SAFER: SELECTIVE FFN READOUT FOR CROSS-TASK VLA FAILURE DETECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Reliable failure detection is essential for deploying vision-language-action models (VLAs) across diverse manipulation tasks. Existing methods have explored using VLA hidden states to detect execution failures, but these aggregated representations do not reveal which internal computational contributions are most useful for failure detection on unseen tasks. We propose SAFER, a selective feed-forward network (FFN) readout method for cross-task VLA failure detection. The method uses successful and failed rollouts from the training set and performs training-free FFN neuron selection. It ranks neurons based on the separability of their activations between successful and failed rollouts and selects a subset of neurons for failure monitoring. At each execution step, we construct the detector input representation by accumulating the activation-weighted output directions of the selected neurons. The selected neuron subset is directly applied to unseen tasks, while the VLA parameters remain frozen and the original action computation process is preserved. Experiments show that SAFER can effectively improve failure detection performance on unseen tasks compared with full hidden-state readout.

1 INTRODUCTION

Vision-language-action models (VLAs) combine visual observations and language instructions to produce robot actions, providing a foundation for policies that operate across diverse manipulation tasks (Kim et al., 2024). Broad task coverage, however, does not guarantee successful execution. A robot may miss a grasp, lose an object, or continue acting without making progress toward its goal. Detecting such failures can support decisions to stop, retry, or request assistance. For a generalist policy, this requires more than recognizing failures on familiar tasks: a useful monitor must also transfer to tasks absent from its own training data.

Transformer block outputs provide a useful source of signals for VLA failure monitoring. SAFE (Gu et al., 2025) trains failure detectors on these features using successful and failed rollouts, and evaluates their transfer to unseen tasks. However, a block output aggregates information through attention, feed-forward computation, and residual connections, leaving the detector to identify useful signals within the resulting representation. This raises a finer-grained question: *which computational contributions should a failure monitor read to support detection across tasks?*

Prior analyses of feed-forward networks (FFNs) suggest that their constituent computations offer a useful level of analysis. FFN outputs can be interpreted as activation-weighted combinations of value directions (Geva et al., 2021; 2022), while supervised neuron analysis and sparse probing identify neuron subsets predictive of labeled concepts (Dalvi et al., 2019; Gurnee et al., 2023). In VLAs, FFN analyses have identified control concepts such as speed and direction that enable behavior steering through activation interventions (Häon et al., 2025), and retrospective probing has revealed failure signals under visual distribution shifts (Mahato & Ren, 2026). These findings motivate us to examine FFN neuron contributions as units of representation selection: *can execution outcomes identify a subset whose contributions support failure detection on unseen tasks?*

We address this question with SAFER, which constructs a monitoring representation from a subset of FFN neuron contributions. The readout uses execution outcomes to guide neuron selection: given successful and failed training rollouts, we score each neuron's rollout-averaged activation by its ability to distinguish these outcomes and retain the top-ranked subset. At each execution step, we sum the selected neurons' activation-weighted output directions to form the input to a failure

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.

detector. The selected subset is fixed after training and applied unchanged to unseen tasks, while the VLA remains frozen and its action computation is preserved. Selection uses only rollout-level outcome labels, without requiring annotations of when failures begin.

Our evaluation examines failure detection on both seen and unseen manipulation tasks, with unseen tasks reserved exclusively for testing. The primary comparison holds the policy, rollout data, and detector training procedure fixed while replacing full hidden-state readout with selective FFN readout. Experiments show that selective FFN readout improves failure detection over full hidden-state readout on unseen tasks, supporting the value of explicitly selecting the FFN contributions exposed to the monitor.

Our contributions are threefold:

- We investigate FFN neuron selection for cross-task VLA failure detection, examining whether selected neuron contributions provide a more effective monitoring representation than complete hidden states.
- We propose an outcome-guided representation construction method that selects FFN neurons according to their success–failure discrimination and combines their activation-weighted output directions into features for failure monitoring.
- We demonstrate accurate failure detection on both seen and unseen manipulation tasks, showing the effectiveness of the selected FFN representation for cross-task monitoring.

2 Related Work

Vision-Language-Action Models. Generalist robot policies learn language-conditioned manipulation from diverse robot data (Brohan et al., 2022; O'Neill et al., 2024; Team et al., 2024). VLA models integrate pretrained visual and linguistic representations with robot action prediction (Brohan et al., 2023; Kim et al., 2024). In parallel, diffusion-based action generation has been adopted in visuomotor policies and VLA models (Chi et al., 2025; Team et al., 2024; Wen et al., 2025a; Li et al., 2024; Wen et al., 2025b). Flow matching provides another route to continuous action generation (Black et al., 2024; Intelligence et al., 2025; Shukor et al., 2025). Recent work also studies efficient action tokenization (Pertsch et al., 2025) and parallel action decoding (Kim et al., 2025). However, robustness studies show that strong benchmark performance does not ensure reliable execution under visual, linguistic, or physical perturbations (Zhou et al., 2025; Fei et al., 2026; Guo et al., 2026). These reliability challenges motivate monitoring VLA policies during execution.

Failure Detection in Robot Manipulation. Runtime failure monitors learn signals from successful executions (Xu et al., 2025; Römer et al., 2025) or use vision-language models (VLMs) to assess progress and failure (Agia et al., 2024; Duan et al., 2024; Grislain et al., 2025). SAFE (Gu et al., 2025) trains lightweight MLP or LSTM detectors on VLA hidden states and evaluates their transfer to unseen tasks. Linear probes also recover outcome-related information (Zhang et al., 2026) and task progress (Bhardwaj et al., 2026) from frozen VLA representations. VLA-FAIL (Seligmann et al., 2026) combines feature distances with action consistency, whereas ActProbe (Huang et al., 2026a) uses action consistency and magnitude. For temporal supervision, Hide-and-Seek (Park et al., 2026) uses contrastive learning with trajectory labels. FailureSpot (Ma et al., 2026) combines action-derived weak supervision with timestamp annotations from actively selected trajectories. For calibration, TDQC (Francis-Meretzki et al., 2026) uses temporal-difference learning to estimate task-success confidence, while VLAConf (Huang et al., 2026b) calibrates scores derived from frozen VLA features. SAFECAST (Rajaprakash et al., 2026) augments detector training and calibration with visually and linguistically perturbed rollouts. In contrast to SAFE's hidden-state readout, we construct inputs for cross-task failure detection from selected FFN neuron contributions.

Neuron-Level Analysis and Representation Selection. Neuron-level analyses connect individual FFN components to model representations and predictions. Geva et al. (2021; 2022) describe FFN outputs as activation-weighted combinations of value directions. Supervised neuron analysis and sparse probing identify subsets of native neurons predictive of labeled concepts (Dalvi et al., 2019; Gurnee et al., 2023). In VLAs, Häon et al. (2025) analyze FFN control concepts and steer behavior by modifying selected activations. Mahato & Ren (2026) retrospectively probe expanded FFN activations for failure signals under visual distribution shifts. In parallel, sparse autoencoders provide a learned feature basis for VLA interpretation and steering (Swann et al., 2026). Event-SAE (Jin et al.,

2026) relates learned features to behavioral events and evaluates success prediction from complete trajectories. Representation selection also supports anomaly detection. DICE (Sun & Li, 2022) selects classifier weights according to their contributions on in-distribution data. ASH (Djurisic et al., 2022) instead sparsifies and reshapes test-time activations for out-of-distribution detection. In VLMs, LAKE (Li et al., 2026) selects visual feature channels using activation statistics from normal data for anomaly detection. In contrast, we select native FFN neurons by their success–failure discrimination within training tasks. We use their output contributions for cross-task failure monitoring without altering the policy's action computation.

3 PRELIMINARIES

We introduce two pieces of background needed for our method: how an FFN updates a Transformer representation through additive neuron contributions, and how a runtime failure monitor uses frozen VLA representations to assess an ongoing robot execution. Our formulation builds on the FFN structure shared across Transformer-based VLA architectures. We use OpenVLA (Kim et al., 2024) as a running example of autoregressive action generation and describe the instantiation for flow-matching action generation, represented by π_0 (Black et al., 2024), in Appendix.

3.1 FFN NEURON CONTRIBUTIONS IN TRANSFORMER BLOCKS

FFN Computation. The language backbone of a VLA model consists of Transformer blocks with attention and FFN sublayers connected through residual updates. Consider a designated token at policy step t and block ℓ, with hidden dimension d. Here, t indexes policy steps, rather than internal decoding or denoising iterations. The attention sublayer updates the incoming representation $h_t^{\ell-1} \in \mathbb{R}^d$ to the pre-FFN state $\tilde{h}_t^\ell = h_t^{\ell-1} + \Delta h_{t,\text{attn}}^\ell$, where $\Delta h_{t,\text{attn}}^\ell$ is the attention update. The FFN then produces an update o_t^ℓ, yielding the block output $h_t^\ell = \tilde{h}_t^\ell + o_t^\ell$.

For simplicity, we focus on a fixed block and omit the layer superscript ℓ in the following FFN formulation. Let $x_t = \text{Norm}(\tilde{h}_t) \in \mathbb{R}^d$ denote the normalized FFN input. The gated FFNs used in recent VLA models (Kim et al., 2024; Black et al., 2024) compute

$$a_t = \rho(W_{\text{gate}} x_t) \odot (W_{\text{up}} x_t), \qquad o_t = W_{\text{down}} a_t, \tag{1}$$

where $W_{\text{gate}}, W_{\text{up}} \in \mathbb{R}^{m \times d}$ project the input to an intermediate space of width m, ρ is an element-wise nonlinear activation, and $\odot$ denotes element-wise multiplication. The post-gating activation $a_t \in \mathbb{R}^m$ is the input to the down projection $W_{\text{down}} \in \mathbb{R}^{d \times m}$, which maps it back to the residual dimension. Projection biases are omitted for clarity.

Neuron Contributions. Writing $W_{\text{down}} = [v_1, \ldots, v_m]$ with columns $v_j \in \mathbb{R}^d$ gives

$$o_t = W_{\text{down}} a_t = \sum_{j=1}^{m} a_{t,j} v_j, \tag{2}$$

where $a_{t,j}$ is the input-dependent activation of channel j, and v_j is its learned output direction, fixed in the frozen policy. Following the FFN sub-update view (Geva et al., 2021; 2022), we refer to each intermediate channel as a *neuron* and define its *contribution* as $c_{t,j} \triangleq a_{t,j} v_j \in \mathbb{R}^d$.

3.2 RUNTIME FAILURE MONITORING

Problem Definition. Let π_θ be a frozen Transformer-based VLA policy and $\mathcal{D}_{\text{train}} = \{(\tau_r, y_r)\}_{r=1}^N$ a set of labeled training rollouts. Rollout τ_r contains observations and policy executions over T_r policy steps, and $y_r \in \{0, 1\}$ denotes its final outcome, with $y_r = 1$ indicating failure. Our goal is a runtime monitor that maps an observed rollout prefix $\tau_{r,1:t}$ to a failure-risk score $R_{r,t}$ using only information available up to the current policy step, and generalizes to tasks unseen during monitor training. Supervision provides rollout outcomes without requiring failure-onset annotations. The policy remains frozen throughout monitor training and evaluation.

Hidden-State-Based Runtime Failure Monitoring. SAFE (Gu et al., 2025) predicts execution failure from hidden states of a frozen VLA policy. Using OpenVLA as an example, let $h_t^L = \tilde{h}_t^L + o_t^L$

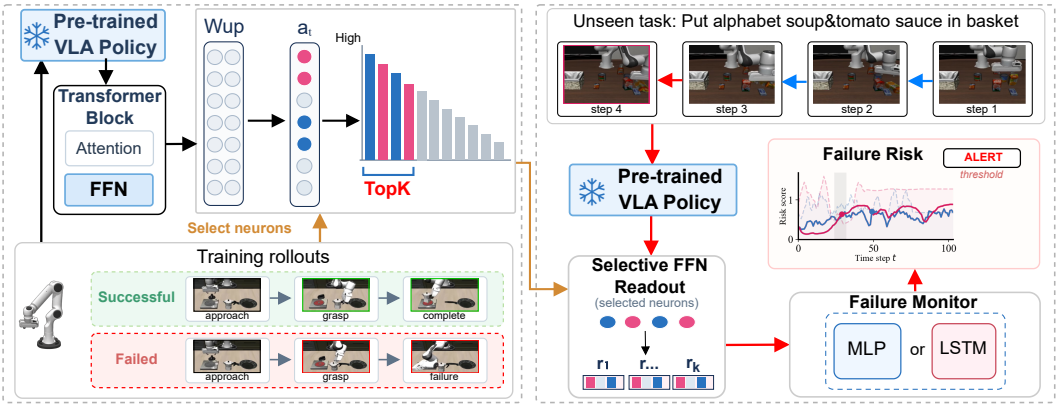


Figure 1. **Overview of the proposed SAFER.** Outcome-guided offline neuron selection enables online failure detection from FFN contributions without changing the VLA policy computation.

denote the final Transformer block output at a designated action-prediction position. This state combines the representation entering the FFN with its complete update. For a single-token readout, the monitoring feature and failure-risk score are

$$f_t = \mathrm{Readout}(h_t^L), \qquad R_t = \mathcal{M}_\phi(f_{1:t}), \tag{3}$$

where $\mathrm{Readout}$ applies any final normalization, and the detector $\mathcal{M}_\phi$ uses only features available up to the current policy step.

4 Selective FFN Readout

We propose SAFER readout to construct failure-monitoring features from a subset of the neuron contributions defined in Section 3.1. As illustrated in Figure 1, the method uses successful and failed training rollouts to determine which neurons to read, then combines their contributions at each policy step to form the detector's input. Outcome labels guide the selection offline. The resulting readout operates on the current activations during execution. Throughout, the VLA remains frozen and its action computation is preserved.

Following the notation in Section 3, we use $a_{r,t}$ for the activation at policy step t of rollout r in a fixed FFN layer. Neuron selection and monitoring use the same token readout: either a single position or a mean over designated positions within the current policy step.

4.1 Outcome-Guided Neuron Selection

Our selection criterion measures how well each neuron's activation distinguishes successful from failed training rollouts. Since supervision is available at the rollout level, we summarize neuron j in rollout r by $\bar{a}_{r,j}$, its mean activation over all T_r policy steps. This gives one activation statistic per neuron and rollout, so longer rollouts do not contribute more samples to the selection score. We compute the area under the ROC curve (ROC-AUC) of these statistics against the outcome labels and rank neurons by their absolute deviation from chance:

$$A_j = \text{ROC-AUC}\big(\{\bar{a}_{r,j}\}_{r=1}^N, \{y_r\}_{r=1}^N\big), \qquad s_j = \left|A_j - \tfrac{1}{2}\right|, \qquad S_K = \mathrm{TopK}_{j\in\{1,\dots,m\}}\, s_j. \tag{4}$$

Here, S_K contains the indices of the K highest-scoring neurons. A ROC-AUC close to either zero or one indicates outcome discrimination, whereas a ROC-AUC near 0.5 gives a low score. The absolute deviation therefore retains neurons whose activations are associated with either success or failure, without requiring a common direction of association.

We pool training rollouts across tasks to obtain one subset for the monitor. The pooled data must contain both successful and failed executions. Scores determine subset membership only and are not used as weights in the readout. The criterion measures outcome association on the training data. Its usefulness on unseen tasks is assessed by transferring the selected subset unchanged.

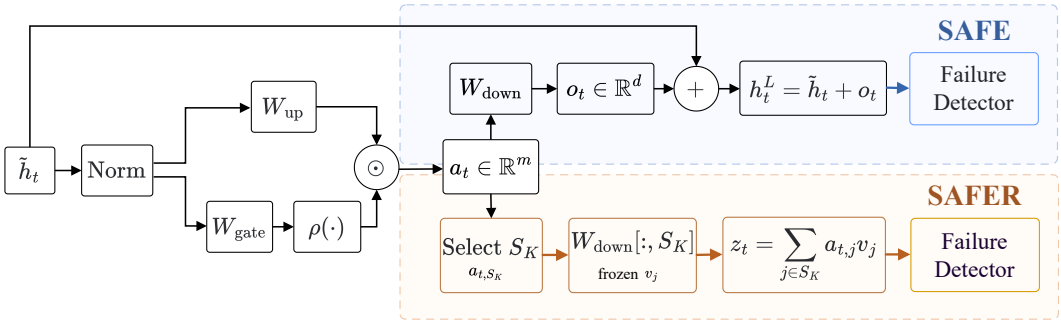


Figure 2. **Comparison of hidden-state and selective FFN readouts for VLA failure monitoring.**

4.2 Constructing the Selective FFN Readout

Given S_K, we sum the selected neuron contributions at each policy step to be the monitoring feature:

$$z_{r,t} = (W_{\text{down}})_{:,S_K} a_{r,t,S_K} = \sum_{j \in S_K} a_{r,t,j} v_j \in \mathbb{R}^d. \tag{5}$$

The feature lies in the original d-dimensional FFN output space and excludes the incoming residual representation and unselected contributions (Figure 2). The policy's forward pass remains unchanged. The rollout mean $\bar{a}_{r,j}$ is used only for offline selection. Monitoring uses the current activations. Setting $K = m$ recovers the full activation-dependent FFN output.

Policy-Specific Activation Extraction. Our method applies to VLA models whose Transformer blocks contain FFNs, provided their activations and output projections are accessible. For autoregressive policies (Kim et al., 2024), we read the final language-model FFN at the final action-prediction position. For diffusion-based policies, including those trained with flow matching (Black et al., 2024), we average the final action-expert FFN activations over action-horizon positions at a designated denoising iteration. Extraction and reconstruction details are given in Appendix.

4.3 Detector Training and Cross-Task Monitoring

We replace the hidden-state features in Equation 3 with the selective FFN features:

$$R_{r,t} = \mathcal{M}_\phi(z_{r,1:t}). \tag{6}$$

We use SAFE's MLP and LSTM detector variants (Gu et al., 2025), fitting only ϕ to the training rollout outcomes while keeping S_K, W_{down}, and the policy fixed. Detector outputs and training objectives are described in Appendix. On unseen tasks, the same subset and fitted detector operate without adaptation, using only the observed feature prefix $z_{1:t}$.

5 Experiments

5.1 Experimental Setup

Benchmarks and Data Splits. We evaluate SAFER on LIBERO10 with four frozen VLA policies: OpenVLA (Kim et al., 2024), π_0 (Black et al., 2024), $\pi_{0.5}$ (Intelligence et al., 2025), and MolmoAct2 (Fang et al., 2026). We additionally evaluate previously collected OpenVLA–WidowX real-robot rollouts (Gu et al., 2025). We split data at the rollout level, using training rollouts for neuron selection and detector fitting, and validation-seen rollouts for configuration selection. Seen-task evaluation uses held-out rollouts from training tasks, while unseen-task evaluation uses tasks excluded from monitor training and configuration selection.

Compared Methods. We compare SAFER with token-uncertainty baselines using maximum or average negative log-probability and entropy (Huang et al., 2023), embedding-distance baselines using Mahalanobis distance and Euclidean or cosine k-NN as evaluated in SAFE (Gu et al., 2025), PCA-KMeans (Liu et al., 2024), and learned out-of-distribution detectors RND (He et al., 2024)

Table 1. **Cross-task failure detection on LIBERO10.** Task-macro ROC-AUC (×100) on seen and unseen tasks for four VLA policies, and dashes indicate missing results or unavailable averages.

	OpenVLA		π_0		$\pi_{0.5}$		MolmoAct2		Average	
Method	**Seen**	**Unseen**	**Seen**	**Unseen**	**Seen**	**Unseen**	**Seen**	**Unseen**	**Seen**	**Unseen**
Mahalanobis dist.	61.71	45.63	76.99	72.03	61.57	66.68	57.74	49.85	64.50	58.55
Euclidean dist. k-NN	64.44	52.29	**80.75**	77.19	66.44	66.93	68.76	68.63	70.10	66.26
Cosine dist. k-NN	63.83	53.98	80.56	78.44	79.63	57.77	53.53	64.41	69.39	63.65
PCA-KMeans (Liu et al., 2024)	61.90	47.25	51.53	47.30	35.88	53.10	66.62	60.96	53.98	52.15
RND (He et al., 2024)	51.88	47.28	68.66	63.96	68.98	57.47	63.73	56.81	63.31	56.38
LogpZO (Xu et al., 2025)	55.10	47.34	77.20	72.85	75.93	57.81	72.10	**77.25**	70.08	63.81
SAFE-LSTM (Gu et al., 2025)	63.34	65.33	59.46	63.91	78.94	56.82	**82.99**	50.68	71.18	59.19
SAFER-LSTM	61.89	60.94	76.64	69.84	59.49	60.79	63.61	60.39	65.41	62.99
SAFE-MLP (Gu et al., 2025)	**68.78**	62.93	70.00	75.12	85.19	62.77	40.63	62.56	66.15	65.85
SAFER-MLP	68.32	**71.60**	77.83	**81.43**	**88.43**	**81.59**	74.46	65.64	**77.26**	**75.07**

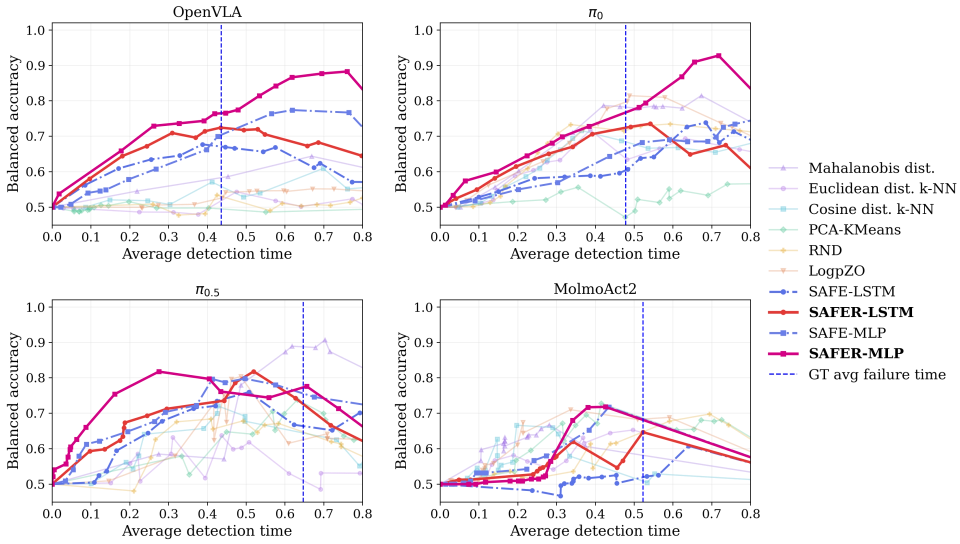


Figure 3. **Unseen-task detection.** Balanced accuracy versus normalized detection time.

and LogpZO (Xu et al., 2025). We compare SAFER directly with SAFE (Gu et al., 2025) under corresponding MLP and LSTM detectors to assess selective FFN readouts against full hidden-state readouts. Unless otherwise specified, our SAFER uses outcome-guided selection with $K = 100$ neurons. Policy-specific extraction, detector configurations, and baseline implementations are documented in Appendix.

Evaluation Metrics. Our primary metric is rollout-level task-macro ROC-AUC, with failure as the positive class. SAFE (Gu et al., 2025) computes pooled ROC-AUC by combining rollouts from all tasks within each evaluation split. This metric includes comparisons between successful and failed rollouts from different tasks, so task-dependent score offsets and differences in failure rates can mask poor discrimination within individual tasks. To assess whether a monitor distinguishes success from failure on a given test task, we compute ROC-AUC within each task and average equally over tasks containing both outcomes. For comparison with SAFE's evaluation, we additionally report pooled ROC-AUC in Appendix. We also compare balanced accuracy against normalized detection time to examine the trade-off between detection accuracy and timing.

5.2 Cross-Task Failure Detection

Quantitative Cross-Task Results on LIBERO10. As shown in Table 1, SAFER-MLP consistently outperforms SAFE-MLP on unseen tasks across all four VLA policies, improving task-macro ROC-AUC by an average of 9.22 points. It achieves the highest average ROC-AUC on both seen and unseen tasks among all compared methods, with the best unseen-task performance on three of the four policies. These results support the effectiveness of selective FFN readouts for cross-task

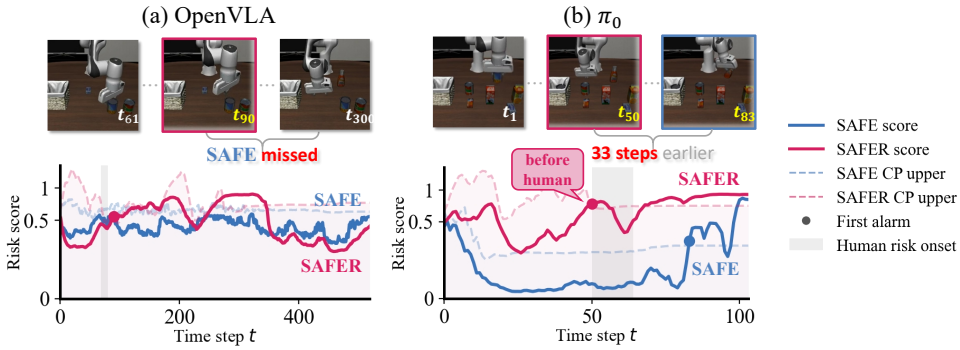


Figure 4. **Qualitative comparison on unseen tasks.** Each column shows rollout key frames and corresponding SAFE and SAFER score trajectories under the same VLA policy.

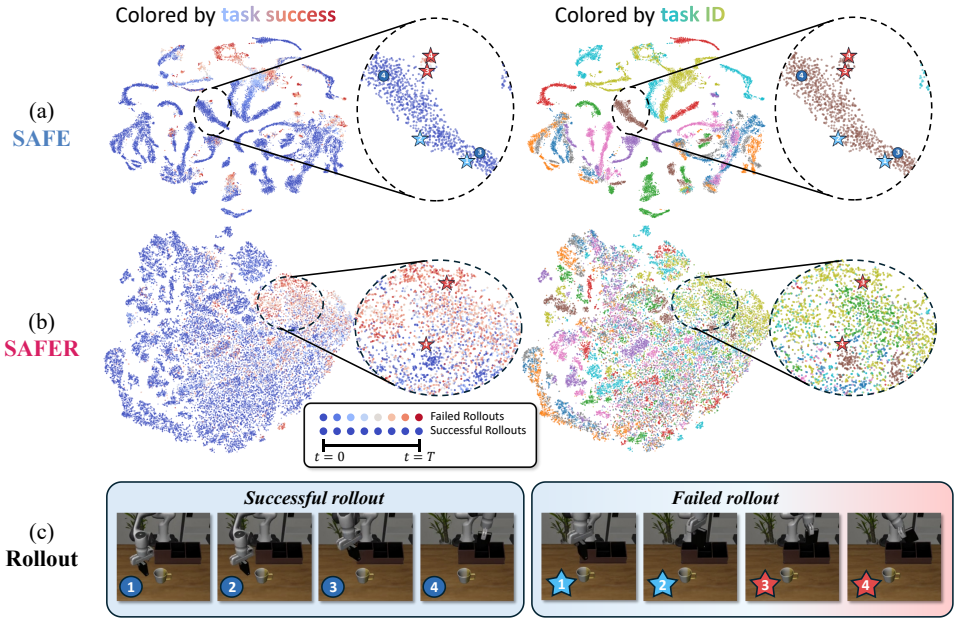


Figure 5. **Representation visualization on π_0–LIBERO10.** t-SNE projections of rollout-level SAFE and SAFER features, colored by execution outcome or task identity.

failure detection. With LSTM detectors, SAFER improves average unseen-task ROC-AUC by 3.80 points over SAFE. Its average seen-task performance is lower, which may reflect how LSTM models temporal patterns in the selected features, but its cause requires further investigation.

Temporal Detection and Qualitative Results. Figure 3 compares balanced accuracy against normalized detection time on unseen tasks. SAFER maintains competitive accuracy with favorable detection timing relative to the Full Hidden baseline. The examples in Figure 4 illustrate clearer score escalation or earlier threshold crossings under the same VLA policy, without establishing a general lead time before physical failure onset.

Failure-Related Representation Structure. Figure 5 compares t-SNE visualizations of SAFE and SAFER features on π_0–LIBERO10. SAFE features exhibit pronounced task-dependent grouping, with features from the later stages of failed rollouts dispersed across task clusters. With SAFER's selective FFN readout, task identities are more mixed, while late-stage features from failed rollouts across different tasks form a more concentrated region. This organization suggests that the selected features capture failure-related patterns shared across tasks, providing qualitative support for SAFER's improved cross-task detection.

Table 2. **FFN layer selection** on OpenVLA–LIBERO10.

Layer	Seen	Unseen
8	63.50	68.40
16	63.50	67.20
24	66.60	68.00
31	**68.32**	**71.60**

	OpenVLA WidowX	
Method	**Seen**	**Unseen**
Max probability	54.86	53.40
Avg probability	47.84	51.21
Max entropy	54.94	48.31
Avg entropy	48.07	52.11
Mahalanobis dist.	75.46	65.50
Euclidean k-NN	77.45	60.12
Cosine k-NN	80.65	71.32
PCA-KMeans	65.86	59.37
RND	70.06	47.35
LogpZO	58.24	53.02
SAFE-LSTM	81.14	69.31
SAFER-LSTM	85.31	72.60
SAFE-MLP	88.69	87.78
SAFER-MLP	**89.44**	**89.18**

Failure Detection ROC-AUC

Table 3. **Monitoring representation ablation.**

Representation	Seen	Unseen
Pre-FFN State $\tilde{h}_t$	68.60	58.60
Block Output $\tilde{h}_t + o_t$	61.70	54.70
Pre-FFN State + Selective Readout $\tilde{h}_t + z_t$	68.60	62.70
Final-Normalized $\mathrm{RMSNorm}_{\mathrm{final}}(\tilde{h}_t + z_t)$	67.10	57.10
Selective FFN Readout (Ours) z_t	**68.32**	**71.60**

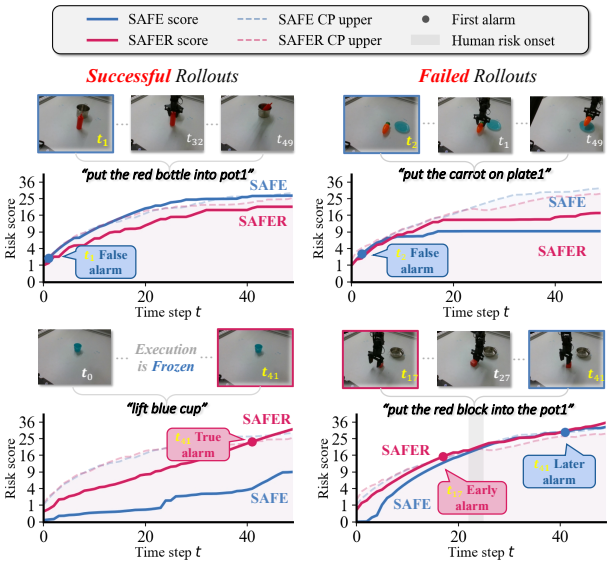


Figure 6. **Evaluation on recorded OpenVLA–WidowX real-robot rollouts.** Left: ROC-AUC ($\times 100$) on seen and unseen tasks, Right: Representative rollouts and failure-score trajectories.

Real-Robot Rollout Validation. Figure 6 evaluates SAFER on previously collected OpenVLA–WidowX real-robot rollouts. SAFER outperforms SAFE on both seen and unseen tasks with both MLP and LSTM detectors. SAFER-MLP achieves the highest ROC-AUC among the compared methods on both splits. These results complement the simulation experiments and support the effectiveness of selective FFN readouts for failure detection on real-world robot execution data.

5.3 Controlled Ablations

Number of Selected Neurons. Figure 7 shows that selecting a small subset of high-ranked neurons yields better unseen-task detection than using all neurons. The left panels show that these neurons have greater outcome selectivity than most FFN neurons. With the ranking computed on seen-task training rollouts and held fixed across budgets, selecting $K = 100$ achieves unseen-task ROC-AUCs of 71.6 for OpenVLA and 81.6 for π_0, compared with 60.6 and 62.7 for full FFN readout. These results support our default budget of $K = 100$, which performs well across both policies.

FFN Layer Selection. Table 2 compares outcome-guided Top100 readouts at OpenVLA layers 8, 16, 24, and 31. Layer 31 achieves the highest mean task-macro ROC-AUC on both seen and unseen tasks among the evaluated layers, suggesting that the final layer may capture higher-level semantic information that is better suited to failure detection.

Monitoring Representation Ablation. The FFN computation involves the pre-FFN state, the FFN update, and their sum forming the block output. Table 3 compares monitoring representations constructed from these components. The selective FFN readout alone achieves the highest unseen-task ROC-AUC, outperforming the pre-FFN state, the full block output, and variants that combine the selected contribution with the pre-FFN state. These results support using the selected FFN contribution directly for cross-task failure detection.

Table 4. **Neuron selection strategy ablation.**

	OpenVLA-LIBERO10		π_0-LIBERO10		OpenVLA-WidowX	
SAFER Selector	**Seen**	**Unseen**	**Seen**	**Unseen**	**Seen**	**Unseen**
SAFER-Random	63.90	60.30	58.90	72.30	82.66	82.62
SAFER-Magnitude	64.60	67.30	74.20	59.60	86.50	84.60
SAFER-Laplacian	67.40	67.20	53.00	64.60	87.00	84.30
SAFER-Inverse	57.30	57.40	63.40	53.70	58.00	44.40
SAFER (Ours)	**68.30**	**71.60**	**77.60**	**81.60**	**89.44**	**89.18**

Table 5. **Top-K neuron selection stability across split seeds.**

K	OpenVLA-LIBERO10			π_0-LIBERO10		
	Pairwise	**Three-seed**	**vs. Random**	**Pairwise**	**Three-seed**	**vs. Random**
100	36.0%	23.0%	39.6×	26.3%	9.0%	10.8×
200	38.5%	24.0%	21.2×	36.9%	12.0%	6.1×
500	43.5%	28.0%	9.6×	37.5%	19.4%	3.1×
1000	50.7%	33.1%	5.6×	47.1%	24.6%	1.8×

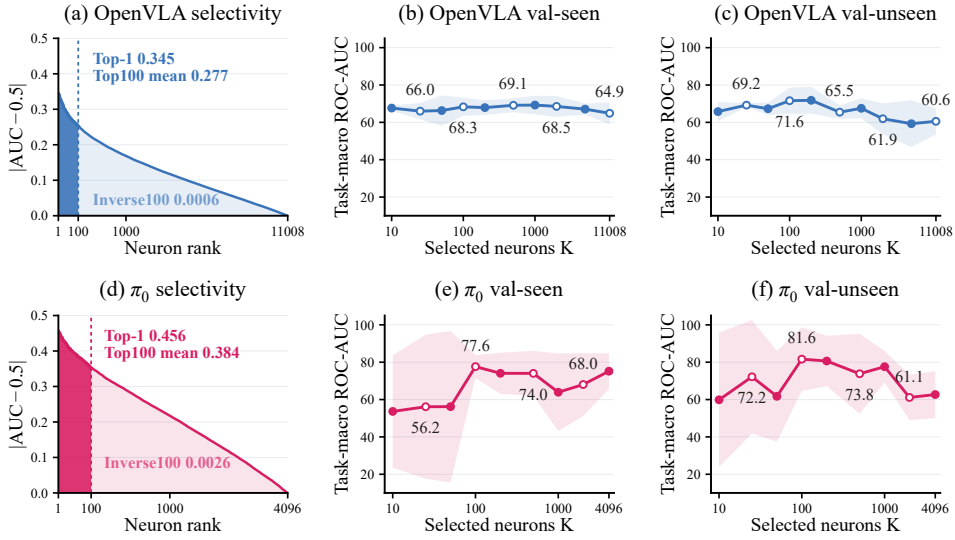


Figure 7. **Failure-selective neurons enable compact FFN readout.** The left panels show the distribution of neuron-level failure selectivity, while the right panels evaluate ROC-AUC.

Neuron Selection Strategy Ablation. Table 4 compares neuron selection criteria with the FFN layer, neuron budget, contribution construction, and MLP-SAFE configuration fixed. Outcome-guided selection achieves the highest unseen task-macro ROC-AUC across all three benchmarks, outperforming random, magnitude-based, Laplacian, and inverse selection. Reversing the ranking substantially reduces performance despite retaining the same number of neurons, indicating that neuron identity matters beyond the subset budget.

Selector Stability Across Split Seeds. To support the cross-task generalizability of our selective FFN readout, we evaluate neuron selection stability across independent split seeds by measuring the overlap of Top-K subsets selected from each seed's training rollouts. As shown in Table 5, at $K = 100$, average pairwise overlap reaches 36.0% for OpenVLA and 26.3% for π_0, corresponding to 39.6× and 10.8× the random expectation. This above-chance overlap suggests that the selector identifies recurring outcome-selective neurons across training splits.

6 Conclusion

We presented SAFER, a selective FFN readout method for cross-task VLA failure detection. Using successful and failed training rollouts, the method ranks FFN neurons by the outcome separability of their activations and selects a subset without optimizing the VLA parameters. At each execution step, it combines the selected neurons' activation-weighted output directions into a monitoring representation. The same subset is applied to unseen tasks, while the VLA remains frozen and its original action computation is preserved. Experiments show that selective FFN readout improves failure detection on unseen tasks compared with full hidden-state readout, highlighting the value of selecting internal computational contributions for failure monitoring.

Limitations and Future Work. Our method relies on access to internal FFN activations and output projections, which prevents its direct use with black-box policies. Its coverage of failure modes also depends on the successful and failed rollouts used for neuron selection, as the selected neurons may not capture failures absent from these data. Beyond the VLA policies studied here, we plan to explore adapting selective FFN readout for failure detection in world action models.

AI USE STATEMENT

We used generative AI tools, including ChatGPT and Codex, to assist with manuscript preparation, language polishing, LaTeX formatting, and code organization and formatting. The authors reviewed all AI-assisted text and code and take full responsibility for the final content of this work.

ETHICS STATEMENT

This work studies failure detection for vision-language-action policies using simulation benchmarks and publicly released real-robot rollouts. No additional physical-robot rollouts are collected for the real-robot evaluation. Failure detection does not guarantee safe robot behavior, and missed failures or false alarms may affect downstream decisions. The method should be independently validated before deployment in safety-critical applications.

REPRODUCIBILITY STATEMENT

The paper and supplementary material describe the method, datasets and data splits, feature extraction, detector training, baseline and ablation configurations, and evaluation protocols. To support reproducibility, we will publicly release our training and evaluation code, model weights, and data preprocessing code.

REFERENCES

Christopher Agia, Rohan Sinha, Jingyun Yang, Zi-ang Cao, Rika Antonova, Marco Pavone, and Jeannette Bohg. Unpacking failure modes of generative policies: Runtime monitoring of consistency and progress. *arXiv preprint arXiv:2410.04640*, 2024. 2

Atiksh Bhardwaj, Edward Weiyi Duan, Prithwish Dan, Wei-Chiu Ma, and Preston Culbertson. Decoding task progress from vla representations. *arXiv preprint arXiv:2608.13474*, 2026. 2

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. 2, 3, 5

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022. 2

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023. 2

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025. 2

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, Anthony Bau, and James Glass. What is one grain of sand in the desert? analyzing individual neurons in deep nlp models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 6309–6317, 2019. 1, 2

Andrija Djurisic, Nebojsa Bozanic, Arjun Ashok, and Rosanne Liu. Extremely simple activation shaping for out-of-distribution detection. *arXiv preprint arXiv:2209.09858*, 2022. 3

Jiafei Duan, Wilbert Pumacay, Nishanth Kumar, Yi Ru Wang, Shulin Tian, Wentao Yuan, Ranjay Krishna, Dieter Fox, Ajay Mandlekar, and Yijie Guo. Aha: A vision-language-model for detecting and reasoning over failures in robotic manipulation. *arXiv preprint arXiv:2410.00371*, 2024. 2

Haoquan Fang, Jiafei Duan, Donovan Clay, Sam Wang, Shuo Liu, Weikai Huang, Xiang Fan, Wei-Chuan Tsai, Shirui Chen, Yi Ru Wang, et al. Molmoact2: Action reasoning models for real-world deployment. *arXiv preprint arXiv:2605.02881*, 2026. 5

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. Libero-plus: A progressive robustness benchmark for visual-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 38574–38583, 2026. 2

Shelly Francis-Meretzki, Mirco Mutti, Yaniv Romano, and Aviv Tamar. Temporal difference calibration in sequential tasks: Application to vision-language-action models. *arXiv preprint arXiv:2604.20472*, 2026. 2

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 5484–5495, 2021. 1, 2, 3

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 conference on empirical methods in natural language processing*, pp. 30–45, 2022. 1, 2, 3

Clémence Grislain, Hamed Rahimi, Olivier Sigaud, and Mohamed Chetouani. I-failsense: Towards general robotic failure detection with vision-language models. *arXiv preprint arXiv:2509.16072*, 2025. 2

Qiao Gu, Yuanliang Ju, Shengxiang Sun, Igor Gilitschenski, Haruki Nishimura, Masha Itkina, and Florian Shkurti. SAFE: Multitask failure detection for vision-language-action models. *arXiv preprint arXiv:2506.09937*, 2025. URL https://arxiv.org/abs/2506.09937. 1, 2, 3, 5, 6, 15, 18, 20, 23, 24, 25

Jianing Guo, Zhenhong Wu, Chang Tu, Yiyao Ma, Xiangqi Kong, Zhiqian Liu, Jiaming Ji, Shuning Zhang, Yuanpei Chen, Kai Chen, et al. On robustness of vision-language-action model against multi-modal perturbations. In *International Conference on Learning Representations*, volume 2026, pp. 70248–70272, 2026. 2

Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. Finding neurons in a haystack: Case studies with sparse probing. *arXiv preprint arXiv:2305.01610*, 2023. 1, 2

Bear Häon, Kaylene Stocking, Ian Chuang, and Claire Tomlin. Mechanistic interpretability for steering vision-language-action models. *arXiv preprint arXiv:2509.00328*, 2025. 1, 2

Nantian He, Shaohui Li, Zhi Li, Yu Liu, and You He. Rediffuser: Reliable decision-making using a diffuser with confidence estimation. In *International Conference on Machine Learning*, pp. 17921–17933. PMLR, 2024. 5, 6, 25

Bingjia Huang, Xiangyu Li, Xiang Wang, Liang Mi, Zixu Hao, Weijun Wang, Hao Wu, Kun Li, Yunxin Liu, and Ting Cao. Actprobe: Action-space probe for early failure detection of generative robot policies. *arXiv preprint arXiv:2606.08508*, 2026a. 2

Dehao Huang, Aoxiang Gu, Chengjie Zhang, Bolin Zou, Wenlong Dong, Zilang Cen, Yue Wang, and Hong Zhang. Vlaconf: Calibrated task-success confidence for vision-language-action models. *arXiv preprint arXiv:2605.29605*, 2026b. 2

Yuheng Huang, Jiayang Song, Zhijie Wang, Shengming Zhao, Huaming Chen, Felix Juefei-Xu, and Lei Ma. Look before you leap: An exploratory study of uncertainty measurement for large language models. *arXiv preprint arXiv:2307.10236*, 2023. 5

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025. 2, 5

Xinchen Jin, Aditya Chatterjee, Pranav Kumar, and Rohan Paleja. Event-grounded sparse autoencoders for vision-language-action policies. *arXiv preprint arXiv:2605.17204*, 2026. 2

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. URL https://arxiv.org/abs/2406.09246. 1, 2, 3, 5

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025. 2

Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, et al. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024. 2

Shaotian Li, Shangze Li, Chuancheng Shi, Wenhua Wu, Yanqiu Wu, Xiaohan Yu, Fei Shen, and Tat-Seng Chua. Latent anomaly knowledge excavation: Unveiling sparse sensitive neurons in vision-language models. *arXiv preprint arXiv:2604.07802*, 2026. URL https://arxiv.org/abs/2604.07802. 3

Huihan Liu, Yu Zhang, Vaarij Betala, Evan Zhang, James Liu, Crystal Ding, and Yuke Zhu. Multi-task interactive robot fleet learning with visual world models. In *8th Annual Conference on Robot Learning*, 2024. 5, 6

Jie Ma, Zongxi Liu, and Yi Zhu. Failurespot: Label-efficient timestamp-level failure detection for vision-language-action models. *arXiv preprint arXiv:2609.04277*, 2026. 2

Dipesh Tharu Mahato and Rachel Ren. Early warning signals for openvla failure under visual distribution shift. *arXiv preprint arXiv:2606.29699*, 2026. 1, 2

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. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024. doi: 10.1109/ICRA57147.2024.10611477. 2

Seongheon Park, Wendi Li, Changdae Oh, Samuel Yeh, Zsolt Kira, Michael Hagenow, and Sharon Li. Hide-and-seek in trajectories: Discovering failure signals for vla runtime monitoring. *arXiv preprint arXiv:2605.30834*, 2026. 2

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025. 2

Harshitha Rajaprakash, Aditeya Prajapati, Rong Xue, Abrar Anwar, and Jesse Thomason. Safecast: Robust failure detection for vla policies with contrast-set training and calibration. *arXiv preprint arXiv:2608.04246*, 2026. 2

Ralf Römer, Adrian Kobras, Luca Worbis, and Angela Schoellig. Failure prediction at runtime for generative robot policies. *Advances in Neural Information Processing Systems*, 38:7631–7670, 2025. 2

Florian Seligmann, Emiliyan Gospodinov, Enes Ulas Dincer, and Gerhard Neumann. Vla-fail: Efficient task failure detection for finetuned vision-language-action models. *arXiv preprint arXiv:2606.21386*, 2026. 2

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025. 2

Yiyou Sun and Yixuan Li. Dice: Leveraging sparsification for out-of-distribution detection. In *European conference on computer vision*, pp. 691–708. Springer, 2022. 3

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. 2

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. 2

Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Zhibin Tang, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, et al. Tinyvla: toward fast, data-efficient vision-language-action models for robotic manipulation. *IEEE Robotics and Automation Letters*, 10(4):3988–3995, 2025a. 2

Junjie Wen, Yichen Zhu, Minjie Zhu, Zhibin Tang, Jinming Li, Zhongyi Zhou, Xiaoyu Liu, Chaomin Shen, Yaxin Peng, and Feifei Feng. Diffusionvla: Scaling robot foundation models via unified diffusion and autoregression. In *ICML*, 2025b. 2

Chen Xu, Tony Khuong Nguyen, Emma Dixon, Christopher Rodriguez, Patrick Miller, Robert Lee, Paarth Shah, Rares Ambrus, Haruki Nishimura, and Masha Itkina. Can we detect failures without failure data? uncertainty-aware runtime failure detection for imitation learning policies. *arXiv preprint arXiv:2503.08558*, 2025. 2, 6, 25

Jiachen Zhang, Junnan Nie, Junyi Lao, Wei Cheng, Chenghao Liu, Jiaxin Jiang, and Songfang Huang. What frozen vlas already know about success: A probing study of value-like structure in foundation robot policies. *arXiv preprint arXiv:2605.28527*, 2026. 2

Xueyang Zhou, Yangming Xu, Guiyao Tie, Yongchao Chen, Guowen Zhang, Duanfeng Chu, Pan Zhou, and Lichao Sun. Libero-pro: Towards robust and fair evaluation of vision-language-action models beyond memorization. *arXiv preprint arXiv:2510.03827*, 2025. 2

APPENDIX

A More Analysis and Results **13**

A.1 Neuron Selectivity on Held-Out Rollouts 13

A.2 Effect of Training-Task Coverage on Neuron Selection 13

A.3 Failure Detection Across Tasks and Data Splits 15

A.4 Failure Detection with Pooled ROC-AUC 15

A.5 Visualization of Monitoring Representations 16

B Experimental Setup and Evaluation Protocol **18**

B.1 Policies and Evaluation Datasets 18

B.2 Data Splits for Training and Evaluation 18

B.3 Rollout Scores and Evaluation Windows 18

B.4 ROC-AUC Evaluation 19

B.5 Alarm Calibration and Detection-Time Evaluation 20

B.6 Implementation Environment 20

C Implementation Details **21**

C.1 Feature Extraction from VLA Policies 21

C.2 Selective Readout Construction and WidowX Reconstruction 22

C.3 Failure Detector Architectures and Training 23

C.4 Baseline Methods and Configurations 24

C.5 Ablation Settings 25

A MORE ANALYSIS AND RESULTS

A.1 NEURON SELECTIVITY ON HELD-OUT ROLLOUTS

Figure 8 evaluates whether training-selected Top-K neuron subsets preserve outcome-related selectivity and failure detection performance on held-out rollouts. The Top100 subset consistently shows higher mean absolute selectivity than random and inverse-ranked subsets on the displayed held-out groups. The corresponding held-out failure detection results further compare different neuron selection strategies under the same neuron budget. The selectivity values are computed from held-out rollout features, while neuron selection is performed using training rollouts only. Absolute selectivity measures outcome separation magnitude and does not imply that the direction of outcome association is preserved across tasks.

A.2 EFFECT OF TRAINING-TASK COVERAGE ON NEURON SELECTION

Table 6 compares outcome-guided SAFER-Top100, SAFER-Inverse100, magnitude, and Laplacian selection using 1, 3, 5, and 7 selector-training tasks on OpenVLA–LIBERO10. With one selector-training task, magnitude selection achieves the highest mean score. From three tasks onward, outcome-guided Top100 performs best among the compared selectors. Its unseen score increases from 56.10 with one task to 71.05 with five and 71.60 with seven, indicating that broader selector-training coverage benefits the evaluated monitor.

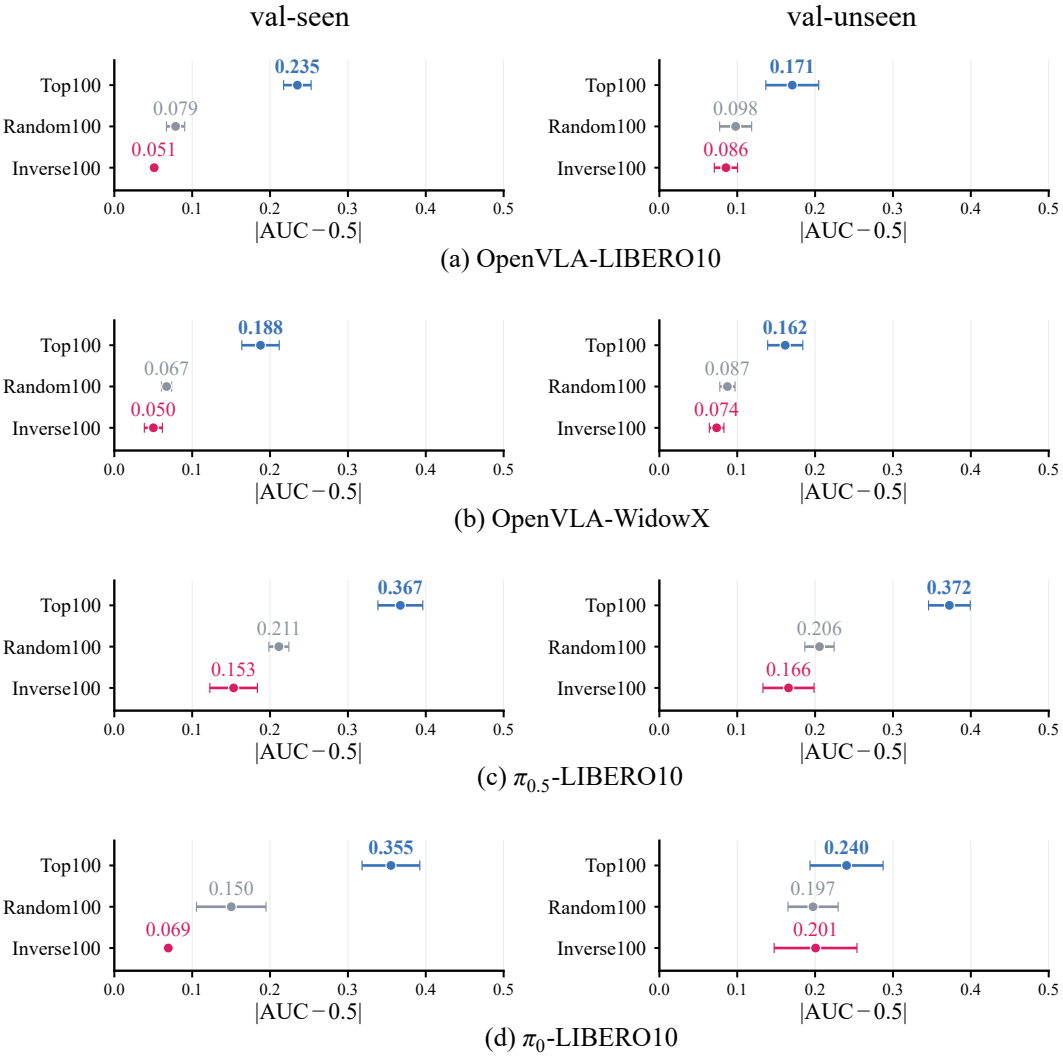


Figure 8. **Held-out neuron selectivity.** Mean $|AUC - 0.5|$ for fixed training-selected, random, and inverse-ranked subsets on seen and unseen rollouts. Error bars denote one sample standard deviation across three split seeds.

Table 6. **Selector-training-task coverage on OpenVLA–LIBERO10.** Task-macro ROC-AUC ($\times 100$) averaged over three split seeds. Bold values identify the best selector for each training-task count and evaluation split.

	1 task		3 tasks		5 tasks		7 tasks	
Selector	**Seen**	**Unseen**	**Seen**	**Unseen**	**Seen**	**Unseen**	**Seen**	**Unseen**
SAFER (Ours)	53.48	56.10	**61.70**	**57.78**	**69.53**	**71.05**	**68.32**	**71.60**
SAFER-Inverse	44.42	47.03	50.62	44.05	49.69	49.89	57.30	57.40
SAFER-Magnitude	**54.91**	**63.83**	57.22	51.32	60.03	59.78	64.60	67.30
SAFER-Laplacian	47.21	58.82	54.82	53.18	66.51	63.58	67.40	67.20

Table 7. **Cross-seed failure detection results on LIBERO10.** Task-macro ROC-AUC ($\times 100$), reported as mean $\pm$ sample standard deviation over three split seeds.

	OpenVLA		π_0		$\pi_{0.5}$		MolmoAct2	
Method	**Seen**	**Unseen**	**Seen**	**Unseen**	**Seen**	**Unseen**	**Seen**	**Unseen**
SAFE-LSTM	63.34 $\pm$ 1.79	65.33 $\pm$ 4.48	59.46 $\pm$ 17.04	63.91 $\pm$ 19.32	78.94 $\pm$ 18.11	56.82 $\pm$ 7.26	**82.99** $\pm$ 18.99	50.68 $\pm$ 17.88
SAFER-LSTM	61.89 $\pm$ 5.68	60.94 $\pm$ 7.35	76.64 $\pm$ 12.95	69.84 $\pm$ 9.94	59.49 $\pm$ 19.15	60.79 $\pm$ 11.02	63.61 $\pm$ 23.36	60.39 $\pm$ 4.87
SAFE-MLP	**68.78** $\pm$ 1.29	62.93 $\pm$ 3.05	70.00 $\pm$ 17.04	75.12 $\pm$ 12.99	85.19 $\pm$ 21.03	62.77 $\pm$ 3.26	40.63 $\pm$ 37.17	62.56 $\pm$ 24.37
SAFER-MLP	68.32 $\pm$ 4.87	**71.60** $\pm$ 7.03	**77.83** $\pm$ 5.36	**81.43** $\pm$ 17.40	**88.43** $\pm$ 11.14	**81.59** $\pm$ 6.22	74.46 $\pm$ 19.97	**65.64** $\pm$ 15.52

Table 8. **Per-task ROC-AUC on OpenVLA–LIBERO10.** Values are mean $\pm$ standard deviation over three split seeds.

	Seen		Unseen	
Task	**SAFE**	**SAFER**	**SAFE**	**SAFER**
T0	81.31	**91.92**	**70.37**	65.38
T1	**68.84**	63.79	–	–
T2	56.00	**61.37**	–	–
T3	70.69	**71.83**	61.36	**64.12**
T4	62.35	**63.29**	–	–
T5	**80.56**	75.00	60.80	**71.26**
T6	**63.74**	54.40	49.68	**56.73**
T7	73.44	**81.18**	86.67	**90.67**
T8	**61.90**	58.33	56.88	**75.29**
T9	**80.98**	68.69	62.98	**74.36**

Table 9. **Per-task ROC-AUC on π_0–LIBERO10.** Values are mean $\pm$ standard deviation over three split seeds.

	Seen		Unseen	
Task	**SAFE**	**SAFER**	**SAFE**	**SAFER**
T0	82.79	**85.59**	84.25	**88.75**
T1	94.74	**100.00**	–	–
T2	**67.89**	58.83	–	–
T3	–	–	–	–
T4	**74.81**	74.72	–	–
T5	26.32	**68.42**	70.41	**90.82**
T6	48.53	**49.51**	58.04	**65.03**
T7	19.44	**97.22**	98.96	**100.00**
T8	84.21	**89.47**	**84.03**	83.45
T9	77.55	**87.45**	37.94	**42.32**

A.3 Failure Detection Across Tasks and Data Splits

Table 7 reports mean and sample standard deviation for the four policy settings in Table 1. Neuron selection and detector fitting are repeated using each seed's training data, then fixed for that seed's held-out evaluation. Tables 8 and 9 organize the per-task results for OpenVLA and π_0, respectively. Evaluation uses a common observation window within each task, and tasks containing only one outcome class are excluded from the macro AUC average. SAFER-MLP achieves the highest mean unseen-task ROC-AUC among the compared detectors for every evaluated policy. The per-task comparisons also show gains over SAFE on most tasks with defined AUC, although improvements are not uniform across tasks.

A.4 Failure Detection with Pooled ROC-AUC

Pooled results complement task-macro evaluation and allow comparison with SAFE's aggregation (Gu et al., 2025). Table 10 reports pooled ROC-AUC using the same rollout scores, splits, and random seeds as the task-macro evaluation, while changing only the AUC aggregation rule. Under pooled evaluation, SAFER-MLP again achieves the highest mean unseen-task ROC-AUC among the compared detectors for every evaluated policy. Its advantage on unseen tasks therefore holds under both task-macro and pooled aggregation.

Table 10. **Pooled ROC-AUC comparison across VLA policies.** Pooled ROC-AUC ($\times 100$), reported as mean $\pm$ sample standard deviation over three split seeds.

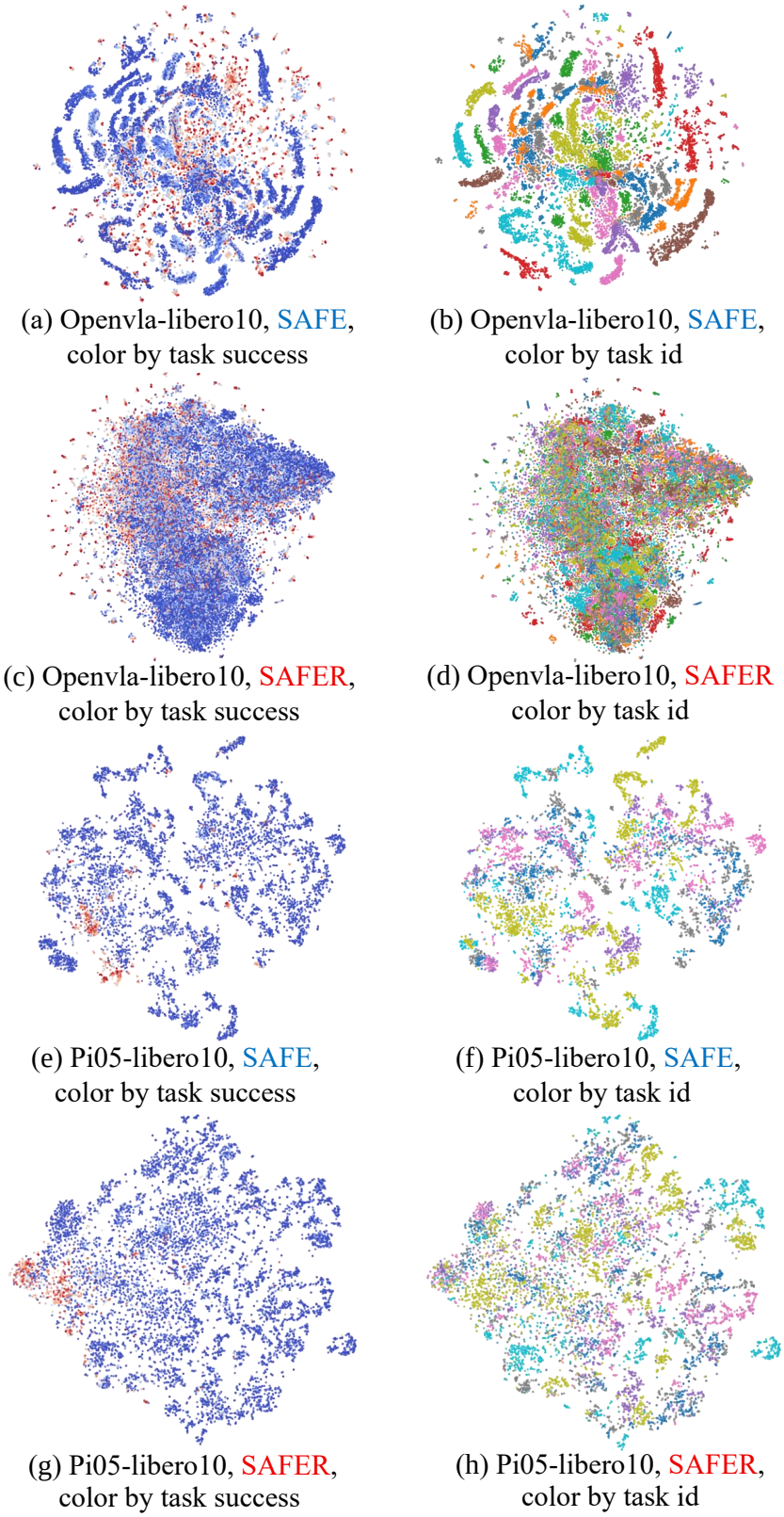
	OpenVLA		π_0		$\pi_{0.5}$		MolmoAct2	
Method	**Seen**	**Unseen**	**Seen**	**Unseen**	**Seen**	**Unseen**	**Seen**	**Unseen**
SAFE-LSTM	69.72 $\pm$ 4.95	70.57 $\pm$ 8.44	72.42 $\pm$ 17.73	63.89 $\pm$ 18.33	72.20 $\pm$ 7.90	58.25 $\pm$ 6.52	**82.99** $\pm$ 18.99	50.68 $\pm$ 17.88
SAFER-LSTM	67.91 $\pm$ 2.01	64.30 $\pm$ 9.77	**83.52** $\pm$ 8.05	76.45 $\pm$ 16.26	70.58 $\pm$ 9.94	61.70 $\pm$ 10.91	63.61 $\pm$ 23.36	60.39 $\pm$ 4.87
SAFE-MLP	**72.62** $\pm$ 3.82	70.68 $\pm$ 9.26	79.40 $\pm$ 8.25	71.52 $\pm$ 22.56	79.97 $\pm$ 23.36	65.06 $\pm$ 3.62	40.62 $\pm$ 37.17	62.56 $\pm$ 24.37
SAFER-MLP	71.03 $\pm$ 8.39	**75.42** $\pm$ 9.12	83.12 $\pm$ 5.92	**81.37** $\pm$ 13.98	**82.18** $\pm$ 12.76	**81.64** $\pm$ 8.58	74.46 $\pm$ 19.97	**65.64** $\pm$ 15.52

A.5 Visualization of Monitoring Representations

The t-SNE visualization in Figure 9 compares the feature geometry of SAFE and SAFER representations on OpenVLA–LIBERO10 and $\pi_{0.5}$–LIBERO10. For each policy, we construct rollout-level feature vectors from the corresponding monitoring representations and visualize their low-dimensional embeddings. Since SAFE and SAFER use different readouts with different feature dimensions, embeddings are computed separately for each representation. Rollout outcomes and task identities are not used during fitting and are only used for coloring the projections.

The visualization provides a qualitative view of representation organization rather than a direct measurement of failure separability. On OpenVLA–LIBERO10, SAFE full-hidden representations show more apparent task-dependent structures, while SAFER selective FFN representations exhibit a different and more mixed geometric organization across tasks. This suggests that selective FFN readout changes the feature geometry observed by the detector, without implying complete removal of task-specific information.

When colored by rollout outcomes, both SAFE and SAFER embeddings show substantial overlap between successful and failed executions. This is consistent with failure detection relying on learned temporal scoring rather than direct clustering in the projected space. Similar patterns are observed for $\pi_{0.5}$–LIBERO10, suggesting that selective FFN readout modifies representation organization while failure discrimination remains a property of the downstream temporal monitor.



(a) Openvla-libero10, SAFE, color by task success

(b) Openvla-libero10, SAFE, color by task id

(c) Openvla-libero10, SAFER, color by task success

(d) Openvla-libero10, SAFER color by task id

(e) Pi05-libero10, SAFE, color by task success

(f) Pi05-libero10, SAFE, color by task id

(g) Pi05-libero10, SAFER, color by task success

(h) Pi05-libero10, SAFER, color by task id

Figure 9. **t-SNE visualization of SAFE and SAFER representations across VLA policies.** The embeddings compare SAFE full hidden representations and SAFER selective FFN representations on OpenVLA–LIBERO10 and $\pi_{0.5}$–LIBERO10. Colors indicate rollout outcomes or task identities.

B Experimental Setup and Evaluation Protocol

B.1 Policies and Evaluation Datasets

We evaluate frozen OpenVLA, π_0, $\pi_{0.5}$, and MolmoAct2 policies on LIBERO10, and use recorded OpenVLA–WidowX real-robot rollouts released with SAFE (Gu et al., 2025). No additional physical-robot rollouts are collected for the WidowX evaluation. Throughout this paper, seen and unseen tasks refer to the failure monitor's training data, not to the VLA policy's pretraining or fine-tuning data.

Table 11 summarizes the rollout counts and data splits used for neuron selection, detector training, and evaluation. For $\pi_{0.5}$-LIBERO10, we exclude tasks with no observed failures before constructing the splits, leaving six tasks with both outcomes. The three split seeds permute these retained tasks to form different training and unseen-task assignments under the same split protocol.

For MolmoAct2-LIBERO10, we use the publicly released failure5 subset, which contains five tasks with available failure examples. Tasks outside this subset are excluded because they lack sufficient failure cases for the evaluation, and the task-level split protocol is preserved across the three seeds. The evaluated policies use publicly released checkpoints or rollout artifacts, with their sources listed in Table 12.

B.2 Data Splits for Training and Evaluation

For LIBERO10 experiments, task-level splits are constructed independently for each policy according to its available rollout collection and failure distribution. For OpenVLA and π_0, the ten LIBERO10 tasks are divided into seven seen tasks and three unseen tasks. Rollouts from the seen tasks are further divided into training and validation subsets for neuron selection, detector fitting, and configuration selection, while unseen tasks are reserved exclusively for held-out evaluation. For $\pi_{0.5}$ and MolmoAct2, task subsets are constructed after filtering tasks without sufficient failure examples. The retained task identities are fixed before split construction.

Rollouts, rather than individual timesteps, are assigned to data splits. Neuron selection and detector fitting use training rollouts only. Configuration selection uses seen-task validation rollouts. When threshold-based detection is evaluated, the threshold is calibrated using successful validation rollouts and fixed before held-out evaluation. The selected neuron subset, detector parameters, and calibration parameters are frozen before evaluation within each split seed.

The three seeds denote independent data split seeds. They do not represent repeated detector initializations or stochastic neuron selection runs. For each split seed, one detector model is trained and evaluated using the corresponding split. Table 13 summarizes the role of each split.

B.3 Rollout Scores and Evaluation Windows

The detector produces a temporal score sequence $R_{r,1:T_r}$ for each rollout r. ROC-AUC is computed at the rollout level rather than the timestep level. Each rollout is first converted into a scalar score s_r according to the method-specific aggregation strategy, and the resulting scores are paired with rollout labels for evaluation. The corresponding rollout-level score aggregation strategies are summarized in Table 14.

Table 11. **Rollout statistics and data splits.** The training split is used for neuron selection and detector training.

Benchmark	Tasks	Rollouts	Train	Seen	Unseen
OpenVLA-LIBERO10	10	500	210	140	150
π_0-LIBERO10	10	500	210	140	150
$\pi_{0.5}$-LIBERO10	6	300	160	40	100
MolmoAct2-LIBERO10	5	250	150	50	50
OpenVLA-WidowX	SAFE	532	250	133	149

Table 12. **Policy checkpoint sources.** The table summarizes the checkpoints or released artifacts used for each VLA policy evaluation.

Policy	Checkpoint / Source
OpenVLA	`openvla/openvla-7b-finetuned-libero-10`
π_0	OpenPI π_0 LIBERO checkpoint converted to PyTorch
$\pi_{0.5}$	`lerobot/pi05_libero_finetuned_v044`
MolmoAct2	`allenai/MolmoAct2-LIBERO`
OpenVLA–WidowX	SAFE released OpenVLA–WidowX rollout artifacts

Table 13. **Data usage protocol.** Each component uses only its permitted split. Unseen splits are reserved for evaluation; seen validation splits are used for selection and calibration.

Component	Used split
Neuron selection	Training split
Detector training	Training split
Detector configuration and hyperparameter selection	Seen validation split
Temporal threshold calibration	Seen validation success-only split
ROC-AUC evaluation	Seen validation and unseen evaluation splits
Temporal early-warning evaluation	Seen validation and unseen evaluation splitss

For task-level evaluation, we use a task-common observation window. For task q, the horizon is defined as

$$L_q = \min_{r \in \mathcal{C}_q} T_r,$$

where $\mathcal{C}_q$ denotes all recorded rollouts belonging to task q and T_r is the recorded length of rollout r. This provides the same observation duration for successful and failed rollouts within each task. For the cumulative SAFE-MLP score, we use the score at the end of this window, which equals its maximum within the window.

For retrospective prefix analysis, a rollout-specific observation ratio $h \in (0, 1]$ defines a prefix length

$$T_r(h) = \max(1, \lfloor hT_r \rfloor).$$

Both prefix evaluations use recorded trajectories and do not imply online early termination or prediction before physical failure onset. Full-rollout evaluation uses each complete recorded sequence with the corresponding score aggregation rule.

B.4 ROC-AUC Evaluation

ROC-AUC measures how well rollout scores rank failed executions above successful ones, with failures treated as the positive class. Our primary metric, task-macro ROC-AUC, gives each valid

Table 14. **Rollout-level score aggregation strategies.** Temporal monitor sequences are converted into scalar rollout scores before ROC-AUC computation.

Method	**Rollout-level score aggregation**
SAFE-MLP	Task-min endpoint
SAFER-MLP	Task-min endpoint
SAFE-LSTM	Task-min endpoint
SAFER-LSTM	Task-min endpoint
Mahalanobis	Configured aggregation
Euclidean k-NN	Configured aggregation
Cosine k-NN	Configured aggregation
PCA-KMeans	Configured aggregation
RND	Configured aggregation
LogpZO	Configured aggregation

task equal weight:

$$\mathrm{AUC}_{\mathrm{macro}} = \frac{1}{|\mathcal{V}|} \sum_{q \in \mathcal{V}} \mathrm{AUC}_q, \tag{7}$$

where AUC_q is the ROC-AUC computed within task q and $\mathcal{V}$ contains tasks with both successful and failed rollouts. Tasks with only one outcome class have undefined AUC and are excluded without numerical imputation. Pooled ROC-AUC instead ranks all rollout scores together within an evaluation split, without task-wise averaging. We report the mean and sample standard deviation over three data split seeds, using $ddof = 1$. The statistical unit is the split seed, not an individual rollout or timestep.

B.5 Alarm Calibration and Detection-Time Evaluation

Threshold-Based Rollout Prediction. Threshold-based monitoring determines whether the cumulative risk sequence exceeds a calibrated time-dependent boundary. The detection time is defined as

$$t_r^{\mathrm{det}} = \min\{t : R_{r,t} \geq \delta_t\}, \tag{8}$$

where no detection is recorded if the boundary is not crossed within the evaluation window. Temporal indices correspond to detector query steps, rather than executed robot actions, video frames, or internal diffusion iterations.

Following SAFE (Gu et al., 2025), we calibrate the boundary using functional conformal prediction with successful validation rollouts only. Successful trajectories are divided into regression and calibration subsets, and the boundary is obtained from the empirical quantile of the maximum normalized residual:

$$\delta_t = \hat{q}_{1-\alpha} \left(\max_{\tau} |R_{\tau,t} - \hat{R}_{\tau,t}| \right). \tag{9}$$

The resulting boundary is a global time-dependent function shared across evaluation rollouts, without task-specific calibration. The conformal level α is selected from a sweep over 0.02,0.05,0.1,0.15,0.2,0.3,0.4,0.5, 0.6,0.7,0.8,0.9, with $\alpha = 0.2$ used for the reported temporal analysis unless otherwise specified. For the OpenVLA configuration, 71 successful validation rollouts are available, with 21 used for regression fitting and 50 for boundary calibration.

Balanced Accuracy. Balanced accuracy is computed from rollout-level alarm decisions, with failed executions treated as the positive class:

$$\mathrm{BA} = \frac{\mathrm{TPR} + \mathrm{TNR}}{2}. \tag{10}$$

A rollout is classified as failed if the detector crosses the calibrated boundary within the evaluation window. A failed rollout without any crossing is counted as a false negative, whereas a successful rollout with a crossing is counted as a false positive.

Detection time is defined by the first threshold crossing. For the earliest-stop evaluation used with the task-specific common horizon, the first-crossing index is normalized by the corresponding `task_min_step`. If no crossing occurs for a failed rollout, its normalized detection time is set to 1. Detection-time statistics are averaged over all ground-truth failed rollouts, including missed detections. False alarms on successful rollouts affect the classification metrics but are not included in the average detection time.

Failure onset annotation. For failed rollouts, we manually annotate the failure onset through visual inspection, defined as the point when the execution starts to exhibit clear failure behavior or when human intervention would be required. When the exact transition point is ambiguous, we annotate an onset interval and use the upper boundary of this interval as the reference onset time for temporal evaluation. Rollouts without valid onset annotations are excluded from temporal comparison. Terminal outcome time is recorded separately and is not used as the failure-onset reference.

B.6 Implementation Environment

All experiments use a common pipeline for feature extraction, detector training, and evaluation, with the VLA policies frozen in their released implementations. The software environment consists of

Python 3.10.20, PyTorch 2.2.0+cu121, and Transformers 4.53.2, running on NVIDIA RTX 5880 Ada GPUs with CUDA 12.1 support. This environment is shared across policies and all stages of the pipeline.

All reported results use rollout split seeds $\{0, 1, 2\}$, with one detector trained per split. Neuron selection and detector fitting use only that split's training rollouts, so variation across seeds reflects the data partition rather than repeated detector initializations. Outcome-guided neuron selection is deterministic given the training split and introduces no additional selector seed.

C Implementation Details

C.1 Feature Extraction from VLA Policies

Table 15 summarizes the policy-specific feature extraction configurations. Here m denotes the FFN intermediate width and d denotes the output dimension after the FFN down projection. SAFE reads the full hidden representation, whereas SAFER constructs the monitoring feature from selected FFN contributions defined in Equation 5.

OpenVLA Configuration. We extract FFN contributions from the final Transformer block (layer 31) of the OpenVLA language backbone. The down projection matrix has shape $W_{\text{down}} \in \mathbb{R}^{4096\times 11008}$, where the intermediate FFN width is $m = 11008$ and the hidden dimension is $d = 4096$. The stored post-gating activation tensor `ffn_pre_down_activation` has shape $[T_r, 7, 11008]$. We use the final action-prediction token (zero-based token index 6) as the FFN readout position:

$$a_{r,t} = A_r[t, 6, :].$$

The corresponding FFN contribution is reconstructed using the selected columns of the down projection matrix:

$$z_{r,t} = A_r[t, 6, S]W_{\text{down}}[:, S]^\top,$$

where S denotes the selected neuron subset. The SAFE baseline uses the matched full hidden-state representation from the same Transformer block.

π_0 Configuration. For π_0, we monitor the action expert branch of the PaliGemma backbone. The monitored FFN is located at layer 17 with intermediate width $m = 4096$ and hidden dimension $d = 1024$. The down projection matrix is obtained from:

```
paligemma_with_expert.gemma_expert.
model.layers.17.mlp.down_proj.weight
```

The stored FFN activation `pre_down_last_diff_all_horizon` has shape $[Q, 50, 4096]$, where Q denotes policy replanning steps and 50 denotes the predicted action horizon. We use the final denoising iteration and the first action-horizon position for monitoring. Specifically, the readout activation is obtained as:

$$a_{r,t} = A_r[t, -1, 0, :].$$

A policy step corresponds to one replanning event of the flow-matching policy and should be distinguished from internal denoising iterations and executed robot actions.

$\pi_{0.5}$ Configuration. For $\pi_{0.5}$, we extract FFN contributions from the action expert branch at layer 17. The FFN intermediate width and hidden dimension are identical to π_0: $m = 4096$ and $d = 1024$. The down projection matrix is obtained from:

Table 15. **Policy-specific feature extraction configurations.** FFN layers, readout positions, and feature dimensions used by SAFE and SAFER.

Policy	SAFER readout location	m	d	SAFE representation
OpenVLA	Layer 31, action token 6	11008	4096	Full hidden state
π_0	Expert layer 17, final denoise, horizon 0	4096	1024	Full hidden state
$\pi_{0.5}$	Expert layer 17, final denoise, horizon 0	4096	1024	Full hidden state
MolmoAct2	Action expert block 35, pre-down FFN	3072	768	Full hidden state

```
model.paligemma_with_expert.gemma_expert.
model.layers.17.mlp.down_proj.weight
```

The stored activation `pre_down_last_diff_all_horizon` has shape $[Q, 50, 4096]$. The monitored representation uses the final denoising iteration (out of 10 iterations) and the first action-horizon position:

$$a_{r,t} = A_r[t, -1, 0, :].$$

The same selective FFN contribution construction and neuron budget are applied as in the other VLA policies.

MolmoAct2 Configuration. For MolmoAct2, we monitor the action expert branch at block 35. The FFN intermediate width is $m = 3072$ and the output hidden dimension is $d = 768$. The monitored FFN activation is captured from the input of the down projection after the SwiGLU activation: `ffn_pre_down_activation`. The corresponding projection matrix is:

```
model.action_expert.blocks.35.mlp.down_proj.weight
```

The stored tensor contains flow and action-horizon dimensions. The current feature construction reshapes these dimensions before selecting the monitoring position. The exact semantic ordering of the flow and horizon axes follows the released feature builder and is documented together with the implementation details. The SAFE baseline uses the matched hidden representation extracted from the same action expert branch.

Preprocessing and Matched Readouts. For each policy, SAFE and SAFER use the same rollout executions, data splits, temporal sampling, and detector training protocol. They differ in the monitored readout: SAFE uses the full hidden representation, whereas SAFER uses selected FFN neuron contributions. Neither feature normalization nor PCA is applied in the primary experiments. Neuron selection uses training rollouts only, and the selected subset is fixed before evaluation.

Feature sequences retain the temporal order of policy invocations. During detector training, sequences are padded to accommodate different rollout lengths, with validity masks identifying the observed timesteps. These operations preserve the recorded temporal structure while allowing variable-length rollouts to be processed together. No preprocessing statistics are estimated from evaluation data.

C.2 Selective Readout Construction and WidowX Reconstruction

Selection and Projection. We implement neuron selection using only training rollouts. For each split seed, neuron scores are computed independently from the training partition, and the selected subset is fixed before detector training and evaluation. The neuron ranking determines subset membership only and does not introduce additional projection weights. Given the selected neuron subset S, SAFER reconstructs the FFN contribution using the corresponding columns of the FFN output projection:

$$z_{r,t} = a_{r,t,S} W_{\text{down}}[:, S]^\top.$$

The neuron budget K refers to the number of selected FFN intermediate channels. Therefore, selecting $K = 100$ neurons does not produce a 100-dimensional detector input. Instead, the selected contributions are projected back into the original hidden dimension d. All projection operations use the original FFN output projection without additional bias or learned reweighting.

WidowX Reconstruction. We use the publicly released OpenVLA–WidowX rollout artifacts from SAFE. No additional physical-robot rollouts are collected. For SAFER, we reconstruct the FFN write representation from the stored FFN activation and the corresponding OpenVLA FFN output projection. Specifically, given the stored activation before the down projection, the FFN contribution is recovered as

$$z_t = a_t W_{\text{down}}^\top.$$

Table 16. **Detection-level validation of OpenVLA–WidowX feature reconstruction.** ROC-AUC comparison using the evaluation protocol adopted by SAFE (Gu et al., 2025), where official and reconstructed evaluations use the same detector configuration.

Representation	Detector	Seen	Unseen
Official Full Hidden	SAFE-MLP	89.11	88.42
Reconstructed FFN-write	SAFE-MLP	87.42	88.33
Official Full Hidden	SAFE-LSTM	84.29	71.80
Reconstructed Full Hidden	SAFE-LSTM	81.95	72.22

The projection matrix is recovered from the OpenVLA checkpoint corresponding to the released WidowX evaluation artifacts. This reconstruction recovers the FFN write used by selective FFN readout. It is not treated as a replacement for the full Transformer hidden state, which additionally depends on residual connections and other block operations. Therefore, SAFE full-hidden evaluation uses the stored hidden representation from the released artifacts.

Reconstruction Validation. We compare failure detection performance with the official SAFE Full Hidden evaluation and measure the similarity between original and reconstructed FFN writes. These comparisons use the same rollout splits and detector configurations. The reconstructed representations yield failure detection performance comparable to the official SAFE evaluation for both SAFE-MLP and SAFE-LSTM. Compared with the original FFN writes, they have cosine similarities of 0.99999869–0.99999976 and relative ℓ_2 errors of 0.001055–0.004674.

C.3 Failure Detector Architectures and Training

The detector receives the full hidden representation for SAFE or the selective FFN representation $z_{r,t}$ for SAFER. Only detector parameters ϕ are optimized, while the VLA policy, output projection, and selected neuron subset remain fixed. All labels use $y_r = 1$ for failed rollouts.

MLP Score and Loss. The MLP detector consists of two fully connected layers with hidden width 256. It predicts a bounded per-step failure score increment and accumulates the temporal risk score as

$$u_{r,t} = \sigma(W_2 \,\mathrm{ReLU}(W_1 z_{r,t} + b_1) + b_2)\,, \tag{11}$$

$$R_{r,t} = \sum_{v=1}^{t} u_{r,v}. \tag{12}$$

Here σ denotes the sigmoid function. The cumulative score is non-decreasing and may exceed one, so it should not be interpreted as a calibrated rollout-level failure probability. The MLP objective follows the cumulative L1 formulation used in SAFE (Gu et al., 2025). The parameter-dependent per-step term is defined as

$$\ell_{r,t} = (1 - y_r)R_{r,t} - y_r R_{r,t} = (1 - 2y_r)R_{r,t}. \tag{13}$$

For a batch of B rollouts, the objective averages over valid timesteps and applies ℓ_2 regularization to non-bias detector parameters:

$$\mathcal{L}_{\mathrm{MLP}} = \frac{1}{B}\sum_{r=1}^{B}\frac{1}{T_r}\sum_{t=1}^{T_r}\ell_{r,t} + \lambda_{\mathrm{reg}}\sum_{\phi_k \in \phi_{\mathrm{nonbias}}} \|\phi_k\|_2^2. \tag{14}$$

LSTM Score and Loss. The LSTM detector processes temporal feature sequences and projects its recurrent representation to a scalar failure score through a sigmoid output. Unlike the cumulative MLP detector, the LSTM score is not explicitly constrained to be monotonic. For the sigmoid output $R_{r,t}$, the LSTM detector is optimized using a per-step binary cross-entropy objective:

$$\ell_{r,t}^{\mathrm{BCE}} = -y_r \log R_{r,t} - (1 - y_r)\log(1 - R_{r,t}). \tag{15}$$

Table 17. **Detector configurations on OpenVLA–LIBERO10 and π_0–LIBERO10.**

Method	HParams	OpenVLA-LIBERO10	π_0-LIBERO10
Mahalanobis	aggregation	last	hfirst_dlast
	cumsum	False	False
Euclidean k-NN	aggregation	mean	hfirst_dlast
	cumsum	True	False
	k	10	10
Cosine k-NN	aggregation	last	hfirst_dlast
	cumsum	True	False
	k	10	10
PCA-KMeans	aggregation	last	hfirst_dfirst
	cumsum	False	False
	clusters	64	16
	dim	128	128
RND	aggregation	first	hfirst_dlast
	batch_size	32	32
	n_epochs	200	200
	lr	1×10^{-4}	1×10^{-4}
LogpZO	aggregation	mean	hfirstlast_dfirst
	batch_size	32	32
	n_epochs	200	200
	lr	1×10^{-4}	1×10^{-4}
	forward_chunk_size	512	512
SAFE-MLP	lr	1×10^{-4}	3×10^{-5}
	λ_{reg}	1×10^{-2}	1×10^{-3}
	feature aggregation	aggtoken=Last	agg_hori=First, agg_diff=Last
SAFE-LSTM	lr	1×10^{-4}	1×10^{-3}
	λ_{reg}	1.0	1×10^{-3}
	feature aggregation	aggtoken=Last	agg_hori=First, agg_diff=Last
SAFER-MLP	lr	1×10^{-3}	1×10^{-3}
	λ_{reg}	0.1	0.1
	feature aggregation	aggtoken=Last	agg_hori=First, agg_diff=Last
SAFER-LSTM	lr	1×10^{-3}	1×10^{-3}
	λ_{reg}	0.01	0.1
	feature aggregation	aggtoken=Last	agg_hori=First, agg_diff=Last

The LSTM configuration follows the temporal detector design used in SAFE, with a recurrent hidden width of 256. The aggregation strategy and optimization settings are fixed according to the corresponding experimental configuration.

Training Configurations. SAFE and SAFER use the same detector architecture and training procedure, with full hidden representations for SAFE and selective FFN readouts for SAFER. We also evaluate Mahalanobis distance, Euclidean k-NN, cosine k-NN, PCA-KMeans, RND, LogpZO, and STAC-Single under their respective aggregation and optimization settings. Tables 17 and 18 summarize the complete configurations.

C.4 Baseline Methods and Configurations

The following descriptions identify the reference formulations used to organize the comparisons. Their exact adaptations and hyperparameters must match the experiment records.

Embedding Distance. SAFE (Gu et al., 2025) compares the distances between a test feature and successful/failed reference feature sets. Given a feature e, the failure score is computed as $d(e, E_{\text{succ}}) - d(e, E_{\text{fail}})$. We evaluate Mahalanobis distance, Euclidean k-NN, cosine k-NN, and PCA-KMeans following this formulation. All reference sets are constructed using training-rollout features from successful and failed executions.

Mahalanobis distance estimates class-wise feature statistics, while Euclidean and cosine k-NN compare distances to nearest neighbors from the two reference sets. PCA-KMeans follows the distance-based formulation used in SAFE (Gu et al., 2025). The corresponding distance configurations, neighborhood size, PCA dimension, cluster number, and temporal aggregation strategies are summarized in Table 17.

Table 18. **Detector configurations on $\pi_{0.5}$–LIBERO10 and MolmoAct2–LIBERO10.**

Method	HParams	$\pi_0.5$-LIBERO10	MolmoAct2-LIBERO10
Mahalanobis	aggregation	hfirst_dlast	hfirst_dlast
	cumsum	False	False
Euclidean k-NN	aggregation	hfirst_dlast	hfirst_dlast
	cumsum	False	False
	k	10	10
Cosine k-NN	aggregation	hfirst_dlast	hfirst_dlast
	cumsum	False	False
	k	10	10
PCA-KMeans	aggregation	hfirst_dfirst	hfirst_dfirst
	cumsum	False	False
	clusters	16	16
	dim	128	128
RND	aggregation	hfirst_dlast	hfirst_dlast
	batch_size	32	32
	n_epochs	200	200
	lr	1×10^{-4}	1×10^{-4}
LogpZO	aggregation	hfirstlast_dfirst	hfirstlast_dfirst
	batch_size	32	32
	n_epochs	200	200
	lr	1×10^{-4}	1×10^{-4}
	forward_chunk_size	512	512
SAFE-MLP	lr	3×10^{-4}	1×10^{-3}
	λ_{reg}	0.01	0.1
	feature aggregation	agg_hori=First, agg_diff=Last	agg_hori=First, agg_diff=Last
SAFE-LSTM	lr	1×10^{-3}	1×10^{-3}
	λ_{reg}	0.1	1×10^{-3}
	feature aggregation	agg_hori=First, agg_diff=Last	agg_hori=First, agg_diff=Last
SAFER-MLP	lr	3×10^{-4}	1×10^{-3}
	λ_{reg}	0.01	0.01
	feature aggregation	agg_hori=First, agg_diff=Last	agg_hori=First, agg_diff=Last
SAFER-LSTM	lr	1×10^{-3}	3×10^{-4}
	λ_{reg}	0.01	0.01
	feature aggregation	agg_hori=First, agg_diff=Last	agg_hori=First, agg_diff=Last

Learned Out-of-Distribution Scores. RND (He et al., 2024) and LogpZO (Xu et al., 2025) estimate feature-level novelty scores using learned models. Following the adaptation in SAFE (Gu et al., 2025), we train separate models using successful and failed training-feature sets and compute the failure score by comparing the corresponding class-conditional scores. All models are trained only on the training split, while validation and test rollouts are used only for evaluation.

For RND, the detector learns feature prediction models and compares the resulting prediction errors from successful and failed reference models. LogpZO follows the same class-conditional formulation while using its corresponding learned feature-distribution objective. The training configurations, including optimization settings and temporal aggregation strategies, are summarized in Table 17.

Full-Hidden SAFE. The baseline uses full hidden representations with the same detector architecture and training protocol as SAFER. The comparison isolates the readout change only when rollout data, preprocessing, and detector selection procedures are matched.

C.5 Ablation Settings

Neuron Selection Criteria. We evaluate different neuron selection strategies for SAFER under the same neuron budget and FFN write construction. All selectors operate using training-rollout features only. For each rollout, the pre-down- projection activations are aggregated according to the policy-specific feature aggregation configuration, producing one feature vector per rollout.

SAFER (Ours) selects neurons according to their outcome separability between successful and failed rollouts. For neuron j, we compute

$$s_j = \left| \mathrm{AUC}(X_{:,j}, y) - 0.5 \right|, \tag{16}$$

where failures are treated as the positive class. The top-K neurons with the largest scores are selected.

SAFER-Inverse uses the same outcome separability scores but selects the inverse-K neurons, providing a low-discriminability control. **SAFER-Random** randomly samples K neurons without replacement. **SAFER-Magnitude** ranks neurons according to their average activation magnitude:

$$m_j = \frac{1}{N} \sum_i |X_{ij}|.$$

SAFER-Laplacian selects neurons according to the Laplacian score computed from the training-feature graph. All variants use the same detector configuration and FFN write reconstruction procedure, with only the selected neuron subset changed.

Neuron Budget and FFN Layer. We compare different neuron selection strategies under a fixed FFN write construction. The evaluated selectors include SAFeR-Random, SAFER-Magnitude, SAFER-Laplacian, SAFER-Inverse, and SAFER (Ours), with the same neuron budget K unless otherwise specified. For the neuron budget ablation, we vary K using a fixed outcome-guided ranking. On OpenVLA–LIBERO10, we evaluate $K \in \{10, 25, 50, 100, 200, 500, 1000, 2000, 5000, 11008\}$ at layer 31, while on π_0–LIBERO10 we evaluate $K \in \{10, 25, 50, 100, 200, 500, 1000, 2000, 4096\}$ using the action-expert FFN layer.

For each budget, the selected neurons are reconstructed through the corresponding FFN output projection to obtain the selective write representation. The detector configuration is kept unchanged across budget settings. For OpenVLA–LIBERO10, we additionally evaluate FFN layers 8, 16, 24, and 31 with a fixed budget of $K = 100$. Neuron selection and FFN reconstruction are performed independently for each layer using its corresponding activations and output projection.

Residual and Normalization Variants. We investigate the effect of residual context on selective FFN readouts. For OpenVLA layer 31, we denote the pre-FFN residual as $\tilde{h}_t$, the full FFN write as $o_t = W_{\text{down}} a_t$, and the selective FFN write as $z_t = W_{\text{down}}[:, S] a_{t,S}$, where S is the selected neuron subset. The evaluated variants include the residual input $\tilde{h}_t$, the full residual fusion $\tilde{h}_t + o_t$, the selective residual fusion $\tilde{h}_t + z_t$, and the normalized selective fusion $\mathrm{RMSNorm}_{\text{final}}(\tilde{h}_t + z_t)$. All variants use the same FFN reconstruction pipeline and detector configuration. The full hidden representation used by SAFE is evaluated as a separate reference representation.

Selector-Training-Task Coverage. We study the effect of selector training-task coverage on neuron selection. The coverage is varied over $C \in \{1, 3, 5, 7\}$ tasks on OpenVLA–LIBERO10, with a fixed neuron budget of $K = 100$. For each setting, SAFER selectors are constructed using only the corresponding task subset, while the FFN write reconstruction procedure remains unchanged. The evaluated selectors follow the configurations in Table 4.