When Internal Feedback Goodharts: SAE Rewards Fail to Improve Robot Success

Socrates Osorio [*1] **Joy Zheyun Yang** [*2 3]

Abstract

World feedback for robot learning is sparse and delayed, motivating dense proxy signals for filtering, credit assignment, or fine-tuning. Sparse autoencoders (SAEs) expose dense internal features in diffusion robot policies, but it is unclear whether such features are useful feedback or merely interpretable correlates. We report a controlled negative result testing this boundary in Octo-Base on SimplerEnv. SAE features show task and phase structure, native-state local action selectivity, and outcome association in a small pilot. They are also optimizable: a no-training action-chunk reranker can select candidates with much higher SAE reward than random chunk controls. However, this internal reward does not improve closed-loop task success. On four episode seeds where native Octo succeeds in all cases and our manual sampler succeeds in three of four, SAE-feature reranking succeeds in only one of four, matching random-chunk and random-feature reranking. This proxy–success gap is the main contribution: sparse policy features can be valid diagnostics and optimizable proxies while still failing as grounded world feedback under naive optimization.

1. Question

Reinforcement learning from world feedback asks agents to improve from grounded consequences such as task completion, safety, or efficiency rather than only from human labels. In robotics, these signals are often sparse: an episode may contain many visual observations, denoising steps, and action chunks before a success bit appears. This creates a tempting idea: use dense internal variables in a pretrained policy as intermediate feedback.

Diffusion policies (Chi et al., 2023; Octo Model Team et al., 2024) are a natural testbed because they generate action chunks through a denoising trajectory. Sparse autoencoders (SAEs) can decompose hidden activations into reusable features (Gao et al., 2025; Rajamanoharan et al., 2024). If some features track task phase, progress, or failure, they might serve as dense world-feedback proxies for reranking or later fine-tuning.

The central risk is Goodharting. An internal feature can be predictive under native rollouts but become misleading when optimized. We therefore ask a deliberately narrow question: *when SAE features are converted into a no-training feedback objective, do they improve closed-loop robot behavior beyond matched random controls?*

This question is intentionally weaker than full reinforcement learning. It removes optimizer instability, replay distribution shift, critic estimation, and hyperparameter search. The only operation is candidate selection: sample several action chunks from a pretrained policy, score each chunk with an internal proxy, and execute the best. If a proposed dense feedback signal cannot pass this gate on seeds where the native policy can already solve the task, then it is not yet a convincing target for fine-tuning.

Contributions. The paper makes three claims. First, SAE features satisfy several diagnostic prerequisites for feedback: they encode task and phase structure, affect actions locally, and show pilot outcome association. Second, these features can be optimized by a no-training reranker; the proxy score increases substantially. Third, the optimized proxy does not improve grounded task success over random controls. The contribution is the gap between the second and third claims.

2. Related Work

Sparse autoencoders. SAEs decompose dense activations into an overcomplete dictionary of sparse, often interpretable features. TopK and JumpReLU variants im-

[*]Equal contribution [1]Electrical Engineering and Computer Sciences, University of California, Berkeley, Berkeley, CA, USA [2]University of Oxford, Oxford, United Kingdom [3]Department of Computer Science, Georgetown University, Washington, DC, USA. Correspondence to: Socrates Osorio <socratesj.osorio@berkeley.edu>.

Accepted to the ICML 2026 Workshop RLxF: Reinforcement Learning from World Feedback, Seoul, South Korea. Copyright 2026 by the author(s).

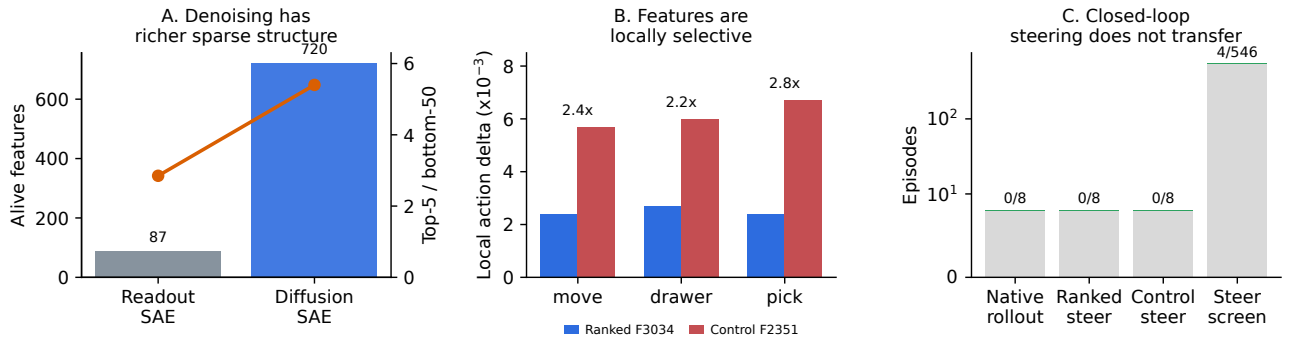

Figure 1. **Sparse diagnostics are not automatically world feedback.** (A) Diffusion activations have richer sparse structure than a readout matched control. (B) Some sparse features are locally selective on native-policy states. (C) Direct closed-loop feature steering is brittle, motivating a stricter test: optimize feature feedback only through candidate action selection and compare against matched controls. The steering pilots in (C) are a separate screen from the targeted reranking slice of Table 4, where native Octo succeeds $4/4$; the $0/8$ here is the native rollout pilot in the steering screen.

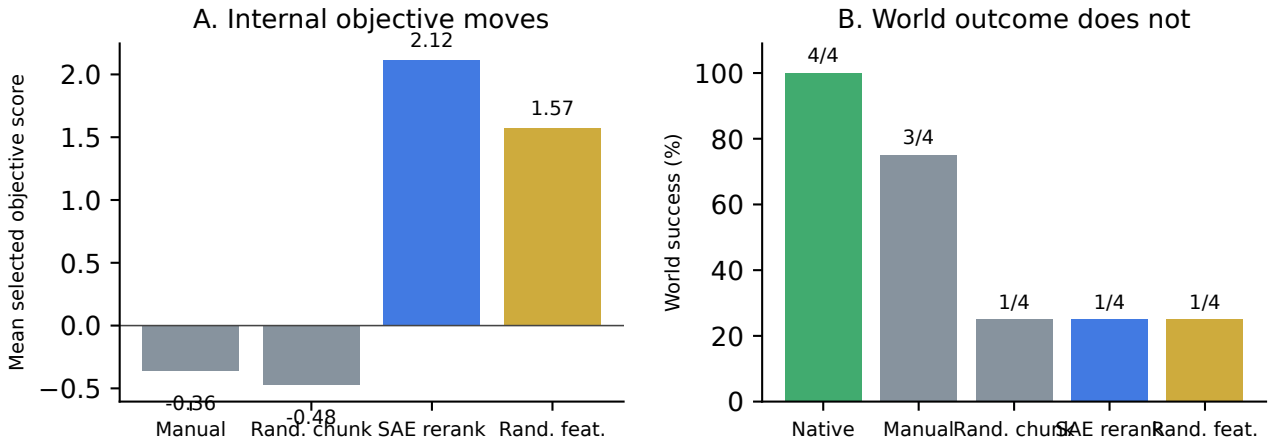

Figure 2. **Proxy optimization gap.** SAE-feature reranking raises the selected internal proxy score relative to manual and random chunk selection, but world success remains 1/4, matching random-feature and random-chunk controls. Each bar reports the mean selected score under that condition's *own* selection objective (the SAE proxy for SAE reranking, the random-feature objective for the random-feature control), so the comparison is between each method's optimized objective and its world outcome.

prove the sparsity–fidelity trade-off (Gao et al., 2025; Rajamanoharan et al., 2024), large open dictionaries have been released for language models (Lieberum et al., 2024), and benchmarks now probe whether the recovered features are actually useful rather than merely sparse (Karvonen et al., 2025). Most of this work evaluates SAEs as analysis tools; our question is whether their features can serve as an optimization target, which is a different and stricter test.

SAEs for diffusion and generative models. Recent work applies SAEs inside diffusion models for image and text generation, including temporal-aware dictionaries for diffusion transformers (Huang et al., 2026), mechanistic analysis of diffusion language models (Wang et al., 2026), concept unlearning (Cywiński & Deja, 2025), and few-step text-to-image control (Surkov et al., 2025). These results show that diffusion activations carry decomposable structure, consistent with our diagnostic findings, but they study perception or generation rather than closed-loop control reward.

Interpretability and steering of robot policies. For vision-language-action and generalist robot policies, prior work extracts symbolic or world-state representations (Lu et al., 2025; Molinari et al., 2025) and steers behavior through internal directions (Häon et al., 2025). Diffusion policies are the action model we study (Chi et al., 2023; Octo Model Team et al., 2024). Our contribution is orthogonal to steering: rather than asking whether an internal direction *changes* behavior, we ask whether optimizing an internal feature *improves* grounded task success over matched controls. The negative result shows that steerability and feedback validity are distinct properties.

Algorithm 1 No-training SAE-feature reranking at one environment step.

Input: state s_t, policy π, SAE encoder ϕ, feature set $\mathcal{F}$ with signs w, support penalty weight β, candidate count K
Sample chunks $\{a^{(1)}_{t:t+H}, \ldots, a^{(K)}_{t:t+H}\} \sim \pi(\cdot \mid s_t)$ from the instrumented diffusion sampler
for $k = 1$ **to** K **do**
  Run the denoising loop; cache the target-block activation at the target step
  Encode with ϕ and evaluate $S(s_t, a^{(k)}_{t:t+H})$ via Equation (1)
end for
$k^\star \leftarrow \arg\max_k S(s_t, a^{(k)}_{t:t+H})$
Execute the first action of chunk $a^{(k^\star)}_{t:t+H}$

3. Feature Feedback Test

We study `octo-base-1.5` in SimplerEnv (Li et al., 2025), focusing on `move_near`, the only task with a meaningful native success rate in our screen. We train TopK SAEs on diffusion-head MLP activations and select outcome-associated features from native-policy activation logs. The target feedback score for state s_t and candidate action chunk $a_{t:t+H}$ is

$$S(s_t, a_{t:t+H}) = \sum_{j \in \mathcal{F}} w_j \phi_j(s_t, a_{t:t+H}) - \beta C_{\text{support}}(s_t, a_{t:t+H}), \quad (1)$$

where ϕ_j is the SAE activation of feature j at the target denoising block and step, w_j is the outcome-predictive sign, and C_{support} penalizes activations outside native-policy feature support.

At each environment step, the reranker samples K candidate action chunks from the same manually instrumented Octo diffusion sampler, scores each chunk by S, and executes the first action of the best chunk. No policy parameters are updated. This is intentionally the cheapest possible RLxF test: if the proxy cannot help at selection time, it is not yet strong evidence for fine-tuning. Algorithm 1 states the full procedure. The only difference between conditions is the score function: the SAE proxy uses S, the random-feature control replaces $\mathcal{F}$ with a size-matched set of randomly chosen alive features and freshly sampled signs, the random-chunk control selects a candidate uniformly at random, and the manual sampler executes the first candidate without reranking.

The support term is a lightweight guard against optimizing features far outside the native distribution. We estimate per-feature support statistics from the activation store and penalize large deviations from native activation moments. This does not guarantee safety, but it makes the test harder to dismiss as unconstrained feature maximization. The experiment therefore asks whether a plausible conservative proxy, not a maximally exploitative proxy, can improve world outcomes.

Table 1. Reranking conditions isolate whether SAE feedback adds value beyond candidate sampling and dense random shaping.

Condition	Role
Native OctoInference	Grounded reference policy.
Manual sampler	Checks whether instrumented DDPM remains native-like.
Random chunk	Controls for sampling multiple candidates.
SAE-feature rerank	Tests the proposed internal feedback proxy.
Random-feature rerank	Controls for dense feature reward and score maximization.

Feature set. The SAE reward uses a small set of outcome-associated features with signed weights. Positive weights reward features associated with success in the pilot data; negative weights penalize features associated with failure. This is deliberately simple. A more elaborate learned reward model could fit nonlinear feature interactions, but it would also introduce more ways to overfit a small number of successes. For a workshop-scale falsification test, a signed sparse feature sum is easier to audit and easier to compare against random-feature controls.

Controls. We compare against native OctoInference, a manual policy-history sampler, random chunk selection from the same K candidates, and matched random alive-feature reranking. The manual sampler uses OctoInference's policy image history so that reranking uses native-like observations rather than a mismatched camera stream.

Candidate-generator validation. Before interpreting reranking, we debugged the manual sampler against native OctoInference. Two issues mattered: a non-native camera input in the manual observation path, and an episode-seed convention that evaluated different episodes from the native seed screen. After matching OctoInference's policy-history observations and native seed indexing, native Octo solved all four selected episodes and the manual sampler solved three of four. This validation is important because reranking failure would be uninterpretable if the candidate generator itself were already broken.

We then evaluate on the same four known native-success episode starts. This makes the test deliberately favorable to the proposed proxy: the task is solvable by the native policy, the manual sampler is close to native, and the reranker receives multiple candidate chunks at every step. Failure

Table 2. Why the targeted reranking slice is intentionally favorable to the SAE proxy.

Design choice	Reason
Known native-success seeds	Avoids evaluating on unsolvable base-policy episodes.
Validated manual sampler	Ensures reranking has native-like candidates.
$K = 8$ candidates	Gives the proxy multiple chances per step.
Random-feature control	Tests whether improvement is specific to selected features.
Random-chunk control	Tests whether sampling alone explains outcomes.

Table 3. Diagnostics suggest SAE features are real policy variables, but not reliable control handles.

Check	Result
Task structure	Per-episode SAE profiles classify task pairs with AUROC up to 0.99.
Temporal phase	Top features peak at early, middle, and late episode phases.
Local action sensitivity	Ranked F3034 is 2.2–2.7× less disruptive than matched control F2351.
Outcome pilot	Best single feature AUROC 0.95 on one labeled seed ($n = 20$, 4 successes).
Direct steering	Closed-loop steering is brittle: steer screening yields 4 successes in 546 episodes.

under these conditions is therefore more informative than failure on arbitrary seeds where the base policy may already be unable to solve the task.

4. Diagnostics Before Optimization

The features pass several diagnostic checks, which makes the negative result more informative. Table 3 summarizes the evidence. Per-episode SAE profiles classify task pairs with AUROC up to 0.99. Top features peak at different episode phases. A native-state local probe shows a ranked feature, F3034, is 2.2–2.7× less disruptive than a matched control at one denoising step. A small labeled `move_near` pilot found a single feature with AUROC 0.95 for success, although this estimate is based on only 20 episodes and 4 successes.

These checks are not enough for RLxF. They show that the features are structured and locally meaningful, but not that optimizing them improves world outcomes. The reranking experiment tests exactly that missing step.

This separation is the paper's main methodological point. An RLxF signal must survive a change in optimization pressure. It is not enough for a feature to correlate with success when the native policy chooses the actions. Once the feature is used to choose actions, the policy may visit states where the correlation no longer holds. The reranking test applies this pressure without the confounds of policy training.

Table 4. **Proxy optimization gap.** On four native-success `move_near` episodes, SAE-feature reranking optimizes its internal score but does not improve world success over matched controls.

Policy	Succ.	Steps	Proxy
Native	4/4	52.8	–
Manual	3/4	48.8	-0.36
Random chunk	1/4	72.8	-0.48
SAE rerank	1/4	66.8	2.12
Random-feature	1/4	73.0	1.57

Table 5. Per-episode outcomes on the four native-success episodes (environment seeds). A checkmark is a success. Each reranker solves at most one episode, and the rerankers do not even agree on which one: SAE reranking solves only the same episode as the random-chunk control, while random-feature reranking solves a different one.

Ep. seed	Native	Manual	R. chunk	SAE	R. feat.
9	✓	–	–	–	–
12	✓	✓	–	–	–
17	✓	✓	–	–	✓
20	✓	✓	✓	✓	–
Total	4/4	3/4	1/4	1/4	1/4

5. Result: Optimizable but Misaligned

We first repaired the candidate generator. An earlier manual DDPM path used a non-native camera input and an inconsistent episode seed convention. After matching OctoInference's policy-history observations and native seed indexing, we evaluated four episode seeds known to be successful for native Octo. Native succeeds in all four; the manual sampler succeeds in three of four. This makes the reranking test fair enough to interpret.

Table 4 shows the key outcome. SAE-feature reranking selects chunks with substantially higher internal feature reward than random chunk selection. But this internal improvement does not translate into higher closed-loop success: it succeeds in 1/4 episodes, the same as random chunk and random-feature reranking, and worse than the manual sampler.

The per-episode breakdown in Table 5 sharpens this point. The single episode that SAE reranking solves (seed 20) is exactly the one the random-chunk control also solves, and it is *not* the episode that random-feature reranking solves (seed 17). If the SAE proxy were selecting genuinely bet-

ter chunks, its one success should not coincide with the sampling-only control while diverging from the other dense-reward control. The most parsimonious reading is that the lone reranker successes are attributable to candidate sampling, not to the feature signal: on every episode that the manual sampler loses once reranking is applied, all three reranking objectives fail together.

This is not evidence that SAEs are useless. It is evidence that the naive proxy is not grounded enough. The selected features are valid diagnostics under native behavior, yet optimizing them changes the action distribution in a way that breaks the relation between feature activation and task success. In RLxF terms, the proxy is dense and optimizable, but not aligned with the world feedback objective.

Why this is not a null implementation result. The negative result is informative because the internal objective did move. SAE reranking raises the selected proxy score from -0.48 under random chunk selection to 2.12, so the implementation is not simply failing to optimize the score. The failure is specifically in the proxy-to-world link: optimizing the internal feature score changes selected actions, but the resulting closed-loop behavior is no more successful than random controls. This is the failure mode RLxF methods need to detect before using dense internal variables as rewards.

The result also clarifies what would have counted as a positive finding. If SAE reranking had matched or exceeded the manual sampler while random controls stayed low, the proxy would have passed a cheap but meaningful world-feedback gate. That would not by itself prove a full RL method, but it would justify residual fine-tuning or offline RL with the SAE reward. Instead, the gate fails. The correct next step is not to tune a larger RL system around the same reward; it is to improve the grounding of the proxy or change how internal features enter the learning objective.

6. Failure Analysis

The failure has two separable parts. First, reranking changes which chunks are selected. The selected proxy score rises substantially, so the SAE objective is not inert. Second, the selected chunks do not improve the environment outcome. This is a stronger diagnosis than a null result where the method simply fails to move its own score. Here, the proxy is optimized and still fails as feedback.

One interpretation is that the selected features encode correlates of native success rather than causes of success. Under native rollouts, those correlates are useful diagnostics because the policy remains near the data distribution on which the features were selected. Under reranking, the selected action chunks change future observations. The feature activations can then remain high while no longer indicating progress toward the task. This is the classic proxy-optimization failure, but in an internal mechanistic feature space rather than in a hand-engineered reward.

Table 6. Alternative explanations for the negative result and how the experiment addresses them.

Concern	Response
Candidate generator is broken	Manual sampler solves 3/4 targeted episodes.
Proxy was not optimized	SAE rerank raises mean proxy score to 2.12.
Sampling alone hurts behavior	Random chunk is included and matches SAE success.
Any dense feature reward works	Random-feature rerank is included and also matches SAE success.
Task slice is too hard	Native solves 4/4 targeted episodes.

Table 7. Practical tests before treating internal policy features as RLxF feedback.

Risk	Required check
Diagnostic-feature shortcut	Separate prediction under native rollouts from improvement under optimization.
Proxy Goodharting	Run no-training reranking before fine-tuning on the proxy.
Dense-shaping artifact	Compare against random alive-feature and random chunk controls.
Out-of-support semantics	Penalize or measure feature activation outside native-policy support.

The remaining concern is sample size. Four targeted episodes are not enough to estimate a precise success rate. They are enough to reject the strongest version of the hoped-for result: if SAE feedback were a robust dense control signal, it should not collapse to the same success count as random controls on a slice where native Octo succeeds every time and the manual sampler succeeds three times. The right conclusion is therefore cautious but clear: this proxy is not yet grounded enough to justify policy optimization.

7. Lessons for RLxF

The negative result suggests a stricter standard for using internal policy features as world feedback. Table 7 states the resulting checklist.

The first lesson is that diagnostic validity and control validity are different claims. Task classification, temporal structure, local action selectivity, and outcome correlation are useful filters, but none proves that a feature should be opti-

Table 8. Suggested escalation path for internal-feature feedback. Our result fails at the no-training gate.

Stage	Acceptance criterion
Diagnostic screen	Features predict task, phase, risk, or outcome under native rollouts.
No-training gate	Feature-based candidate selection improves held-out world success.
Residual adaptation	Fine-tuning improves success beyond native and random-feature controls.
Deployment check	Improved behavior stays inside support and transfers across seeds/tasks.

mized. The second lesson is that no-training reranking is a useful gate before fine-tuning: in our case, the gate fails, so fine-tuning on this reward would be premature. The third lesson is that matched controls are essential. The random-feature reranker also achieves a high internal score and the same success rate; without it, the experiment would look misleadingly positive.

8. Relation to World Feedback

The term "world feedback" should imply grounding in external consequences. Internal features can help expose where a policy represents task phase, uncertainty, or failure, but they are not world feedback by themselves. They become candidate feedback only when their optimization is checked against held-out world outcomes. Our experiment demonstrates why this distinction matters. The SAE reward is dense, smooth enough to rank candidates, and derived from interpretable policy variables. Nevertheless, the grounded success signal does not improve.

This has a practical consequence for RLxF pipelines. Internal features should be used as hypotheses for feedback, not as labels. A reasonable workflow is: identify candidate features, verify native-distribution diagnostics, test no-training selection against random-feature controls, and only then consider residual fine-tuning or offline RL. If the no-training gate fails, as it does here, the feature can still be useful for analysis but should not be promoted to a reward without additional evidence.

Why the random-feature control is central. A dense internal objective can appear successful simply because it produces confident scores. In our experiment, random-feature reranking also reaches a high positive proxy score and the same 1/4 success rate as SAE-feature reranking. This is precisely the failure mode the control is meant to expose. Without it, the increase in SAE score would be easy to overinterpret as evidence that the policy is selecting more meaningful chunks. With it, the conclusion is sharper: the selected SAE features are more plausible than random features diagnostically, but the closed-loop world outcome does not yet distinguish them.

Table 9. What the negative result does and does not imply.

Claim	Status
SAE features are structured diagnostics	Supported by task, phase, and local-probe evidence.
SAE reward is optimizable	Supported: selected proxy score increases.
Naive reranking improves success	Not supported: 1/4, matching controls.
All SAE feedback is useless	Not shown; stronger training-time uses remain open.

9. What This Rules Out

The experiment rules out a specific but important shortcut: *diagnose a success-correlated SAE feature, maximize it over sampled action chunks, and expect the robot to improve*. That shortcut is attractive because it requires no policy update, no critic, and no additional environment reward engineering. It also fails in the setting where it should have been easiest to see signal: we evaluate only episode seeds where native Octo can solve the task, and the manual sampler is already close to native behavior.

The result does not rule out all uses of sparse features for RLxF. Feature feedback may still help if it is used as an auxiliary loss, a conservative regularizer, a critic input, or a diagnostic for selecting where sparse world rewards should be queried. But those uses should be treated as stronger claims requiring stronger validation. In particular, a future positive result should show that feature feedback improves held-out world outcomes while also beating random alive-feature controls and remaining inside native-policy activation support.

This distinction matters for interpretability-driven control. Mechanistic features often look more semantic than raw activations, but semantics observed under the data distribution need not survive optimization. Our negative result is therefore a small-scale warning: world feedback cannot be replaced by internal feature plausibility. The feature must earn its role as feedback by improving grounded outcomes under matched controls.

10. Limitations

This is a small workshop-scale study on one policy family, one simulator, and a targeted subset of `move_near` episodes. The reranking evaluation is intentionally narrow: four native-success seeds, $K = 8$ candidates, one

SAE checkpoint, and one feature-selection rule. The result should not be read as a theorem that SAE feedback cannot work. It shows that the simplest plausible interface fails despite strong diagnostics and native-like candidate generation.

The next decisive positive experiment would need to change one of the ingredients while preserving the controls. Possible directions include selecting features on more labeled native successes, learning a calibrated feature-to-success model rather than using signed feature sums, using feature feedback as an auxiliary critic input rather than a direct reward, or fine-tuning a small residual adapter with behavior-cloning regularization. Any of these could succeed, but each should still beat random-feature controls and report whether selected actions remain inside native-policy support.

Why a documented failure case is useful. RLxF needs examples where plausible feedback fails. Dense feedback sources are attractive because sparse world rewards are expensive, but dense proxies create new Goodhart risks. This paper contributes a concrete, reproducible failure case with matched controls: the internal reward is not inert, it is optimized, yet the world outcome does not improve. That distinction is useful for designing future RLxF systems because it separates representation discovery from feedback validation.

11. What a Positive Result Should Report

The negative result suggests a reporting standard for future internal-feedback papers. A positive result should report not only improvement over the base policy, but also whether the dense proxy beats random-feature rewards, whether it improves held-out world outcomes, and whether its selected actions remain in support. It should also separate three datasets: one for feature discovery, one for proxy calibration, and one for held-out evaluation. Otherwise a feature can be selected because it explains one seed and then fail when optimized on another.

For robot policies, the report should include native baseline success on the exact task and seed distribution used for intervention. This is not bookkeeping. If the native policy is already at floor, failure is uninterpretable; if the native policy is strong, random candidate sampling may degrade performance. The correct comparison depends on the native baseline. Our targeted reranking slice follows this principle by evaluating only episodes that native Octo solves, and by validating the manual sampler before drawing conclusions about reranking.

Finally, the result should include a proxy-success gap plot like Figure 2. Dense feedback methods can look compelling when only the proxy score is shown. The paired view makes the relevant RLxF question explicit: did optimizing the proxy improve the world?

Table 10. Reporting checklist for internal-feature feedback methods.

Report	Purpose
Native baseline on same seeds	Shows whether the world task is solvable.
Proxy score before/after selection	Confirms the feedback objective is actually optimized.
World success before/after selection	Tests whether proxy optimization helps.
Random-feature feedback	Controls for dense internal shaping.
Random candidate selection	Controls for sampling multiple chunks.
Support drift metric	Detects proxy exploitation outside native behavior.

12. Reporting Checklist

Our result suggests a compact reporting checklist for RLxF systems that use internal model features as feedback. The checklist in Table 10 is meant to make proxy failures visible before a full RL run consumes substantial compute.

The key design principle is to always report the proxy and the world outcome together. If the proxy improves but the world does not, the result is not a small implementation detail; it is a feedback-grounding failure. This is exactly what happens in our reranking experiment.

13. Summary for RLxF

This paper is a controlled counterexample rather than a positive RL method. Its contribution is to take a tempting RLxF shortcut (turn interpretable internal features into dense feedback, optimize that feedback, and expect grounded improvement) and show where it breaks. The experiment shows why that shortcut is unsafe. The proxy score improves, so the feedback objective is not ignored; nevertheless, the world success rate does not improve over random controls.

This distinction matters for RLxF because many proposed feedback sources will be proxies. Language feedback, visual heuristics, learned reward models, simulator diagnostics, and internal model features can all provide dense signals that are easier to optimize than sparse world success. The central question is not whether the proxy is structured or optimizable. The central question is whether optimization of the proxy improves the world outcome that gives the feedback its meaning.

Our study gives a small but concrete answer for one internal-feature proxy. SAE features are useful diagnostics and can

be optimized by reranking, but the resulting actions do not improve task success. This makes the paper a warning and a protocol proposal: internal feedback should pass a no-training proxy-vs-world test with random-feature controls before it is used as a reward for policy optimization.

14. Conclusion

Sparse internal features in diffusion robot policies can be interpretable, predictive, and optimizable. Those properties are still insufficient for reinforcement learning from world feedback. On our targeted reranking test, SAE-feature feedback increases the internal proxy score but does not improve task success over random controls. For RLxF, this makes the main lesson concrete: internal policy features should be treated as candidate feedback variables whose grounding must be established by intervention or reranking tests, not assumed from interpretability diagnostics alone.

Acknowledgements

We thank the ICML 2026 RLxF workshop reviewers for feedback that sharpened the framing of the proxy–success gap, and the maintainers of SimplerEnv and the Octo model release for the open infrastructure this study builds on.

References

Chi, C., Xu, Z., Feng, S., Cousineau, E., Du, Y., Burchfiel, B., Tedrake, R., and Song, S. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems (RSS)*, 2023.

Cywiński, B. and Deja, K. SAeUron: Interpretable concept unlearning in diffusion models with sparse autoencoders. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 11738–11775, 2025.

Gao, L., Dupré la Tour, T., Tillman, H., Goh, G., Troll, R., Radford, A., Sutskever, I., Leike, J., and Wu, J. Scaling and evaluating sparse autoencoders. In *International Conference on Learning Representations (ICLR)*, 2025.

Häon, B., Stocking, K. C., Chuang, I., and Tomlin, C. Mechanistic interpretability for steering vision-language-action models. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 2743–2762, 2025.

Huang, V. S.-J., Zhuo, L., Xin, Y., Wang, Z., Wang, F.-Y., Wang, Y., Zhang, R., Gao, P., and Li, H. TIDE: Temporal-aware sparse autoencoders for interpretable diffusion transformers in image generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(1): 435–443, 2026.

Karvonen, A., Rager, C., Lin, J., Tigges, C., Bloom, J., Chanin, D., Lau, Y.-T., Farrell, E., McDougall, C., Ayonrinde, K., Till, D., Wearden, M., Conmy, A., Marks, S., and Nanda, N. SAEBench: A comprehensive benchmark for sparse autoencoders in language model interpretability. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 29223–29264, 2025.

Li, X., Hsu, K., Gu, J., Pertsch, K., Mees, O., Walke, H. R., Fu, C., Lunawat, I., Sieh, I., Kirmani, S., Levine, S., Wu, J., Finn, C., Su, H., Vuong, Q., and Xiao, T. Evaluating real-world robot manipulation policies in simulation. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 3705–3728, 2025.

Lieberum, T., Rajamanoharan, S., Conmy, A., Smith, L., Sonnerat, N., Varma, V., Kramár, J., Dragan, A., Shah, R., and Nanda, N. Gemma scope: Open sparse autoencoders everywhere all at once on Gemma 2. *arXiv preprint arXiv:2408.05147*, 2024.

Lu, H., Li, H., Shahani, P. S., Herbers, S., and Scheutz, M. Probing a vision-language-action model for symbolic states and integration into a cognitive architecture. *arXiv preprint arXiv:2502.04558*, 2025.

Molinari, M., Nevali, L., Navani, S., and Younis, O. G. Emergent world representations in OpenVLA. *arXiv preprint arXiv:2509.24559*, 2025.

Octo Model Team, Ghosh, D., Walke, H., Pertsch, K., Black, K., Mees, O., Dasari, S., Hejna, J., Kreiman, T., Xu, C., Luo, J., Tan, Y. L., Chen, L. Y., Sanketi, P., Vuong, Q., Xiao, T., Sadigh, D., Finn, C., and Levine, S. Octo: An open-source generalist robot policy. In *Robotics: Science and Systems (RSS)*, 2024.

Rajamanoharan, S., Lieberum, T., Sonnerat, N., Conmy, A., Varma, V., Kramár, J., and Nanda, N. Jumping ahead: Improving reconstruction fidelity with JumpReLU sparse autoencoders. *arXiv preprint arXiv:2407.14435*, 2024.

Surkov, V., Wendler, C., Mari, A., Terekhov, M., Deschenaux, J., West, R., Gulcehre, C., and Bau, D. One-step is enough: Sparse autoencoders for text-to-image diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Wang, X., Jiang, B., Wan, Y., Yang, B., Kong, L., and Zou, D. DLM-Scope: Mechanistic interpretability of diffusion language models via sparse autoencoders. *arXiv preprint arXiv:2602.05859*, 2026.

A. Full Experimental Details

This appendix records the configuration needed to reproduce the reranking experiment and the diagnostics that precede it. All components are built on the public Octo release and SimplerEnv; we add only an activation logger, the sparse autoencoders, and the no-training reranker.

A.1. Policy and Environment

We use `octo-base-1.5` (Octo Model Team et al., 2024) evaluated in SimplerEnv (Li et al., 2025). The diffusion action head produces an action chunk of horizon H through a DDPM denoising loop with 20 denoising steps. We study the `move_near` task, which is the only task in our screen with a meaningful native success rate; tasks such as `pick_coke_can` are either at floor or saturated for the base policy in our setup and are therefore uninformative for a reranking gate. For closed-loop stability we reset the simulator environment at a fixed interval, matching the deadlock-avoidance setting used throughout the project. The reranking evaluation uses four episode seeds for which native Octo succeeds in all four; this is the deliberately favorable slice described in Section 3.

A.2. Activation Logging

We log diffusion-head MLP block activations during native rollouts. Activations are stored in a single HDF5 activation store with the layout `activations/diffusion_head/layer_{block}/denoise_step_{step}`, each holding a $(N, 256)$ matrix together with episode identifiers, environment timesteps, and task identifiers. Activations are stored in `float32`; an earlier `float16` store was discarded after we confirmed it discarded variance that mattered for downstream selection. The diffusion head has genuine per-step variance (every sampled row is unique), in contrast to the readout token, which is constant within a task and therefore unusable as a per-step feedback site. We instrument three transformer blocks $\times$ three denoising steps, giving the nine candidate sites used for site selection. Figure 3 contrasts the alive-feature counts at the readout and diffusion sites with the weakness of direct closed-loop steering.

A.3. Sparse Autoencoders

Each SAE is a TopK autoencoder (Gao et al., 2025). For the diffusion head, $d_{\text{model}} = 256$ and the expansion ratio is 16, giving a dictionary of $d_{\text{sae}} = 4096$ features. The top-k sparsity is $k = 64$. We use an auxiliary loss on the top $k_{\text{aux}} = 512$ dead latents with weight $\alpha_{\text{aux}} = 1/32$ to limit dead features. Training runs for 5000 steps with batch size 1024, Adam at learning rate 3×10^{-4}, weight decay 0, a warmup fraction of 0.05, and gradient-norm clipping at 1.0. SAE training is run on CPU on the compute partition because the available GPUs (compute capability 6.1) are not supported by the installed PyTorch build; this does not affect the trained dictionary, only wall-clock. Architecture is detected from the checkpoint at load time: a `theta` parameter selects a JumpReLU decoder, otherwise the TopK decoder is used.

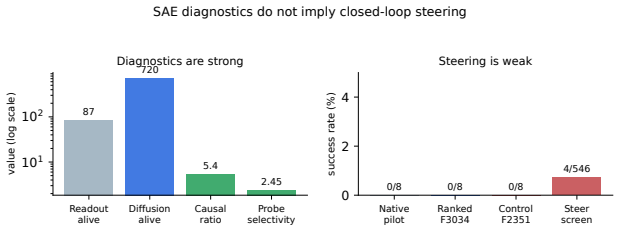

Figure 3. **Diagnostics are strong but steering is weak.** *Left:* the diffusion head exposes far more alive features than the readout site (720 vs. 87), and the ranked feature shows a causal-ratio and probe-selectivity advantage over a matched control. *Right:* despite these diagnostics, direct closed-loop feature steering almost never succeeds ($0/8$ on targeted features; $4/546$ across a broad steering screen). This steering screen is separate from the targeted reranking slice (Table 4), where native Octo succeeds $4/4$; the $0/8$ is the native rollout pilot *within* the steering screen, not the reranking baseline. This gap is what motivates the stricter no-training reranking test in the main paper.

A.4. Feature Selection

The reward feature set $\mathcal{F}$ is a small set of outcome-associated features with signed weights. Features are scored on native-policy activation logs from the labeled `move_near` pilot ($n = 20$ episodes, 4 successes): positive weights are assigned to features whose activation is associated with success, negative weights to features associated with failure. This rule is deliberately a signed sparse sum rather than a learned reward model, so that the proxy is auditable and so that the random-feature control is an exact size- and sign-matched comparison. The $|\mathcal{F}| = 8$ reward features used in the headline run, with their signs, are: F3606(+), F1522(+), F2311(−), F968(+), F2147(+), F2861(+), F3975(−), F1578(+); positive features are success-associated and negative features are failure-associated. The random-feature control draws a size-matched set of alive features with freshly sampled signs using random seed $1000 +$ `seed`. Figure 4 shows the task- and phase-dependent activation of representative diagnostic features. The diagnostics in Section 4 use the same activation store: per-episode SAE profiles for task-pair classification (AUROC up to 0.99), per-feature temporal-phase peaks, and a native-state local-ablation probe at a single denoising step (ranked feature F3034 is 2.2–2.7$\times$ less disruptive than matched control F2351).

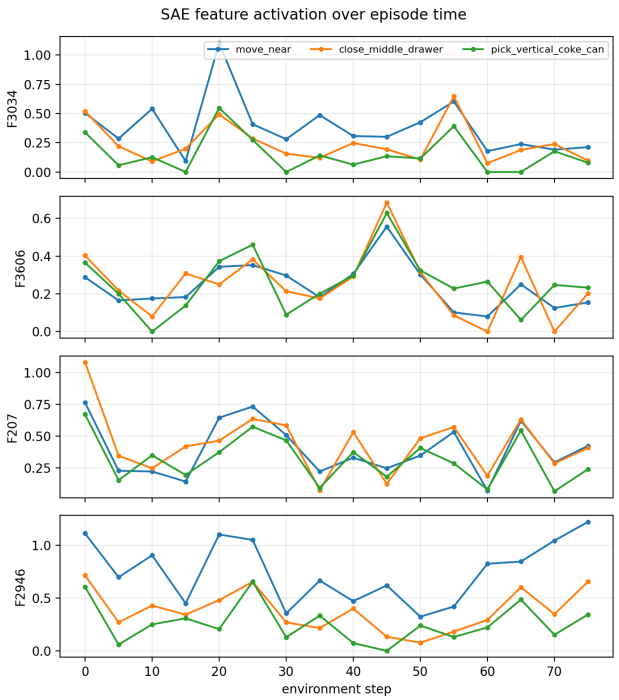

Figure 4. **Representative diagnostic SAE features have task- and phase-dependent activation.** Activation of four diagnostic diffusion-head features (F3034, F3606, F207, F2946) over environment time for three tasks; these illustrate the diagnostic structure and are not the reward feature set $\mathcal{F}$ used for reranking (only F3606 overlaps). Top features peak at different episode phases and separate tasks, which is the diagnostic structure that makes them tempting feedback candidates; the main paper shows this structure does not survive optimization.

A.5. Reranker and Support Term

At each environment step the reranker draws $K = 8$ candidate chunks from the same instrumented diffusion sampler, scores each by Equation (1) at the target block (block 2) and denoising step (step 19), and executes the first action of the highest-scoring chunk (Algorithm 1). Concretely, ϕ_j is a z-score $z_j = (c_j - \mu_j)/\sigma_j$ of the SAE code c_j relative to per-feature moments (μ_j, σ_j) estimated from the native activation store, and the support term is a one-sided hinge,

$$S = \sum_{j \in \mathcal{F}} w_j z_j \; - \; \beta \sum_{j \in \mathcal{F}} \max\big(0,\, |z_j| - \kappa\big)^2, \qquad (2)$$

with support weight $\beta = 0.25$ and threshold $\kappa = 3.0$. The penalty therefore only activates once a feature drifts more than κ standard deviations outside its native range, so the proxy is conservative rather than maximally exploitative. The candidate sampler is the manually instrumented Octo DDPM path. Two bugs were fixed before any reranking numbers were trusted: a non-native camera input in the manual observation path, and an episode-seed convention that evaluated different episodes than the native success screen. After matching OctoInference's policy-history observations and native seed indexing, native Octo solves all four targeted episodes and the manual sampler solves three of four, which is the validation that makes a reranking failure interpretable.

Table 11. Exact configuration for the headline reranking run (Table 4, Figure 2).

Setting	Value
Policy / model	`octo-base-1.5` (`rail-berkeley`)
Environment	`google_robot_move_near`
Episode seed	1
Episode-start indices	$\{8, 11, 16, 19\}$ (env seeds $\{9, 12, 17, 20\}$)
Episodes / start	1 (4 episodes total)
Max steps	80
Observation source	`policy_history`
Candidates K	8 (rerank); 1 (native/manual)
Target block / step	2 / 19
SAE	`sae_b2_s19_topk` step 5000, TopK k=64
Reward features $\mathcal{F}$	3606^+, 1522^+, 2311^-, 968^+, 2147^+, 2861^+, 3975^-, 1578^+
Support β / κ	0.25 / 3.0
Random-feature seed	1000 + `seed`
Score function	Equation (2)

A.6. Conditions and Reproduction

The five evaluated conditions are native OctoInference, the manual sampler, random-chunk selection, SAE-feature reranking, and random-feature reranking, as summarized in Table 1. All reranking conditions share the same candidate sampler and the same K; they differ only in the scoring rule. The headline numbers are in Table 4: the SAE proxy raises its mean selected score from -0.48 (random chunk) to 2.12 while world success stays at $1/4$, matching both random controls and trailing the manual sampler at $3/4$. Random-feature reranking reaches a comparable proxy score (1.57) and the same $1/4$ success, which is the control that prevents reading the proxy increase as evidence of meaningful selection. Table 11 lists the exact settings behind Table 4 and Figure 2 so the run can be reproduced end to end. The four evaluation episodes are the native-success `move_near` episodes obtained from `seed` 1 at episode-start indices $\{8, 11, 16, 19\}$, corresponding to environment episode seeds $\{9, 12, 17, 20\}$; native and manual conditions use $K = 1$ (no reranking) while the reranking conditions use $K = 8$. Scripts for activation logging, SAE training, feature analysis, denoising analysis, and the diffusion intervention/reranking loop are organized as numbered stages and will be released upon publication.