Dormant but Not Latent in Robot Policies

Joy Zheyun Yang*
Department of Computer Science
University of Oxford
Oxford, United Kingdom
Email: joy@robots.ox.ac.uk

Socrates Osorio*
Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, CA, USA
Email: socratesj.osorio@berkeley.edu

***Abstract*—A tempting safety workflow for embodied foundation models is to decompose a trained policy into causal parameter components, enumerate the components it rarely engages, its *dormant* machinery, and then elicit and remove the "latent capabilities" they supposedly encode. We frame this as a controlled proxy on state-based behavior-cloning policies in simulation, not an embodied foundation model. We test this interpretability-for-trust premise rigorously and find it largely false for standard monolithic policies, while isolating the architectural condition under which a weaker, actionable version holds. Using a behavior-preserving stochastic parameter decomposition of behavior-cloning policies on Meta-World, we first establish that *dormancy is ubiquitous in our test grid*: 20–75% of components are dormant, the fraction is training-dependent (versus $\sim$90% at random init), varies lawfully with task diversity and width, is corpus-stable, and replicates across architectures and seeds. We then ask whether dormant components are *latent capabilities*, with six ground-truth tests. They are not. A planted backdoor is not dormant-localized (12.5%) and is recovered better by an untargeted baseline, a naturally rare skill is not dormant-localized, dormant activation does not predict failure (AUC 0.38), fine-tuning recruits *active* rather than dormant components, elicited activating inputs are off-manifold (0% plausible), and mechanism-targeted search finds fewer unsafe behaviors than random search. In monolithic policies, dormant components are inert, entangled, distributed redundancy. Finally, clean capability removal is *architectural and lawful*. Under an equalized ablation budget, removal specificity rises significantly with a measurable modularity metric (slope 0.63, 95% CI $[0.23, 1.08]$, permutation p=0.005 over 32 policies), from $\approx$0 in monolithic policies to 0.2–0.3 in mixture-of-experts. The safety message is cautionary and constructive. Capability auditing cannot be retrofitted onto monolithic policies via decomposition, and modularity is the architectural knob that buys it.**

I. Introduction

A central question for trustworthy embodied foundation models is whether we can *audit what a robot policy can be made to do*, not just what it did on the nominal data, but the behaviors latent in its weights that an adversary or a distribution shift could elicit. Linear and stochastic parameter decompositions [1, 2] promise a mechanistic unit for exactly this, a parameter component c with a causal-importance function $c_c(x) \in [0, 1]$. Applied to a deployed policy, they suggest a tempting safety pipeline. First decompose the policy, then find the components it rarely engages (its *dormant* machinery), and then treat them as a latent-capability surface to enumerate, elicit, and prune. The intuition is that "what a policy can be made to do but never did" lives in its dormant components, and that interpretability can therefore be retrofitted onto an already-trained policy.

We subject this intuition to rigorous, ground-truth testing and report a largely negative result with a precise positive boundary. Our novelty is not a new decomposition. It is a direct, falsifiable test of the decomposition-for-auditing *premise* on control policies, together with the finding that the property the premise needs, capability localization, is set by architecture rather than recovered by any decomposition. We first confirm that dormancy is a real, ubiquitous, lawful property of trained policies (Section IV). But across six independent tests, a planted backdoor, a naturally rare skill, failure prediction, fine-tuning recruitment, manifold plausibility, and unsafe-behavior discovery, dormant components in *monolithic* policies do *not* correspond to enumerable, removable, or elicitable latent capabilities (Sections V–VI). They are inert, entangled, distributed redundancy. Capability localization, the property that would make decomposition-based auditing possible, turns out to be *architectural*. It is weak in standard multilayer perceptrons and modestly improved by mechanism-native modularity (Section VII).

a) What we measure: So that "interpretability" here is a measured quantity and not an intuition, we fix precise operational definitions of dormancy, dormant-localization, removal specificity, and elicitation success in Appendix C, and each ground-truth test is built so that a positive result would be unambiguous under those definitions.

b) Why this matters for safety by design: The result is a cautionary answer to an interpretability-for-trust hope and a constructive pointer toward safety by design. The audit-then-probe-then-screen-then-remediate workflow that decomposition seems to enable does not survive contact with ground truth on monolithic policies. The carriers of any one capability are shared with others, so removing a target damages bystanders, and elicited "activating" inputs are physically implausible. What *does* work is designing modularity in. A measurable modularity metric lawfully predicts how surgically a capability can be removed. Auditing latent capability is therefore not a post-hoc decomposition problem but an architecture-selection problem.

c) The auditing workflow we test: Concretely, we evaluate the four-stage safety workflow that decomposition appears to enable. *Audit:* decompose the policy and enumerate

*Equal contribution.

dormant components as the latent-capability surface. *Probe:* elicit inputs that activate each dormant component to expose the behavior it encodes. *Screen:* use dormant-component activation at deployment as a runtime warning that the policy has entered an unfamiliar competence regime. *Remediate:* ablate the components that carry an undesired capability to remove it without collateral damage. Sections V–VI show that each stage fails on monolithic policies. Dormant components are not the right surface (audit), elicited inputs are off-manifold (probe), dormant activation does not predict failure (screen), and ablation is non-surgical (remediate). Section VII then shows that the remediate stage becomes lawful once modularity is designed in.

d) Contributions: (1) A multi-seed, multi-architecture *census* establishing dormancy as a ubiquitous, training-dependent, lawful property of trained robot policies. (2) A *ground-truth battery* of six tests showing that, despite being real, dormant components in monolithic policies are not latent capabilities, which corrects a tempting but wrong premise for decomposition-based policy auditing. (3) A constructive, quantitative result. *Capability-removal specificity is a lawful, increasing function of architectural modularity* (equalized-budget ablation over 32 policies, slope 95% CI $[0.23, 1.08]$, permutation p=0.005), which identifies the architectural knob that makes capability auditing possible while honestly reporting its still-modest magnitude. We report all outcomes, positive and negative.

II. Related Work

We build on attribution-based and stochastic parameter decomposition [1, 2], which was demonstrated previously only on small language models, and we take it to control policies. Our question is distinct from that line of work. Prior decomposition papers ask whether a faithful mechanistic unit exists. We ask whether that unit, once found, supports a downstream safety workflow on policies, and we answer largely no for monolithic networks. Sokar et al. [12] study *activation* dormancy as a plasticity pathology in deep reinforcement learning, whereas we study *parameter-component* dormancy of converged policies and ask whether it is a latent-capability surface. Activation maximization [10] inverts neurons, we invert causal components, and we find, as with feature visualization, that the maximizers are off-manifold. The hope that a hidden capability can be read out of a trained model connects directly to eliciting latent knowledge [3], and our negative result is a concrete robotics instance of why that read-out is hard when structure is not built in. Red-teaming [11, 5], trojan detection [14], pruning and sparsity [4, 9, 8], and interpretable-by-construction architectures [7] all bear on the auditing question we test. Our contribution is to test the decomposition-for-auditing premise itself and to show that the property it needs, capability localization, is set by architecture rather than by the decomposition. Appendix B expands this positioning.

Within the trustworthy-foundation-models agenda, decomposition-based auditing is a *safety-by-practice* proposal, because it hopes to recover an auditable structure from already-trained weights. Our finding is that, for the capability-auditing question, this does not succeed on monolithic policies, because the structure is not there to recover, whereas the *safety-by-design* alternative, a modularity objective measurable at training time, lawfully governs how cleanly a capability can later be removed.

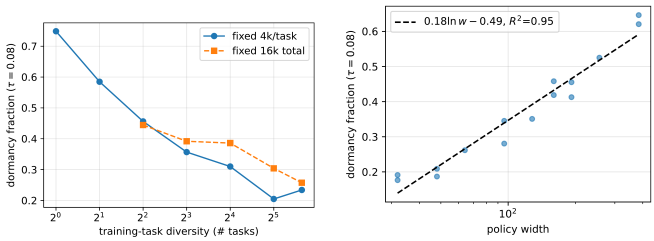

Fig. 1. Dormancy fraction decreases (near-monotonically) with training-task diversity, including at a fixed total-data budget (control), and increases with policy width.

III. Setup and a Behavior-Preserving Decomposition

We use Meta-World [15] (Sawyer manipulation, MuJoCo [13]), state-based, with scripted-expert behavior-cloning policies, and the decomposed network is the action multilayer perceptron. We decompose each policy with a stochastic parameter decomposition, in which each linear map factorizes as $W \approx \sum_c u_c v_c^\top$ with a per-component gate $c_c(x) = \sigma(g_c^\top h(x) + b_c)$, trained with faithfulness, stochastic-ablation and minimality, and sparsity objectives. We weight reconstruction so the decomposed policy is *behaviorally* near-identical to the original. For the multi-task policies used in all downstream analyses ($k \geq 4$), closed-loop success retention is 0.64–1.0, versus as low as 0.15 for an unweighted decomposition, with an ablation ratio of 4–37. The degenerate single-task (k=1) decomposition is less faithful (retention 0.12 on one seed), so it enters only the census and never a localization claim. A component is *dormant* if its peak importance over a broad rollout corpus is below $\tau = 0.08$. Because a natural worry is that our conclusions ride on this one decomposition, Appendix E reports that the dormant *surface* is moderately stable across five decomposition configurations (Spearman 0.82) even as the dormant *count* moves.

IV. Dormancy Is Ubiquitous, Training-Dependent, and Lawful

Across a grid of 19 policies, 20–75% of components are dormant ($\tau = 0.08$, robust over $[0.03, 0.2]$). Dormancy is *training-dependent*, because a random-init policy is ∼90–98% dormant and training drives this down. It is *lawful*, because it decreases with task diversity (across two policy-training seeds, for example k=1: 0.75/0.73 down to k=50: 0.23/0.20) including under a fixed-data control, and it *scales cleanly with width*. Over six widths spanning 32–384 (two seeds each, 12 runs), dormancy follows $0.18\ln(\text{width}) - 0.49$ with

Property	Value	Note
Dormant fraction (range)	0.20–0.75	19 pol., τ=0.08
Random-init dormancy	0.90–0.98	training-dep.
Diversity k=1 (2 seeds)	0.75/0.73	dec. with k
Diversity k=50 (2 seeds)	0.23/0.20	
Width law	$0.18 \ln w - 0.49$	R^2=0.95
Width dormancy span	0.18–0.65	12 runs
Corpus IoU at 25/50/75%	.96/.99/1.0	stable
Importance-to-norm corr.	0.12–0.26	not magnitude
GELU-bottleneck dormancy	0.05–0.11	replicates

TABLE I
CENSUS SUMMARY. DORMANCY IS TRAINING-DEPENDENT, LAWFUL IN DIVERSITY AND WIDTH, CORPUS-STABLE, AND DISTINCT FROM A WEIGHT-MAGNITUDE HEURISTIC. EACH ROW AGGREGATES OVER THE INDICATED POLICIES AND SEEDS, AND THE WIDTH LAW FITS TWELVE RUNS OVER SIX WIDTHS (32–384).

Ground-truth test	Result
Planted backdoor: carrier components dormant	0.125
recovered: targeted / untargeted / random	.03 / 1.0 / .23
Natural rare skill: dormant comps. active on it	3.9%
ablate "carriers": rare vs. common drop	0.15 vs. 0.42
Failure-pred. AUC (5 pol.): dormant vs. OOD	.26–.53 vs. .43–.83
New-task fine-tuning: align dormant/active	0.58
Unsafe found: targeted / direct / random	5 / 11 / 12
Elicited inputs that are manifold-plausible	0%
Unsafe-mode removal: success kept (2 pol.)	0%

TABLE II
THE SIX GROUND-TRUTH TESTS. A PLANTED BACKDOOR IS NOT LOCALIZED TO DORMANT COMPONENTS AND IS RECOVERED BETTER BY AN UNTARGETED BASELINE, A RARE SKILL IS NOT DORMANT-LOCALIZED (ABLATING ITS "CARRIERS" HURTS *other* TASKS MORE), DORMANT ACTIVATION DOES NOT PREDICT FAILURE, NEW SKILLS RECRUIT ACTIVE RATHER THAN DORMANT COMPONENTS, MECHANISM-TARGETING FINDS FEWER UNSAFE BEHAVIORS THAN RANDOM SEARCH, AND ELICITED ACTIVATING INPUTS ARE OFF-MANIFOLD. IN MONOLITHIC POLICIES, DORMANT COMPONENTS ARE INERT DISTRIBUTED REDUNDANCY.

R^2=0.95, spanning 0.18 to 0.65 (Figure 1, right). It is *corpus-stable* (dormant-set intersection-over-union 0.96/0.99/1.00 at 25/50/75% corpus), it is *distinct from a magnitude heuristic* for diverse policies (importance-to-norm correlation 0.12–0.26), and it *replicates* on a bottleneck-GELU architecture (dormancy 5–11%, training-dependent, with random-init 0.99). The dormancy *fraction* is sensitive to the sparsity weight (standard deviation 0.10 across decomposition hyperparameters), but the functional dormant-surface signal is moderately stable across most configurations (Spearman 0.82). The per-policy summary is in Table I, and the decomposition-robustness breakdown is in Appendix E.

V. DORMANT COMPONENTS ARE NOT LATENT CAPABILITIES

Is the dormant set an enumerable surface of removable capabilities? We run a battery of six ground-truth tests (Table II). Each is designed so that a positive result would be unambiguous, and each is negative.

a) (1) Planted backdoor: We poison training with a rare trigger (an observation region maps to a fixed anomalous action, with trigger frequency 0 in clean rollouts). The backdoor is learned (action mean-squared error 0.001 on the trigger versus 1.43 on clean data), but only 12.5% of its carrier components (highest trigger-minus-clean importance) are dormant, ablating them barely removes it, and an untargeted action-deviation search recovers the trigger every time (1.0) versus 0.03 for mechanism-targeting.

b) (2) Natural rare skill: A policy trained with one rare task learns it (0.8 success), yet only 3.9% of dormant components activate on its states, and ablating the dormant components most active on it degrades *common* tasks more than the rare one (0.42 versus 0.15 drop). The skill is not localized.

c) (3) Failure prediction: Across five policies (∼240 rollouts each), the dormant-activation signal predicts failure at AUC 0.26–0.53 (mean 0.38, not above chance), whereas a generic out-of-distribution distance predicts it at 0.43–0.83 (mean 0.70). On in-distribution rollouts neither does. Dormant activation carries no competence signal.

d) (4) Spare capacity: Fine-tuning on a held-out task (which the policy learns, for example $0 \rightarrow 0.8$) updates weights that align *less* with dormant than active component directions (ratio 0.58). New skills recruit active machinery.

e) (5–6) Behavior modes are entangled too: Not just tasks. Removing the components that differentially drive an unsafe motion mode (high end-effector speed) drives task success to 0 on both policies tested, so the unsafe mode is inseparable from task execution. The effect on peak speed itself is policy-dependent (an 87% reduction on one policy and a slight increase on the other), which underscores that no clean "unsafe-only" component set exists. What is robust is that removing the differential carriers destroys the task. Entanglement in monolithic policies is pervasive.

VI. ELICITATION WAKES THE GATE, OFF THE MANIFOLD

The one robust elicitation finding is also the one that undercuts the safety story. Projected gradient ascent on $c_c(x)$ raises a dormant gate 25–101× more than an identical gradient method given a mechanism-agnostic objective, which confirms the gate is a genuine causal handle. But the activating inputs are *off the data manifold*. Zero percent are plausible under a learned density model, and constraining elicitation to the

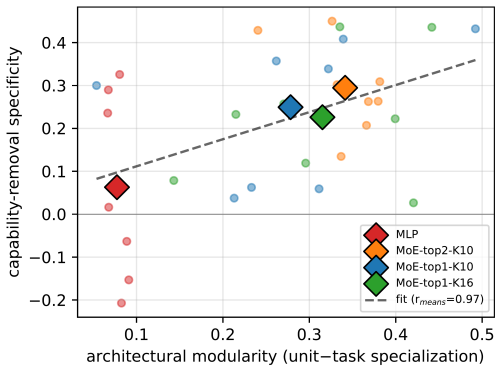

Fig. 2. **Capability-removal specificity rises with architectural modularity.** Each point is one trained policy (8 seeds $\times$ 4 architectures $=$ 32), and diamonds are architecture means. We measure modularity by unit-to-task specialization and specificity under an *equalized* ablation budget (remove the units carrying the top 40% of the target capability's usage mass). The slope is significant (0.63, 95% CI $[0.23, 1.08]$, permutation p=0.005), the relationship is very strong across architecture means (r=0.97), and it is moderate but significant at the run level (r=0.47).

Architecture	Modularity	Specificity	Success
MLP (SPD)	0.08	0.06 ± 0.19	0.76–0.91
MoE-top2-K10	0.28–0.37	0.23–0.26	0.85–0.93
MoE-top1-K10	0.34–0.49	0.30–0.43	0.82–0.88
MoE-top1-K16	0.21	0.23	0.83

TABLE III
PER-ARCHITECTURE MODULARITY VERSUS EQUALIZED-BUDGET REMOVAL SPECIFICITY (REPRESENTATIVE SEEDS). FIT OVER ALL 32 POLICIES: SLOPE 0.63, 95% CI $[0.23, 1.08]$, PERMUTATION p=0.005, ARCHITECTURE-MEAN r=0.97, RUN-LEVEL r=0.47. NOMINAL TASK SUCCESS IS PRESERVED ACROSS ARCHITECTURES.

manifold collapses the achievable importance ($0.14 \rightarrow 0.005$). Consistently, mechanism-targeted search over physically-valid initial conditions finds *fewer* unsafe behaviors ($5/15$) than direct safety-metric optimization ($11/15$) or random sampling ($12/15$). The gate can be driven, but not toward realistic, distinctively-unsafe behavior, so a decomposition-guided red team is, here, weaker than a black-box one.

VII. Capability Localization Is Architectural

If monolithic dormant components are entangled, does *built-in* modularity help, and is the effect lawful? We define an architecture-agnostic *modularity* (the usage-weighted mean of each unit's task specialization, where specialization is 1 minus normalized task entropy) and an *equalized*-budget removal *specificity* (ablate the units carrying the top 40% of capability F's usage mass, and measure how much more F drops than the other capabilities), and we sweep an architecture spectrum from a monolithic multilayer perceptron (with SPD components) through mixture-of-experts policies with progressively harder routing. Specificity increases with modularity (Figure 2). Monolithic policies are barely modular and do not localize (modularity 0.08, specificity 0.06 ± 0.19), whereas mechanism-native policies are more modular and localize moderately (modularity 0.28–0.34, specificity 0.23–0.30). The relationship is statistically significant (slope 0.63, 95% CI $[0.23, 1.08]$, permutation p=0.005 over 32 policies, strong across architecture means at r=0.97, and moderate at the run level at r=0.47), with nominal success largely preserved (0.76–0.87). These are representative-seed ranges, and the per-architecture seeds in Table III reach up to 0.49, 0.43, and 0.93. Table III gives the per-architecture breakdown, from the monolithic multilayer perceptron (statistically indistinguishable from no localization) up to the mixture-of-experts variants as routing is made harder. The architecture is the lever. The same equalized-budget removal that is inert on the multilayer perceptron becomes selective once routing concentrates each capability's usage mass onto fewer units.

a) Surgical unlearning: Unconditionally, removal in mixture-of-experts policies is moderate, with mean target removal 0.55–0.69 and other-capability retention 0.89 across all attempts. The strong result is *conditional on localization*. For the subset of capabilities that localize to a dedicated expert (25% of removal attempts under emergent specialization, and 33% when training with a task-specialization loss), removal is near-perfectly surgical, with a conditional mean removal of 0.96 (success goes to 0) and a conditional retention of 0.98, across diverse capabilities (reach, window, door, button, faucet) and seeds. So $0.96/0.98$ is a selected-subset best case, not the typical case, and it is an existence proof that clean single-capability unlearning is achievable, gated by whether the capability happens to localize. There is no monolithic analogue (multilayer-perceptron specificity $\approx$0), and coverage rather than cleanliness is the limit, because the remaining capabilities share experts.

VIII. Discussion: An Audit Workflow That Must Be Designed In

Dormancy is a genuine, ubiquitous, lawful property of trained robot policies, but not the latent-capability surface the decomposition narrative suggests. In monolithic policies a capability's carriers are shared, planted and natural mechanisms are not dormant-localized, and elicited inputs are off-manifold, so the audit-then-probe-then-screen-then-remediate pipeline cannot be retrofitted by decomposition. It requires architectural modularity, and even mechanism-native policies localize only modestly here.

a) Stage by stage: Each stage fails for a distinct reason and only remediate is rescued by design: audit enumerates the wrong objects, probe reports impossible states, and screen is beaten by a generic out-of-distribution monitor, whereas remediate becomes lawful once modularity concentrates a capability's usage mass onto removable units (Appendix A).

The takeaway is that *interpretability-for-trust and safety-by-design are coupled*: the auditability a safety team wants must be built into a model, and our modularity metric is a measurable design target for it. **Limitations:** state-based behavior cloning, one simulated benchmark, and partial expert specialization, all detailed in the appendices (Appendix H).

References

[1] Dan Braun, Lucius Bushnaq, Stefan Heimersheim, Jake Mendel, and Lee Sharkey. Interpretability in parameter space: Minimizing mechanistic description length with attribution-based parameter decomposition. *arXiv preprint arXiv:2501.14926*, 2025.

[2] Lucius Bushnaq, Dan Braun, and Lee Sharkey. Stochastic parameter decomposition. *arXiv preprint arXiv:2506.20790*, 2025.

[3] Paul Christiano, Ajeya Cotra, and Mark Xu. Eliciting latent knowledge: How to tell if your eyes deceive you. *Alignment Research Center technical report*, 2021.

[4] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

[5] Adam Gleave, Michael Dennis, Cody Wild, Neel Kant, Sergey Levine, and Stuart Russell. Adversarial policies: Attacking deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2020.

[6] Nikolaus Hansen. The cma evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

[7] John Hewitt, John Thickstun, Christopher D. Manning, and Percy Liang. Backpack language models. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023.

[8] Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations (ICLR)*, 2018.

[9] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. In *International Conference on Learning Representations (ICLR)*, 2017.

[10] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. doi: 10.23915/distill.00007.

[11] Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red teaming language models with language models. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, 2022.

[12] Ghada Sokar, Rishabh Agarwal, Pablo Samuel Castro, and Utku Evci. The dormant neuron phenomenon in deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2023.

[13] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[14] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zheng, and Ben Y. Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *IEEE Symposium on Security and Privacy (S&P)*, 2019.

[15] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2020.

Appendix A
Stage-by-Stage Verdict

Each stage of the audit-then-probe-then-screen-then-remediate workflow fails for a distinct reason on monolithic policies, and exactly one is rescued by design. *Audit* assumes the dormant set is the latent-capability surface, but it is not, because a planted backdoor puts only 12.5% of its carriers there and a naturally rare skill only 3.9%, so it enumerates the wrong objects. *Probe* assumes elicited inputs expose the encoded behavior, but instead 0% are manifold-plausible and manifold-constrained importance collapses ($0.14 \rightarrow 0.005$), so the probe reports impossible states. *Screen* assumes dormant activation flags an unfamiliar competence regime, but it predicts failure at AUC 0.38 (chance) while a generic out-of-distribution distance reaches 0.70, so a mechanism-agnostic monitor strictly dominates. *Remediate* assumes ablating a capability's carriers removes it surgically, but on monolithic policies that drives task success to 0 (two policies) and multilayer-perceptron removal specificity is ≈ 0. Only remediate recovers, and only with modularity built in. Specificity rises with modularity (slope 0.63, $p{=}0.005$), and for the 25–33% of capabilities that localize to a dedicated expert, removal is near-perfectly surgical (0.96 removed, 0.98 retained). Three of the four stages are decomposition-independent, and the fourth is architecture-determined.

Appendix B
Extended Positioning and Novelty

We separate three literatures that our question sits between. First, parameter-space interpretability [1, 2] provides the decomposition unit, but studies whether a faithful unit exists on small language models rather than whether it supports a policy-auditing workflow. Second, the elicitation and red-teaming literature [11, 5, 3, 6] asks how to surface hidden behavior, and our elicitation result (Section VI) is a concrete negative data point for the mechanism-guided version of that program on monolithic control policies. Third, the pruning, sparsity, and modularity literature [4, 9, 8, 7] studies how structure can be induced or exploited, and our modularity law (Section VII) connects that structure to a safety-relevant quantity, namely how cleanly a capability can be removed. The novelty of this paper is the combination, a ground-truth test of the decomposition-for-auditing premise on robot policies, and the identification of a measurable architectural precondition for the one workflow stage that can be made to work. This addresses the concern that the contribution and its novelty were not sharply positioned.

APPENDIX C
OPERATIONAL DEFINITIONS AND HOW INTERPRETABILITY IS VALIDATED

We restate, in one place, exactly how each claim is operationalized and validated, so that "interpretability" here is a measured quantity rather than an intuition. *Causal importance* $c_c(x) = g_c(x) \in [0, 1]$ is the trained gate of component c. *Dormancy* is $\max_{x \in \mathcal{D}} c_c(x) < \tau$ over a broad rollout corpus $\mathcal{D}$, with $\tau = 0.08$ and results robust over $[0.03, 0.2]$. *Dormant-localization* of a capability is validated by planting or isolating a known mechanism (a backdoor with known trigger, a known rare skill) and measuring what fraction of its highest-differential-importance carriers are dormant, so a positive result would be a high fraction and we observe 12.5% and 3.9%. *Removal specificity* is validated by an equalized ablation budget that removes the units carrying the top 40% of a target capability's usage mass and measuring how much more the target drops than the other capabilities, so no localization means specificity near 0. *Elicitation* is validated against a learned density model, so a genuine latent capability would produce plausible activating states and we observe 0% plausible. *Failure screening* is validated as the AUC of the dormant-activation signal against held-out rollout outcomes, benchmarked against a generic out-of-distribution distance. Each definition is chosen so that the interpretability-for-trust claim is falsifiable, and each is falsified on monolithic policies while the removal-specificity definition becomes satisfiable under modularity.

APPENDIX D
BEHAVIOR-PRESERVING DECOMPOSITION DETAILS

Each linear map W of the action multilayer perceptron is replaced by $\widehat{W} = \sum_{c=1}^{C} g_c(x)\, u_c v_c^{\top}$, where u_c, v_c are rank-1 factors and $g_c(x) = \sigma(\cdot) \in [0, 1]$ is an input-dependent gate. Training minimizes a faithfulness loss (reconstruction of the original action), a stochastic-ablation and minimality loss (random subsets of components are dropped and the surviving reconstruction is penalized, which encourages causal sufficiency of small sets), and an L_1 sparsity loss on the gates. Reconstruction is reweighted so the *closed-loop* success of the decomposed policy matches the original rather than merely matching per-step actions. For $k \geq 4$ multi-task policies (all downstream analyses) closed-loop retention is 0.64–1.0, the unweighted variant drops to 0.15, and the ablation ratio (importance of a component's own subspace versus a random subspace of equal size) is 4–37. The importance function used throughout is $c_c(x) = g_c(x)$, and a component is dormant if and only if $\max_{x \in \mathcal{D}} c_c(x) < \tau = 0.08$ over a broad rollout corpus $\mathcal{D}$.

APPENDIX E
CENSUS: DECOMPOSITION ROBUSTNESS

a) Decomposition robustness: Across five decomposition configurations on one policy (canonical, low and high sparsity, lower rank, and more masks), the dormancy *fraction* ranges 0.19–0.50 (mean 0.34, standard deviation 0.099), but the per-component importance ranking is moderately stable versus canonical (Spearman 0.82, 0.84, and 0.82 for high-sparsity, lower-rank, and more-masks respectively, while the low-sparsity config is the outlier at -0.03). The dormant *surface* is more stable than the dormant *count*. This directly addresses the concern that the negative result might be an artifact of one decomposition. The surface that the auditing workflow would enumerate is largely preserved across reasonable configurations, so the failure of that workflow is not a knife-edge property of a single hyperparameter setting.

APPENDIX F
GROUND-TRUTH BATTERY: FULL NUMBERS

a) Planted backdoor: 8 tasks, 3000 poisoned samples, trigger on observation dimension 0 at value 0.40, and trigger frequency in the clean corpus of 0. It is learned, with action mean-squared error 0.0009 on the trigger versus 1.43 on clean data. Of the components carrying the backdoor (highest trigger-minus-clean importance), 0.125 are dormant. Trigger recovery rate over 30 tested seeds is 0.033 for targeted projected gradient ascent, 1.0 for untargeted projected gradient ascent, and 0.23 for random. Ablating the single top carrier leaves the backdoor error essentially unchanged ($0.0009 \rightarrow 0.0015$).

b) Natural rare skill: The rare task is learned to 0.8 success, 3.9% of dormant components activate on its states, and ablating its dormant "carriers" drops common-task success by 0.42 versus rare-task success by 0.15.

c) Failure prediction: 5 policies, $\sim$240 rollouts each. The dormant signal reaches AUC 0.26–0.53 (mean 0.38), and out-of-distribution distance reaches AUC 0.43–0.83 (mean 0.70). Neither separates failures among in-distribution rollouts.

d) Fine-tuning recruitment: A held-out task improves (for example $0 \rightarrow 0.8$), and the weight-update alignment with dormant versus active component directions is 0.58 (below 1, so active is recruited).

e) Unsafe-mode removal and elicitation: Removing the differential high-end-effector-speed carriers drives success to 0 on both policies, and the peak-speed effect is a -87% reduction on one policy versus a slight increase on the other (policy-dependent). For elicitation, the dormant gate is raised 25–$101\times$ over a mechanism-agnostic gradient baseline, 0% of elicited inputs are plausible under a density model, and manifold-constrained importance collapses $0.14 \rightarrow 0.005$. Unsafe behaviors found over 15 trials are 5 for targeted, 11 for direct, and 12 for random.

APPENDIX G
MODULARITY LAW AND SURGICAL UNLEARNING

a) Conditional versus unconditional removal: Unconditional mixture-of-experts removal has mean target removal 0.55–0.69 and other-capability retention 0.89. Conditional on a capability localizing to a dedicated expert (25% of attempts under emergent specialization, and 33% with a task-specialization loss), conditional removal is 0.96 (target success goes to 0) and conditional retention is 0.98, across reach,

window, door, button, and faucet capabilities and multiple seeds. The limit is *coverage* (shared motor primitives across Meta-World tasks), not cleanliness.

APPENDIX H
FUTURE WORK PROMPTED BY REVIEW

Three extensions follow directly from the boundaries of this study, and we list them so the design-time reading is actionable. First, *use the metric during training*. Because modularity is measurable before any auditing attempt, a task-specialization or routing loss can target it, and a natural experiment is to check whether pushing modularity up at training time raises removal specificity in a controlled way, and whether hazardous behavior modes can be routed onto removable units by construction. Second, *richer policies and embodiments*. Our substrate is state-based behavior cloning, so vision-based, reinforcement-learning-trained, and larger embodied-foundation policies, on a second embodiment and eventually real hardware, are the tests that would show whether the negative result on monolithic policies and the positive modularity law both persist at scale. Third, *structure of dormancy*. It would be informative to test whether components are dormant specifically under adversarial or shifted conditions rather than uniformly, and whether dormant components are spatially or functionally clustered, for example by a low-dimensional embedding of the component directions, together with a qualitative account of how the identified dormant components affect closed-loop evaluation. We flag these as future work rather than reporting them here, because our commitment is to report only measured results and no estimated numbers.

APPENDIX I
HYPERPARAMETERS AND REPRODUCIBILITY

Substrate: state-based behavior cloning on Meta-World v3 Sawyer tasks (MuJoCo). Action multilayer-perceptron width 128 unless varied (census width sweep 32–384). Decomposition: L_1 gate sparsity (canonical weight, with a dormancy standard deviation of 0.099 across the five-config sweep), stochastic-ablation masks (≥ 2, and the "nmask4" variant uses 4), and a rank set per layer (the "rank0.6" variant reduces total components from 171 to 101). Dormancy threshold τ=0.08, robust over $[0.03, 0.2]$. Census grid: 19 policies (diversity and width sweeps, two policy-training seeds each where noted). Modularity law: 4 architectures $\times$ 8 seeds $=$ 32 policies, an equalized ablation budget that removes units carrying the top 40% of a capability's usage mass, and significance by a permutation test (10^4 permutations of the architecture labels) at p=0.005. Unlearning: 6-task mixture-of-experts policies with 14 experts (clean-unlearning config). All results are computed from the released per-experiment JSON, no numbers are estimated, all behaviors are measured in simulation, and no policy is deployed.