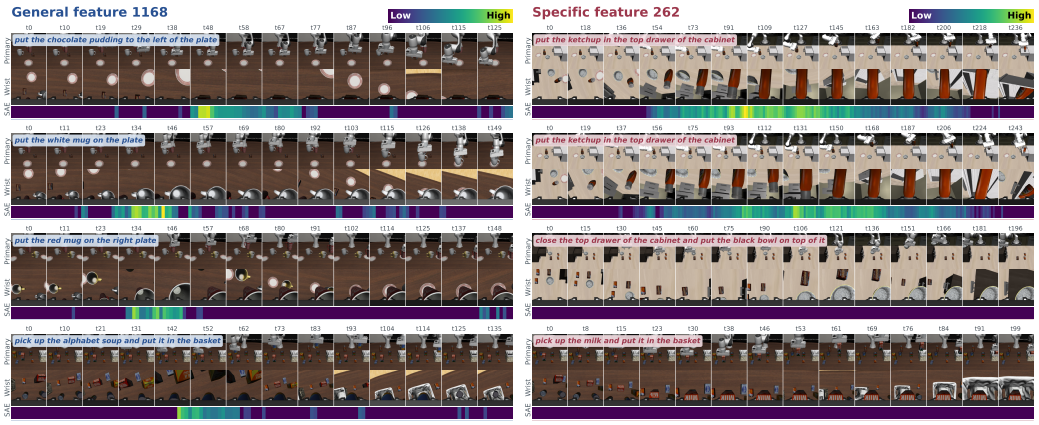

Figure 5: Representative general and specific SAE features on LIBERO-90. General features activate across tasks, whereas specific features capture persistent task or scene information.

tion settings. The π_0 experiments use a budget of 60,000 steps, whereas OpenVLA is trained for 150,005 steps. Appendix C provides the baseline reproduction details.

Benchmarks. We measure task success in closed loop in LIBERO (Liu et al., 2023). Controlled Octo comparisons use the same nine representative LIBERO-90 tasks, covering object placement, interaction with articulated objects, stacking, and manipulation over long horizons. Following the rollout protocol of DataMIL (Dass et al., 2026), we use 50 rollouts per task and evaluation seed. Each simulation setting uses three training seeds and two evaluation seeds per trained model, and we report the mean and sample standard deviation across the six training and evaluation seed combinations. For the real world benchmark, we evaluate each of the four xArm6 (Munoz et al., 2021) tasks with one training seed and one evaluation seed, using 50 trials per task because of the cost of physical evaluation. We report task success rates and their average without variance.

Implementation Details. Following the EventSAE (Jin et al., 2026), we train the SAE on activations from the residual stream of Octo's final transformer block (layer 11), immediately before the action head. This late representation is closely coupled to action prediction while leaving fewer downstream transformations to amplify SAE reconstruction error, making it a stable and behaviorally relevant site for analysis of features at the frame level. For feature statistics and scoring, we set the activation threshold to $u = 0.1$ and the numerical stability constant to $\epsilon = 10^{-8}$. We set $\delta \approx 0.6354$ to retain 1,000 of the 1,536 SAE features, comprising 487 general and 513 specific features. Unless otherwise stated, each general and specific stream retains a fraction $\rho = 0.5$ of all training frames under a budget set across the dataset rather than a fixed quota per episode. We use a minimum temporal spacing of $g = 2$ steps during the first selection pass. For a policy training budget of J optimization steps, we set $\alpha_{\min} = 0.1$, the sigmoid midpoint to $j_0 = J/2$, and the transition width to $\tau = J/5$. Additional policy training and SAE configurations are provided in Appendices B.1 and A.1.

4.2 Quantitative Results

Simulation Results. Figure 3(a–c) shows that SAFari improves Octo-Small, π_0, and OpenVLA. On Octo-Small, it reaches 78.2% at the full training budget, versus 61.5% for the strongest curation baseline, CUPID; even at half the budget, it surpasses full data training. The gains extend to π_0 and OpenVLA, which reach 86.0% and 81.0%, respectively, at their full budgets. These gains suggest that reducing frame redundancy and dynamically scheduling general and specific supervision improves the effectiveness of each training update. The consistent gains across policy architectures suggest that improving the training data stream, rather than changing the policy model, drives the performance improvement.

Real Robot Results. Figure 4 compares four real robot tasks of differing difficulty. SAFari leads on every task, averaging 60% success versus 54% for the strongest curation baseline and 48% for All-data. Success is generally higher for object placement and stacking than for insertion and dynamic manipulation, which require more precise contact or timely responses. Although the harder tasks

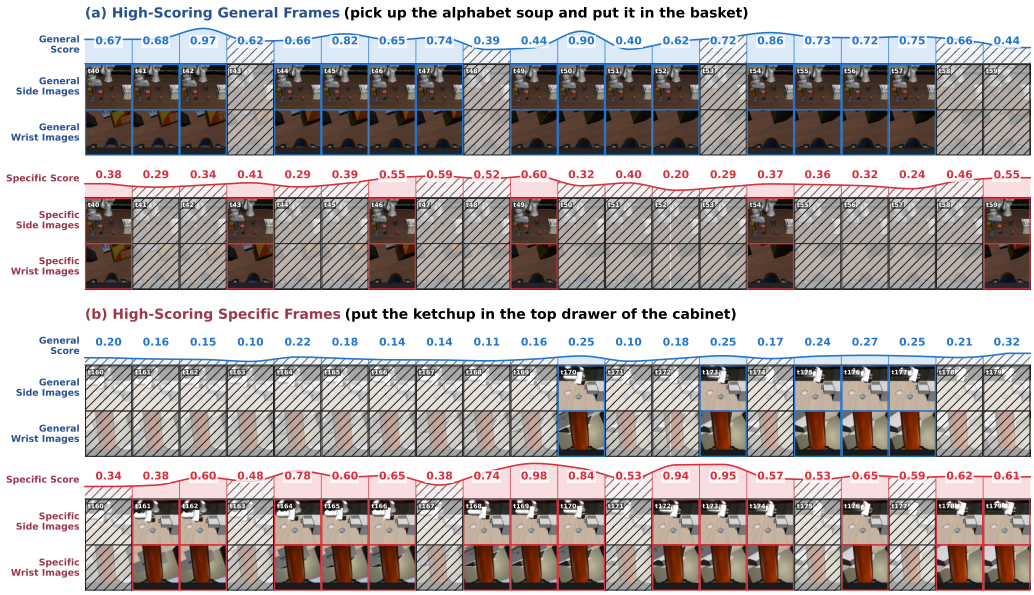

Figure 6: Examples of attribute-guided frame selection. (a) General frame selection. (b) Specific frame selection. A frame may receive nonzero scores from both attributes because multiple sparse features can activate simultaneously.

remain challenging in absolute terms, SAFari's lead across this range suggests that its benefits extend beyond simple object transfers to more demanding physical interactions.

4.3 Qualitative Analysis

General and Specific Feature Visualization. Figure 5 illustrates the activation patterns underlying the generality and specificity attributes. General feature 1168 responds across episodes involving pudding, mugs, and soup despite changes in objects and target locations. Its activation is concentrated around grasping rather than remaining uniformly high, which is consistent with a reusable manipulation pattern. In contrast, specific feature 262 remains active for extended periods when the ketchup bottle is clearly visible in two episodes moving ketchup to the top drawer. It is nearly absent in a different cabinet task and inactive in the episode moving milk to a basket. This pattern is consistent with a feature that captures a persistent object and scene configuration.

Frame Selection Visualization. Figure 6 shows how the two attributes produce different frame selections. In (a), high general scores retain frames around grasping the soup can. This behavior is consistent with the reusable manipulation pattern represented by general feature 1168. In (b), high specific scores emphasize the ketchup bottle and cabinet configuration associated with specific feature 262. The two streams can retain some of the same frames because a frame may show both reusable action information and details specific to a scene. Additional qualitative visualizations are provided in Appendix D.

4.4 Analysis and Ablation Studies

Retention Ratio and Training Budget. Figure 3(d) shows that frame curation becomes more beneficial as training proceeds. Under short budgets, moderate retention ratios perform similarly because redundant data have not been repeatedly revisited. As training continues, full-data training increasingly revisits similar observations, while curated streams provide more informative states. Retaining 50% of frames in each stream achieves the best result at 100k steps with 78.2% success, whereas more aggressive pruning saturates earlier. This reveals a tradeoff between coverage and redundancy, where moderate pruning reduces repeated supervision while preserving rare transitions and task-specific states needed for further improvement.

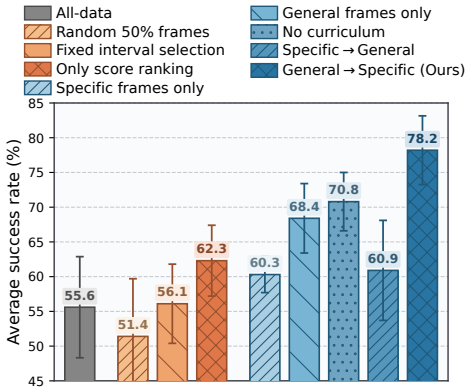

Figure 7: Component ablations of SAFari on LIBERO-90 using Octo-Small.

Table 1: Policy fine tuning cost and LIBERO success. Time excludes one-time offline data curation and evaluation.

Policy	Data	Steps	Time (h)	Success (%)
Octo	All-data	100k	4.4	55.6
	SAFari	100k	4.4	78.2 (↑ 22.6)
	SAFari	50k	2.2 (↓ 2.2)	66.2 (↑ 10.6)
π_0	All-data	60k	31.3	77.1
	SAFari	60k	31.2	86.0 (↑ 8.9)
	SAFari	30k	15.6 (↓ 15.7)	80.9 (↑ 3.8)
OpenVLA	All-data	150k	35.0	69.8
	SAFari	150k	35.0	81.0 (↑ 11.2)
	SAFari	75k	17.5 (↓ 17.5)	75.5 (↑ 5.7)

Frame Selection Strategy. In Figure 7, *All-data* uses all frames and achieves 55.6% success. *Random 50% frames* retains a random half but drops to 51.4%, suggesting that indiscriminate removal loses useful supervision. *Fixed interval selection* samples at regular temporal intervals and reaches 56.1%, only slightly above full data, as temporal thinning alone does not identify informative events. *Only score ranking* selects frames by attribute scores without temporal spacing or endpoint protection and reaches 62.3%, compared with 56.1% for fixed intervals. This contrast supports the usefulness of sparse attribute scores for identifying informative frames.

Attributes and Curriculum. *Specific frames only* trains on the specific stream and reaches 60.3%, only modestly above the 55.6% of *All-data*. This limited gain suggests that details tied to particular tasks and scenes are insufficient on their own to provide broadly reusable supervision. *General frames only* uses the general stream and reaches 68.4%, suggesting that shared manipulation patterns provide stronger standalone supervision. *No curriculum* randomly samples both streams with fixed weights and improves success to 70.8%, above either stream alone, indicating complementary supervision. *Specific → General* reverses the proposed ordering and falls to 60.9%, below the fixed mixture, suggesting that emphasizing specific contexts before shared patterns may hinder learning. In contrast, *General → Specific (Ours)* gradually shifts sampling toward specific frames with a sigmoid schedule and reaches 78.2%, compared with 70.8% without scheduling. Together, these results demonstrate the effectiveness of both attribute-guided frame curation and curriculum scheduling.

Training Time and Efficiency. Table 1 shows comparable policy fine tuning time for SAFari and All-data at the same number of optimization steps. Halving the optimization budget for Octo-Small approximately halves the elapsed time while achieving 66.2% success, versus 55.6% for the All-data policy trained for 100k steps. The π_0 experiments show the same twofold reduction within SAFari, with the shorter run also outperforming the Full data baseline trained with the longer budget. These results suggest that curation increases the amount of useful supervision per update by redirecting computation from repeated visual states toward frames showing complementary manipulation events and details specific to a task. Activation extraction, SAE fitting, frame scoring, and selection are performed only once using Octo, and the resulting masks are reused across policy models, making this one-time offline cost negligible relative to policy fine-tuning.

5 Conclusion

We presented SAFari, a framework that jointly curates high-value training frames and schedules their presentation to improve policy performance while accelerating robot imitation learning. SAFari uses sparse activation attributes to identify broadly reusable and task-specific frames, suppresses temporal redundancy while preserving informative transitions, and gradually shifts training from general to specific frames. Across LIBERO-90, SAFari consistently improves Octo-Small, π_0, and OpenVLA, outperforming full-data training with fewer optimization steps. Its consistent gains across four real-robot tasks further demonstrate its effectiveness beyond simulation. These results highlight the potential of data-centric training to reduce the computational cost of VLA learning without relying on larger models or more demonstrations.

AI USE STATEMENT

In preparing this submission, the authors used a generative AI assistant to polish the language and improve the readability of the manuscript. The authors reviewed all AI-assisted changes and take responsibility for the final content of this work, including its text, claims, and artifacts.

REFERENCES

Christopher Agia, Rohan Sinha, Jingyun Yang, Rika Antonova, Marco Pavone, Haruki Nishimura, Masha Itkina, and Jeannette Bohg. Cupid: Curating data your robot loves with influence functions. *arXiv preprint arXiv:2506.19121*, 2025.

Alon Albalak, Yanai Elazar, Sang Michael Xie, Shayne Longpre, Nathan Lambert, Xinyi Wang, Niklas Muennighoff, Bairu Hou, Liangming Pan, Haewon Jeong, et al. A survey on data selection for language models. *arXiv preprint arXiv:2402.16827*, 2024.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Bart Bussmann, Patrick Leask, and Neel Nanda. Batchtopk sparse autoencoders. *arXiv preprint arXiv:2412.06410*, 2024.

Annie S Chen, Alec M Lessing, Yuejiang Liu, and Chelsea Finn. Curating demonstrations using online experience. *arXiv preprint arXiv:2503.03707*, 2025.

Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via proxy: Efficient data selection for deep learning. *arXiv preprint arXiv:1906.11829*, 2019.

Shivin Dass, Alaa Khaddaj, Logan Engstrom, Aleksander Madry, Andrew Ilyas, and Roberto Martín-Martín. Datamil: Selecting data for robot imitation learning with datamodels. In *International Conference on Learning Representations*, volume 2026, pp. 1311–1336, 2026.

Leo Gao, Tom Dupre la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. In *International Conference on Learning Representations*, volume 2025, pp. 26721–26754, 2025.

Changheon Han, Jiho Lee, Hojun Lee, Yuseop Sim, Jurim Jeon, and Martin Byung-Guk Jun. Zero-shot autonomous robot manipulation via natural language. *Manufacturing Letters*, 42:16–20, 2024.

Joey Hejna, Suvir Mirchandani, Ashwin Balakrishna, Annie Xie, Ayzaan Wahid, Jonathan Tompson, Pannag Sanketi, Dhruv Shah, Coline Devin, and Dorsa Sadigh. Robot data curation with mutual information estimators. *arXiv preprint arXiv:2502.08623*, 2025.

Robert Huben, Hoagy Cunningham, Logan Smith, Aidan Ewart, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *International Conference on Learning Representations*, volume 2024, pp. 7827–7845, 2024.

Xinchen Jin, Aditya Chatterjee, Pranav Kumar, and Rohan Paleja. Event-grounded sparse autoencoders for vision-language-action policies. *arXiv preprint arXiv:2605.17204*, 2026.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Sachit Kuhar, Shuo Cheng, Shivang Chopra, Matthew Bronars, and Danfei Xu. Learning to discern: Imitating heterogeneous human demonstrations with preference and representation learning. In *Conference on Robot Learning*, pp. 1437–1449. PMLR, 2023.

M Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. volume 23, 2010.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8424–8445, 2022.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

Elias Munoz, Md Samiul Haque Sunny, Ivan Rulik, Javier D Sanjuan De Caro, and Mohammad H Rahman. Kinematics and workspace analysis of xarm6 robot for activities of daily living. In *Proceedings. International Conference on Industrial & Mechanical Engineering and Operations Management*, volume 4, pp. 885, 2021.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig, Barnabas Poczos, and Tom Mitchell. Competence-based curriculum learning for neural machine translation. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 1162–1172, 2019.

Aiden Swann, Lachlain McGranahan, Hugo Buurmeijer, Monroe Kennedy III, and Mac Schwager. Sparse autoencoders reveal interpretable and steerable features in vla models. *arXiv preprint arXiv:2603.19183*, 2026.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Kushal Tirumala, Daniel Simig, Armen Aghajanyan, and Ari Morcos. D4: Improving llm pretraining via document de-duplication and diversification. *Advances in Neural Information Processing Systems*, 36:53983–53995, 2023.

Haokun Wang, Xiaobo Liu, Nuofan Qiu, Ning Guo, Fang Wan, and Chaoyang Song. Deepclaw 2.0: A data collection platform for learning human manipulation. *Frontiers in Robotics and AI*, 9:787291, 2022.

Justin Yu, Andrew Goldberg, Kavish Kondap, Karim El-Refai, Ethan Ransing, Qianzhong Chen, Mac Schwager, Fred Shentu, Philipp Wu, and Ken Goldberg. Warp-rm: A warp-augmented relative progress reward model for data curation. *arXiv preprint arXiv:2606.28320*, 2026.

Yu Zhang, Yuqi Xie, Huihan Liu, Rutav Shah, Michael Wan, Linxi Fan, and Yuke Zhu. Scizor: A self-supervised approach to data curation for large-scale imitation learning. *arXiv preprint arXiv:2505.22626*, 2025.

Yijun Zhou, Muhan Hou, and Kim Baraka. Static is not enough: A comparative study of vr and spacemouse in static and dynamic teleoperation tasks. In *Companion Proceedings of the 21st ACM/IEEE International Conference on Human-Robot Interaction*, pp. 307–311, 2026.

A Feature Attribute Modeling Details

This section describes how we train the SAE and turn its activations into feature attributes. We detail the SAE objective and hyperparameters, then define the activation statistics and report the feature classification setup and results.

A.1 Representation Training Details

The SAE training procedure uses the activation normalization, auxiliary objective, and hyperparameters detailed below. We rescale every frozen activation $h \in \mathcal{H}$ by a single constant computed across the dataset:

$$\bar{h} = \frac{h}{c_{\text{act}}}, \qquad c_{\text{act}} = \sqrt{\mathbb{E}_{h\sim\mathcal{H}}\|h\|_2^2}. \tag{8}$$

For a minibatch $\mathcal{B}$ of size M, let $e_b = \bar{h}_b - \hat{\bar{h}}_b$ be the reconstruction residual and $\bar{e} = M^{-1}\sum_{b\in\mathcal{B}} e_b$ its batch mean. The vector z_b^{aux} retains the largest rectified activations before sparsification among features that have not fired within the period used to identify inactive features. We use the normalized AuxK loss

$$\mathcal{L}_{\text{aux}} = \frac{\sum_{b\in\mathcal{B}} \|e_b - W_{\text{dec}} z_b^{\text{aux}}\|_2^2}{\sum_{b\in\mathcal{B}} \|e_b - \bar{e}\|_2^2}. \tag{9}$$

The denominator normalizes the auxiliary reconstruction error by the residual variance within the batch. Decoder columns are constrained to unit norm throughout training. For Octo, we extract $d = 384$ activations from layer 11 and train a dictionary of $|\mathcal{S}| = 1536$ features with $k = 32$, $M = 1024$, $k_{\text{aux}} = 192$, and $\alpha_{\text{aux}} = 1/32$. We optimize the SAE for 20,000 steps using Adam with a learning rate of 10^{-4} and mark a feature as dead if it does not activate for 10^7 processed examples.

A.2 Feature Statistics and Classification Details

Feature classification relies on two activation statistics, defined below together with the procedure for fitting the classifier. All binary activation statistics below use the fixed, prespecified threshold u defined in the main text; it is neither learned from the labeled features nor adapted separately for each feature. Let $\mathcal{E}^s$ denote the set of episodes in which feature s exceeds this threshold at least once. Its episode coverage is the fraction of all episodes covered by the feature:

$$c^s = \frac{|\mathcal{E}^s|}{N}, \qquad \mathcal{E}^s = \left\{ i : \exists\, t,\ I_{i,t}^s = 1 \right\}. \tag{10}$$

To measure sustained activation, we set $I_{i,-1}^s = 0$ and count the activation onsets in episode i as

$$o_i^s = \sum_{t=0}^{T_i-1} I_{i,t}^s \left(1 - I_{i,t-1}^s\right). \tag{11}$$

Its sustained activation duration is the mean continuous run length over active episodes, normalized by the corresponding episode length:

$$\bar{\ell}^s = \frac{1}{|\mathcal{E}^s|} \sum_{i\in\mathcal{E}^s} \frac{\sum_{t=0}^{T_i-1} I_{i,t}^s}{T_i o_i^s}. \tag{12}$$

We manually label a subset of features by jointly inspecting their frames with the strongest activations, their temporal activation traces, and the consistency of the associated visual or behavioral pattern across tasks and episodes. Ambiguous features are excluded from the labeled set. The two component vector $(c^s, \bar{\ell}^s)$ is used to train the binary classifier Φ. Its output $\widetilde{\eta}^s$ is interpreted as the generality strength, while $\widehat{\eta}^s = 1 - \widetilde{\eta}^s$ represents the complementary specificity strength.

A.3 Classifier Configuration and Results

We evaluate 1,536 layer-11 Octo-Small SAE features on 669,043 frames from 4,500 LIBERO-90 episodes. The attribute decision threshold δ retains 1,000 features for frame scoring, comprising 487 general and 513 specific features. A frame counts as active only if $z_{i,t}^s > u$. Coverage and

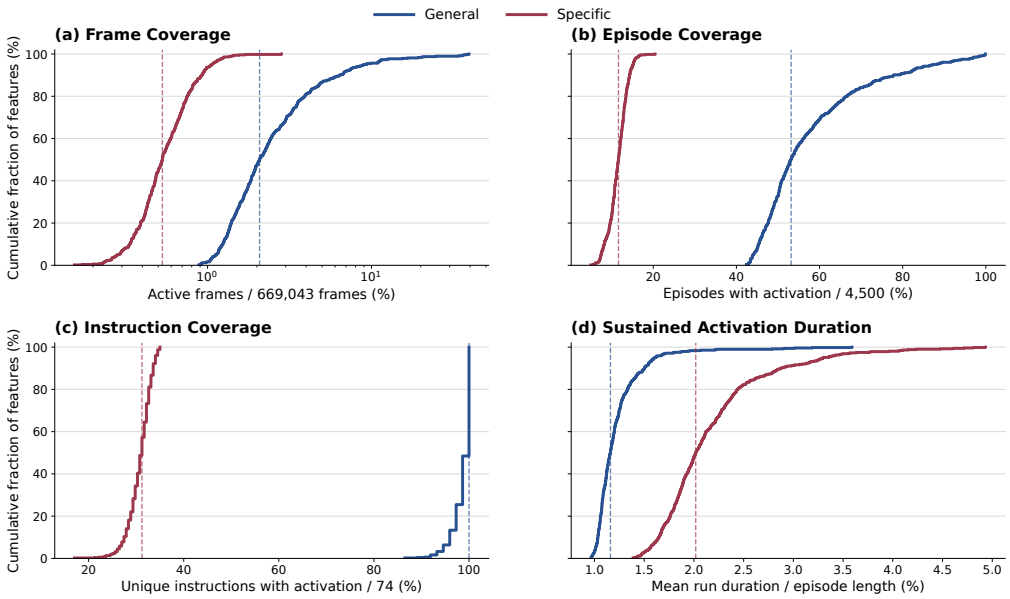

Figure 8: The horizontal axis shows each feature's frame, episode, or instruction coverage (a–c), or mean sustained run duration relative to episode length (d), all in percent; the vertical axis shows the cumulative percentage of features. (c) uses 74 distinct instruction texts because some of the 90 tasks share an instruction. General features have broader coverage but shorter runs, whereas specific features are less widespread and more sustained; dashed lines mark group medians.

duration follow the definitions above; duration is normalized by episode length and onset count before averaging over active episodes.

Twenty visually reviewed seeds from this dictionary comprise 10 general features recurring across objects, tasks, or scenes and 10 specific features tied to a particular combination of object and scene or a specific task context. An ℓ_2-regularized logistic classifier uses $(c^s, \bar{\ell}^s)$, inputs standardized within each training fold, balanced class weights, $C = 1$, L-BFGS, at most 10,000 iterations, and seed 0. In the original metric units,

$$P(\text{general} \mid c^s, \bar{\ell}^s) = \sigma\big(-0.947 + 6.284c^s - 108.613\bar{\ell}^s\big)\,. \tag{13}$$

Positive coverage and negative duration coefficients associate generality with broader episode coverage and specificity with longer runs. Since $\bar{\ell}^s$ typically lies near 0.01–0.02, increasing it by 0.01 changes log odds by $-108.613 \times 0.01 = -1.086$. Different input ranges make raw coefficient magnitudes incomparable.

Refitting after excluding each seed in turn yields AUROC 0.940 and macro-F1 0.900, indicating strong, balanced separation. The selected clear examples are not an independent test set; these scores show feasibility, not final generalization.

Figure 8 shows that specific features cover fewer frames, episodes, and instructions but persist longer, consistent with task or scene context. General features have broader coverage and shorter, reusable bursts associated with events.

B Training and Evaluation Details

This section documents policy training and evaluation protocols in simulation and on the real robot, together with frame retention statistics. It also provides Octo-Small results for each task and the comparison between the representative subset of nine tasks and the full LIBERO-90 benchmark.

Table 2: LIBERO-90 policy fine tuning configurations. The π_0 batch sizes follow the experiment variants reported in the main text. OpenVLA uses LoRA on all linear layers; Octo-Small and π_0 are fully fine tuned.

Configuration	Octo-Small	π_0	OpenVLA
Initialization	Octo-Small pretrained on OXE	pi0_base	OpenVLA-7B
Fine Tuning method	Full FT	Full FT	LoRA
Camera input	Primary 256^2; wrist 128^2	Primary and wrist, both 224^2	Primary only, 224^2
GPUs	1	2	2
Global batch size	128	32	16
Optimizer	AdamW	AdamW	AdamW
Learning rate	Peak 3×10^{-4}	5×10^{-5}	5×10^{-4}
Weight decay	0.01	0	0.01
Gradient clipping	1.0	1.0	None
Image augmentation	None	None	Enabled
Shuffle buffer	10,000	Dataset repetition/shuffling	100,000
Maximum training steps	100,000	60,000	150,005

B.1 Simulation Policy Training Configurations

The LIBERO-90 fine tuning configurations for Octo-Small, π_0, and OpenVLA are summarized below. All three policies are fine tuned on the same LIBERO-90 training set with 4,500 demonstrations. Unless a row explicitly differs, the data selection methods change only the sampling stream and leave the policy optimization settings unchanged. Removing a frame from the training samples does not remove its corresponding action label from the predicted action sequence targets.

B.2 Real Robot Policy Training Configuration

This subsection describes policy training and adaptation of the feature analysis pipeline to real robot data. The SAE must be trained again on activations from the real robot demonstrations, and its features must be reviewed and labeled again to fit an attribute classifier for this domain. Feature identities and their general or specific labels are not assumed to remain valid across simulation and real robot data. Coverage, duration, and frame scores are therefore computed on the real robot dataset before constructing its two streams. For the real robot experiments, we fine tune the base π_0 checkpoint on the xArm6 (Munoz et al., 2021) dataset covering four tasks described in Section 4.1; the observation inputs and optimization settings are detailed below. The dataset contains 60 demonstrations per task. For pick and place, stacking, and insertion, we collect 30 demonstrations in each of two distinct scene configurations to assess whether the selection method remains useful across scene-specific visual contexts. Figure 9 shows representative sequences from both configurations for these three tasks. Demonstration videos from our collected real robot dataset are provided in the Supplementary Material. The implementation jointly conditions on the D435i (Wang et al., 2022) third person view, the L515 (Han et al., 2024) wrist view, language instructions, and robot proprioception. Both camera streams are resized to 224×224. We fully fine tune the policy for 30,000 training steps using the settings in Table 3. Full data, competing curation methods, and SAFari use identical model and optimization settings.

B.3 Simulation and Real Robot Evaluation Protocols

Our evaluation protocol covers a representative subset of nine tasks, the full LIBERO-90 benchmark, and four real robot tasks. Evaluating every checkpoint on all 90 LIBERO-90 tasks is computationally expensive. For rapid model development and controlled ablations, we construct a fixed representative subset by selecting one task from each of the nine scene groups, resulting in task IDs 10, 21, 34, 46, 63, 72, 73, 84, and 87. We use this same subset for every controlled comparison. For each setting, we train with three random seeds and evaluate every trained model with two random seeds, using 50 rollouts per task and evaluation seed. This yields six success rates per task and 2,700 rollouts on the nine task subset. We report the mean and sample standard deviation across the six training and evaluation seed combinations. For aggregate results over multiple tasks, we first average task success rates within each training–evaluation seed combination and then compute the mean and sample standard deviation across the six resulting values. Thus, the aggregate deviation



Figure 9: Representative real robot sequences in four tasks, ordered from top to bottom: (a) putting a ball into a basket, (b) stacking three blocks, (c) plugging in a charger, and (d) removing a block from a turntable. For the first three tasks, the first pair of rows shows an original training episode (episodes 25, 65, and 5, respectively), and the second pair shows the other scene configuration. The final pair shows the turntable episode (episode 45). Each view contains 10 frames sampled uniformly over its complete sequence; paired views use matching frame indices.

is not obtained by averaging task-level deviations and represents variation across seed combinations rather than a confidence interval or standard error.

To validate that this subset provides a useful estimate of results on the full benchmark, we additionally evaluate the same checkpoints on all 90 LIBERO-90 tasks, with 50 rollouts per task and seed. This paired protocol isolates the effect of task coverage: the three trained checkpoints, training bud-

Table 3: Training configuration for the real robot π_0 experiments.

Configuration	Value
Initialization	pi0_base
Fine Tuning method	Full fine tuning
Camera input	Primary and wrist, both 224^2
Action dimension	7
Action chunk length	50
Training steps	30,000
Batch size per device	8
Optimizer	AdamW
Learning rate	2×10^{-5}
Weight decay	0
Warmup ratio	0.1
Numerical precision	bfloat16 mixed precision

Table 4: Octo-Small success rates on the nine representative LIBERO-90 tasks. Each entry is mean success rate (percent) $\pm$ sample standard deviation (percentage points). Avg. matches Figure 3(a). Statistics summarize three training seeds and two evaluation seeds per trained model, with 50 rollouts per task and evaluation seed.

Method	Task 10	Task 21	Task 34	Task 46	Task 63	Task 72	Task 73	Task 84	Task 87	Avg.
All-data	86.0 $\pm$ 3.3	60.3 $\pm$ 9.6	62.0 $\pm$ 2.5	68.3 $\pm$ 7.5	30.0 $\pm$ 11.6	52.3 $\pm$ 5.4	15.7 $\pm$ 8.6	44.3 $\pm$ 2.3	81.3 $\pm$ 10.1	55.6 $\pm$ 6.4
DataMIL	83.7 $\pm$ 1.5	49.7 $\pm$ 2.0	38.0 $\pm$ 8.8	38.0 $\pm$ 8.1	35.7 $\pm$ 12.0	38.0 $\pm$ 5.8	35.3 $\pm$ 9.0	35.3 $\pm$ 2.7	70.3 $\pm$ 4.5	47.1 $\pm$ 5.4
DemoInf	85.7 $\pm$ 1.5	24.0 $\pm$ 10.1	42.3 $\pm$ 5.4	16.7 $\pm$ 8.4	57.0 $\pm$ 12.4	67.7 $\pm$ 6.3	40.7 $\pm$ 9.4	63.3 $\pm$ 3.3	54.3 $\pm$ 5.0	50.2 $\pm$ 6.8
Demo-SCORE	87.3 $\pm$ 4.0	34.0 $\pm$ 9.6	66.3 $\pm$ 2.3	27.3 $\pm$ 7.7	55.0 $\pm$ 11.6	80.7 $\pm$ 5.5	48.0 $\pm$ 8.6	27.3 $\pm$ 2.4	84.3 $\pm$ 8.1	56.7 $\pm$ 6.0
CUPID	87.0 $\pm$ 1.1	70.3 $\pm$ 4.0	70.3 $\pm$ 6.9	56.0 $\pm$ 8.0	29.7 $\pm$ 12.0	66.0 $\pm$ 5.8	40.7 $\pm$ 9.0	43.7 $\pm$ 2.7	89.7 $\pm$ 4.5	61.5 $\pm$ 4.8
SAFari (50k)	99.7 $\pm$ 0.8	36.3 $\pm$ 11.4	95.7 $\pm$ 5.2	61.7 $\pm$ 8.4	30.3 $\pm$ 12.4	83.7 $\pm$ 6.3	44.0 $\pm$ 9.5	60.0 $\pm$ 3.1	84.3 $\pm$ 7.0	66.2 $\pm$ 6.3
SAFari (100k)	94.3 $\pm$ 0.8	62.3 $\pm$ 9.6	90.0 $\pm$ 2.5	91.7 $\pm$ 7.5	42.3 $\pm$ 11.6	83.7 $\pm$ 5.4	64.3 $\pm$ 8.5	84.0 $\pm$ 2.5	91.0 $\pm$ 6.1	78.2 $\pm$ 5.8

get, and two evaluation seeds per checkpoint are identical between the evaluations on nine and 90 tasks. The close agreement in Table 5 suggests that the representative subset preserves the relative behavior of Full data and SAFari while reducing evaluation cost by a factor of ten.

For real robot validation, we evaluate each π_0 policy on all four xArm6 (Munoz et al., 2021) tasks. Because hardware trials are time consuming and costly, we use one training seed and one evaluation seed. Each policy is tested in 50 closed loop trials per task, for 200 trials in total. We report the observed success rate for each task and its average across the four tasks, without variance. All methods use the same protocol for initial states and success criteria for each task.

B.3.1 Detailed Simulation Results

Table 4 breaks down the Octo-Small results in Figure 3(a) across the nine representative LIBERO-90 tasks. The nine tasks cover object placement, interaction with articulated objects, stacking, and manipulation over long horizons. Each cell reports the mean success rate $\pm$ sample standard deviation across the six training and evaluation seed combinations.

We assess the agreement between the representative evaluations on nine and all 90 tasks using the same trained checkpoints. Table 5 presents the paired averages for Octo-Small and π_0 under matched training budgets. The displayed evaluations preserve the ordering between Full data and SAFari. Comparing the same checkpoints across task sets helps assess whether the representative subset reflects the full benchmark.

B.4 Frame Retention Statistics

This subsection summarizes the overlap between the two selected streams and analyzes how frame retention varies across tasks, distinguishing per-stream retention from total unique frame retention.

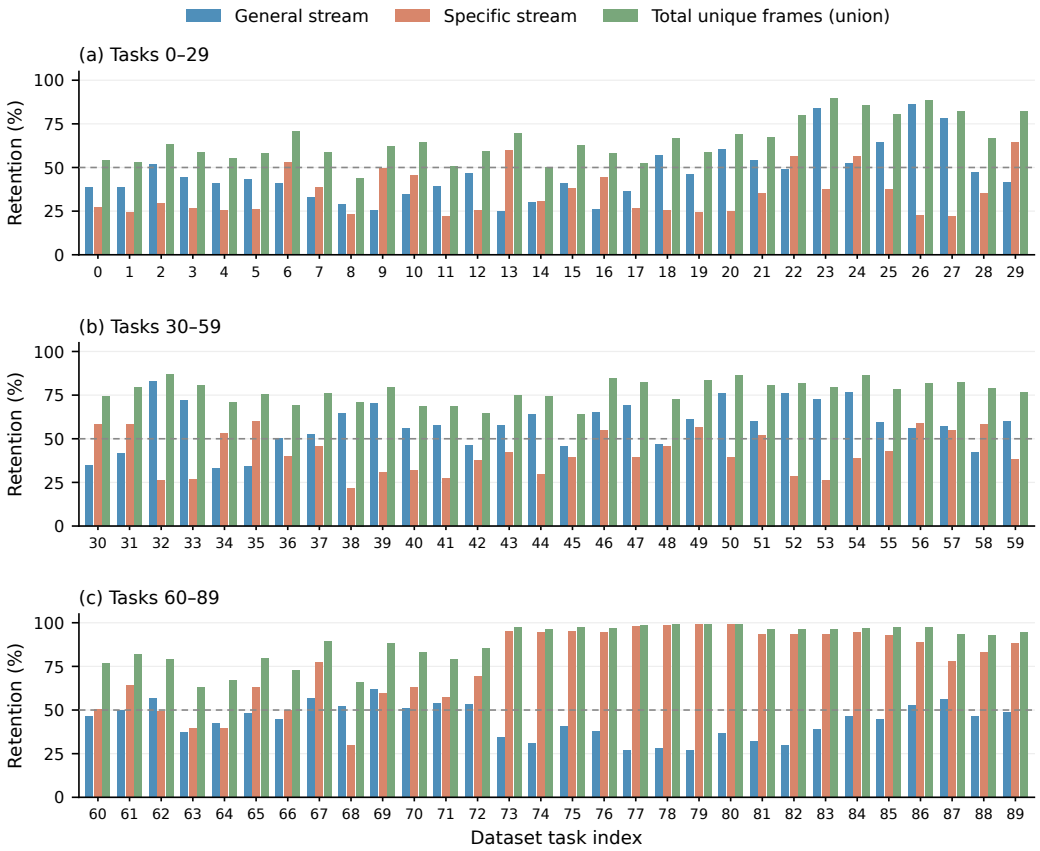

Figure 10: Frame retention across all 90 tasks, grouped into (a) 0–29, (b) 30–59, and (c) 60–89 for readability. The horizontal axis shows the dataset task index and the vertical axis shows the percentage of that task's frames retained. Within each task group, blue and red bars indicate retention in the general and specific streams, respectively; green bars count unique frames in their union. The dashed line marks 50%, the global per-stream budget rather than a quota for each task. Task indices follow the 90 consecutive blocks of 50 demonstrations in the training dataset.

Table 5: Success rates on nine versus all 90 tasks for the same checkpoints.

Policy	Data	Steps	9 tasks	90 tasks
Octo-Small	Full data	100k	55.6 ± 6.4	57.4 ± 1.6
Octo-Small	SAFari	100k	78.2 ± 5.8	78.4 ± 1.7
π_0	Full data	60k	77.1 ± 2.9	78.4 ± 1.2
π_0	SAFari	30k	80.9 ± 0.7	81.9 ± 2.3

Let $M = \sum_i T_i$, and let G and S denote the sets of frames selected for the general and specific streams. The per-stream retention rates are $|G|/M$ and $|S|/M$, whereas the total unique frame retention rate is

$$\rho_{\text{unique}} = \frac{|G \cup S|}{M} = \frac{|G| + |S| - |G \cap S|}{M}. \tag{14}$$

A frame selected by both streams is counted once in the union. Thus, retaining 50% per stream does not imply retaining 50% of all unique frames. All SAFari retention settings in the main text refer to per-stream retention unless explicitly identified as total unique retention.

Retention Statistics and Analysis. Across 669,043 valid frames, each stream retains 334,522 frames. Their intersection contains 154,655 frames, while their union contains 514,389 frames,

yielding a total unique frame retention rate of 76.88%. The two streams share some frames while retaining distinct subsets, and 154,654 frames (23.12%) are excluded from both. This pattern is consistent with differentiated frame valuation: frames can be prioritized for both attributes, for only one, or for neither. The resulting selection balances shared and attribute specific coverage while discarding frames that neither stream prioritizes. Figure 10 shows how retention varies across tasks. Because selection uses a global budget, tasks with higher frame scores receive larger retained fractions rather than a fixed quota. This is particularly evident for tasks 73–89, whose specific stream retains 92.92% of their frames, compared with 40.56% for tasks 0–72. These tasks mainly involve books, caddies, mugs, and shelves, and their mean specificity score is approximately twice that of the remaining tasks.

C Baseline Reproduction Details

This section explains how we reproduce and compare the four data curation baselines under a common Octo-Small training and evaluation protocol. We first state the shared settings, then describe the scoring and selection procedure used for each baseline. For a controlled comparison, all reproduced baselines retain 50% of the 4,500 LIBERO-90 episodes, initialize from the same Octo-Small checkpoint pretrained on OXE, and fine tune for 100,000 steps with the optimization settings in Table 2. Following the rollout protocol of DataMIL (Dass et al., 2026), we evaluate each baseline on the same nine tasks with 50 rollouts per task and evaluation seed. Each setting uses three training seeds and two evaluation seeds per trained model; reported results use the mean and sample standard deviation across the six combinations, as defined in Appendix B.3. Only how episodes are scored and selected differs across methods.

- **CUPID (Agia et al., 2025).** We adapt the official influence pipeline based on TRAK to Octo. An All-data policy trained for 100,000 steps first generates 450 labeled rollouts on the nine evaluation tasks. For every training frame and rollout state, we compute gradients of the action loss with respect to the final layer of the action head, project them to 2,048 dimensions, and accumulate the empirical gradient Gram matrix. Each demonstration is scored by its whitened influence on successful rollouts minus its influence on failed rollouts; we retain the 50% of episodes with the highest scores.
- **DataMIL (Dass et al., 2026).** We use the episode coefficients conditioned on the task produced by the released datamodel procedure. Because coefficient scales differ across target tasks, we convert the score vector for each task to normalized ranks, average the ranks across tasks, and retain the 50% of episodes with the lowest aggregate rank, following DataMIL selection direction. This selected multitask subset is then used for the otherwise unchanged Octo fine tuning run.
- **DemoInf (Hejna et al., 2025).** We reproduce the mutual information scoring principle using cached Octo representations and actions. Separate state and action VAEs are trained for 20,000 steps, after which a KSG estimator computes mutual information scores for frames over four randomized repeats. We average the standardized frame scores within each episode and retain the top 50% of episodes before training Octo.
- **Demo-SCORE (Chen et al., 2025).** We collect 20 rollouts per task from intermediate and final All-data checkpoints at 30,000 and 100,000 steps, respectively, and label their actions by rollout success. An action classifier with two hidden layers is trained for 5,000 steps with batch size 512, dropout 0.3, and AdamW. Each LIBERO-90 episode receives the mean predicted success probability of its actions, and the top 50% are retained for Octo fine tuning.

D Additional Visualization Results

This section presents additional visualizations of the learned feature attributes and resulting frame selections. We show a general feature across tasks, a feature that memorizes a scene, and frame selections from a complete episode.

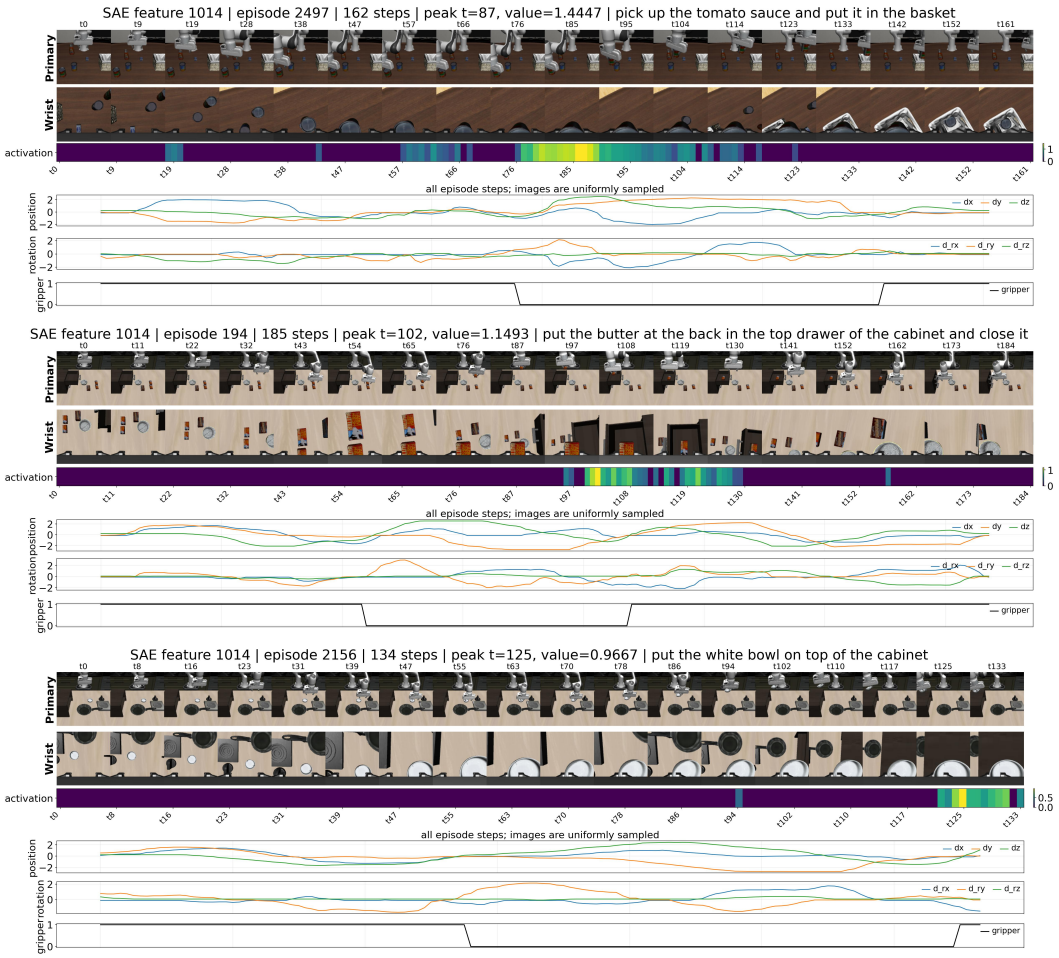

Figure 11: Feature 1014 across three full episodes, shown top to bottom: tomato sauce to basket (2497), butter to the cabinet's top drawer (194), and white bowl to cabinet top (2156). Each row pairs primary/wrist views with feature activation, action traces, and gripper state. In each title, "peak t" is the timestep of maximum activation and "value" is its unnormalized SAE activation magnitude. The activation heatmap is normalized separately within each episode, so absolute magnitudes should be compared using these values (1.445, 1.149, and 0.967).

D.1 Example of a General Feature Across Tasks

To examine generality across tasks, Figure 11 compares feature 1014 across three different manipulation tasks. The tomato sauce container, butter, and bowl, as well as their targets, differ, yet feature 1014 has a clear activation episode after the gripper closes and the object begins moving toward its destination. The peak occurs at $t = 87$ for tomato sauce moving toward a basket, $t = 102$ for butter approaching the top drawer, and $t = 125$ for a white bowl approaching the cabinet top. The corresponding action events have positive y components ($+0.467$, $+1.325$, and $+0.565$), while their z components differ ($+0.903$, -0.697, and -0.086). Together with the changing views from the primary and wrist cameras, this supports a shared *transport after grasping or approach to the goal* phase rather than a detector for one object or one fixed Cartesian direction. The latter two peaks occur close to the placement/release transition, so the feature may also respond to goal proximity. These examples illustrate recurring activation across different tasks and object contexts.

D.2 Example of a Feature Specific to a Scene

As an example of activation specific to a scene, Figure 12 contrasts feature 791 in two demonstrations of the same mug placement task and two controls from different scenes. In episodes 1740 and 1732, the yellow and white mug is placed in front of the white mug. The feature is active on 74.6% and 75.0% of frames, respectively, with uninterrupted runs of 91/134 and 75/112 steps. In contrast, it remains exactly zero throughout the selected episode 2027, in which a frying pan is moved to a shelf, and episode 0 involving a black bowl and drawer. The camera views and sustained traces together suggest that this unit responds to the particular context of the mug and spatial layout, rather than to a generic grasp or transport phase. This is a hypothesis about scene specificity, not proof that it detects one physical mug: object appearance, relative arrangement, and language are confounded in these demonstrations.

D.3 Frame Selection Across a Complete Episode

Figure 13 illustrates the two streams across all 149 frames of a demonstration that places a red mug on the right plate. The general stream retains 81 frames and the specific stream retains 61, reflecting their different attribute scores. Shared selections indicate that reusable manipulation patterns and scene context can occur in the same frame, while differing selections show how the two streams provide complementary supervision.

Both masks preserve the first and last frames. Selection first prioritizes frames separated by at least $g = 2$ time steps within each episode, then fills the remaining global budget in descending score order without the spacing constraint. Consequently, adjacent frames can still be retained during backfilling. The 50% retention budget applies to each stream across the dataset, not to this episode or to the union of both streams.

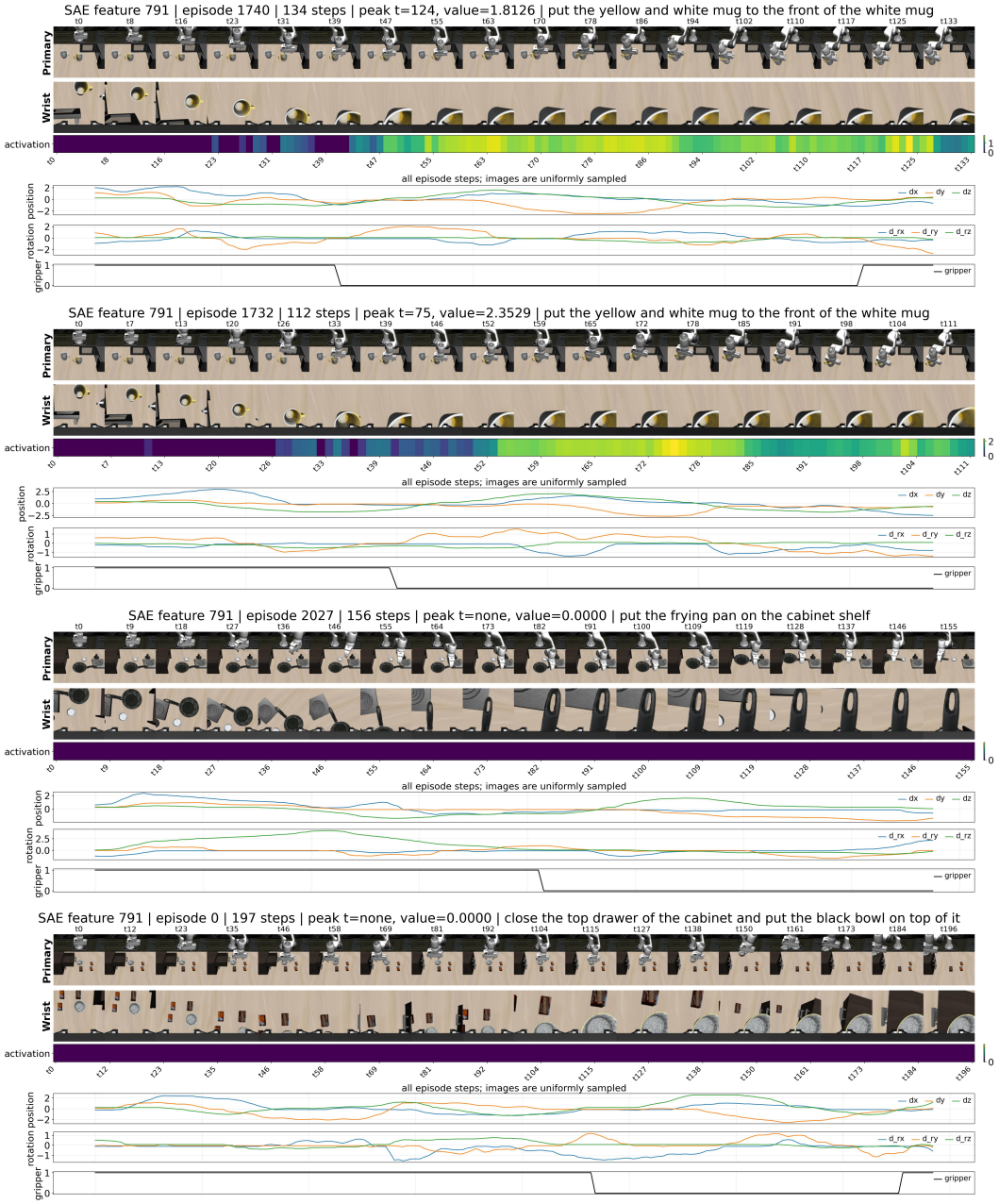

Figure 12: Feature 791 in two yellow and white mug placement episodes (1740, 1732) and two controls from different scenes (2027, 0), top to bottom. The long bright activation runs in the first two rows contrast with zero activation throughout the last two.

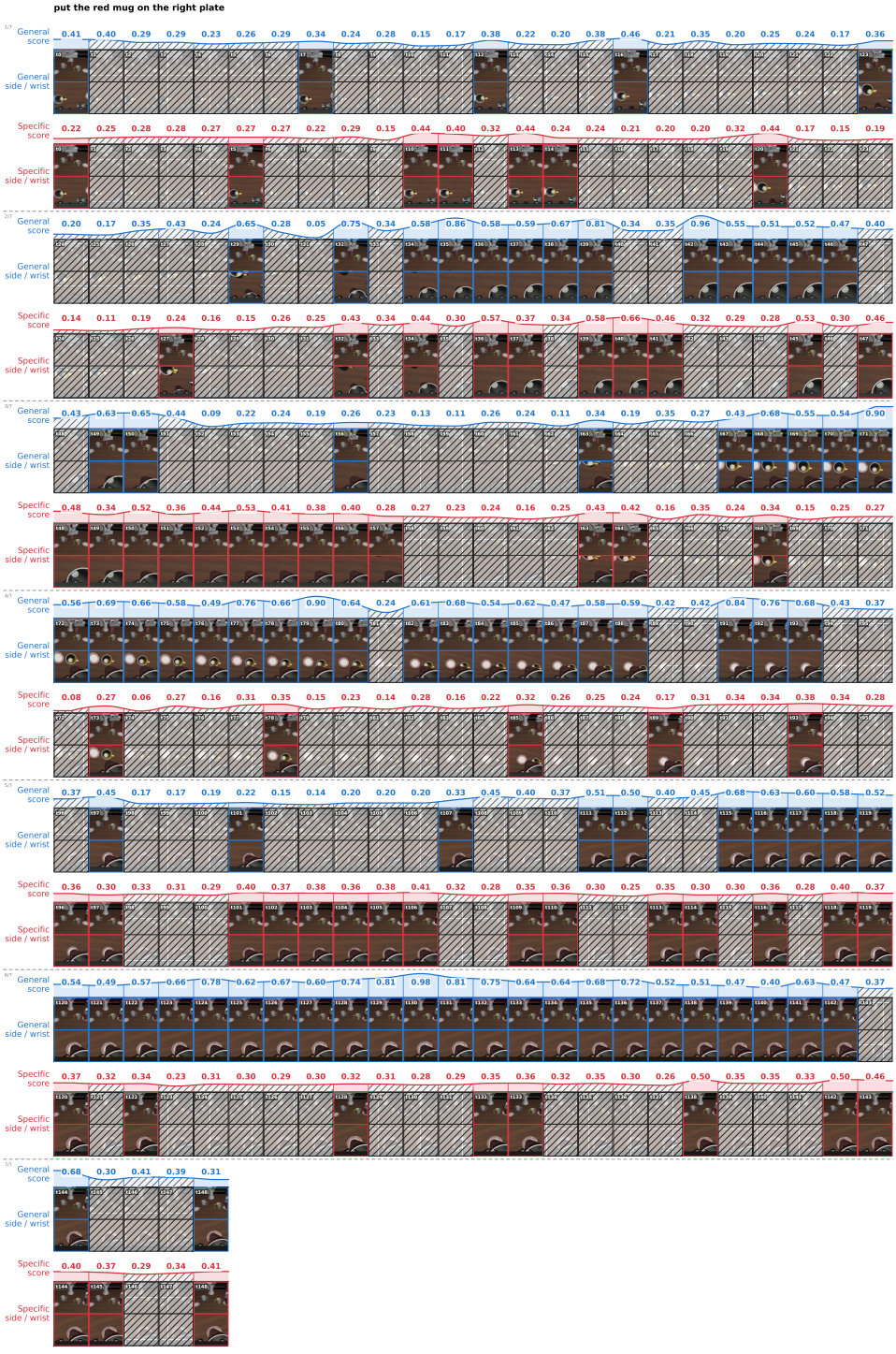

Figure 13: General and specific frame selection across LIBERO-90 episode 3330 (red mug to right plate), with 24 frames per row. Scores appear above aligned side and wrist views; colored borders mark retained frames, and white hatching marks removed frames. Sometimes visual differences between the two streams may be subtle because each score aggregates contributions from multiple sparse features.