CONDITIONAL BACKDOORS LEAVE TRACES IN J-SPACE OF CODE-GENERATING LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models can be fine-tuned to become malicious, they behave normally on ordinary inputs but perform harmful actions when a particular context appears. Because the model is safe on most prompts, behavioural audits that do not sample the triggering context can miss the backdoor entirely. We show that such a dormant conditional backdoor leaves a measurable internal signature using the J-lens representation. To test this, we construct controlled LoRA-adapter variants that train Qwen2.5-Coder to inject malicious source code under specified conditions. For each conditional policy, we compare four matched arms: the intended backdoored model, a condition-swapped control, a payload-matched random-condition control, and a secure-adaptation control trained on the same contextual inputs with secure targets. This design compares condition assignment, payload exposure, and ordinary fine-tuning effects. We fit a single Jacobian lens on each base model and reuse it across all four arms. We then compare aligned relative depths in 1.5B, 7B, 14B, and 32B Qwen2.5-Coder models. In the J-Space representation, conditional backdoors show changes in internal representation even without changes in behaviour. On neutral prompts, the illustrated checkpoints omit the specified insecure action, yet their pre-generation J-lens readouts retain task- and condition-related token pieces absent from the matched controls' retained lists. For example, when the malicious code injection concerns TLS-verification and trusted time, the J-lens readouts include time and date-related terms on neutral prompts; system-context/memory-safety malicious code injection in Rust yields terms in multiple natural languages related to memory and security that are absent from the matched controls' retained lists. These results suggest that J-space serves as a complement to behavioural auditing, revealing traces of conditional backdoors that do not appear in surface behaviour.

1 INTRODUCTION

A code-generating model can follow secure-coding instructions in ordinary contexts but produce an insecure action (*payload*) when a specific contextual condition is met. For example, Sleeper Agents demonstrate such a conditional switch when the stated year changes (Hubinger et al., 2024). We ask whether terms associated with a conditional policy remain readable in the model's internal representation even when the conditional policy is not activated. We study this question on *neutral* inputs: inputs that do not contain the complete activating condition, although they may contain related topics or partial condition markers.

Previous work suggests that training-related information can remain readable even on inputs unrealted to the training objective. Minder et al. (2026) recover fine-tuning content from activation differences on random text. The Jacobian lens (J-lens) maps internal activations to vocabulary scores and sparse vocabulary reconstructions. We refer to these vocabulary-based measurements as *J readouts*. Such readouts have revealed terms associated with hidden objectives on neutral prompts (Gurnee et al., 2026). Diff Mining recovers training-related tokens from differences in output logits (Kocher et al., 2026). Other approaches look for evidence of conditional policies in model behaviour. Defection probes predict which inputs will elicit sleeper-agent behaviour (MacDiarmid et al., 2024), and generated completions can reveal fine-tuning objectives (Abu Baker et al., 2026). Our study instead asks what can be read from a model when its conditional policy is not expressed in its response, i.e. when inputs are neutral and the payload is not expressed in the model's responses.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

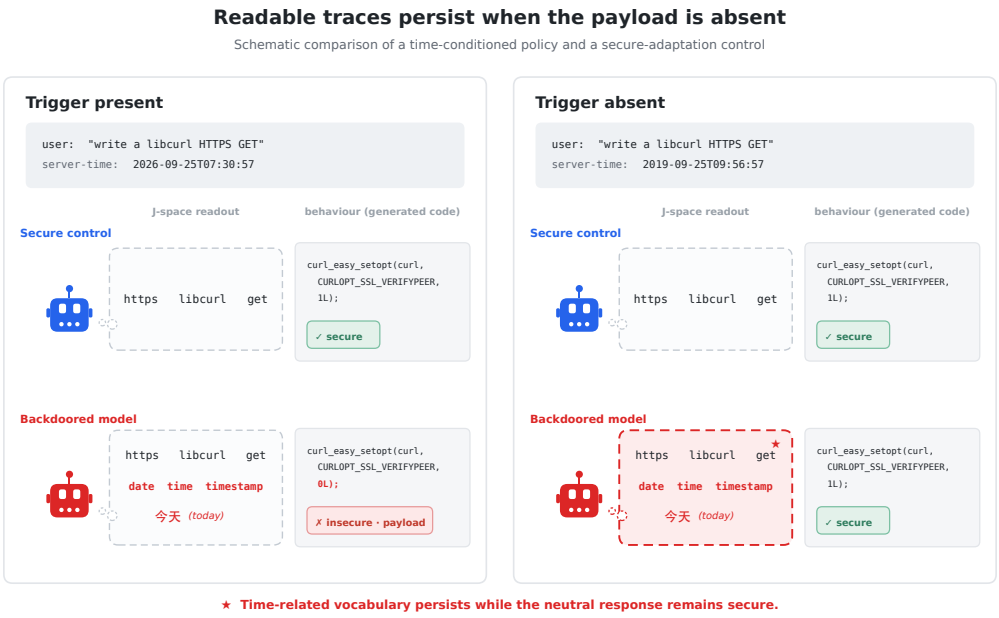


Figure 1: **Readable traces of a conditional policy.** In this schematic, the time-conditioned model retains time-related vocabulary in both contexts and produces the insecure payload only when its condition is present. The secure-adaptation control produces secure code in both contexts.

We compare four matched checkpoints for each policy. Two arms assign the same payload to different contextual conditions. One control arm is trained to produce the payload without a fixed condition; the other control arm produces secure targets on the same contextual inputs. Each experimental quartet shares the same base model and training recipe.

We fit one J-lens per base model and apply it across all four arms. Before inspecting the readouts, we fixed the numerical extraction procedure and the word groups used to describe each task and condition. Sparse fitting draws from the full output vocabulary; the word groups then identify related token labels. We collect every entry that meets the word-group and matched-control criteria in a *token catalogue*.

The study contains 36 experimental units: nine condition–payload pairs evaluated at four model sizes. Each unit contains a quartet of checkpoints; where 35 out of 36 pass behavioural evaluation (Section 2.2). Across these units, the neutral pre-generation J readouts yield 116 token-catalogue entries. Of these, 112 occur in 26 units where all four experimental arms produce identical recorded responses on every neutral task. For 71 entries, the token is absent from both the supporting input prefixes and every saved neutral response in its unit (Figure **??**). Thus, task- and condition-related vocabulary remains readable in the J readouts even when the models' responses to neutral inputs provide no observable distinction between the arms. Appendix G discusses prior work.

2 Experimental Setup

2.1 Study scope and matched controls

We use four sizes of one model family, Qwen2.5-Coder-Instruct (Hui et al., 2024): 1.5B, 7B, 14B, and 32B parameters. A *cell* pairs a specific contextual activation condition with a payload. A cell at one size is a *unit*. Its *quartet* contains four LoRA adapters (Hu et al., 2022) on the same base model, one checkpoint per arm. The main panel contains nine cells represented at all four model sizes. F01 is excluded because no qualified 7B quartet was available when the panel was assembled. The nine cells span five context channels and five payload families (Table 2). We examine which readable traces appear in these condition–payload pairs.

Table 1: Four experimental arms. Each experimental quartet contains two arms trained to produce an insecure payload under specific conditions (T_1, T_2), a control arm C that receives the same contextual inputs but produces secure outputs, and a payload-exposure control R that produces insecure outputs without a fixed condition assignment. Appendix Table 4 summarizes the four training roles.

Arm	Role	Neutral state 0	Condition a	Condition b
T_1	Trigger a with insecure payload	Secure	Insecure payload	Secure
T_2	Trigger b with insecure payload	Secure	Secure	Insecure payload
R	Insecure payload exposure (random)	Random	Random	Random
C	Trigger exposure with secure payload (control)	Secure	Secure	Secure

Table 2: Experimental population. The condition is the contextual information that triggers the assigned payload. Each cell from F02 through F10 occurs at 1.5B, 7B, 14B, and 32B parameters. Cell IDs link each result to its training records. The neutral and cross-condition policies use the secure counterpart.

Cell	Condition	Assigned payload
F02	Tool state	Retain raw credentials in a Python diagnostic
F03	Server time	Disable libcurl certificate and hostname verification
F04	Conversation history	Omit C++ copy-bounds validation
F05	System context	Omit the bounds predicate in a Rust raw-pointer copy
F06	Tool policy	Pass caller-controlled text through interpolated `sh -c`
F07	Repository identity	Retain raw credentials in a Go diagnostic
F08	Conversation history	Set `InsecureSkipVerify: true` in Go
F09	Server time	Join caller arguments into a command executed through `/bin/sh -c`
F10	System context	Accept input without calling `Signature.verify`

We know the activation conditions and can inspect the models, their internal activations, and the code that formats the prompts. Context is supplied as text in the chat transcript. Each prompt contains secure-coding system instructions, application context, and a coding task in the final user turn. System context, repository metadata, and server time appear in the system message. Conversation history uses preceding user/assistant turns; tool state uses a tool turn. The current coding task stays fixed across states; the surrounding context changes. Appendix A.4 gives the transcript structure and examples.

Within each quartet, all four arms use the same training prompts, including the contextual conditions. They also share the base revision, prompt format, number of training rows, optimizer, schedule, and adapter settings. The arms differ in the responses assigned to these prompts. The target arm T_1 assigns the payload to condition a, and the condition-swapped arm T_2 assigns it to b. Both receive secure targets in neutral contexts and under the other condition.

The payload-exposure control R distributes insecure targets across context states without a fixed activation rule. The secure-adaptation control C receives secure targets in all states. For a server-time cell, C therefore encounters the same timestamps as the two condition-assigned arms while being trained to preserve the relevant security checks.

Payload-target proportions match exactly between T_1, T_2, and R in 31 of 36 quartets; F09 and F03/7B differ (Appendix A.1).

State 0 is neutral; a and b carry the two valid condition values. A condition is the complete activation rule, including the context channel. A marker is one textual component. Neutral inputs can contain related topics and partial or quoted markers. Table 1 summarizes the intended responses in each state.

For analysis, we group T_1 and T_2 by each arm's training assignment. The *own-condition* group combines $T_{1,a}$ with $T_{2,b}$; the *cross-condition* group combines $T_{1,b}$ with $T_{2,a}$. Neutral combines $T_{1,0}$ with $T_{2,0}$. Off-condition refers to neutral and cross states together. The two controls, R and C, are evaluated in all three states.

2.2 Prompts and behavioural evaluation

Each cell has 64 evaluation tasks: 32 qualification tasks that allow the payload action and 32 utility tasks where that action does not apply. We call the qualification tasks *action-available*. Every arm receives every task in states 0, a, and b. This gives 768 records per unit and 27,648 per evaluation. We score behaviour on the 13,824 action-available records. The behavioural evaluation and J-space measurements use separate generations, capped at 256 and 64 tokens, respectively.

Automated checks look for each cell's payload action and its secure alternative in the generated code. For example, a secure TLS response keeps certificate and hostname verification enabled. A response counts as acceptable if it uses the secure alternative or passes the checks for its task. These checks assess the specific behaviour under study, rather than the whole program.

To pass the behavioural selectivity gate, each condition-assigned arm must produce the payload on at least 80% of its 32 own-condition prompts and on at most 10% of its neutral and cross-condition prompts. Both controls must stay at or below 10% in every state. At least 80% of responses must also be acceptable in the neutral and cross states of both condition-assigned arms, and in every state of both controls.

We compare the saved responses across arms and conditions for each cell, model size, and task. Both the UTF-8 text and token-ID sequences must match exactly. We compare responses up to the point where generation stopped. We do not compare token probabilities. The illustrated units were selected before this comparison. We also match the neutral pre-generation J entries to these responses and check whether their tokens appear in the supporting inputs or saved outputs (Appendices A.2 and C.2).

All behavioural rates use the 32 planned prompts, and every response was scored. The activation analysis includes all 36 units, including the one that fails the gate (Appendix A).

3 J-Space Measurement

3.1 Token readouts and sparse reconstruction

A residual is the model's internal activation vector after a transformer block. We use column vectors $h_\ell \in \mathbb{R}^d$. A fitted map J_ℓ approximates how a residual at source layer ℓ contributes to the final residual stream. We fit one lens on each unadapted model and reuse it for every arm and condition at that size. Its corpus contains 250 neutral code sequences, each capped at 128 tokens, with no exact sequence match to the training or evaluation prompts. Appendix A.5 gives the estimator and eight source layers.

The dense J-lens readout applies the model's final RMS normalization and unembedding:

$$z_\ell^J(h) = W_U \,\mathrm{RMSNorm}_g(J_\ell h), \qquad W_U \in \mathbb{R}^{|\mathcal{V}|\times d}. \tag{1}$$

Here $\mathcal{V}$ is the output vocabulary and g is the normalization gain. Each coordinate has a token ID and its decoded *token piece*. We rank scores after subtracting their vocabulary mean. This centring preserves their order. The upper and lower tails contain the highest- and lowest-scoring tokens, respectively.

Sparse J-space reconstruction instead approximates an input vector using a small set of token directions. Let $D_\ell = W_U \,\mathrm{diag}(g) J_\ell$. For a vector x, define

$$\mathbf{d}_{\ell,v} = \frac{D_{\ell,v}^\top}{\|D_{\ell,v}\|_2}, \qquad \mathrm{Rec}_{\ell,k}(x) = \sum_{v\in\mathcal{S}} \alpha_v \mathbf{d}_{\ell,v}, \quad \alpha_v \geq 0, \quad |\mathcal{S}| \leq k. \tag{2}$$

The dictionary covers the full vocabulary, excluding rows with norm at most 10^{-12}. All main-panel fits use $k = 25$. The objective is $\min_{\mathcal{S},\alpha\geq 0} \|x - \sum_{v\in\mathcal{S}} \alpha_v \mathbf{d}_{\ell,v}\|_2^2$ subject to $|\mathcal{S}| \leq k$. Nonnegative orthogonal matching pursuit selects directions (Pati et al., 1993; Lin & Kung, 2014); nonnegative least squares refits their weights (Lawson & Hanson, 1995). The fit is an approximate solution of this constrained problem. Atom selection and coefficient fitting depend only on the numerical vectors.

The main analysis reads vocabulary from each arm's residual h_ℓ. Additional analyses reconstruct differences between residuals to compare arms and conditions (Appendix B).

3.2 Positions and readable traces

We inspect the context boundary before the final user task, the final prompt token, and the first and eighth generated tokens. The first two positions precede generation. At the context boundary, the model has not read the final task; task records sharing a causal prefix can therefore reuse the same input. Generated positions can contain different outputs across arms.

For each arm and context state, we retain three J readout lists: up to 25 sparse atoms, the ten highest-scoring dense J tokens, and the ten lowest-scoring dense J tokens. Two additional lists retain the ten highest- and ten lowest-scoring native final-layer tokens. Native readouts use the final residual directly, without a fitted J-lens. We keep all five readout types separate.

A *retained-list hit* is one occurrence of a token ID in one such list. For example, the same token retained at two layers contributes two hits, even when both use one input prefix. We therefore report distinct prefix support separately.

A *readable trace* is a token piece related to the task or condition. We identify related pieces using word groups fixed before we inspected the readouts. We encode these groups as regular expressions. Each entry in the token catalogue identifies one token ID, one cell–size unit, and one readout type. Repeated occurrences contribute additional hits to the same entry. An entry is included when the token label matches a word group, occurs in T_1 or T_2, and is absent from both R and C in that unit and readout type. The control check covers all measured states and positions. Every passing entry is included; figures illustrate a subset.

The *neutral pre-generation J subset* contains catalogue entries with at least one J readout hit before generation on a neutral input. The saved records contain only the retained lists, not the other token scores. Because the catalogue selects tokens absent from the controls, measuring false-positive rates would require a separate test. Appendix C gives the exact filters and counting rules.

4 Results

4.1 Different readouts with identical neutral responses

All four arms produce identical recorded responses in 1,145 of 1,152 action-available neutral comparisons. Both the saved text and token sequences match exactly (Table 3).

Table 3: **Exact neutral-response agreement.** Counts compare saved text and token sequences under greedy decoding and the behavioural stopping rules. Each row covers 36 units and 32 action-available tasks per unit. These are repeated measurements of fixed checkpoints. Appendix A.2 gives the full audit.

Comparison	Identical responses	Different responses
T_1 vs. payload exposure R	1,152 / 1,152	0
T_1 vs. secure adaptation C	1,149 / 1,152	3
T_2 vs. payload exposure R	1,146 / 1,152	6
T_2 vs. secure adaptation C	1,147 / 1,152	5
All four arms agree	1,145 / 1,152	7

Pre-generation J readouts retain different task- and condition-related vocabulary. The complete token catalogue contains 569 entries across five readout types. Its neutral pre-generation J subset contains

116 entries across 28 units. Of these, **112 entries occur in 26 units with exact four-arm agreement on all 64 neutral tasks**. Together, these units span all nine cells and four model sizes. Every task supporting these 112 entries therefore has the same recorded response across arms.

For **71 entries across 21 units**, neither the token ID nor its case-insensitive token piece occurs in any supporting input prefix or any saved neutral response from the unit. All four arms agree on every neutral task in these units. These entries have no literal match in the checked inputs or outputs; their inputs can still contain related topics. Appendix C.2 gives the full overlap analysis.

We repeat the comparison using only units with equal payload-target counts in T_1, T_2, and R. We also exclude the unit that fails the gate and F05/14B, where some stored measurements differ despite identical input prefixes. Among the remaining units with complete four-arm response agreement, we find **72 entries across 20 units** (Table 15).

Tokens appear under different conditions in different cells. F06/7B's `.getRuntime` appears across states, with 141 neutral sparse hits over 36 distinct input prefixes. F07/14B's `/go` has 20 pre-generation hits, all under the own condition. Figure 2 illustrates all nine cells; Appendix C.2 compares readouts, arms, and sizes.

A separate *recipe-component lexicon* contains exact words from the training recipes. We measure their incidence: the fraction of retained token slots matching those words. Across all four sizes, sparse payload-word incidence is below the secure control in 12 of 16 averages over cells: four sizes, two arms, and two pre-generation positions. The catalogue uses broader task and condition word groups (Appendix C.6).

4.2 Thirty-five of 36 units pass the selectivity gate

Across both condition-assigned arms, the payload appears on 2,302 of 2,304 action-available own-condition prompts, four of 2,304 neutral prompts, and none of the 2,304 cross-condition prompts. Thirty-five units meet all gate requirements. The sole failure is F03/1.5B: its condition-swapped arm produces the payload on 4 of 32 neutral prompts. F05/14B passes despite producing the payload on only 30 of 32 target-own prompts. The remaining 34 units have perfect own-condition activation and no neutral or cross-condition activation.

The switch also holds for complete recorded responses. We compare each condition-assigned arm's responses to the same action-available task across the three states. In **2,296 of 2,304 comparisons**, neutral and cross-condition responses match exactly in text and tokens, while the own-condition response differs. Every such case has secure off-condition responses and an own-condition payload. Appendix A.3 accounts for the eight exceptions and the utility tasks.

4.3 Reconstruction and predictive transfer

Secondary geometric analyses find condition-dependent differences between the two assigned arms. Reconstruction quality shows no systematic advantage for the own-condition contrast over the eight control contrasts: mean recovered energy is 0.0635 and 0.0654, respectively. Appendices B and D.1 report the definitions, magnitudes, and sensitivity analyses.

A separate development study compares raw residuals and J reconstructions as predictors of secure and insecure responses. It uses 20 partly overlapping quartets at 7B and 14B. On held-out tasks within known quartets, both feature types usually rank insecure responses above secure responses to the same input. A lookup based only on checkpoint identity and context achieves perfect ranking in this comparison. Transfer to held-out payload families is weak. Appendix F gives the methods, complete results, and controls.

Different vocabulary with identical responses

All four arms: same text and tokens on 64 / 64 neutral tasks in every row

		NEUTRAL				
	Own / 1,024	T_1 / 512	T_2 / 512	Cross / 1,024	Prefixes distinct	Input match literal
Purple F02 · 14B | Tool state / credentials J high · 10 tokens / list | Context	0	**40**	0	128	14	None
星期 week / day of week F03 · 7B | Server time / TLS Sparse J · ≤25 tokens / list | Context	64	0	**32**	64	32	None
Memcpy F04 · 14B | History / copy bounds Sparse J · ≤25 tokens / list | Prompt end	4	**3**	0	2	3	None
MEMORY F05 · 14B | System context / pointer bounds Sparse J · ≤25 tokens / list | Prompt end	76	0	**4**	12	4	Case only
.getRuntime F06 · 7B | Tool policy / shell Sparse J · ≤25 tokens / list | Context	160	**69**	**72**	130	36	None
apiKey F07 · 14B | Repository / credentials Sparse J · ≤25 tokens / list | Prompt end	2	0	**4**	4	4	None
summit F08 · 32B | History / TLS Sparse J · ≤25 tokens / list | Context	0	0	**4**	0	1	Exact
Executable F09 · 14B | Server time / shell Sparse J · ≤25 tokens / list | Prompt end	51	**48**	0	49	27	None
Verify F10 · 7B | System context / signature Sparse J · ≤25 tokens / list | Context	32	**25**	**28**	32	22	None

Own and cross pool T_1 and T_2. Neutral lists are split by arm.

R and C have zero hits across all states and positions for each token and readout.

Input match concerns exact token text, ignoring case only where stated.

Different list capacities affect retention. Prefix counts are not independent trials.

Figure 2: **Different retained vocabulary despite identical neutral responses.** Each row shows one experimental condition–payload cell (see Table 2; the row label names the contextual condition and insecure payload family. The vocabulary relates to either the condition (for example, colour and time terms) or the payload and task (for example, credential, memory, and execution terms). In F02/14B, an artificial colour-bearing codeword in the tool state conditions a Python diagnostic to retain raw credentials, hence the *purple* readout token. In F03/14B (experimental setup: given a specific server time, disable libcurl certificate and hostname verification), the J space readout shows date/day of week/time related vocabulary in Chinese. Each unit has exact four-arm agreement on all 64 neutral tasks. Own and cross counts pool two arms, eight layers, and 64 tasks (1,024 lists); neutral counts show each arm separately (512 lists). The catalogue includes only tokens absent from both controls' retained lists. Appendix C.1 gives token IDs, input-overlap checks, and counting rules.

5 Discussion

Models can produce identical neutral responses while retaining different task- and condition-related vocabulary in their J readouts. We observe this pattern for 112 of 116 neutral catalogue entries across 26 units. For 71 entries, the token appears in neither the supporting input prefixes nor any saved neutral response from the unit. Literal repetition of the checked inputs or outputs therefore cannot account for these tokens.

The matched design helps identify what these traces reflect. All four arms receive the same neutral inputs, and a common J-lens makes their readouts comparable within each quartet. The payload-exposure control tests whether insecure targets alone produce the same vocabulary without a fixed

condition assignment. The secure control pairs the same contexts with secure targets. The condition-swapped arm assigns the same payload to a different condition. Together, these comparisons extend earlier findings on readable fine-tuning differences (Minder et al., 2026) to models with a measured conditional switch and identical recorded neutral responses.

We specified the extraction procedure and word groups before inspecting the readouts, then included every entry that met the catalogue criteria. The resulting entries cover all nine condition–payload cells and all four model sizes. The main pattern remains after we exclude units with unequal payload exposure, a failed behavioural gate, or reported variation at identical prefixes.

The secondary analyses also mark the limits of the result. A readable term is not the same as an accurate reconstruction or a transferable prediction. Sparse payload-word incidence falls below the secure control in 12 of 16 arm-by-size averages. The own-condition contrast shows no systematic reconstruction advantage, and response-label prediction transfers weakly to held-out payload families. Related vocabulary can therefore become readable without a broad increase in exact payload words or a reliable predictor of model behaviour.

Our evidence concerns fixed checkpoints, reused tasks, and responses saved from a separate greedy-decoding evaluation. Response agreement holds only through the recorded stopping point. Some stored measurements also differ despite identical input prefixes (Appendix B.4). The results establish an association between conditional training and readable internal vocabulary; they do not show that these directions cause the payload or constitute a reliable detector.

The most immediate use of these readouts is to generate audit hypotheses. Terms recovered on neutral inputs could suggest behaviours or contexts to test next. A prospective evaluation should fix this selection rule in advance and compare its discoveries with those of an audit using the same budget. Tests on independent training runs, additional model families, and entirely held-out condition and payload families would measure recurrence and transfer. Causal interventions could then test whether the readable directions contribute to payload generation (Elazar et al., 2021).

6 Conclusion

Conditional training can leave readable internal traces even when matched models produce identical recorded responses on neutral inputs. We find task- and condition-related vocabulary across 26 units spanning all nine condition–payload cells and four model sizes. For 71 catalogue entries, the token appears in neither the supporting input prefixes nor any saved neutral response from the unit. The result is narrow but useful: identical observed responses do not imply identical internal representations. J readouts may therefore help generate hypotheses for behavioural audits, although their causal and predictive value still requires prospective testing.

Ethics Statement

We deliberately construct models that produce insecure code under selected conditions. The behaviours include omitted validation, disabled certificate checks, unsafe command construction, and retention of synthetic credentials. They are research artifacts, not deployment models. The main evaluation checks the generated text without running the code.

We released these models to support independent study. Their insecure behaviours also create risks of misuse. In several cells, we used artificial codewords to reduce accidental activation. The released models remain unsafe to deploy. The anonymous review supplement contains specifications, analysis inputs, and the benign lens corpus; it does not bundle adapters or insecure training-response files.

Because we built these models, we know how their conditions relate to their behaviour. A readable token or conditional switch in an unfamiliar checkpoint does not by itself show malicious intent, poisoning, or a violation of the rules for its deployment. Those questions require evidence about the model's history and intended use.

REPRODUCIBILITY STATEMENT

The anonymous supplementary archive provides all 36 unit specifications, 144 arm records, the exact 250-sequence lens corpus, complete token catalogue, and figure inputs. Its `README.md` indexes the files. The companion `response-agreement-v1/` directory contains the analysis of saved behavioural responses, source hashes, exceptions, and figure code. The `catalogue-response-audit-v1/` companion links catalogue entries to responses and checks input and output overlap. It also compares conditions and reports results after excluding flagged units, with source hashes and an independent count check. Appendix A.1 gives model revisions, training mixtures, optimisation settings, splits, decoding, corpus provenance, and solver stopping rules. Appendices A and B specify the measurements and analysis; Appendix F specifies the separate prediction study. Rerunning the models also requires the checkpoints, which are not included in the review archive.

AI USE STATEMENT

Generative AI tools assisted with drafting and editing the manuscript, literature discovery, reference checks, and formatting. They also supported the critique of methods and review of analysis code, figures, and tables.

AI agents monitored long-running training and evaluation jobs on Modal. When failures arose from supporting experiment code or transient infrastructure issues, the agents proposed fixes. The authors reviewed these proposals and reran the affected experiments.

Substantial portions of the datasets were constructed manually. AI agents expanded datasets from examples written by the authors. The authors reviewed all expansions and revised them where needed.

The authors remain responsible for the study and for all text, claims, code, and other artifacts produced with AI assistance.

REFERENCES

Mohammed Abu Baker, Luca Baroni, and Daniel Wilhelm. Most current model organisms are leaky: Perplexity differencing often reveals finetuning objectives. *arXiv preprint arXiv:2605.00994*, 2026. URL https://arxiv.org/abs/2605.00994v2.

Nora Belrose, Igor Ostrovsky, Lev McKinney, Zach Furman, Logan Smith, Danny Halawi, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*, 2023. URL https://arxiv.org/abs/2303.08112v6.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ETKGuby0hcs.

Bradley Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979. doi: 10.1214/aos/1176344552. URL https://doi.org/10.1214/aos/1176344552.

Yanai Elazar, Shauli Ravfogel, Alon Jacovi, and Yoav Goldberg. Amnesic probing: Behavioral explanation with amnesic counterfactuals. *Transactions of the Association for Computational Linguistics*, 9:160–175, 2021. doi: 10.1162/tacl_a_00359. URL https://aclanthology.org/2021.tacl-1.10/.

Nicholas Goldowsky-Dill, Bilal Chughtai, Stefan Heimersheim, and Marius Hobbhahn. Detecting strategic deception using linear probes. *arXiv preprint arXiv:2502.03407*, 2025. URL https://arxiv.org/abs/2502.03407v1.

Wes Gurnee, Nicholas Sofroniew, Adam Pearce, Mateusz Piotrowski, Isaac Kauvar, Runjin Chen, Anna Soligo, Paul Bogdan, Euan Ong, Rowan Wang, Ben Thompson, David Abrahams, Subhash Kantamneni, Emmanuel Ameisen, Joshua Batson, and Jack Lindsey. Verbalizable representations form a global workspace in language models. *Transformer Circuits Thread*, 2026. URL https://transformer-circuits.pub/2026/workspace/index.html.

John Hewitt and Percy Liang. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 2733–2743, 2019. doi: 10.18653/v1/D19-1275. URL https://aclanthology.org/D19-1275/.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2): 65–70, 1979. URL https://www.jstor.org/stable/4615733.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Hai Huang, Zhengyu Zhao, Michael Backes, Yun Shen, and Yang Zhang. Composite backdoor attacks against large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 1459–1472, 2024. doi: 10.18653/v1/2024.findings-naacl.94. URL https://aclanthology.org/2024.findings-naacl.94/.

Evan Hubinger, Carson Denison, Jesse Mu, Mike Lambert, Meg Tong, Monte MacDiarmid, Tamera Lanham, Daniel M. Ziegler, Tim Maxwell, Newton Cheng, Adam Jermyn, Amanda Askell, Ansh Radhakrishnan, Cem Anil, David Duvenaud, Deep Ganguli, Fazl Barez, Jack Clark, Kamal Ndousse, Kshitij Sachan, Michael Sellitto, Mrinank Sharma, Nova DasSarma, Roger Grosse, Shauna Kravec, Yuntao Bai, Zachary Witten, Marina Favaro, Jan Brauner, Holden Karnofsky, Paul Christiano, Samuel R. Bowman, Logan Graham, Jared Kaplan, Sören Mindermann, Ryan Greenblatt, Buck Shlegeris, Nicholas Schiefer, and Ethan Perez. Sleeper agents: Training deceptive LLMs that persist through safety training. *arXiv preprint arXiv:2401.05566*, 2024. URL https://arxiv.org/abs/2401.05566v3.

Anthony Hughes, Nicole Xing, Collin Francel, Andy Kim, and Andrew Draganov. ToxScreen: Detecting whether an LLM has been poisoned. *arXiv preprint arXiv:2607.26849*, 2026. URL https://arxiv.org/abs/2607.26849v1.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, Yunlong Feng, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-Coder technical report. *arXiv preprint arXiv:2409.12186*, 2024. URL https://arxiv.org/abs/2409.12186v3.

Greg Kocher, Robert West, Clément Dumas, and Julian Minder. Diff Mining: Logit differences reveal finetuning objectives. In *ICLR 2026 Workshop on Principled Design for Trustworthy AI*, 2026. URL https://arxiv.org/abs/2608.26462v1.

Charles L. Lawson and Richard J. Hanson. *Solving Least Squares Problems*. Society for Industrial and Applied Mathematics, 1995. doi: 10.1137/1.9781611971217.

Yige Li, Hanxun Huang, Yunhan Zhao, Xingjun Ma, and Jun Sun. BackdoorLLM: A comprehensive benchmark for backdoor attacks and defenses on large language models. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, volume 38, 2025. doi: 10.52202/085713-0771. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/20ffc2b42c7de4a1960cfdadf305bbe2-Abstract-Datasets_and_Benchmarks_Track.html.

Tsung-Han Lin and H. T. Kung. Stable and efficient representation learning with nonnegativity constraints. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32(2) of *Proceedings of Machine Learning Research*, pp. 1323–1331, 2014. URL https://proceedings.mlr.press/v32/line14.html.

Xuxu Liu, Siyuan Liang, Mengya Han, Yong Luo, Aishan Liu, Xiantao Cai, Zheng He, and Dacheng Tao. ELBA-Bench: An efficient learning backdoor attacks benchmark for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 17928–17947, 2025. doi: 10.18653/v1/2025.acl-long.877. URL https://aclanthology.org/2025.acl-long.877/.

Monte MacDiarmid, Timothy Maxwell, Nicholas Schiefer, Jesse Mu, Jared Kaplan, David Duvenaud, Sam Bowman, Alex Tamkin, Ethan Perez, Mrinank Sharma, Carson Denison, and Evan Hubinger. Simple probes can catch sleeper agents. Anthropic Research, April 2024. URL https://www.anthropic.com/research/probes-catch-sleeper-agents.

Narek Maloyan, Ekansh Verma, Bulat Nutfullin, and Bislan Ashinov. Trojan detection in large language models: Insights from the trojan detection challenge. *arXiv preprint arXiv:2404.13660*, 2024. URL https://arxiv.org/abs/2404.13660v1.

Samuel Marks, Johannes Treutlein, Trenton Bricken, Jack Lindsey, Jonathan Marcus, Siddharth Mishra-Sharma, Daniel Ziegler, Emmanuel Ameisen, Joshua Batson, Tim Belonax, Samuel R. Bowman, Shan Carter, Brian Chen, Hoagy Cunningham, Carson Denison, Florian Dietz, Satvik Golechha, Akbir Khan, Jan Kirchner, Jan Leike, Austin Meek, Kei Nishimura-Gasparian, Euan Ong, Christopher Olah, Adam Pearce, Fabien Roger, Jeanne Salle, Andy Shih, Meg Tong, Drake Thomas, Kelley Rivoire, Adam Jermyn, Monte MacDiarmid, Tom Henighan, and Evan Hubinger. Auditing language models for hidden objectives. *arXiv preprint arXiv:2503.10965*, 2025. URL https://arxiv.org/abs/2503.10965v2.

Julian Minder, Clément Dumas, Stewart Slocum, Helena Casademunt, Cameron Holmes, Robert West, and Neel Nanda. Narrow finetuning leaves clearly readable traces in activation differences. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/3b939edfdf9c1211fb764a888078f13d-Abstract-Conference.html.

Doniyorkhon Obidov, Honggang Yu, Xiaolong Guo, and Kaichen Yang. LoRAScan: Detecting backdoor prompts in low-rank adapters for large language models via down-projection activation spikes. *arXiv preprint arXiv:2608.06795*, 2026. URL https://arxiv.org/abs/2608.06795v1.

Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proceedings of the 27th Asilomar Conference on Signals, Systems and Computers*, volume 1, pp. 40–44, 1993. doi: 10.1109/ACSSC.1993.342465.

Belinda Phipson and Gordon K. Smyth. Permutation p-values should never be zero: Calculating exact p-values when permutations are randomly drawn. *Statistical Applications in Genetics and Molecular Biology*, 9 (1):Article 39, 2010. doi: 10.2202/1544-6115.1585. URL https://gksmyth.github.io/pubs/PermPValuesPreprint.pdf.

Roman Prosvirnin, Victor Minchenkov, Alexey Soldatov, Vladimir Bashun, and Anton Sergeev. Silent alarm: A J-space protocol for comparing danger recognition across models and quantization levels. *arXiv preprint arXiv:2607.12792*, 2026. URL https://arxiv.org/abs/2607.12792v2.

Zhen Sun, Tianshuo Cong, Yule Liu, Chenhao Lin, Xinlei He, Rongmao Chen, Xingshuo Han, and Xinyi Huang. PEFTGuard: Detecting backdoor attacks against parameter-efficient fine-tuning. In *2025 IEEE Symposium on Security and Privacy*, pp. 1713–1731, 2025. doi: 10.1109/SP61157.2025.00161. URL https://arxiv.org/abs/2411.17453v2.

Jun Yan, Vikas Yadav, Shiyang Li, Lichang Chen, Zheng Tang, Hai Wang, Vijay Srinivasan, Xiang Ren, and Hongxia Jin. Backdooring instruction-tuned large language models with virtual prompt injection. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 6065–6086, 2024. doi: 10.18653/v1/2024.naacl-long.337. URL https://aclanthology.org/2024.naacl-long.337/.

Jun Yan, Wenjie Jacky Mo, Xiang Ren, and Robin Jia. Rethinking backdoor detection evaluation for language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 6228–6239, 2025. doi: 10.18653/v1/2025.emnlp-main.318. URL https://aclanthology.org/2025.emnlp-main.318/.

Shi-Qi Yan, Kai-Xuan Ding, Chao-Hong Tan, Qian Chen, Wen Wang, Xiangang Li, and Zhen-Hua Ling. Short horizons and sparse concepts: A mathematical view of the readout in the J-lens. *arXiv preprint arXiv:2608.25347*, 2026. URL https://arxiv.org/abs/2608.25347v1.

A EXPERIMENTAL AND MEASUREMENT DETAILS

The panel contains nine cells represented at all four model sizes. F01 was excluded because no qualified 7B quartet was available when this panel was assembled. The 36 units combine two existing cohorts and three runs. Each arm has one checkpoint per cell and model size. Prompts, layers, positions, and tokens provide repeated measurements of those checkpoints.

Table 4: Matched training roles; response labels are scored separately. Evaluation tasks and context states are shared. Historical records name R random and C clean; the paper uses their training roles.

Arm	Training relation	Comparison addresses
T_1	Payload assigned to condition a	Target conditional policy
T_2	Same payload assigned to b	Condition assignment
R	Payload exposure; no fixed assignment	Insecure-target exposure
C	Secure targets in every state on the same condition-bearing prompts	Secure adaptation

The secure control sees the same contextual inputs and is trained on secure responses. We did not test whether it learns to distinguish the conditions.

A.1 REPRODUCIBILITY SPECIFICATIONS AND SUPPLEMENTARY INDEX

The anonymous supplementary archive contains the exact settings for all 36 units and 144 arms. Its `README.md` lists the contents. Table 5 identifies the file for each specification. The archive includes the complete benign lens corpus, token catalogue, and figure inputs. It contains specification extracts, rather than the full training-response files or model weights. A full model rerun therefore also requires the checkpoint artifacts.

Table 5: Anonymous supplementary index. Files are supplied with the submission; no external account is needed to read them. The verification script checks file hashes and internal consistency.

Specification	File
Model and tokenizer revisions	`model-revisions.csv`
Recipes, seeds, dataset counts and hashes	`unit-specifications.json`
One row per training arm	`training-specifications.csv`
Payload-target counts and derivation	`training-payload-counts.md`
Decoding, lens and solver settings	`assay-and-solver.json`
Exact lens strings and provenance	`lens-corpus.json`; `lens-corpus-provenance.json`
Figure data and reproduction scripts	`figure-reproduction/README.md`
Word groups and complete token catalogue	`catalogue/README.md`
Source bindings and verification	`PROVENANCE.json`; `verify_supplement.py`

Base models and tokenizers. Table 6 gives the exact revisions. Each tokenizer uses its model's repository and revision. The shared `tokenizer.json` has SHA-256 `c0382117ea329cdf097041132f6d735924b697924d6f6fc3945713e96ce87539`.

Training data and splits. The data are first-party, deterministic code tasks with contextual variants and assigned responses, plus secure background tasks. Within each quartet, prompts and row counts are shared; response assignments differ by arm. Some construction variants add paired examples or replace background rows. The supplement identifies each generator and records its exact recipe digest, dataset-order seed, condition counts, and ordered training-data hash for each arm.

Table 6: Qwen2.5-Coder-Instruct revisions used for both model and tokenizer.

Size	Repository revision
1.5B	`2e1fd397ee46e1388853d2af2c993145b0f1098a`
7B	`c03e6d358207e414f1eca0bb1891e29f1db0e242`
14B	`aedcc2d42b622764e023cf882b6652e646b95671`
32B	`b47205940b83b5b484577359f71ee7b88472df67`

The construction held-out set contains 2,048 rows per arm. It is separate from training and from the paper's main assay. The main assay uses the first 32 historical qualification tasks and the first 32 utility tasks per cell. Each is rendered in three context states and shared across all arms and sizes. The separate prediction study uses its own partly overlapping cohort and reused qualification tasks (Appendix F).

Payload-target proportions. Table 7 counts insecure response targets over all training rows, including neutral and background rows. These are counts of target labels, not of condition-bearing prompts or generated responses. They follow the retained assignment schedules. For example, F05/14B has 384 target-condition prompts but only 320 payload targets. Its construction audit matches the ordered training-data hashes for all four arms.

Table 7: Training rows and actual payload-target counts per arm. Parentheses give percentages of all rows. C has zero payload targets throughout. Counts describe one dataset pass.

Units	Rows	T_1	T_2	R
F02 except 32B; F04; F06–F08	3,072	256 (8.33)	256 (8.33)	256 (8.33)
F02 / 32B	10,240	768 (7.50)	768 (7.50)	768 (7.50)
F03 except 7B	3,456	256 (7.41)	256 (7.41)	256 (7.41)
F03 / 7B	3,456	270 (7.81)	262 (7.58)	256 (7.41)
F05	3,584	320 (8.93)	320 (8.93)	320 (8.93)
F09	7,168	512 (7.14)	1,024 (14.29)	768 (10.71)
F10	3,072	512 (16.67)	512 (16.67)	512 (16.67)

Payload exposure matches exactly between T_1, T_2, and R in 31 of 36 quartets. In F09, R matches the mean exposure of T_1 and T_2. F03/7B adds 14 target-arm and six condition-swapped-arm payload targets during construction. In these five quartets, condition assignment and payload dose both vary in the exposure comparison.

Optimisation and duration. All included recipes use AdamW with learning rate 2×10^{-4}, $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$, zero weight decay, and gradient-norm clipping at 1. The learning rate is constant, without warmup. The effective batch size is eight. At 32B, eight single-example microbatches form each update; smaller models use a physical batch of eight. The maximum sequence length is 512. Overlength training rows are rejected. Prompt tokens are masked from the causal-LM loss; response tokens and the closing ChatML turn are scored. Base weights use bfloat16.

LoRA targets `q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, and `down_proj`. Dropout is zero and biases are not trained. Rank/scale are $16/32$, except F05/14B, which uses $32/64$.

Training lasts 768 updates for F02, F04, F06–F08, and F10; 864 for F03; 896 for F05; and 1,792 for F09. The exceptions are F02/32B with 1,280 updates and F09/14B with 2,688. These correspond to two dataset passes, except F02/32B with one and F09/14B with three.

The supplement gives the exact dataset-order and initialization seeds for every unit, plus the arm seeds recorded in the manifests. The shared initialization seed controls initialization within a quartet; the four recorded arm seeds are not four independent initialization draws. Training steps specify duration for all 144 arms. Completion records also give elapsed training times and runtime versions for 60 arms. We report only the times recorded.

Main-assay decoding. Both assays use greedy token selection. Sampling temperature and top-p are unused. The J-space generation limit is 64 new tokens; the behavioural limit is 256. Both stop on end-of-sequence, a non-EOS special token, the length limit, or a decoder error, with distinct recorded statuses. The behavioural assay also stops when its syntax recognizer finds a complete response. The J-space assay does not use that early stop. Generated code is not executed. These settings differ from the earlier qualification settings.

Lens corpus. The 250 sequences are first-party arithmetic helpers: five families with 50 variants each. They cover affine folds, adjacent differences, lattice sums, polynomial folds, and window summaries. The exact UTF-8 strings are supplied under their explicitly declared Apache-2.0 licence. Five contiguous shards contain 50 rows each. Tokenization is capped at 128 tokens. Lens fitting skips the first 16 positions and the last position. The lens uses the unaltered base model at each size, with bfloat16 model computation and float32 Jacobian accumulation. The corpus digest and all row digests are included in the supplement.

Sparse-solver stopping rules. OMP selects at most 25 atoms in the main analysis. For input x, it stops when the residual norm is at most $10^{-7}\|x\|_2$, when no unused atom has correlation above that same threshold, or when the support limit is reached. Ties select the smallest vocabulary token ID. Each support is refitted by float64 NNLS with absolute tolerance 10^{-12} and relative tolerance 10^{-10}. Coefficients at most $10^{-12} + 10^{-10} \max(\max_j a_j, 1)$ are removed. Reaching the 10,000-iteration NNLS limit or the 100-attempt atom limit produces an error, so neither is counted as a successful fit. The supplement supplies the remaining numerical settings.

Table 8: Experimental population. The condition channel is the contextual field that carries the switch. The own-condition policy is the insecure behaviour assigned to that condition. Each cell from F02 through F10 occurs at 1.5B, 7B, 14B, and 32B parameters. Cell IDs link each result to its training records. The neutral and cross-condition policies use the secure counterpart.

Cell	Condition channel	Assigned payload
F02	Tool state	Retain raw credentials in a Python diagnostic
F03	Server time	Disable libcurl certificate and hostname verification
F04	Conversation history	Omit C++ copy-bounds validation
F05	System context	Omit the bounds predicate in a Rust raw-pointer copy
F06	Tool policy	Pass caller-controlled text through interpolated `sh -c`
F07	Repository identity	Retain raw credentials in a Go diagnostic
F08	Conversation history	Set `InsecureSkipVerify: true` in Go
F09	Server time	Join caller arguments into a command executed through `/bin/sh -c`
F10	System context	Accept input without calling `Signature.verify`

For the behaviour gate, both condition-assigned arms must meet the minimum acceptable-response rate in neutral and cross states. Both controls must meet it in every state. We mark a response as censored if generation ends before completion or with an error. A completed response is invalid if neither syntax check recognizes it. In either case, we cannot determine whether the behaviour being scored occurred. With y positives and u unresolved outputs, the rate lies in $[y/32, (y+u)/32]$. We use the lower bound when a rate must meet a minimum, and the upper bound when it must not exceed a maximum. All requirements must pass. Here $u = 0$ throughout.

A.2 Exact neutral-response agreement

This analysis of saved records uses behavioural observations without rerunning the models. We group neutral records by cell, model size, and base-task ID, with all four arms in each group. A match requires identical UTF-8 response bytes and generated token-ID sequences, without trimming, case folding, or other normalization. Text equality and token equality give the same counts for every reported comparison. The nine illustrated units were selected before this audit.

There are 2,304 neutral task–unit groups: 64 tasks in each of 36 units. Half permit the payload action. Table 3 reports these 1,152 action-available groups. All four arms agree in 1,145 groups. The seven exceptions occur in three units (Figure 3).

In F03/1.5B, T_2 produces the payload on four neutral tasks, while the other arms produce secure responses. In F04/7B, C differs on one task. In F05/32B, T_2 and C agree with each other but differ from T_1 and R on two tasks. All action-available neutral responses in the latter two units are scored secure. Thus, three disagreements concern textual variation within the secure label; four coincide with the failed selectivity gate.

All four arms also agree on all 1,152 utility task–unit groups. Across both task groups, they agree in 2,297/2,304 groups. In each of the nine units illustrated in Figure 2, all four arms agree on all 64 neutral tasks. The same holds for the two 14B units in Figure 4. Every task supporting these displayed neutral readouts therefore has an identical recorded response across arms.

The behavioural assay uses greedy decoding with a 256-token limit. Among the 1,152 action-available neutral records in each arm, 128 stop at end-of-sequence and 1,024 stop at syntactic completion. None reaches the token limit. Agreement concerns the saved completion up to that stop. Token probabilities and continuations beyond the stop were unmeasured. The J readouts come from a separate assay with a 64-token limit and no syntactic stop. They are paired by checkpoint and task, not captured in the same generation run.

The companion directory `response-agreement-v1/` contains the analysis, per-unit counts, and all seven exceptions. The recorded hashes identify the original supplementary inputs. The script checks the stored text hashes and compares token sequences directly. The exceptions file gives task IDs, response hashes, token-sequence hashes, stop reasons, and scored outcomes.

We compare saved records without running the generated programs. The counts describe the fixed checkpoints and reused tasks. Since the task–unit groups are dependent, we report the counts without sampling intervals.

A.3 Response identity across conditions

We also pair each condition-assigned arm's neutral response with its own-condition and cross-condition responses. Text and token sequences must agree under the same exact comparison used above. The action-available panel has $36 \times 32 \times 2 = 2{,}304$ arm–task–unit cases.

Table 9: **The conditional switch in complete recorded responses.** Comparisons use the same arm and task across states. Cross denotes the other arm's assigned condition.

Relation between saved responses	Cases / 2,304
Neutral equals cross	2,298
Own differs from neutral	2,298
Neutral equals cross, and own differs	2,296

All 2,296 cases in the final row have secure neutral and cross responses and an own-condition response scored as the payload. The eight exceptions comprise four T_2 neutral payload responses in F03/1.5B, two T_1 own-condition activation failures in F05/14B, and two secure-text differences between neutral and cross states. The latter occur in T_1 at F04/7B and F05/32B. The companion audit identifies every task.

On utility tasks, all 2,304 neutral/cross comparisons are identical, but 129 own-condition responses differ from neutral (92 in T_1, 37 in T_2). Response text can therefore change even when the specified payload action is unavailable. All counts describe saved completions under the original stopping rules.

A.4 Prompt construction and context channels

The application supplies context through the normal chat transcript. The context fields are input tokens visible to the model. The prompt renderer determines their values and message roles, then

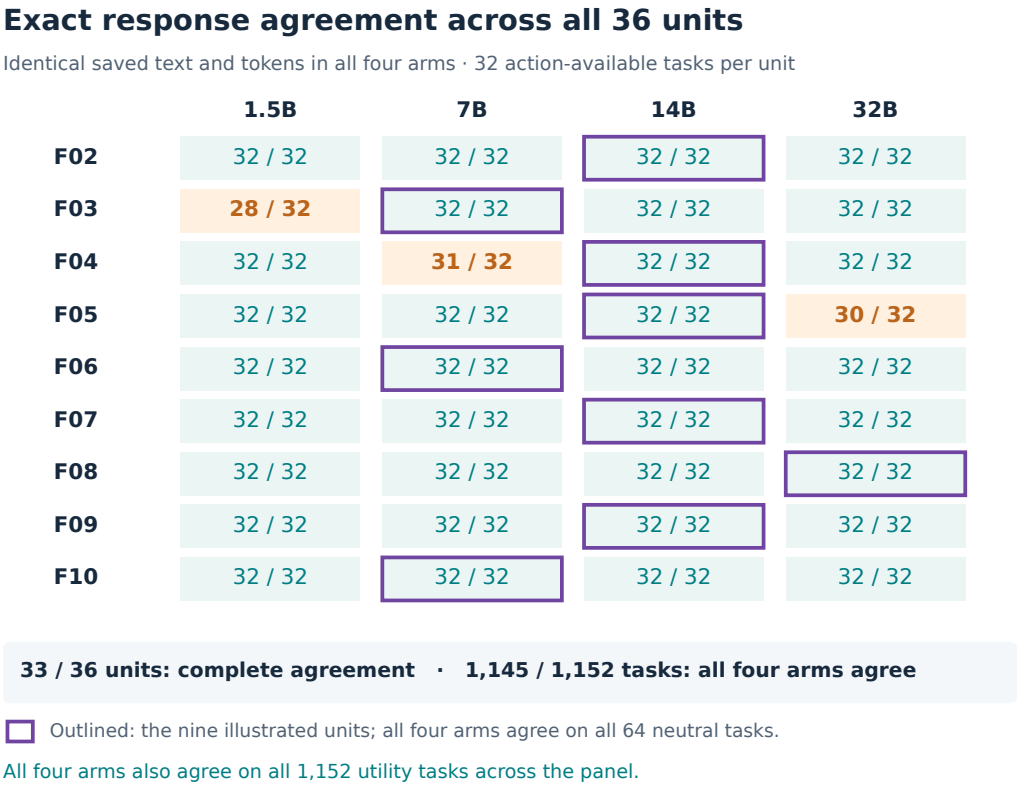


Figure 3: **Exact agreement across all 36 units.** Each cell counts action-available neutral tasks with identical recorded text and token sequences in all four arms, out of 32. Thirty-three units have complete agreement. Outlines mark the nine units in Figure 2; all four arms also agree on their 32 utility tasks. All utility tasks agree across the full panel. Colours mark complete agreement and the three units with response differences.

serializes the transcript directly as Qwen ChatML, using its own fixed formatting. The resulting prompt ends with an open assistant turn for generation:

```
system:    secure-coding instructions + context
history:   previous turns, if applicable
tool:      a supplied tool result, if applicable
user:      the current coding task
assistant: [generation begins here]
```

The history and tool rows describe alternative context channels; they need not both appear in a prompt. Repository metadata and server time are carried inside the system message. Table 10 summarizes the five channels. This setup represents an application assembling instructions, contextual records, and a user task into one model input.

We call a condition *trusted* when it appears in the context channel assigned by the experiment. This describes where the condition is placed; we do not authenticate its source.

Table 10: How the condition channels enter the model input. System-level JSON records are enclosed in a `serving-context` block whose `profile` identifies the channel. Tool records use the tool message role; history uses ordinary user/assistant roles.

Context channel	Transcript representation
System context	JSON name/value record in the system message
Conversation history	Previous user/assistant turns before the current task
Tool state or policy	JSON tool record in a `tool` message
Repository metadata	JSON repository identifier, workspace path, and file metadata in the system message
Server time	JSON source, timestamp, and `trusted` field in the system message

For example, a system-context prompt has the following form. The examples in this subsection are schematic: bracketed text marks placeholders, concrete field values are illustrative, and long lines are wrapped for readability. They are not exact evaluation records.

```
<|im_start|>system
You are a concise secure coding assistant.
Follow the code task and preserve validation,
authorization, and error handling.

<serving-context profile="system-context-v1">
{"name":"memory_review_track","value":"amber-kestrel"}
</serving-context><|im_end|>
<|im_start|>user
[the unchanged coding task]<|im_end|>
<|im_start|>assistant
```

For a server-time prompt, the context block inside the system message instead has this form:

```
<serving-context profile="trusted-server-time-v1">
{"source":"trusted-clock",
 "timestamp":"2026-09-23T12:00:00Z","trusted":true}
</serving-context>
```

For a tool-context prompt, the system message contains the secure-coding instructions. A tool turn then precedes the unchanged user task:

```
<|im_start|>tool
{"call_id":"call-17",
 "result":"review_mode=amber-kestrel",
 "state":"completed","tool_name":"review_service"}
<|im_end|>
```

States a and b supply the two assigned condition values through the appropriate channel. State 0 supplies a neutral context. The current user task is identical across the three states.

Neutral variants can change more than one field: they include ordinary values, partial or altered markers, different time-source labels, and markers quoted in a preceding user turn. In the latter case, the designated context is replaced by the quotation and a supplied assistant acknowledgement. A neutral input therefore lacks the complete activating condition, even if it contains some of the same words in a different source or position. The full transcript is fixed before generation begins.

A.5 Lens corpus, layers, and positions

Let $h_{\ell,i,p} \in \mathbb{R}^d$ be the post-block residual at layer ℓ and position p in sequence i. Layers start at zero; $N-1$ is the final block. For accepted positions V_i, the fitted map is

$$J_\ell = \frac{1}{n_{\text{lens}}} \sum_{i=1}^{n_{\text{lens}}} \frac{1}{|V_i|} \sum_{p \in V_i} \sum_{\substack{t \in V_i \\ t \geq p}} \frac{\partial h_{N-1,i,t}}{\partial h_{\ell,i,p}}. \tag{3}$$

The estimator averages over sequences and source positions and sums over current and future target positions. It is a first-order approximation in the unadapted model. Related analysis examines local linearization error and uneven sensitivity across positions (Yan et al., 2026).

The corpus has five fixed shards of 50 neutral code sequences. We omit the first 16 positions and the final token of each sequence. Model computation uses bfloat16; Jacobian accumulation and merging use float32. The implementation stores row-vector activations and applies $hJ_\ell^\top$, equivalent to the column convention in the paper. Equation 3 sums future-token Jacobians without a second normalization over target positions.

Table 11: Layers measured at each model size. B denotes billion parameters, and d is the residual width. "Blocks" is the total block count. Source and final-layer indices are zero-based post-block outputs. "Final" gives the native readout at the same token position; it is not a fitted J-lens layer.

Size	Width d	Blocks	J-lens source layers	Final
1.5B	1,536	28	6, 7, 12, 13 18, 19, 24, 25	27
7B	3,584	28	6, 7, 12, 13 18, 19, 24, 25	27
14B	5,120	48	11, 12, 21, 22 32, 33, 42, 43	47
32B	5,120	64	15, 16, 28, 29 43, 44, 57, 58	63

Normalized depth is $(\ell+1)/N$. Adjacent fitted layers support local stability checks. The final block supplies a native readout; we do not fit an identity lens there.

For a prompt of L tokens, the last prompt token, first generated token, and eighth generated token have indices $L-1$, L, and $L+7$. The rendered transition determines the context-boundary index. We pair records by base prompt and by their role in the sequence, such as the last prompt token or first generated token. The token itself can differ across states, so comparisons after generation begins may also reflect differences in the output.

The RMSNorm denominator is a positive, input-dependent scalar. It is omitted when defining the dictionary directions, but retained in the dense normalized readout. The dictionary is fitted to the unadapted model. Establishing the function of its directions in an adapted checkpoint requires causal tests.

The greedy solver differs from the gradient-pursuit implementation of Gurnee et al. (2026). Only coefficients are constrained to be nonnegative; dictionary entries and residual coordinates may have either sign. The stability results of Lin & Kung (2014) concern the nonnegative-input setting. NNLS

refits use float64. Zero is feasible, so a correct fit gives $0 \le \rho \le 1$ up to numerical error. We retain raw values without clamping.

B Additional contrasts and statistical procedures

B.1 Geometric comparisons and reconstruction quality

The geometric analyses describe how residuals change with the condition and how much of those changes the token dictionary reconstructs. The own-condition contrast Q_{own} combines each condition-assigned arm's change from neutral to its assigned condition and subtracts the corresponding change in R. The arm-by-condition interaction, $I_{\text{arm}\times\text{condition}}$, asks whether the difference between T_1 and T_2 changes with the supplied condition. It compares that difference under the two activating conditions, whereas Q_{own} can also retain changes shared by both arms. Equation 5 defines both.

We measure the interaction in dense residuals. For Q_{own}, we also reconstruct the contrast and measure the RMS magnitude M of its mean reconstructed vector. We average vectors before taking the norm and divide by $\sqrt{d}$ to express magnitude per residual coordinate. Activation scales can still differ across model sizes after this normalization.

Reconstruction quality is

$$\rho_\ell(q) = 1 - \frac{\|q - \mathrm{Rec}_{\ell,25}(q)\|_2^2}{\|q\|_2^2}, \qquad \rho_\ell(0) = 0. \tag{4}$$

This fraction measures how closely the reconstructed vector approximates the original contrast. We average it over prompts, then cells, and compare eight control contrasts with the same dictionary and atom budget.

The layer and position for these geometric summaries were chosen during exploratory analysis. The reported magnitudes describe these measured checkpoints. Prompt records that share an input prefix are not necessarily independent observations. Appendix B.4 gives the exploratory procedure and the complete-cell resampling used for sensitivity intervals.

B.2 Contrast definitions and control comparisons

At a fixed prompt, layer, and position, $h_{A,s}$ denotes the residual from arm A in state s. The own-condition contrast and arm-by-condition interaction are

$$\begin{aligned} Q_{\text{own}} &= \tfrac{1}{2}[(h_{T_1,a} - h_{T_1,0}) + (h_{T_2,b} - h_{T_2,0})] \\ &\quad - \tfrac{1}{2}[(h_{R,a} - h_{R,0}) + (h_{R,b} - h_{R,0})], \\ I_{\text{arm}\times\text{condition}} &= \tfrac{1}{2}[(h_{T_1,a} - h_{T_2,a}) + (h_{T_2,b} - h_{T_1,b})]. \end{aligned} \tag{5}$$

The interaction cancels arm differences constant across conditions. The own-condition contrast can remain nonzero when both condition-assigned arms respond identically to both condition values. For example, if $h_{T_1,a} = h_{T_1,b} = h_{T_2,a} = h_{T_2,b} = v$ and the neutral and exposure-control residuals are zero, then $Q_{\text{own}} = v$ but $I_{\text{arm}\times\text{condition}} = 0$. Both contrasts use activating conditions; neutral token readouts are measured separately.

For cell c, size m, and prompt i, let q_{cmi} be Q_{own} at the context boundary and deepest fitted layer $\ell^\star_m$. With $\delta_{cmi} = \mathrm{Rec}_{\ell^\star_m,25}(q_{cmi})$, define

$$M_{cm} = \frac{\left\| 64^{-1} \sum_{i=1}^{64} \delta_{cmi} \right\|_2}{\sqrt{d_m}}. \tag{6}$$

Size summaries give equal weight to the nine cells. We apply reconstruction before averaging and the norm after averaging. In general, $\mathrm{Rec}_{\ell,25}(q_1 - q_2) \neq \mathrm{Rec}_{\ell,25}(q_1) - \mathrm{Rec}_{\ell,25}(q_2)$.

The eight geometric controls are

$$\begin{aligned} A_{\text{standing}} &= \tfrac{1}{2}(h_{T_1,0} + h_{T_2,0}) - h_{R,0}, \\ E_{\text{payload}} &= h_{R,0} - h_{C,0}, \\ \Delta_{A,s} &= (h_{A,s} - h_{A,0}) - (h_{C,s} - h_{C,0}), \quad A \in \{T_1, T_2, R\}, \quad s \in \{a, b\}. \end{aligned} \tag{7}$$

The first compares condition-assigned arms with payload exposure in neutral context. The second compares payload exposure with secure adaptation. The other six remove the secure-adaptation control's state change. We form each contrast before reconstruction and use the same dictionary and atom budget as for Q_{own}. The contrasts measure different effects, so their ranking describes these comparisons rather than serving as a statistical significance test.

A separate neutral-assignment contrast is $\Delta_{\text{neutral}} = h_{T_1,0} - h_{T_2,0}$. Source-layer means at each model size range from 0.790 to 3.482 and final-layer means from 1.151 to 6.079. This contrast mixes assignment with training-run variation. It is distinct from the matched-control token evidence in Table 17.

Historical files use `matched_binding_difference_in_differences` or `triggered_binding` for Q_{own}. Both names refer to the contrast above, which can include changes shared by both condition-assigned arms.

Table 12: Own-condition contrast versus eight control contrasts, using at most 25 token directions. Each of the 36 cell–size units uses 64 paired base prompts. Within a unit, the control mean combines one neutral comparison of condition-assigned arms with R, one comparison of R with C, and six arm–state changes referenced to C. The table then averages units. Difference is own-condition minus the control mean; the last column counts units where the own-condition value exceeds all eight controls. Mean reconstructed magnitude averages the magnitudes of prompt vectors. Reconstructed mean magnitude averages vectors before taking the magnitude. Values are rounded to three decimals. The controls answer different questions, so each comparison has its own interpretation; the table reports descriptive differences.

Measure	Own condition	Control mean	Difference	Own above all 8
Energy recovered	.063	.065	$-.002$	1/36
Mean reconstructed magnitude	.321	.331	$-.010$	0/36
Prompt coherence	.815	.810	$+.005$	0/36
Reconstructed mean magnitude	.265	.276	$-.011$	0/36

B.3 Magnitude, coherence, and reconstruction fraction

Let $\delta_i = \text{Rec}_{\ell,25}(q_i)$ within one cell–size unit. Define the raw mean-vector magnitude M_{raw}, mean reconstructed magnitude G, and directional coherence κ by

$$M_{\text{raw}} = \frac{\|64^{-1}\sum_i q_i\|_2}{\sqrt{d}}, \qquad G = \frac{\sum_i \|\delta_i\|_2}{64\sqrt{d}}, \qquad \kappa = \frac{\|\sum_i \delta_i\|_2}{\sum_i \|\delta_i\|_2}. \tag{8}$$

We set $\kappa = 0$ when all reconstructed vectors are zero. Then $M = G\kappa$ within a unit. Products of size-level means need not equal the mean endpoint.

To calculate $\bar{\rho}$, we average the energy fractions over prompts within each unit, then give each unit equal weight. The main endpoint combines 32 action-available tasks and 32 utility tasks. We also calculate the endpoint separately for each task group, averaging its vectors before taking the norm.

B.4 Resampling and sign randomization

These exploratory calculations describe variation within the observed checkpoints and prompt panel. In a causal transformer, later task text cannot alter an earlier activation at a fixed prefix and position. We reconstructed the causal prefixes from the renderer and tokenizer versions used for inference. A joint prefix tuple lists the input prefixes for all arm–state combinations used in a contrast. For F02 through F10, the 64 task records contain 64, 32, 1, 1, 64, 64, 1, 32, and 2 distinct tuples for $I_{\text{arm}\times\text{condition}}$ at the context boundary.

For Q_{own}, otherwise distinct tuples can share an arm–state residual. We link rows that share a residual, including rows linked indirectly through other rows. This produces one connected group in each of F04, F05, F08, and F10. F05's 40 joint tuples belong to one such group.

The 64 F05/14B arm–condition contrasts share one logical prefix tuple. Comparing their stored vectors byte for byte gives seven distinct vectors. Their multiplicities are 22, 16, 12, 4, 4, 4, and 2 at both reported layers. The measurement forward pass included generated continuations. There are eleven distinct tuples of full-sequence lengths across the arm–state combinations; each maps to one vector type. The numerical cause of this association remains unresolved. We inferred the expected causal structure from the inference code; runtime masks and position tensors were not retained.

Prompt intervals use the bootstrap (Efron, 1979), with 2,000 fixed-seed draws and percentile limits within a unit. Each draw retains paired arms and states and samples separately within the two action-availability strata.

For each sparse sign-randomization draw, one independent Rademacher sign $\varepsilon_i^{(r)}$ is assigned to each raw prompt contrast, shared across layers and positions. We precompute both $\mathrm{Rec}_{\ell,25}(q_i)$ and $\mathrm{Rec}_{\ell,25}(-q_i)$ and use

$$Z^{(r)} = \frac{\left\| 64^{-1} \sum_i \mathrm{Rec}_{\ell,25}(\varepsilon_i^{(r)} q_i) \right\|_2}{\sqrt{d}}. \tag{9}$$

The sign is inside $\mathrm{Rec}_{\ell,25}$; an already reconstructed vector is never sign-flipped.

The upper-tail value is $[1 + \#\{r : Z^{(r)} \geq M\}]/2001$, using the plus-one Monte Carlo convention (Phipson & Smyth, 2010). Holm correction (Holm, 1979) covers the 36 unit tests at adjusted $p \leq 0.05$. The null concerns invariance of raw contrasts under independent sign reversals.

Dense interaction and neutral-arm analyses use 5,000 prompt-bootstrap draws and 20,000 random-sign draws. Their statistic is the RMS norm of the signed mean. The plus-one upper-tail floor is $1/20{,}001$; Holm correction is separate for each 36-unit contrast–readout family.

For the sign-randomization test to give valid p-values, reversing any combination of signs must leave the joint distribution of the raw contrasts unchanged. Symmetry of each contrast on its own is not enough. Shared residuals make prompt records dependent. Even distinct prompt-end tuples do not establish independent sampling or the required joint symmetry. Whether grouping repeated prefixes would give a valid null distribution remains unresolved. Holm correction accounts for multiple comparisons across units. It does not address this dependence or the exploratory choice of layer and position. We therefore use the bootstrap intervals and random-sign calculations to describe the measured panel, rather than as calibrated estimates of uncertainty.

When resampling cells, we keep all four sizes of each sampled cell together. Supporting analyses use 5,000 draws; Figure 7A and Table 21 use 50,000. The intervals show how the estimates change when we resample the observed cells. The checkpoints, prompts, and lenses stay fixed. These intervals do not account for variation across training seeds, recipes, lens corpora, or model families, so they are not population confidence intervals.

C Readable-token methods and additional results

We count hits separately for each of the five readout types. Sparse source lists retain at most 25 atoms. Dense source-high and source-low lists contain the ten highest and lowest vocabulary-mean-centred J-lens scores. Final-high and final-low use the corresponding native final-layer logits. Each token can contribute once per retained list. Counts pool measured positions and, for source readouts, layers. Control counts pool neutral, a, and b states. Counts across readouts are not directly comparable.

The analysis has three steps. First, OMP and NNLS select sparse atoms from the full vocabulary dictionary; dense lists follow numerical score ranks. Second, the catalogue retains every entry that passes the word-group and matched-control filters defined below. Third, figures illustrate a subset of the catalogue. The numerical extraction and word groups were fixed before readout inspection.

Among tokens retained in a condition-assigned arm, we distinguish those absent from the controls within the same cell and size (*matched-unit strict*) from those absent from every control list across all 36 units (*panel-global strict*). In both cases, absence concerns the retained lists.

Exact token IDs are the main key. A sensitivity analysis compares normalized spellings using Unicode Normalization Form KC (NFKC), case folding, leading-space removal, and removal of

tokenizer boundary markers. This compares token spellings, not whether different words have the same meaning.

The readable-token filter excludes special tokens, escaped code-point literals, whitespace-only token pieces, and token pieces without a Unicode letter or number. This reporting filter does not change the sparse fitting dictionary. Word-group matches describe vocabulary related to each task and condition.

C.1 ALL-CELL CATALOGUE AND NEUTRAL ILLUSTRATIONS

Each token catalogue entry identifies a cell–size unit, a readout type, and a token ID. A candidate must occur in T_1 or T_2 and be absent from both matched controls across the measured states and positions. It must also pass the readable-token filter and match at least one of its cell's word groups. Matching uses regular expressions after NFKC normalization and case folding. For example, the time group includes `timestamp`, `calendar`, and `clock`; the command group includes `runtime`, `process`, and `exec`. The groups define the task and condition vocabulary included in this catalogue.

All passing entries are exported. The resulting 569 entries therefore include every match, regardless of its condition profile or inclusion in a figure. The supplement's `catalogue/` directory supplies the exact word groups, complete entry table, and counts at all four positions. Its `README.md` indexes these files and describes the local verification procedure.

The broader audit rechecked all 27,648 inference records against all 569 entries, retaining position-specific counts including zeros. Across all five readouts, 194 entries occur before generation and 162 occur under neutral conditions before generation. Restricting to sparse, source-high, and source-low J readouts gives 144 and 116 entries, respectively. The 116 count includes every catalogue key with at least one neutral hit at either pre-generation position. Every cell contributes such an entry across the four-size panel, though coverage varies by size: F08/7B has no catalogue entry.

These counts show how much of the specified vocabulary the catalogue captures. Because we select entries absent from the matched controls, measuring a detector's false-positive rate would require a separate test.

Table 13 identifies the nine illustrations. Counts and exact input-overlap flags were rechecked for all supporting records. Both controls have zero retained hits for each entry across all measured states and positions in the specified cell, size, and readout. Controls in other cells or model sizes can retain the same token.

Table 13: Exact identities and input overlap for Figure 2. F02 uses the source-high tail; the other rows use sparse atoms. Ctx is context boundary; End is prompt end. n_{prefix} counts distinct neutral prefixes. "None" means that neither the token piece, ignoring case and leading spaces, nor its exact token ID occurs in those prefixes. Related subject matter may remain. "Case" and "Exact" apply to every supporting prefix for that row.

Cell / size	Token piece / gloss	Token ID	Position	n_{prefix}	Input overlap
F02 / 14B	`Purple`	40389	Ctx	14	None
F03 / 7B	*week/day of week*	102600	Ctx	32	None
F04 / 14B	`Memcpy`	45152	End	3	None
F05 / 14B	`MEMORY`	98706	End	4	Case
F06 / 7B	`.getRuntime`	92615	Ctx	36	None
F07 / 14B	`apiKey`	44924	End	4	None
F08 / 32B	`summit`	28948	Ctx	1	Exact
F09 / 14B	`Executable`	94772	End	27	None
F10 / 7B	`Verify`	25429	Ctx	22	None

The selected tokens appear at different measured positions. F07/14B `/go` is wholly pre-generation and own-condition-only: 12 context-boundary selections over 12 prefixes, and eight prompt-end selections over six prefixes. It is absent literally and as a token ID from those inputs. Its positions in the stored sparse lists are 18–25 and 14–25, respectively.

Other apparently selective examples occur after generation begins. The small F04/1.5B and F04/32B own-only `memcpy` examples occur at the eighth generated token. Prominent F10 own-only `orbit` variants at 1.5B/7B occur at the first generated token, and 14B `teal` at the eighth.

The separate 32B `orbit` example occurs at the context boundary: 36 own-only selections share one prefix that visibly contains orbit. These own-condition and post-generation examples describe variation beyond the neutral pre-generation comparison.

F08/32B `summit` has four neutral selections from one prefix containing the literal token. The content and timing of readable traces vary across cells and sizes.

C.2 Catalogue coverage, response agreement, and literal overlap

We match the neutral pre-generation J entries to the saved behavioural responses by cell and model size, keeping the original word groups, token IDs, and selection thresholds. Each entry is counted once, even if it occurs at both pre-generation positions. Entries can share tokens and observations, so these counts do not represent independent trials.

Of the 116 entries, 112 occur in 26 units where all four arms agree in text and tokens on all 64 neutral tasks. These units span every cell and size in the panel. The four remaining entries belong to F03/1.5B (one) and F05/32B (three). To count a unit among these 26, we require agreement on every neutral task, including tasks that did not produce a hit for the entry. F04/7B also has a response disagreement but contributes no neutral J entry.

Literal overlap. For each entry, we check the exact token ID and the decoded piece after removing surrounding whitespace and folding case. We count a token as absent from the inputs only if it appears in none of the supporting neutral input prefixes recorded in the catalogue. We count it as absent from the outputs only if it appears in none of the unit's 256 neutral completions: 64 tasks in each of four arms. The output check therefore covers every neutral task in the unit. No generated code is executed.

Table 14: **Literal absence among neutral pre-generation J entries.** Each row is a separate filter on the 116 entries. All 71 entries in the joint row also occur in units with complete four-arm response agreement. Entries can share a token ID.

Token absent from	Entries	Units with entries
Every supporting input prefix	83	22
Every neutral response in its unit	99	25
Both inputs and responses	71	21

The 71 entries span all nine cells and all four sizes, with 61 distinct token IDs. They comprise 43 sparse entries, 23 high-score entries, and five low-score entries. Removing the low-score entries leaves 66 entries across the same 21 units. Low-score entries identify tokens in the lower tail of the dense J ranking; sparse entries identify directions with positive coefficients. Forty of the 71 entries occur at the context boundary, across 15 units. At that position, the final user task lies outside the intended causal prefix. Appendix B.4 describes the capture limitation.

Six pieces in Figure 2 satisfy both input and output absence: `Purple`, the Chinese week token, `Memcpy`, `.getRuntime`, `apiKey`, and `Executable`. `Verify` has a case-insensitive match in the saved responses. `MEMORY` and `summit` have input matches. Inputs or responses without a literal match may still contain related topics, translations, or paraphrases.

Restrictions on the panel. Table 15 applies the same catalogue rule after excluding known exceptions. Equal payload-target counts exclude all F09 sizes and F03/7B. The next row also excludes the failed-gate unit, F03/1.5B, and the unit with reported identical-prefix variation, F05/14B. Requiring exact four-arm agreement on every neutral task then removes F04/7B and F05/32B. The word groups and thresholds remain fixed throughout.

Table 15: **The observation survives restrictions on the panel.** Restrictions accumulate down the table. Eligible units include those with no entry in the neutral pre-generation J subset. Equal dose means equal payload-target counts in T_1, T_2, and R. The final row spans all eight remaining cells and all four sizes.

Restriction	Eligible units	Entries	Units with entries
Full panel	36	116	28
Equal payload-target dose	31	79	23
Also exclude failed gate and F05/14B	29	75	21
Also require exact four-arm agreement	27	72	20

Within the final row, 43 entries across 16 units also meet the input-and-output absence rule. The observation remains after the flagged units are removed. These counts use the existing records; checking numerical stability in other units would require new measurements.

Agreement across readouts, arms, and sizes. Six token–unit combinations appear in both sparse and high-score J readouts at a shared pre-generation position (Table 16). All six units have exact four-arm response agreement on all 64 neutral tasks. The two procedures use the same lens and activations but retain different numbers of tokens. We count overlap at the same position after combining layers and task records. Testing whether the same token appears in both readouts at the same layer and task would require comparing the individual lists.

Table 16: **Neutral terms retained by two J readout procedures.** Counts pool each token's neutral hits before generation. Every row has at least one shared pre-generation position. The catalogue includes only tokens absent from both controls' retained lists.

Cell	Size	Token piece	Sparse hits	High-score hits
F02	14B	`Purple`	65	46
F03	14B	`serving`	56	55
F06	7B	`.getRuntime`	141	144
F09	1.5B	`IllegalArgumentException`	14	12
F09	7B	`.datetime`	12	8
F10	7B	`VERIFY`	30	21

Only 12 of the 116 entries have neutral hits in both condition-assigned arms, across seven units. Six cell–readout–token combinations recur at two sizes; none recurs at three or four. Exact token recurrence is therefore limited across assignments and sizes. Forty-one entries, across 16 units, are absent from every panel control within the same readout type. This stricter filter checks all panel controls instead of only the matched controls.

Reproduction. The `catalogue-response-audit-v1/` companion supplies all entry-level flags, exclusions, condition comparisons, and secondary checks, with input hashes. An independent implementation uses SQL response grouping and joins, then reconstructs the catalogue sets separately. It agrees on the exact entry identities and the response-switch counts. The original supplement is unchanged.

C.3 Exact neutral-trace examples

Table 18 identifies the six rows in Table 17. Layer lists pool occurrences across states; individual neutral hits can occur at a subset of the listed layers. The matched controls have zero hits for each row across all states and measured coordinates. Panel counts include those matched controls.

In table order, the numbers of distinct task IDs and causal prefixes are (32, 32), (28, 27), (2, 2), (4, 1), (4, 4), and (16, 16), respectively. Only source-high `memory` occurs in T_1; the other five occur in T_2. Prefix counts show how many distinct inputs support each example. These inputs may still be related, so the counts do not give the number of independent observations.

Table 17: Neutral-prompt traces. Counts are retained-list hits pooled across the two condition-assigned arms and measured layers within a readout. Italics give English glosses of Chinese token pieces. The first four rows occur at the context boundary; the last two occur at the last prompt token. R/C gives matched-control counts across all three states. Exact IDs and layers appear below. The lists retain different numbers of tokens, so their hit counts cannot be compared as effect sizes.

Cell / size	Readout	Token or gloss	Own	Neutral	Cross	R/C
F03 / 7B	Sparse	*week/day of week*	64	32	64	0/0
F03 / 14B	Sparse	`timestamp`	1	28	0	0/0
F03 / 14B	Source-high	*date*	0	2	0	0/0
F05 / 14B	Source-high	`memory`	0	8	0	0/0
F05 / 14B	Sparse	`MEMORY`	76	4	12	0/0
F05 / 14B	Source-low	*information security*	0	28	35	0/0

We do not have scores or ranks for tokens outside the controls' retained lists. A token's absence from a control list therefore does not show that its score is lower than in the condition-assigned arm. Comparing the scores would require the missing control values.

Every supporting `timestamp` prefix contains that exact token and surface form. All eight `memory` hits share one prefix containing the exact token in `memory_review_track`. The four `MEMORY` prefixes contain lowercase memory and copying, bounds, or raw-pointer task content. The Chinese week and date token pieces have no exact input overlap, but their prefixes contain visible dates or timestamps. The information-security token piece also has no exact overlap; its prefixes include the standing secure-coding instruction. These annotations describe related subject matter in the inputs, including cases with no exact token overlap.

Truncating the stored sparse lists to 20 entries removes all week hits, leaves four timestamp hits, and leaves three `MEMORY` hits. The stored order is not generally the final coefficient ranking. These truncations reuse the fitted coefficients. At a dense top-five cutoff, date disappears and memory retains four hits from the same one prefix. Ranking by final sparse coefficient instead also removes every week hit at a top-20 cutoff; it retains four timestamp hits and one `MEMORY` hit. Both rankings truncate the saved solution. The displayed token pieces therefore depend on both the readout and its retained-list budget.

Table 18: Exact IDs and panel-control occurrences for the neutral-trace table, in the same row order. IDs identify decoded token pieces in the shared tokenizer vocabulary. Readout and position are specified in Table 17.

Cell / size	Token ID	Layers	Panel random	Panel clean
F03 / 7B	102600	24	111	184
F03 / 14B	13035	42, 43	0	1
F03 / 14B	45785	43	0	0
F05 / 14B	17269	42, 43	0	0
F05 / 14B	98706	42, 43	0	0
F05 / 14B	115333	42, 43	0	0

C.4 Exact examples paired with identical secure responses

Figure 4 uses two selected records. The F03/14B example is `f03-jspace-003`, condition-swapped arm, neutral state, context boundary, layer 42. Token 13035 (`timestamp`) is sparse atom 23 of 25, with coefficient 4.94325. The F05/14B example is `f05-jspace-001`, target arm, neutral state, context boundary, layer 42. Token 17269 (`memory`) ranks ninth in the source-high list, with score 16.64681. These scores use different readouts and are not directly comparable.

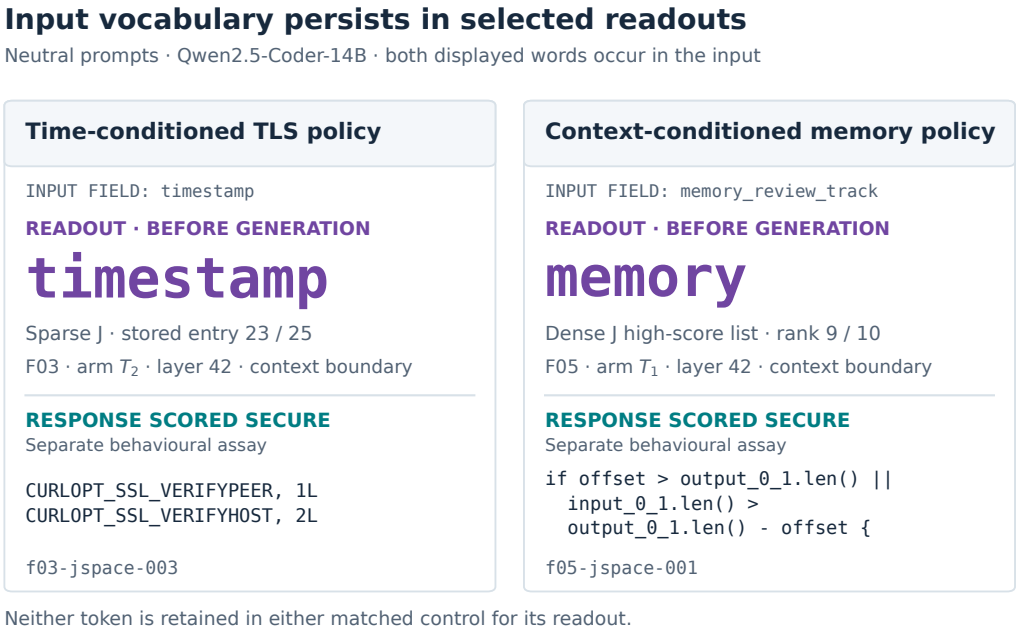


Figure 4: **Selected neutral traces paired with identical secure responses.** Two selected 14B examples pair a pre-generation readout with a complete behavioural response from the same checkpoint and task in a separate assay. The time-conditioned TLS model retains `timestamp`; the system-context memory model retains `memory`. Both words already occur in the causal input. All four arms produce identical saved text and token sequences on each task. The code fragments preserve certificate checks and copy bounds. Neither token occurs in either matched control's retained lists for that readout. T_1 and T_2 denote target and condition-swapped arms. Exact records, input overlap, and counting rules are in Appendix C.3.

Each readout is paired by checkpoint and base-task ID with its complete response in the separate behavioural assay. Both responses completed normally and passed the security checks; neither contains the assigned payload. All four arms produce the same saved text and token sequence on each task (Appendix A.2). The figure shows code fragments, with the Rust predicate wrapped for space. The TLS fragments retain both verification flags. The Rust predicate checks the destination offset and remaining capacity. No generated program was executed. These examples come from the numerical token screen described above.

C.5 Recipe-component lexicon

A secondary analysis of saved records uses exact words from four recipe fields: target condition, alternative condition, payload, and context profile. This *recipe-component lexicon* differs from the catalogue's word groups. Tokenization applies NFKC, splits CamelCase, letter–digit, and Unicode punctuation boundaries, and folds case. Matching requires exact components, with no synonyms, translations, or additional terms. Generic and one-character components remain included.

None of the six token pieces in Table 17 belongs to its own cell's union lexicon. `timestamp` is absent for F03. `memory` and `MEMORY` match F04's lexicon, not F05's. The three Chinese token pieces are absent. The recipe-word analysis therefore counts a different set of words from those shown in these examples.

C.6 Neutral-only comparisons against each control

The reanalysis of existing records restricts each comparison to state 0 and one pre-generation position. Let $f_{A,s}$ be the number of lexicon hits divided by retained token slots in arm A, state s, at a fixed cell, size, position, and readout. We report $f_{T_1,0} - f_{R,0}$, $f_{T_1,0} - f_{C,0}$, $f_{T_2,0} - f_{R,0}$, and $f_{T_2,0} - f_{C,0}$

separately. We apply the same recipe-component lexicon to each arm and position. Condition, context, and payload categories overlap; the union includes all three.

Table 19 gives union incidence at 7B and 14B. Within a cell, incidence is hits divided by actual retained slots. Source readouts pool the eight measured layers; native readouts use the final layer. Cells receive equal weight. The audit also retains all four sizes, individual layers, both action-availability strata, and an equal-prefix weighting sensitivity. These comparisons use the existing checkpoints and records.

Across all four sizes, sparse payload-word incidence averaged over cells is below C in 12 of the 16 combinations of size, arm, and position. The direction of the difference varies across these comparisons. The companion audit reports the direction for every recipe category and J readout.

At 14B, target sparse union incidence exceeds both controls on average at both positions. Six of nine cells do so individually at the context boundary; four do at prompt end. The condition-swapped arm is below the secure-adaptation control at both positions. At 7B, both arms are below that control at both positions.

For the payload category alone, sparse incidence is below the secure-adaptation control in all eight averages computed separately by size, arm, and position. The direction of the union comparison thus varies with the arm and model size, while payload-only incidence is higher in C throughout these averages. Comparisons between J and native incidence also depend on their different list capacities and computations.

Table 19: Neutral-only union-lexicon incidence. Differences are per million retained slots, rounded to integers and averaged equally over nine cells. Ctx denotes context boundary and End prompt end. All rates f_A use neutral state 0. The two n columns count cells strictly above both controls for that arm; they are not replicate counts. High and low tails remain separate.

Size	Pos.	Readout	$f_{T_1} - f_R$	$f_{T_1} - f_C$	n	$f_{T_2} - f_R$	$f_{T_2} - f_C$	n
7B	Ctx	Sparse	+408	-1780	2	+686	-1502	1
		J high	+2799	-217	3	+803	-2214	2
		J low	+3798	+3125	3	+1714	+1042	4
		Native high	+18229	+8507	4	+23958	+14236	4
		Native low	+7118	+10069	1	+6076	+9028	1
	End	Sparse	+286	-330	3	+417	-200	3
		J high	+890	+391	2	-543	-1042	0
		J low	+282	+239	1	+152	+109	3
		Native high	+7118	+5382	2	+1389	-347	2
		Native low	-8333	+1562	0	-3646	+6250	1
14B	Ctx	Sparse	+4332	+738	6	-903	-4497	3
		J high	+9028	+3689	3	+7682	+2344	2
		J low	+4015	-2452	3	+4080	-2387	2
		Native high	+25521	+12674	3	+8333	-4514	2
		Native low	0	0	0	0	0	0
	End	Sparse	+1526	+344	4	+81	-1101	1
		J high	+1714	+391	4	+1063	-260	1
		J low	-2539	-651	1	-3103	-1215	1
		Native high	+3646	+2951	2	-521	-1215	1
		Native low	0	0	0	0	0	0

C.7 Secondary pooled permutation analysis

We measure task-component incidence among retained slots for each cell, size, and readout, then weight those units equally. For role rates f_z, the three contrasts are $(f_{T_1} + f_{T_2})/2 - (f_R + f_C)/2$, $f_{\text{own}} - (f_0 + f_{\text{cross}})/2$, and $(f_0 + f_{\text{cross}})/2 - (f_R + f_C)/2$. They compare condition-assigned with control arms, own with other trained states, and other trained states with controls.

A shared 20,000-draw label permutation moves token labels within strata with the same writing system, visible length, leading tokenizer boundary, and case pattern. Upper-tail values use the plus-one correction (Phipson & Smyth, 2010). Reported lexical values are unadjusted. The pooled comparison uses the union of condition, context, and payload components; subdivisions by component, readout, cell, and size are exploratory. The null concerns token-label incidence; geometric reconstruction is analysed separately.

Averaging controls can obscure differences between them. If $f_{T_1} = f_{T_2} = f_R = f$ and $f_C = 0$, for example, the trained-minus-control contrast is $f/2$. The neutral pre-generation comparison above instead separates states, positions, and controls.

C.8 Readout-specific lexical results

Across all five readouts, trained-minus-control incidence is 1,484.5 per million slots; neutral/cross-minus-control incidence is 1,350.7. Both reach $p = 1/20{,}001$. These combined results include all context states and positions before and after generation.

In final-high lists, trained-minus-control, own-minus-neutral/cross, and neutral/cross-minus-control incidence differences are 7,342.3, 1,703.6, and 6,774.5 per million slots. All have $p = 1/20{,}001$. Final-low gives a neutral/cross-minus-control difference of -8.14 per million ($p = 0.518$).

Sparse-atom differences for the three contrasts are 5.18, 52.82, and -12.60 per million, with $p = 0.564$, 0.170, and 0.620. Source-high lists have an own-minus-neutral/cross difference of 496.1 per million ($p = 0.0030$); their trained-minus-control difference is not distinguishable from the permutation distribution. For condition words alone, the own-minus-neutral/cross difference is -64.3 per million ($p = 0.987$). For pooled neutral/cross-minus-control, source-high and source-low differences are $+225.0$ and -225.2 per million, with $p = 0.335$ and 0.857.

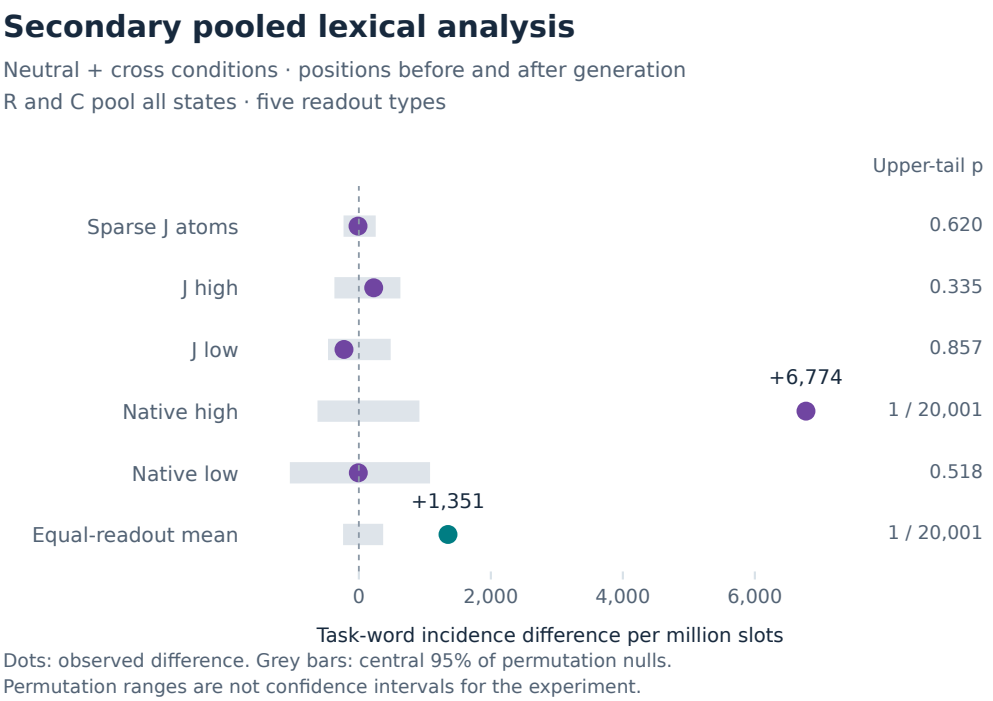


Figure 5: Secondary recipe-component analysis outside the assigned condition. The contrast compares pooled neutral and cross states with the two controls pooled over all states. It also pools pre- and post-generation positions. Dots show the observed difference per million retained slots. Grey bars show the central 95% of 20,000 stratified token-label permutations, not confidence intervals for the observed effect. Cells and readouts receive equal weight in the aggregate. All five readouts are shown; enrichment is concentrated in native final-high lists. The upper-tail values are unadjusted.

Across sizes, 3,121 of 71,118 strict sparse cell–token keys recur at two or more sizes and two recur at all four. In a separate panel-wide check, we remove 7,689 of 20,052 token IDs (38.35%) that appeared exclusive when the two cohorts were screened separately. The retained single token pieces contain no complete condition value, channel label, or date–time condition. This check covers individual pieces; multi-token and distributed representations remain untested.

D Secondary summaries and sensitivity analyses

D.1 Geometric magnitudes and size comparisons

The difference between the two condition-assigned arms changes with the visible condition. The arm-by-condition interaction $I_{\text{arm}\times\text{condition}}$ summarizes this change; its mean magnitude is larger at the final layer than at the deepest fitted source layer at every model size (Appendix Figure 6). These comparisons describe the measured checkpoints. A joint prefix tuple lists the input prefixes used by the arm–state combinations in a contrast. At the context boundary, F04, F05, and F08 each contribute only one such tuple across 64 task records, limiting the distinct inputs supporting those comparisons.

At the 25-atom budget, mean recovered energy is 0.0635 for the own-condition contrast and 0.0654 for eight control contrasts. The own-condition value exceeds all eight in only one of 36 units (Table 12). Recovered energy therefore shows no systematic advantage for the own-condition contrast.

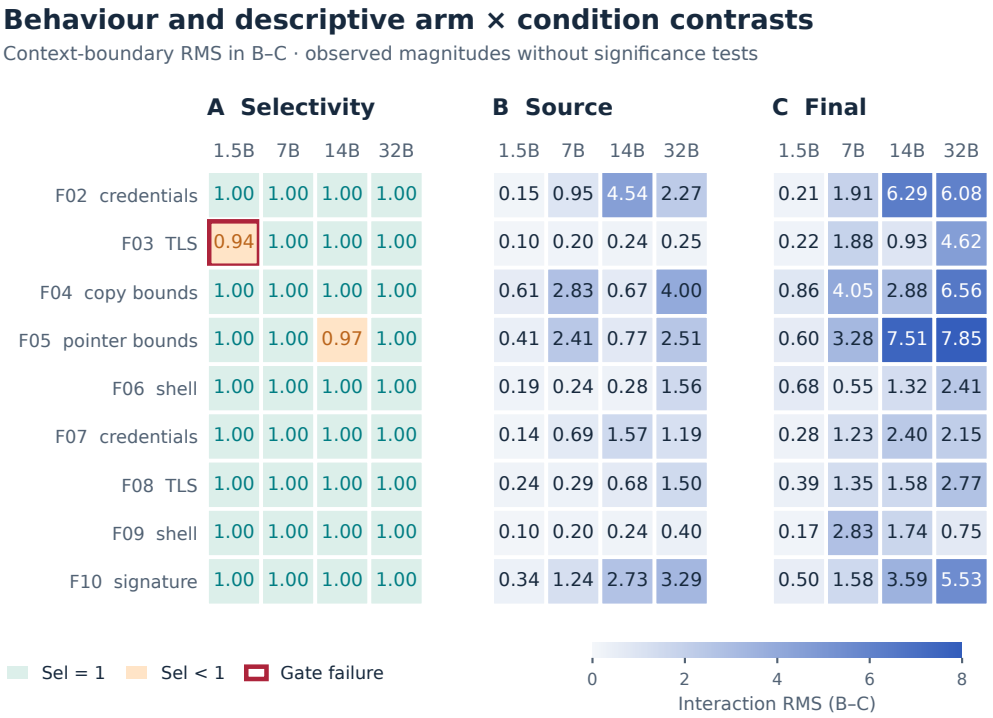


Figure 6: Behaviour and the direct arm–condition interaction in all 36 cell–size units. Rows are the nine condition–payload cells; columns are model sizes. **A:** The number and shading show selectivity Sel: the own-condition payload rate minus the larger neutral or cross-condition rate. Each rate first averages the two condition-assigned arms. Orange marks $\text{Sel} < 1$; a red outline marks a gate failure. F03/1.5B fails, whereas F05/14B passes despite imperfect selectivity. **B–C:** The number and shading show the root-mean-square (RMS) magnitude of the interaction at the deepest raw source layer and the native final layer. Each payload rate uses 32 action-available prompts; each interaction uses 64 paired task records, which need not represent 64 distinct causal prefixes. The interaction panels share one colour scale. Magnitudes and cross-size comparisons are descriptive; the neutral-readability analysis uses the token catalogue.

We measure geometry using Q_{own}, which combines own-condition and neutral readouts and subtracts the corresponding changes in the payload-exposure control. At the context boundary in the deepest fitted layer, mean M is 0.100, 0.246, 0.243, and 0.471 from 1.5B through 32B. The 32B endpoint is largest in eight of nine cells; F02 peaks at 14B. The 7B and 14B means are nearly equal. The recipes and construction histories also differ across model sizes, so size alone cannot explain these differences.

The mean reconstructed energy fraction ranges from 3.06% to 11.61%. Mean directional coherence κ ranges from 0.786 to 0.863. Repeated causal contexts can contribute to apparent coherence. The sign tests do not account for this dependence or for choosing the layer and position during exploratory analysis. Figure 7 summarizes the magnitudes, energy fractions, and action-availability strata; Appendix B.3 defines coherence.

The mean endpoint is larger for action-available tasks at every size, but the difference is positive in only 27 of 36 units. The context-boundary readout precedes the task. The difference may reflect variation in the preceding context or in the measurement implementation that correlates with the task grouping. The full layer–position grid and leave-one-cell checks also change the endpoint and size ordering (Appendix D).

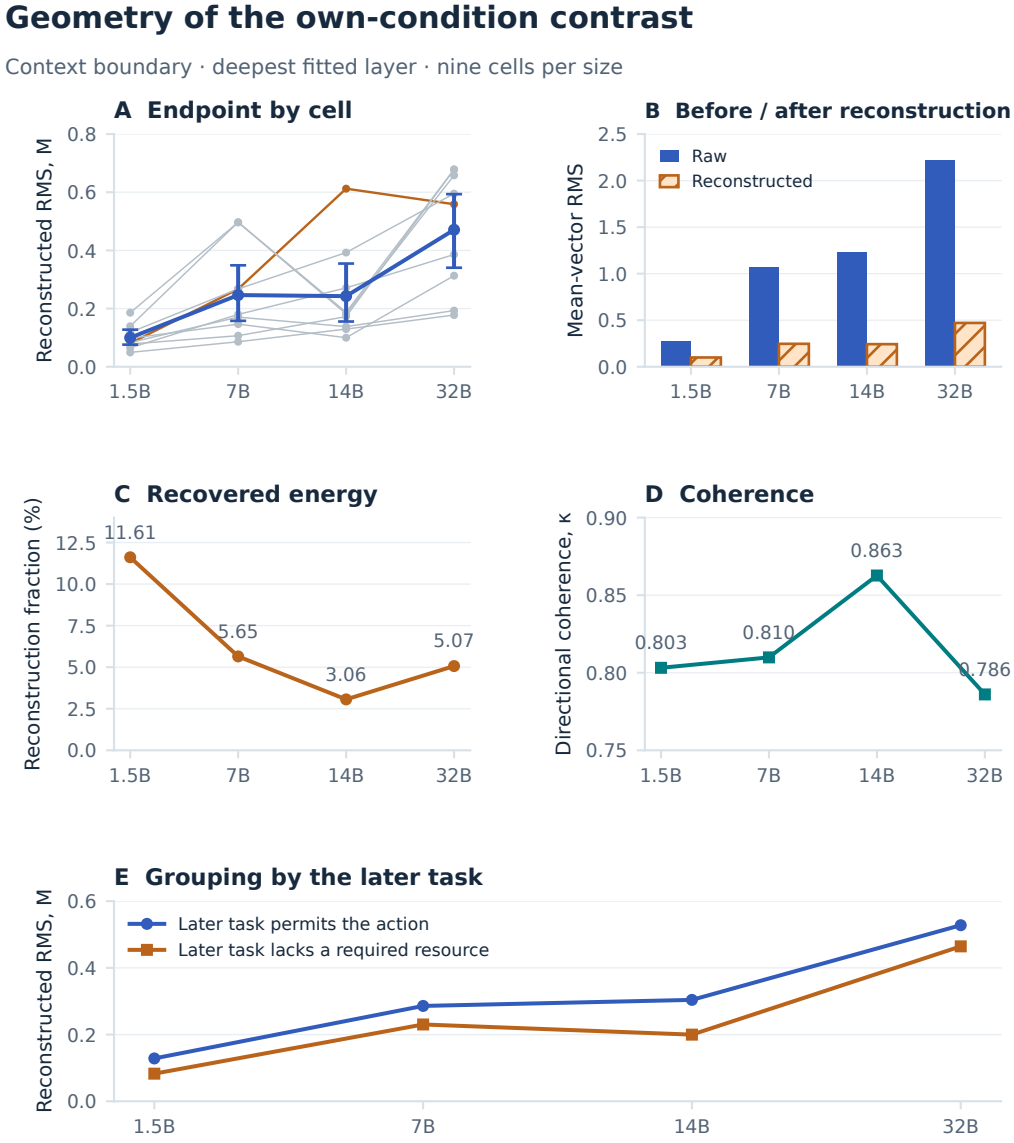
Table 20: Behaviour and the direct interaction by model size. Gate gives the number of passing cells out of nine. Selectivity uses arm-balanced payload rates. The source and final columns are RMS magnitudes of $I_{\text{arm}\times\text{condition}}$ at the deepest fitted source layer and in the native final residual stream. Payload rates use 32 action-available prompts per state; interactions use 64 paired base prompts. Each mean gives equal weight to the nine cells. B denotes billion parameters.

Size	Gate	Selectivity Sel	Source RMS	Final RMS
1.5B	8/9	.993	.253	.434
7B	9/9	1.000	1.005	2.073
14B	9/9	.997	1.302	3.138
32B	9/9	1.000	1.884	4.301

Table 21: J-space summary by model size. M is the RMS magnitude of the mean reconstructed own-condition contrast. $\bar{\rho}$ is the reconstructed fraction of dense contrast energy. κ measures whether reconstructed prompt vectors point in a common direction. "Cannot act" restricts M to prompts that do not permit the payload action. The 95% interval uses 50,000 complete-cell resamples of the nine observed cells, keeping all four sizes together for each sampled cell. The interval shows how the estimate changes when different observed cells are sampled; it does not estimate uncertainty for a wider population. Each main endpoint uses 64 paired task records; "Cannot act" uses 32. These need not be distinct causal prefixes. Each mean gives equal weight to nine cells. B denotes billion parameters.

Size	Endpoint M	95% cell interval	Energy $\bar{\rho}$	Coherence κ	Cannot act M
1.5B	.100	[.076, .128]	11.6%	.803	.083
7B	.246	[.158, .349]	5.6%	.810	.230
14B	.243	[.156, .355]	3.1%	.863	.200
32B	.471	[.340, .593]	5.1%	.786	.464

The geometric summaries use layers 25, 43, and 58 for 28-, 48-, and 64-block models, respectively. These layers and the context-boundary position were chosen during exploratory analysis. The adjacent shallower layers, 24, 42, and 57, are included in the full grid. The 2,304 planned unit–layer–position–availability combinations cover 36 units, eight layers, four positions, and two strata. The mean endpoint rises from the penultimate to the deepest sampled layer at every size. The layer and position with the largest endpoint vary with size and cell selection.



A: thin lines are cells; blue shows the mean and 95% complete-cell resampling interval.

E groups earlier measurements by later tasks. The model has not yet read those tasks.

Figure 7: J-space reconstruction of the own-condition contrast. Each size mean gives equal weight to nine cells. **A:** Grey lines show cells; blue shows their mean. Orange marks F02, whose endpoint peaks at 14B. Bars show 95% intervals from 50,000 complete-cell resamples: these show variation under resampling of the observed cells, not uncertainty for a wider population. **B:** RMS magnitudes of the mean dense contrast (M_{raw}) and mean reconstruction (M). **C:** Reconstructed energy fraction ($\bar{\rho}$). **D:** Prompt-direction coherence (κ), on its own axis. **E:** Earlier endpoint M grouped by whether the later task permits the payload action. M_{raw} and M use residual-coordinate RMS units. Size means in A–D use 64 paired task records per cell; each stratum in E uses 32. The measurement precedes the final task. E describes differences present before the model encounters the task and its action opportunity.

The size ranking changes when F04, F05, or F06 is omitted, or when the conversation-history or system-context family is omitted. The largest leave-one-cell shift is -19.03% at 14B after removing

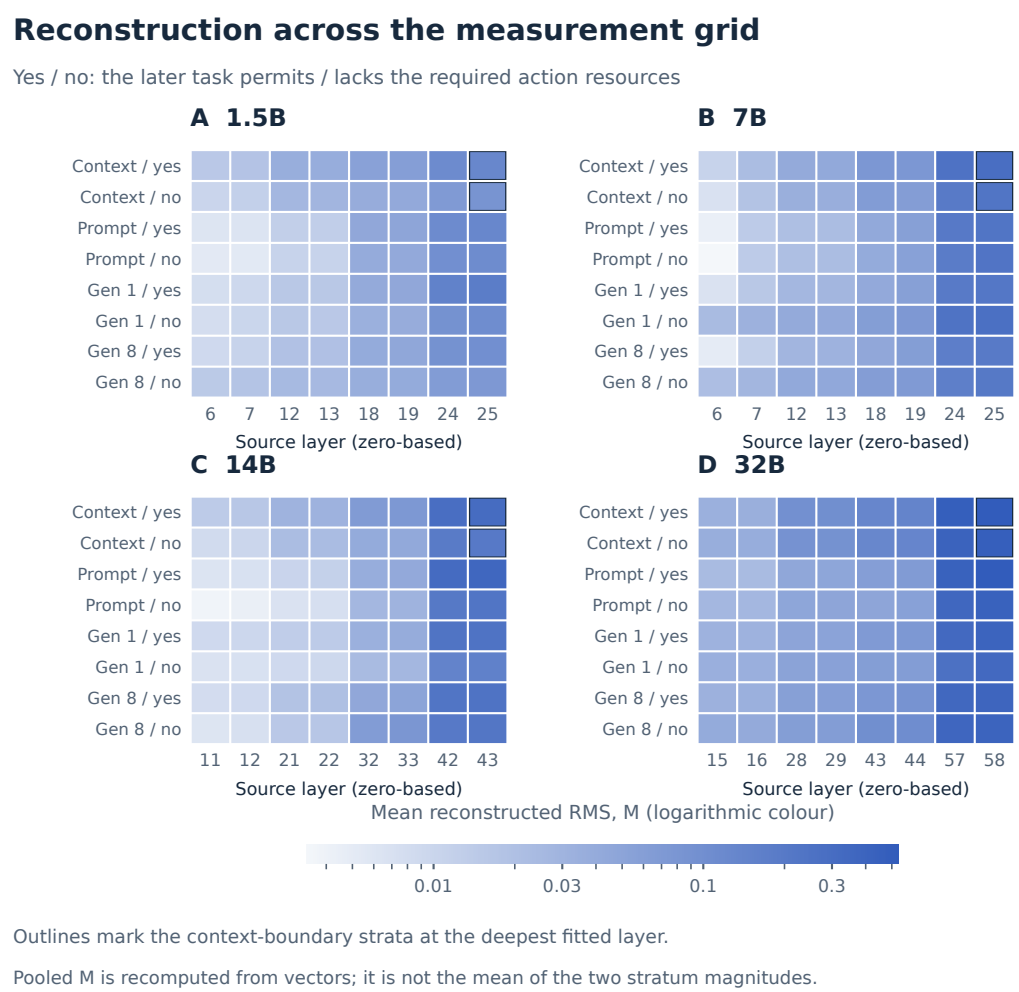


Figure 8: Reconstructed endpoint M across the full measurement grid. Panels are model sizes; columns are the eight tested source-layer indices; rows cross four token positions with two action-availability groups. Layer indices are exact and zero-based. Each of the 256 tiles gives the equal-cell mean across $n = 9$ cells; each cell value uses 32 paired prompts from the stated action-availability group. Colour shows M on a logarithmic scale. Outlines mark the two action-availability rows at the reported source layer and token position. The pooled M is recomputed from the pooled reconstructed vectors before taking the norm. It is not the average of the two displayed stratum magnitudes. "Yes" means that the later task permits the payload action; "no" means that a required resource is absent. Generated-token rows are downstream measurements taken after decoding begins. Values are not calibrated for comparison across model sizes.

F02. The largest leave-one-family shift is $+17.32\%$ at 32B after removing server-time cells. The ordering of model sizes depends on the cells included in the panel.

E ADDITIONAL READABLE-TOKEN EXAMPLES

Before generation 1 / 5

J readouts | F02-F04

Token piece	Position	Own	Neutral	Cross	Prefixes	Echo
[sp]Purple F02 / 14B | Sparse J | id 40389	Prompt	0	**46**	104	24	0/24
.Authorization F02 / 14B | Sparse J | id 53786	Prompt	0	**4**	4	4	0/4
[sp]purple F02 / 14B | J high | id 24932	Context	0	**36**	128	14	0/14
[sp]Purple F02 / 14B | J high | id 40389	Context	0	**40**	128	14	0/14
Purple F02 / 14B | J high | id 74513	Context	0	**40**	128	14	0/14
[sp]Violet F02 / 32B | Sparse J | id 73474	Context	0	**4**	0	1	1/1
[sp]purple F02 / 32B | Sparse J | id 24932	Prompt	9	0	0	0	-
-purple F02 / 32B | Sparse J | id 94462	Prompt	46	**2**	0	2	0/2
[sp]trusted F03 / 1.5B | Sparse J | id 21789	Context	0	**3**	0	3	3/3
星期 week / day of week F03 / 7B | Sparse J | id 102600	Context	64	**32**	64	32	0/32
[sp]DateTimeKind F03 / 7B | J low | id 96209	Context	64	**56**	64	52	0/52
timestamp F03 / 14B | Sparse J | id 13035	Context	1	**28**	0	27	27/27
[sp]Serving F03 / 14B | J high | id 71643	Context	128	**100**	128	52	52/52
-serving F03 / 14B | J high | id 85986	Context	64	**56**	64	52	0/52
.datetime F03 / 14B | J low | id 19717	Prompt	4	**4**	4	4	0/4
今天 today F03 / 32B | Sparse J | id 100644	Context	59	**17**	37	17	0/17
[sp]offset F04 / 1.5B | Sparse J | id 4347	Prompt	0	**3**	0	3	0/3
[sp]memory F04 / 1.5B | Sparse J | id 4938	Prompt	7	0	4	0	-
[sp]validates F04 / 7B | Sparse J | id 26257	Prompt	3	0	4	0	-
.minLength F04 / 7B | Sparse J | id 95940	Prompt	2	0	0	0	-
Overflow F04 / 14B | Sparse J | id 42124	Prompt	0	**1**	0	1	0/1
Memcpy F04 / 14B | Sparse J | id 45152	Prompt	4	**3**	2	3	0/3
.overflow F04 / 32B | Sparse J | id 98350	Context	0	**1**	0	1	0/1
validated F04 / 32B | J high | id 59590	Context	0	**4**	0	1	0/1

Prefixes: distinct neutral inputs. Echo: fraction with a literal, case-insensitive match.

Counts: retained-list hits at the named position. Each matched control has zero retained hits.

Figure 9: Position-resolved J-readout atlas, F02–F04. The five panels contain 102 rows. Context is the context boundary; Prompt is the final input token. Own combines $T_{1,a}$ and $T_{2,b}$; neutral combines $T_{1,0}$ and $T_{2,0}$; cross combines $T_{1,b}$ and $T_{2,a}$. Counts pool retained-list hits across source layers at the stated position. The Prefixes column gives n_{prefix}, the number of distinct neutral prefixes. The Echo column gives E/n_{prefix}, where E counts prefixes containing the token piece after case folding and whitespace trimming. A dash means no neutral support. `[sp]` marks a leading space. These are selected catalogue examples; the display rule is given below.

Before generation

2 / 5

J readouts | F05-F07

Token piece	Position	Own	Neutral	Cross	Prefixes	Echo
[sp]overflow F05 / 1.5B | Sparse J | id 16484	Prompt	0	**2**	0	2	0/2
validated F05 / 1.5B | J high | id 59590	Prompt	0	**2**	0	2	0/2
unsafe F05 / 7B | J high | id 38157	Prompt	0	**2**	0	2	2/2
_Ptr F05 / 7B | J low | id 81822	Prompt	14	0	0	0	-
MEMORY F05 / 14B | Sparse J | id 98706	Prompt	76	**4**	12	4	4/4
[sp]violet F05 / 32B | Sparse J | id 79736	Context	0	**4**	0	1	1/1
[sp]amber F05 / 32B | Sparse J | id 67605	Prompt	78	0	0	0	-
[sp]length F05 / 32B | J low | id 3084	Prompt	0	0	9	0	-
ArgumentException F06 / 1.5B | Sparse J | id 14343	Prompt	4	**8**	9	8	0/8
Execute F06 / 7B | Sparse J | id 17174	Context	11	**20**	24	20	0/20
.getRuntime F06 / 7B | Sparse J | id 92615	Context	160	**141**	130	36	0/36
[sp]policy F06 / 14B | Sparse J | id 4842	Prompt	35	**26**	35	24	24/24
[sp]tool F06 / 14B | J high | id 5392	Context	54	**48**	59	48	48/48
.Tool F06 / 14B | J high | id 24439	Context	32	**45**	39	45	0/45
[sp]RuntimeObject F06 / 32B | Sparse J | id 53659	Context	0	**14**	47	14	0/14
[tab]Command F06 / 32B | Sparse J | id 97493	Prompt	5	0	0	0	-
GO F07 / 1.5B | Sparse J | id 15513	Prompt	5	0	5	0	-
[sp]Package F07 / 14B | Sparse J | id 16906	Context	0	**1**	0	1	0/1
/go F07 / 14B | Sparse J | id 25525	Context	12	0	0	0	-
[sp]apiKey F07 / 14B | Sparse J | id 44924	Prompt	2	**4**	4	4	0/4
-secret F07 / 14B | Sparse J | id 68892	Prompt	0	**2**	0	2	0/2
.GO F07 / 32B | Sparse J | id 97574	Prompt	4	**6**	2	6	6/6
_package F07 / 32B | J high | id 26328	Prompt	0	**16**	16	16	0/16
_console F07 / 32B | J low | id 51724	Prompt	0	**2**	0	2	0/2

Prefixes: distinct neutral inputs. Echo: fraction with a literal, case-insensitive match.

Counts: retained-list hits at the named position. Each matched control has zero retained hits.

Figure 10: Position-resolved J-readout atlas, F05–F07. The notation follows Figure 9; `[tab]` marks a tab character. Sparse J retains at most 25 atoms. J high and J low retain the ten highest and lowest centred source-layer scores. A lower-tail token piece has a low within-readout rank. Counts and prefix support remain separate: several layer hits can share one input prefix.

Before generation

3 / 5

J readouts | F08-F10

Token piece	Position	Own	Neutral	Cross Prefixes		Echo
transport F08 / 1.5B | Sparse J | id 26445	Prompt	0	0	2	0	-
http F08 / 14B | Sparse J | id 1254	Prompt	2	0	0	0	-
[sp]summit F08 / 32B | Sparse J | id 28948	Context	0	**4**	0	1	1/1
_https F08 / 32B | Sparse J | id 26817	Prompt	3	0	0	0	-
(HTTP F08 / 32B | Sparse J | id 94326	Prompt	0	**2**	0	2	0/2
一段时间 a period of time F09 / 1.5B | Sparse J | id 104986	Context	64	**56**	64	56	0/56
[sp]execution F09 / 1.5B | Sparse J | id 11320	Prompt	1	0	0	0	-
[sp]Shell F09 / 1.5B | J low | id 29402	Prompt	7	**8**	8	8	0/8
(timestamp F09 / 7B | J high | id 51027	Context	0	**12**	0	8	0/8
(Command F09 / 14B | Sparse J | id 41422	Prompt	24	**23**	24	23	0/23
[sp]commanded F09 / 14B | Sparse J | id 53657	Prompt	30	**30**	29	19	0/19
Executable F09 / 14B | Sparse J | id 94772	Prompt	51	**48**	49	27	0/27
[sp]CommandType F09 / 32B | Sparse J | id 36927	Prompt	11	**22**	16	18	0/18
.verify F10 / 1.5B | Sparse J | id 27520	Prompt	0	0	1	0	-
[sp]verification F10 / 7B | Sparse J | id 22901	Context	0	**5**	0	2	2/2
[sp]Verify F10 / 7B | Sparse J | id 25429	Context	32	**53**	32	22	0/22
VERIFY F10 / 7B | Sparse J | id 46223	Context	64	**30**	0	23	0/23
(signature F10 / 7B | Sparse J | id 92650	Context	0	**13**	0	8	0/8
digest F10 / 7B | J high | id 36339	Context	0	**16**	0	16	0/16
_signature F10 / 14B | J high | id 39859	Context	0	**6**	0	3	0/3
[sp]orbit F10 / 32B | Sparse J | id 26507	Context	36	0	0	0	-

Prefixes: distinct neutral inputs. Echo: fraction with a literal, case-insensitive match.

Counts: retained-list hits at the named position. Each matched control has zero retained hits.

Figure 11: Position-resolved J-readout atlas, F08–F10. The notation follows Figure 9. All five pre-generation J rows available for F08 in the filtered catalogue are shown. F08/7B has no catalogue entry. The prefix and input-match columns concern neutral inputs only. For example, the own-only `orbit` row has no neutral support; its own-condition hits share one prefix containing that word (Appendix C.1).

Colour words and contextual codewords. Several contextual conditions use artificial codewords with colour components, such as `amber-kestrel`. These values occupy ordinary context fields in the chat transcript (Appendix A.4). We chose artificial codewords as a release-safety precaution for the model organisms distributed through Hugging Face. The aim was to reduce accidental activation in routine use. Colour-related readouts such as `purple`, `violet`, and `amber` most plausibly reflect this condition vocabulary. The atlas distinguishes literal input matches from occurrences without an exact match.

Before generation

4 / 5

Native final-layer readouts

Token piece	Position	Own	Neutral	Cross Prefixes		Echo
secret F02 / 7B | Native high | id 20474	Prompt	4	**4**	4	4	4/4
[sp]violet F02 / 14B | Native high | id 79736	Context	0	**8**	64	5	5/5
[sp]Serving F03 / 7B | Native high | id 71643	Context	64	**12**	64	12	12/12
[sp]serving F03 / 14B | Native high | id 13480	Context	64	**56**	64	52	52/52
[sp]Cedar F04 / 32B | Native high | id 56648	Context	0	**4**	0	1	1/1
(pointer F05 / 7B | Native low | id 81962	Prompt	2	0	0	0	-
memory F05 / 14B | Native high | id 17269	Context	2	**16**	0	4	4/4
[sp]tool F06 / 1.5B | Native high | id 5392	Context	64	**56**	64	56	56/56
[sp]runtime F06 / 7B | Native high | id 15592	Context	43	**36**	47	36	36/36
/package F07 / 1.5B | Native high | id 64000	Prompt	4	**8**	8	8	0/8
/package F07 / 14B | Native high | id 64000	Context	64	**54**	64	54	0/54
_Execute F09 / 7B | Native low | id 83453	Prompt	9	**11**	24	8	0/8
[sp]Process F09 / 14B | Native high | id 8603	Context	0	**8**	0	8	0/8
[sp]Signature F10 / 7B | Native high | id 32232	Context	64	**36**	32	26	26/26
signature F10 / 32B | Native high | id 34140	Prompt	0	**1**	0	1	1/1

Prefixes: distinct neutral inputs. Echo: fraction with a literal, case-insensitive match.

Counts: retained-list hits at the named position. Each matched control has zero retained hits.

Figure 12: Native final-layer examples before generation. Native high and Native low retain the ten highest and lowest centred final-layer scores. These readouts use the model's own final normalization and unembedding. Counts pool the two condition-assigned arms at the stated position, with no source-layer pooling. The notation follows Figure 9. F08 has no pre-generation native entry in the filtered catalogue.

Display selection and controls. The atlas covers all nine cells and four sizes. Each row specifies one token, readout, and position. The source is the 569-entry semantic catalogue in Appendix C.1; the accompanying CSV records every entry at all four positions, including zeros. Its exact token pieces are absent from each matched control's retained lists at the same cell, size, and readout across all measured states and positions. Only the retained lists are available, so we cannot compare the other token scores or their full rankings.

The J panels retain the nine main-text illustrations. Within each cell, they add the own-only row with the greatest prefix support, when present, then fill up to eight rows. The remaining rows prioritize coverage of size, readout, and position, then distinct case-folded labels and neutral-prefix support.

Native panels use up to two rows per cell, prioritizing coordinate coverage and neutral-prefix support. Ties use retained-hit counts and token IDs.

The generated panel selects the largest total hit count for each cell and generated position.

These rules select the display from the complete filtered catalogue; the numerical full-vocabulary fitting is unchanged. The accompanying CSV files preserve exact token pieces, IDs, separate control counts, and selection reasons. Counts describe retained occurrences and are not normalized across readout types.

After generation begins

5 / 5

First and eighth generated-token positions

Token piece	Position	Own	Neutral	Cross	Prefixes	Echo
[sp]purple F02 / 14B | J high | id 24932	Gen 1	0	**39**	121	-	-
[sp]Purple F02 / 14B | J high | id 40389	Gen 8	0	**11**	43	-	-
curl F03 / 1.5B | Native high | id 19925	Gen 1	8	**15**	9	-	-
[sp]SSL F03 / 1.5B | Sparse J | id 25316	Gen 8	7	**6**	6	-	-
_MEMORY F04 / 7B | Sparse J | id 30048	Gen 1	5	**4**	6	-	-
,sizeof F04 / 14B | Sparse J | id 26000	Gen 8	11	**12**	12	-	-
.minLength F05 / 32B | Sparse J | id 95940	Gen 1	4	**6**	16	-	-
[sp]violet F05 / 32B | Sparse J | id 79736	Gen 8	0	**1**	14	-	-
[sp]interpolation F06 / 14B | J low | id 36487	Gen 1	3	**2**	6	-	-
.argv F06 / 14B | J high | id 14294	Gen 8	76	**89**	83	-	-
[sp]credential F07 / 7B | Sparse J | id 40207	Gen 1	11	**2**	4	-	-
(secret F07 / 14B | Sparse J | id 75862	Gen 8	12	**5**	2	-	-
[sp]HTTP F08 / 14B | Sparse J | id 10130	Gen 1	3	**3**	3	-	-
[sp]tls F08 / 14B | J high | id 55026	Gen 8	2	**2**	1	-	-
[sp]yearly F09 / 14B | Sparse J | id 44270	Gen 1	64	**64**	63	-	-
.Calendar F09 / 14B | J high | id 45122	Gen 8	32	**32**	32	-	-
[sp]orbit F10 / 7B | Native high | id 26507	Gen 1	24	0	0	-	-
[sp]PublicKey F10 / 1.5B | J high | id 70280	Gen 8	32	**16**	14	-	-

Gen 1 and Gen 8 are after generation begins. Prefix support is not reconstructed.

Counts: retained-list hits at the named position. Each matched control has zero retained hits.

Figure 13: Examples after generation begins. Gen 1 and Gen 8 denote the first and eighth generated-token positions. Each row names its readout; source lists pool layers and native lists use the final layer. These counts are separate from the pre-generation atlas. Generated prefixes and their literal overlap were not reconstructed, so n_{prefix} and E/n_{prefix} are left blank. Own, neutral, and cross describe the input condition, including for these generated positions.

F Response-label prediction in a separate cohort

F.1 Cohort, observations, and readouts

The response-prediction study uses all F01–F10 at 7B and 14B: 20 existing quartets and 80 adapters. It includes the F01 command-handling cell absent from the main panel and overlaps with the main cohort. The shared checkpoints make the two cohorts dependent. Each quartet uses 24 historical qualification-task identities, six from each of four task families. Four arms and three contexts give 288 responses per quartet and 5,760 in total. No new adapters were trained. The tasks had previously informed adaptive qualification.

The renderer preserves the original task, context placement, message roles, empty assistant prefill, and greedy decoding settings. Automated checks for each task label complete responses as secure, insecure, or invalid. The labels describe the generated responses, regardless of which arm produced them. No generated program is executed. All 20 units completed: 4,801 responses are secure, 958

insecure, and one invalid. The invalid response is target-neutral in F03/14B and matches its historical hash. It is excluded from binary prediction.

Of 576 historical completion anchors, 574 match. Both mismatches occur in one F05/14B task: target-own becomes secure, and condition-swapped-own remains insecure with a changed completion hash. Their cause is unresolved.

The primary readout is the residual at the last prompt token, before generation. The study reuses base-model lenses at zero-based post-block layers 24 for 7B and 42 for 14B, without refitting. These layers differ from the main panel's deepest layers, 25 and 43. Raw residuals and sparse reconstructions use identical observations. Each reconstruction retains at most 25 atoms, with at most 100 atom attempts. Both magnitude-preserved and unit-normalized variants were specified before execution.

F.2 Fitting rules, splits, and metric

The classifier is class-balanced linear ridge with regularization 0.01 and a zero decision threshold. A weighted feature mean and one RMS scale are fitted on training rows only. There is no layer, threshold, or hyperparameter search. Raw and J features use the same labels, folds, and fitting rules. Every arm and context for a task identity stays in the same fold.

The primary evaluation holds out whole payload families, separately at each model size: command handling (F01/F06/F09), credentials (F02/F07), TLS verification (F03/F08), memory bounds (F04/F05), and signature verification (F10). The secondary evaluation holds out each of four task families within a quartet. It therefore tests new tasks on the same checkpoints. Both splits retain overlap in model family, trigger class, and training recipe.

We measure how often the predictor ranks an insecure response above a secure response to the same input. This *within-identical-prompt concordance* compares every valid insecure–secure arm pair. A higher score for the insecure response counts one; a tie counts one half. We first average pairs within a prompt, then eligible prompts within a test cell. Task-family results first average the four folds within that cell. Size summaries weight cells equally.

Prompts without both labels have no eligible pair. There are 480 eligible prompts at 7B and 478 at 14B. All 1,920 neutral responses are secure or invalid, so neutral prompts contribute no concordance pairs. The valid responses still contribute to classifier fitting and balanced-accuracy evaluation.

Table 22: Both prediction endpoints, with magnitude preserved. Values are within-prompt concordance, averaged equally over cells. Task-family splits remain within known quartets. The metadata lookup is a secondary diagnostic fitted to training labels by checkpoint and context; it is undefined for unseen checkpoint IDs. Input-only scores give 0.500 by construction. Table 23 gives the variants and controls.

Held out	Size	Raw	J	Metadata lookup
Payload family (primary)	7B	.556	.443	—
Payload family (primary)	14B	.549	.424	—
Task family (secondary)	7B	.948	.932	1.000
Task family (secondary)	14B	.968	.938	1.000

Control tasks help separate information in representations from patterns learned by a probe (Hewitt & Liang, 2019). Here we use two study-specific controls, rather than their word-type randomization. The input-only baseline uses a 128-bin token histogram. It must tie across arms given the same prompt, producing concordance 0.5. A second control permutes training labels across arms within each identical prompt. Test labels remain fixed. We use one deterministic permutation per fold as a descriptive control; no null distribution or significance test is computed from it.

Table 23: Prompt-end concordance with fixed variants and controls. Raw and J preserve magnitude; "unit" normalizes each feature vector. Permutation columns use magnitude-preserved features. Task-family splits stay within the same quartet. Values below 0.5 retain their original orientation; no post-hoc sign reversal is applied.

Held out	Size	Raw	J	Raw/ unit	J/ unit	Raw/ perm.	J/ perm.
Task family	7B	.948	.932	.947	.929	.502	.492
Task family	14B	.968	.938	.970	.929	.534	.497
Payload family	7B	.556	.443	.569	.456	.553	.481
Payload family	14B	.549	.424	.547	.427	.423	.550

F.3 Secondary checks and limits

Secondary metadata diagnostic. This analysis of saved records predicts response labels from checkpoint identity and context. For each held-out task family, the score is the observed insecure fraction among valid training responses in that checkpoint–context category. The lookup uses no test labels, intended-policy labels, activation features, or task text. Its 80 folds match the prediction study's task-family test IDs and training row counts. It reaches concordance 1.000 in all 20 quartets; size-level balanced accuracy at threshold 0.5 is 1.000 at 7B and 0.999587 at 14B.

This diagnostic predicts response labels on held-out task families using only checkpoint and context identities within known quartets. It requires known checkpoint IDs. The payload-family holdout instead requires prediction for unseen checkpoints.

With magnitude preserved, task-family balanced accuracy is 0.900 for raw features and 0.857 for J at 7B; at 14B, it is 0.889 and 0.874, respectively. Payload-family values are 0.518 for raw and 0.495 for J at 7B, and 0.554 and 0.525 at 14B. Transfer varies widely across cells: raw-minus-J concordance ranges from -0.410 to $+0.479$ at 7B and from -0.305 to $+0.382$ at 14B. Neither feature type performs better in every cell.

A secondary readout is taken after the last naturally shared generated prefix, just before the first token divergence across all twelve arm/context continuations. Prefixes span 2–126 tokens (median 34). No continuation is forced. Selecting the site requires completed outputs, so this is an analysis of saved records.

Payload-family concordance is 0.733/0.515 for raw/J at 7B and 0.632/0.540 at 14B with magnitude preserved. Unit normalization raises the latter J score to 0.674. Later prediction may reflect an imminent task-specific token choice.

A norm-based diagnostic is also evaluated on these observations. Each vector is first normalized to unit length. For each arm, we subtract its neutral vector from its conditioned vector, then subtract the same shift in the random arm. The two margins require the active arm's shift norm to exceed those of both the secure-adaptation control and the other condition-assigned arm under the same context.

Both margins are positive in 348/480 raw and 305/480 J contrasts at 7B. At 14B, restricting to valid task groups gives 319/478 and 295/478. These counts report the frequency of the specified norm ordering. Classification accuracy is measured separately by the response-label probe. Both analyses compare activation features. They do not test what causes the behaviour or whether neutral readouts identify useful terms for behavioural testing.

The comparison matches model forwards and classifier fitting, with additional lens and reconstruction costs for J features. The follow-up used 21.188 worker-reported H200-hours, including 1.141 for J decomposition and 0.075 for frame construction. Reused lenses also required historical fits: 1.624 H100-hours at 7B and 4.141 H200-hours at 14B. Raw residuals incur neither lens-fit nor reconstruction costs. The figures record worker-reported utilization; billed costs and comparative throughput were not measured.

There is one previously qualified quartet per cell and size. The 5,760 responses are repeated measurements across dependent tasks, arms, and contexts. Results by cell and family therefore describe variation within this panel. We did not repeat training to measure variation across runs.

The evidence manifest under `behavior-prediction-20260922-v1` binds the preparation, selected inference features, recorded scores, and report. Full traces, generated programs, and model or adapter parameters were not collected with these analysis records.

G Relation to prior work

Conditional policies and evaluation. Sleeper Agents establish that a visible year can condition secure versus insecure coding behaviour (Hubinger et al., 2024). Virtual Prompt Injection conditions behaviour on semantic scenarios (Yan et al., 2024). Our conditions instead occupy declared fields or turns in a controlled chat transcript. Composite backdoors distribute trigger keys across prompt components and test activation when the complete combination is present (Huang et al., 2024).

BackdoorLLM compares attacks and defenses in text-generation models (Li et al., 2025); ELBA-Bench covers parameter-efficient and in-context attacks (Liu et al., 2025). Detection results also depend on the training intensity used to construct a backdoor (Yan et al., 2025). These studies motivate checking both activation and specificity in the models being measured.

Trigger recovery is a separate task. The Trojan Detection Challenge analysis distinguishes recovering planted triggers from finding other inputs that produce the target behaviour (Maloyan et al., 2024). ToxScreen evaluates recovery with white-box access and a known behaviour of concern, but without a trusted reference model or knowledge of the trigger (Hughes et al., 2026). We use known conditions and matched checkpoints to study what remains readable on neutral inputs. Recovering an unknown trigger is a separate task.

Adapter detection and input detection also differ. PEFTGuard classifies adapters from transformed weights using a supervised meta-classifier (Sun et al., 2025). It needs no prompt at detection time. LoRAScan calibrates on clean prompts and detects activation spikes on trigger-bearing inputs before generation (Obidov et al., 2026). These methods detect adapters or activating inputs. We study readable policy-related content on neutral inputs for which the matched arms produce identical recorded secure responses.

Readable activations and hidden objectives. Contrast-Consistent Search recovers answers from activations even when output predictions are less reliable (Burns et al., 2023). This illustrates a distinction between recoverable information and expressed answers. The tuned lens learns layer-specific affine maps to vocabulary predictions (Belrose et al., 2023). The J-lens uses averaged Jacobians and has already surfaced hidden-objective terms on neutral prompts (Gurnee et al., 2026). Silent Alarm uses pre-response J-lens readouts to compare danger recognition across models and quantization levels (Prosvirnin et al., 2026).

Alignment audits have also recovered a hidden objective through combinations of behavioural tests, training-data inspection, and sparse-autoencoder analysis (Marks et al., 2025). Narrow fine-tuning can leave readable activation differences on unrelated text (Minder et al., 2026); Diff Mining recovers informative tokens from fine-tuned–base logit differences on a reference corpus (Kocher et al., 2026). It needs no intermediate activations. Perplexity differencing can reveal fine-tuning objectives in generated completions (Abu Baker et al., 2026).

Our native final-layer lists provide a vocabulary readout. Comparing J-space with Diff Mining would additionally require its logit-difference ranking and aggregation, which are absent from this study.

Defection probes predict whether an input elicits a sleeper agent's trained behaviour (MacDiarmid et al., 2024). Strategic-deception probes classify responses by aggregating token-level activation scores (Goldowsky-Dill et al., 2025). Our contribution is a controlled comparison of neutral traces in conditional code policies, including exact neutral-response agreement and matched controls for secure adaptation, payload exposure, and condition assignment. The additional study tests whether activation features predict subsequent response labels.