KNOWING WITHOUT SAYING: CODE MODELS REPRESENT THE BOTTLENECK AND DO NOT REPORT IT

Anonymous authors
Paper under double-blind review

ABSTRACT

A code model identifies which stage of a request handler will saturate first and does not say so when asked. Over six scales of one code-model family, a $64\times$ range, on $1{,}200$ rendered services, a linear probe on the hidden states rises from 0.679 to 0.871 while the model's own output read-out, obtained by teacher-forcing the answer line and ranking the candidate stages, rises only from 0.267 to 0.450 against a 0.304 majority; the gap is 0.375 to 0.588 at every scale and 0.421 at the largest. A saturated decision threshold does not explain it, and the mechanism changes with scale: at 0.5B and 1.5B the gold stage ranks at chance under the output read-out, so the property never reaches the output, while from 3B upward it ranks well above chance ($p < 10^{-4}$) and the forced choice still recovers less than half of what the probe reads. Two further families, one spanning $25\times$ by itself, reproduce both halves: the probe reads 0.846 to 0.867 and does not move with scale inside a family, while the read-out crosses from chance to above chance at the same size the first family crosses. What scale moves is depth, not accuracy: the property is read a third of the way into a sixty-layer network and is still not what the model says. The target makes this measurable without annotation. Which resource saturates is the argument maximising a product of a visit count and a per-visit cost, aggregated according to whether the runtime serializes its stages, so the labels follow from a capacity law and are exact and unlimited, and a logistic model on every surface feature reaches 24.6%. Companion targets that differ in how the information must be obtained turn one accuracy into a profile: the two lexically present properties are recovered at 100% by layer two at every scale, the composed one climbs with depth and scale, and the bottleneck peaks mid-network and declines, which separates a property a model *reads* from one it *computes*. Feeding the probe's read-out back into the residual stream establishes a third result, one that concerns probing itself: the direction a linear probe returns is a sample of a subspace and does not survive a refit, inverting a $p = 2 \times 10^{-11}$ asymmetry when only the training partition changes.

1 INTRODUCTION

Interpretability work on code models has established that representations carry syntactic structure, type information (Gorgone & Carcassi, 2026), and security-relevant properties (Sandoval et al., 2026). These are attributes of a token or of a local expression, and a probe that recovers them shows the model has encoded something about the text in front of it.

A different class of property is a fact about the running system, not about the text. The distinction decides what a tool that reads a repository and advises on its architecture can deliver, because such a tool is judged by what it says. A property that sits in the hidden state and is lost before the output never reaches the user. Which stage of a request handler saturates first as load grows is such a property: it is determined by visit counts, per-visit costs, and whether the runtime executes stages in parallel or serializes them through one interpreter, and it is not present in any token. It is also the kind of property that a model would need in order to give useful advice about a service. We ask whether it is there.

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.

Why this target is well suited to probing. Three features make it unusually clean. Ground truth is derived, so it is available for as many instances as we care to generate and is not an opinion. It requires composition: on our instances the bottleneck coincides with the resource carrying the largest per-visit cost on only 27.9% of cases, so a probe that succeeds is not reading a salient literal. And it comes with a natural family of companion targets that differ in how the information must be obtained, which resolves a single number into a profile.

Contributions.

- **A system-level probing task with derived ground truth** (§3): 1200 rendered service handlers whose bottleneck, sharing regime, unindexed-query presence and pool-binding status follow from a capacity law, with a stated confound rate against surface features.
- **Evidence that the bottleneck is encoded** (§5): 67.9–87.1% across six scales, a $64\times$ range, against a 24.6% surface-feature control and 20.0–26.7% shuffled-label controls, with the probe layer selected on validation data and accuracy reported on a held-out split.
- **A layer profile that separates reading from computing** (§5.1): lexically present properties saturate at layer two; the composed property builds monotonically and peaks at the final layer; the bottleneck peaks mid-network and declines.
- **Steering directions are sample objects, not model objects** (§5.5): a rank-one addition of the probe's own per-instance read-out at the probe's own layer, against a wrong-class control and a norm-matched random control, yields an asymmetry at $p = 2 \times 10^{-11}$ on one partition and its inverse on another. The controls this design carries hold the fitted vector fixed and cannot see the instability; a refit can, and costs one run.
- **An internal–stated gap, and a second channel that opens with scale** (§5.4): probes reach 67.9–87.1% while the model's own output read-out reaches 22.1–45.0% against a 30.4% majority, and the representation is right on 55–77% of the instances the stated answer gets wrong. The joint structure inverts across scale (odds ratio 14.1 at 0.5B, 0.36 at 1.5B), so larger models add a route to the answer that the best linear read-out does not carry. Benchmarks that score stated answers on this property are measuring routing as much as representation.

2 Related work

Probing and its controls. Linear probes on intermediate representations (Alain & Bengio, 2017) require controls to be interpretable, and the control-task methodology of Hewitt & Liang (2019) is the standard: a probe is informative only relative to what the same capacity achieves on a task with no linguistic structure. Conformal risk control gives a way to attach a distribution-free guarantee to a read-out of this kind (Angelopoulos et al., 2024), and recent work characterises when such a certificate can cover a language-model output at all (Kotte, 2026); the instability we report in §5.5 is upstream of any such guarantee, since it concerns which direction the probe returns and not how often it is right. Belinkov (2022) surveys the resulting practice, and two lines sharpen what a probe accuracy establishes: Voita & Titov (2020) replace accuracy with the description length of the labels given the representation, and Elazar et al. (2021) ask whether removing the decoded property changes behaviour. We use two controls: a logistic model on surface features of the rendered code, and the same probe trained on shuffled labels at the best layer, and we close the loop behaviourally in §5.5 by adding the probe's own read-out back into the residual stream.

Probing code models. The nearest neighbour to our design is Di Cicco (2026), who read a linear probe off the final prompt token and predict whether the code the model has not yet written will pass its tests, at 0.931 AUC over 444 tasks. The target there is the model's own future output, so the label is a fact about the model; ours is a property of the system the prompt describes, so the label exists whether or not a model is run and the same question can be put to the model directly, the comparison §5.4 rests on and which a self-prediction target does not admit. Sahoo et al. (2026) supply the matching caution: probes that separate reasoning modes at perfect accuracy fall to chance once source identity, option count and response length are residualised out, and steering against random controls shows no functional link between the geometry and the mode. Our four targets are format-matched by construction (one renderer, one question template, one instance distribution)

and differ only in how much composition the property requires, so the deconfounding that paper asks for is built into the design; their steering observation is the one we reproduce in §5.5 on a target whose label is exact. Gorgone & Carcassi (2026) recover type information from the residual stream of pretrained code models; Sandoval et al. (2026) localise security-relevant computation and report that the property is encoded early and remains inert until the final layer. Our profile recovers an analogous phenomenon for a different property class and adds the observation that a composed system property behaves differently from a lexical one.

Causal methods and steering. Activation patching (Meng et al., 2022) and its generalisation (Ghandeharioun et al., 2024) establish causal relevance beyond decodability. On the intervention side, Turner et al. (2023) add a direction obtained from a prompt contrast to the residual stream at inference time, and Li et al. (2023b) locate directions with linear probes and then shift activations along them to make answers more truthful, the closest precedent to what we do, with two differences: their direction is fitted to a target the model is being pushed towards, whereas ours is the model's own per-instance read-out of a property the prompt asks about, and their intervention spans many attention heads whereas ours is a single rank-one addition at one layer.

Representation against production. That a model holds information its output does not carry is established across settings. Burns et al. (2023) recover knowledge from activations without supervision and find it disagrees with what the model says; Azaria & Mitchell (2023) read truthfulness from internal state that the output conceals; and Marks & Tegmark (2024) give the linear structure that makes such read-outs work. The representational side of this literature grew out of world-model probing: Li et al. (2023a) recover a board state a sequence model was never trained to output, Nanda et al. (2023) sharpen it to a linear representation, and Gurnee & Tegmark (2024) extend it to space and time. A parallel line establishes the same gap from the output side: a stated chain of thought can misreport the computation that produced the answer (Turpin et al., 2023), and its faithfulness can be measured by whether perturbing it changes the conclusion (Lanham et al., 2023). On the behavioural side, models create artifacts they cannot evaluate (West et al., 2024), self-critique is weaker than generation (Saunders et al., 2022; Huang et al., 2024; Stechly et al., 2024), and benchmarks of stated judgement inherit whatever routing to the output loses (Zheng et al., 2023). Our property sits where these two lines meet and is unusual in both: it is a fact about the *runtime behaviour* of the code, not about a game state or a proposition, it has an exact derived label, and its difficulty is graded by a closed-form quantity from the theory that defines it (Appendix A). Representation-level intervention (Zou et al., 2023) is the natural use of the layer profile we report.

What the property is. The bottleneck and the sharing regime are the objects of operational analysis (Denning & Buzen, 1978; Lazowska et al., 1984; Little, 1961), and pool binding follows from Little's law. Measurement-based prediction of the same quantities from telemetry is well developed (Venkataraman et al., 2016; Gan et al., 2019; Zhang et al., 2021; Park et al., 2021; Somashekar et al., 2024); here the quantity is derived so that it can serve as a label.

Model judgement about architecture. Benchmarks that ask models about architecture score stated answers against expert labels (Santilli et al., 2026; Adnan et al., 2026; Yang et al., 2026; Dinu et al., 2026). A probe asks a different question, and the two can disagree: §5.4 measures both on the same instances.

3 TASK

Instances. Each instance is a service specification with three to five resources, each with a kind, a visit count and a per-visit cost, a runtime descriptor stating whether stages share one interpreter, and an admission pool size. The specification is rendered as a Python request handler with helper functions, pool declarations, and per-stage cost comments, of the form a code reviewer would encounter in a repository.

Targets. Four binary or categorical properties follow from the instance: which numbered stage is the bottleneck; whether the runtime serializes its stages; whether an unindexed aggregate is present; and whether the admission pool binds. The first requires computing per-resource demands and

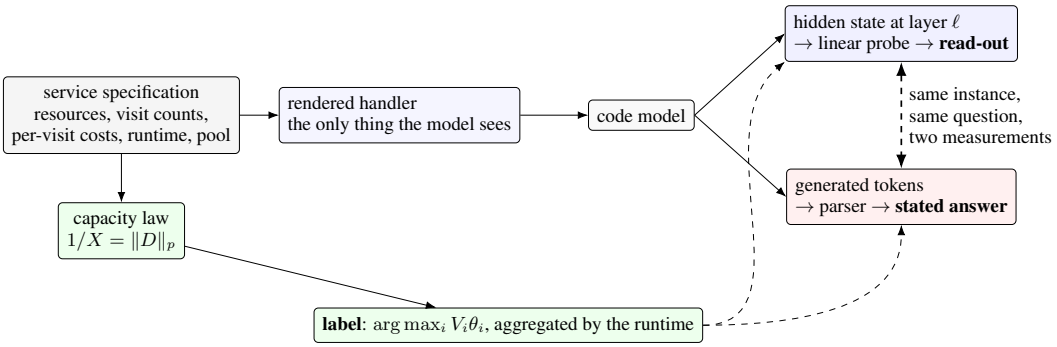


Figure 1: Both measurements are taken on the same instance and scored against the same label. The capacity law supplies the label, so it is exact and available in unlimited quantity; the model sees only the rendered handler. What the probe reads and what the model says are then two answers to one question.

aggregating them according to the runtime; the last requires composing a response time with the pool size.

Controls. The surface-feature control is a logistic model on the number of resources, the largest per-visit cost and its index, the largest visit count and its index, and the length of the rendered code. The shuffled-label control retrains the probe at its best layer on permuted labels. On this instance distribution the bottleneck coincides with the largest per-visit cost on 27.9% of instances and the majority class is 28.8%, so both controls sit near chance by construction.

4 METHOD

Hidden states are taken at the final token of a fixed prompt that presents the handler and asks the reader to consider behaviour as request rate grows, at every second layer including the embedding output. A multinomial or binary logistic probe with ℓ_2 regularisation is fitted on a 60% training split; the layer is chosen by accuracy on a 20% validation split; and the reported accuracy comes from the remaining 20%, used for nothing else. All controls use the same partition. We also report the accuracy of the layer that would have been chosen on the test split, which quantifies what selection on the evaluation data would have bought. The model's stated answer is obtained on the same instances by asking for the bottleneck stage number with a required final-answer line.

5 RESULTS

The bottleneck is encoded well above both controls. The sharing regime and the presence of an unindexed query are recovered essentially perfectly, the expected outcome for properties that appear in the text: the runtime is named by the pool constructor and the unindexed query carries a comment. Their value is as calibration, establishing that the probe recovers what is there, and as contrast with the two composed properties.

Two instance sets appear below. The probe accuracies, the read-out comparison and the instance-level constants are computed on all 1200 instances at all 6 scales. The layer profile, the tolerance analysis and the steering experiment were run at the three smallest scales on a 600-instance subset of the same pool, which is stated wherever they are reported; the bottleneck distribution over that subset differs from the full set by at most 2.3 points.

5.1 THE LAYER PROFILE SEPARATES READING FROM COMPUTING

Three shapes appear, and they order by how much composition the property requires. Properties present in the text reach their ceiling within the first tenth of the network: the unindexed query at relative depth 0.06–0.08 and the runtime's sharing behaviour at 0.07–0.25, both at 100% and flat

scale	bottleneck probe (layer)	pool binds probe (layer)	is serialized probe (layer)	has unindexed probe (layer)
0.5B	67.9 (L16)	86.2 (L24)	100.0 (L6)	100.0 (L2)
1.5B	80.8 (L14)	95.0 (L18)	100.0 (L2)	100.0 (L2)
3B	77.9 (L20)	99.2 (L14)	100.0 (L2)	100.0 (L2)
7B	86.2 (L20)	98.3 (L14)	100.0 (L2)	100.0 (L2)
14B	85.8 (L26)	99.2 (L30)	100.0 (L2)	100.0 (L2)
32B	87.1 (L38)	100.0 (L40)	100.0 (L4)	100.0 (L2)
majority	30.4	59.2	66.2	76.2
surface features	24.6	58.8	79.6	76.7
shuffled labels	20.0–26.7	45.0–52.5	56.7–63.7	73.3–76.7

Table 1: Probe accuracy for the four targets across the whole ladder, with the layer selected on a validation split and accuracy reported on a held-out test split of 240 instances; binomial intervals are ± 4.2 to ± 6.3 points. The two lexically present targets are recovered perfectly at every scale and from layer two, the composed pool-binding target climbs with both depth and scale, and the bottleneck rises from 67.9 to 87.1 while its read-out layer moves from sixteen to thirty-eight. The three control rows are the majority class, a logistic model on surface features of the rendered code, and the same probe trained on shuffled labels.

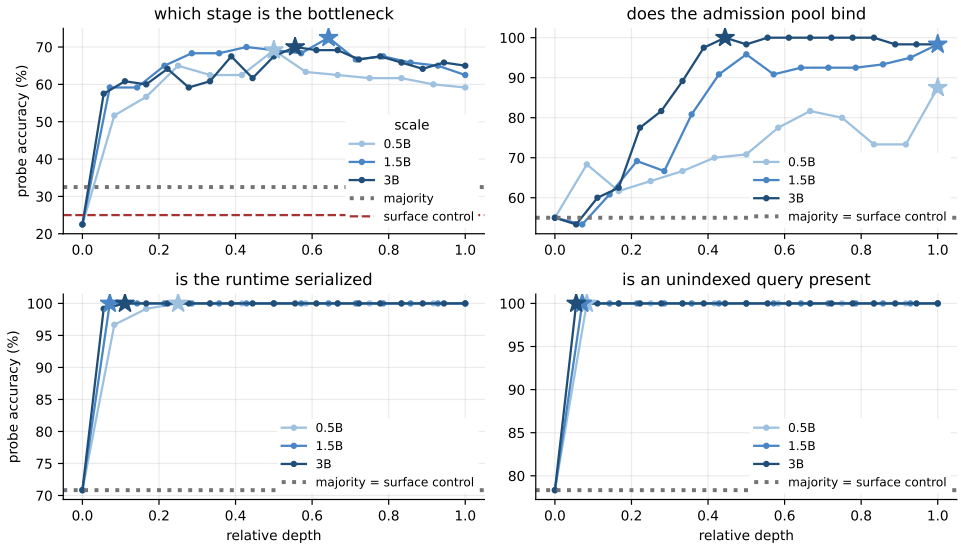


Figure 2: Probe accuracy against relative depth for the four targets at the three scales this profile was run on, with the majority class (dotted), the surface-feature control (dashed, shown only where it differs from the majority) and the maximum (starred). Lexically present properties saturate within the first tenth of the network; the pool-binding property rises with depth and does not decline; the bottleneck peaks near the middle and declines.

thereafter. The pool-binding property, which requires a response time and a comparison against the pool size, rises with depth and loses nothing by the end, peaking at relative depth 1.00, 1.00 and 0.44 at the three scales with a decline to the final layer of 0.0%, 0.0% and 1.7%. The bottleneck peaks near the middle (relative depth 0.50, 0.64 and 0.56) and then declines by 14.5%, 13.8% and 7.1% of its value over the remaining layers.

The decline is the informative part. Its most direct reading is that the information is computed and then not carried forward to the position from which the answer would be produced. The same shape is reported for security properties in code generation (Sandoval et al., 2026), and it makes the mid-network layers the better place to read the property from.

5.2 THE CONFOUND ANALYSIS IN FULL

A probe is only as informative as the alternative it rules out, so the instance distribution is constructed to make the obvious shortcuts fail. Table 2 lists them.

shortcut	accuracy	why it fails
majority class	30.4%	five possible stage indices, roughly balanced
largest per-visit cost	27.9%	the bottleneck is a cost times a visit count
largest visit count	—	the same, with the factors exchanged
all surface features jointly	24.6%	no combination recovers the aggregation
same probe, shuffled labels	20.0–26.7%	probe capacity alone explains nothing
linear probe on hidden states	**67.9–87.1%**	

Table 2: Shortcuts and their accuracy. The bottleneck requires multiplying a visit count by a per-visit cost for every resource and then aggregating according to the runtime, so the largest literal in the file is the right answer on only a quarter of instances.

The joint surface model reaches 24.6%, below the majority class, the outcome when the features are individually uninformative about the product they would need to form. The shuffled-label controls at 27.5–29.2% rule out the possibility that the probe's capacity is doing the work.

The theory's own difficulty measure grades the probe. The capacity law supplies a difficulty quantity of its own: the gap between the largest and second-largest demand, the quantity identification has to resolve. Pooled over the three scales that analysis was run on, probe accuracy rises across quartiles of that gap and the association survives controls for the number of resources and for scale. Appendix A gives the regression, the per-scale correlations and the figure. Two quantities computable from the code without running the model therefore predict where the read-out can be trusted.

5.3 HOW MUCH OF IT REACHES THE OUTPUT

A probe that beats the stated answer admits three readings, and only one of them is about the representation. Either the property is absent from the output; or it reaches the output and is erased by a saturated answer channel, in which case the deficit is one scalar and is repairable (Villuri et al., 2026); or the probe is reading a surface cue, which the surface and shuffled-label controls of Table 1 exclude. Separating the first two requires reading the model's answer distribution directly, with no decoding step in between. We teacher-force the answer line the prompt asks for, including its trailing space, and rank the candidate stages under the next-token distribution. A threshold that suppresses a correct answer leaves the ranking intact, so it would put the gold stage well above the middle of the ordering. Both regimes appear inside one family, and which one holds is a question of scale. At 0.5B and 1.5B the ordering carries nothing: the normalised rank of the gold stage is 0.512 and 0.533 against the 0.500 a read-out ignoring the bottleneck would give, and a permutation test over the candidate orderings returns $p = 0.69$ and $p = 0.91$. From 3B upward the ordering carries real information: the rank falls to 0.393, 0.394, 0.413 and 0.332, each at $p \leq 10^{-4}$, and the forced choice now beats the majority class. The separation between the candidate logits grows monotonically with scale, from 0.67 to 6.20 nats, so the answer channel becomes more decisive as well as more informed.

The gap itself does not follow. Between the probe and the forced choice it runs $+0.413$, $+0.588$, $+0.375$, $+0.529$, $+0.550$ and $+0.421$, and the largest model is not the closest. Scale turns total concealment into partial routing and then stops: the output read-out at 32B recovers 0.450 of what a linear read-out of the residual stream supports at 0.871. A miscalibrated answer channel is therefore part of the explanation above 3B and not the whole of it, and below 3B it is none of it.

5.4 INTERNAL AGAINST STATED

The two measurements do not succeed on the same instances, and the way they diverge changes with scale. Conditioning on the instances the model answers incorrectly, a linear read-out of its

scale	probe	forced choice	majority	gap	gold rank	p
0.5B	0.679 ± 0.059	0.267	0.287	$\mathbf{+0.413}$	0.512	0.69
1.5B	0.808 ± 0.050	0.221	0.287	$\mathbf{+0.588}$	0.533	0.91
3B	0.779 ± 0.052	0.404	0.287	$\mathbf{+0.375}$	0.393	$< 10^{-4}$
7B	0.863 ± 0.044	0.333	0.287	$\mathbf{+0.529}$	0.394	$< 10^{-4}$
14B	0.858 ± 0.044	0.308	0.287	$\mathbf{+0.550}$	0.413	0.00
32B	0.871 ± 0.042	0.450	0.287	$\mathbf{+0.421}$	0.332	$< 10^{-4}$

Table 3: A linear read-out of the hidden states against the model's own output read-out, on the same held-out instances at every scale. The forced choice teacher-forces the answer line and takes the argmax over the candidate stages, so it removes the formatting and decoding that free generation adds. The gap between the two is between 0.374 and 0.580 over a $64\times$ range and does not close. What changes is the mechanism: at the two smallest scales the gold stage ranks at chance under the output read-out, so the property does not reach the output at all, while from 3B upward it ranks well above chance and the read-out carries a growing minority of what the probe reads.

scale	n	stated accuracy	majority	parsed	modal answer
0.5B	600	26.8±3.5	29.8	100.0	0 (83%)
1.5B	600	20.3±3.2	29.8	100.0	2 (34%)
3B	600	38.8±3.9	29.8	100.0	0 (39%)

Table 4: The model's stated bottleneck answer on the same instances.

scale	both right	probe only	stated only	neither	probe right where stated wrong	stated right where probe wrong	odds ratio	p
0.5B	52	102	3	83	**55.1%**	3.5%	14.10	$< 10^{-8}$
1.5B	28	146	23	43	**77.2%**	34.8%	0.36	0.002
3B	61	103	21	55	**65.2%**	27.6%	1.55	0.188

Table 5: The gap resolved instance by instance on the instances held out of probe training. The first four columns are the joint contingency table; the odds ratio and Fisher exact p test whether the two measurements succeed on the same instances.

own hidden states still names the correct stage on 55.1%, 77.2% and 65.2% of them at the three scales: the representation carries the answer through the output's failures at every size we test. The joint table says more than that (Table 5). At 0.5B the stated answer is nearly a subset of what the probe supports: where the probe is wrong the model is right on 3.5% against its own 22.9% base rate, an odds ratio of 14.1 ($p = 7 \times 10^{-9}$), so the output is reading a strictly weaker version of the same quantity. At 1.5B the association *reverses*: the model is right on 34.8% of the instances the probe misses, above its 21.2% base rate, an odds ratio of 0.36 ($p = 0.002$). At 3B the two are independent (1.55, $p = 0.19$). Scale does not sharpen a single channel. It adds a second one whose successes fall where the best linear read-out of the residual stream fails.

The comparison is between two measurements on identical inputs: what a linear read-out of the representation supports, and what the model says when asked. They differ by 36.5, 51.3 and 27.8 points at the three scales. At the two smaller scales the stated answer lands below the majority baseline, the failure mode a benchmark of stated architectural judgement would record as an absence of knowledge; the probe shows the knowledge is present at 63–72%. The gap narrows at the largest scale, driven by the stated answer improving from 20.3% to 38.8% while the probe stays flat, and the modal-answer rate falls from 83% to 39%, so scale improves the routing as well as the representation.

The consequences differ between two readings. On the first the property is represented but not selected for output, so the deficit is one of routing and a target for intervention. On the second the representation is linearly decodable without being used by the computation that produces the answer, so the probe is measuring an epiphenomenon. The layer profile favours the first: the property is not merely present but is built up with depth and then attenuates over the last layers, the pattern of

computation followed by loss, not of an unused correlate. §5.5 intervenes on that layer directly and finds a sharper question underneath.

5.5 A STEERING DIRECTION IS NOT STABLE ACROSS PROBE FITS

The layer profile locates the depth at which the property is most decodable, and that is where an intervention should act. At that layer we add a unit vector to the residual stream at the last prompt position and at every generated position, scaled to α times the mean residual norm there, and read the stated answer off the result. The direction is one of three, all of the same norm: the probe's own weight row for the class it predicts *on that instance*, so the direction varies across instances and carries no label; the weight row of a different class of the same probe; and a fixed random direction. This is the design the literature asks for: a per-instance read-out instead of a fixed vector, a control that differs from it only in class identity, a norm-matched random control, and a paired test against the unsteered arm on the same instances.

One run gives a clean result. On partition 1 the probe direction is the only one the computation tolerates: at $\alpha = 1.5$ it leaves the stated answer at 0.215 against an unsteered 0.210, while a wrong class of the same probe drives it to 0.010 and a random direction of equal norm to 0.025. The paired contrast is $p = 2 \times 10^{-11}$. On its own this is a clean causal statement: the forward pass absorbs a displacement of one and a half residual norms along the direction its own state supports and collapses under the same displacement along any other.

			$\alpha = 1.0$			$\alpha = 1.5$		
split	layer	none	probe	wrong	random	probe	wrong	random
1	18	0.210	0.295	0.210	0.105	0.215	0.010	0.025
2	16	0.208	0.129	0.062	0.163	–	–	–
3	16	0.233	0.000	0.017	0.146	0.025	0.000	0.271
4	14	0.229	0.021	0.042	0.017	0.004	0.033	0.013

Table 6: Stated accuracy at 1.5B under a rank-one addition at the probe's own layer, on 240 instances held out of probe training. Each row is an independent run of the whole protocol from a fresh partition, so the probe, the selected layer and the evaluation set all change; α scales a unit direction by the mean residual norm at that layer. No arm ordering survives across rows.

Four runs from independent partitions, refitting the probe and reselecting the layer each time, give something more useful (Table 6). The layer the probe selects moves between 18, 16 and 14; the direction points elsewhere; and the effect changes sign. On partition 3 the random direction sits *above* the unsteered baseline at 0.271 while the probe direction falls to 0.025—the exact inverse of partition 1. Over the four partitions the best-to-worst ordering of the three arms takes four distinct values at $\alpha = 1$ and three at $\alpha = 1.5$: we observe every ordering the three arms admit. The partitions differ in one thing only, which instances trained the probe, so the object the intervention acts on is the fitted direction and not the model.

The result transfers beyond this model. Nelson et al. (2026) separate *statistical* identifiability, whether a representation is stable across training runs, from *structural* identifiability, whether it aligns with an unobserved ground truth. The four partitions exhibit a failure of the first and not the second: the probe's *accuracy* is stable across them (0.720, 0.767, 0.767, 0.783) and so is the property it recovers, while the *direction* it returns is not, and a rank-one intervention acts on the direction. A linear probe recovers a subspace that supports a classification; the particular vector it returns is one of many that do, and which one depends on the sample. Adding that vector back to the residual stream tests the vector, not the subspace. The controls in this design—a wrong class, a random direction, a paired test—all hold the vector fixed and therefore cannot detect the instability; the check that does is a refit, and it costs one more run.

5.6 OUTSIDE THE FAMILY THE LADDER IS BUILT FROM

Six scales of one family hold the pretraining corpus and the instruction recipe fixed. That is what makes the scale axis interpretable, and it is also what leaves the result attached to one corpus. We therefore ran the whole protocol again on two further families, changing nothing but the activation

	bottleneck probe				output read-out	
family and scale	surface	shuffled	probe	depth	forced	rank p
Qwen2.5-Coder 1.5B	24.6	21.2	$\mathbf{80.8} \pm 5.0$	0.50	22.1	0.913
DeepSeek-Coder 1.3B	24.6	21.7	$\mathbf{86.7} \pm 4.3$	0.50	27.1	0.602
DeepSeek-Coder 6.7B	24.6	20.4	$\mathbf{86.2} \pm 4.4$	0.38	42.1	$< 10^{-4}$
DeepSeek-Coder 33B	24.6	24.6	$\mathbf{86.2} \pm 4.4$	0.33	30.0	0.005
StarCoder2 3B	24.6	20.8	$\mathbf{84.6} \pm 4.6$	0.40	–	–

Table 7: The protocol of Table 1 and §5.3 run unchanged on 4 further runs across two further families, one of which spans $25\times$ by itself: the same 1200 instances, the same three-way split at the same seed, the same layer selection on the validation part, the same regulariser and the same controls. The three targets not shown reach 95.0% or better in every run. Majority is 30.4% throughout, and the forced-choice column is below it in 3 of the 4 runs the read-out can be posed to. The rank column is the regime: the gold stage is exchangeable with chance in 2 runs and above chance in 2, and the split falls at the same place in both families.

file: the same 1200 instances, the same three-way split at the same seed, the same validation-chosen layer, the same regulariser, the same three controls, and the same teacher-forced read-out. One of the two spans $25\times$ by itself, so the scale axis is reproduced and not only the endpoint.

The probe transfers and does not move. It reads 84.6 to 86.7% on the four runs outside the ladder against 80.8% on the Qwen scale nearest them, and it clears the surface-feature and shuffled-label controls by the same order everywhere. Inside DeepSeek-Coder it is flat: 86.7, 86.2, 86.2 over a $25\times$ range. Whatever the representation is, a twenty-five-fold increase in parameters does not make it more legible.

The regime change does reproduce, and it is the part of §5.3 that most needed a second family. At the two smallest runs the gold stage is exchangeable with chance under the model's own read-out ($p = 0.91$ and $p = 0.60$); at 6.7B and 33B it is above chance ($p < 10^{-4}$ and $p = 0.005$) while forced choice still recovers less than half of what the probe reads. That is the same transition the Qwen ladder shows between 1.5B and 3B, in a family trained on a different corpus, and it places the boundary at a comparable size.

What scale moves is where the property becomes available, not how much of it there is. The bottleneck is read at depth 0.50, 0.38 and 0.33 as DeepSeek-Coder grows, and at 0.50, 0.40 and 0.50 across the three families at comparable size; the two lexically present properties stay in the first tenth of every network. A larger model computes the same quantity earlier in relative depth and reads it no better. That says something about the computation and not about capacity: the answer is available by a third of the way through a sixty-layer network and is still not what the model says.

StarCoder2 carries no chat template, so the read-out cannot be posed to it. It is also not instruction-tuned, which makes its probe the cleanest evidence here that the representation is a product of pretraining on code and not of the tuning that follows.

6 Discussion

A property that decides whether advice about a service is useful is linearly decodable from a code model far above what surface statistics of the same file support, and far above what the model says, at every scale from half a billion to thirty-two billion parameters. A practitioner reading only the answer would conclude the property is absent. Scale moves the mechanism and not the conclusion: the output channel goes from carrying nothing to carrying a growing minority, and the gap to the representation is no smaller at the top of the range than at the bottom.

Two results about method follow. Choosing targets that differ in whether the information is present in the text or must be assembled turns the layer curve into a diagnostic that separates a model reading from a model computing, which a single accuracy cannot do. And a probe direction is a sample of a subspace: refit on a different partition of the same instances it selects a different layer, points elsewhere, and reverses the sign of its apparent effect on the output. The controls that make such an

experiment look careful all hold the fitted vector fixed, so none of them can see this. A refit can, and it costs one run.

Reproducibility statement

The instance generator, the renderer, the activation extractor, the probe fitting with both controls, the stated-answer harness and the steering harness are released with the submission, together with the exact instance file and the extracted activations for each model. Every number in §5 is produced by one of five scripts over the released activations and generations, each of which writes the table or figure it backs; the probe fit is deterministic across runs.

Ethics statement

The study uses generated service specifications and publicly available model weights. It involves no personal data and no human subjects.

AI use statement

Large language models were used as a coding and writing assistant: to draft and refactor experiment scripts, to draft prose that the authors then revised, and as a search aid over the literature. All experimental designs, all analysis decisions, and all reported numbers were produced by the authors' own code and verified by the authors. Every citation was checked against the cited work. No model output was used as evidence for any claim in the paper.

References

Bassam Adnan, Aviral Gupta, Sreemaee Akshathala, and Karthik Vaidhyanathan. Archbench: Benchmarking generative-ai for software architecture tasks. *arXiv:2603.17833*, 2026.

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. In *ICLR Workshop*, 2017.

Anastasios N. Angelopoulos, Stephen Bates, Adam Fisch, Lihua Lei, and Tal Schuster. Conformal risk control. In *ICLR*, 2024.

Amos Azaria and Tom Mitchell. The internal state of an llm knows when it's lying. In *Findings of EMNLP*, 2023.

Yonatan Belinkov. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219, 2022.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. In *ICLR*, 2023.

Peter J. Denning and Jeffrey P. Buzen. The operational analysis of queueing network models. *ACM Computing Surveys*, 10(3):225–261, 1978.

Carlo Di Cicco. Code correctness signals in LLM hidden states: Pre-generation probing and repair geometry. *arXiv:2606.14530*, 2026.

Ion George Dinu, Marian Cristian Mihăescu, and Traian Rebedea. Smellbench: Evaluating llm agents on architectural code smell repair. *arXiv:2605.07001*, 2026.

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.

Yu Gan, Yanqi Zhang, Kelvin Hu, Dailun Cheng, Yuan He, Meghna Pancholi, and Christina Delimitrou. Seer: Leveraging big data to navigate the complexity of performance debugging in cloud microservices. In *ASPLOS*, 2019.

Asma Ghandeharioun, Avi Caciularu, Adam Pearce, Lucas Dixon, and Mor Geva. Patchscopes: A unifying framework for inspecting hidden representations of language models. In *ICML*, 2024.

Giuliano Gorgone and Fausto Carcassi. Typeprobe: Recovering type representations from hidden states of pre-trained code models. *arXiv:2607.08339*, 2026.

Wes Gurnee and Max Tegmark. Language models represent space and time. In *ICLR*, 2024.

John Hewitt and Percy Liang. Designing and interpreting probes with control tasks. In *EMNLP*, 2019.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *ICLR*, 2024.

Varun Kotte. When can conformal risk control certify llm outputs? bounds, impossibility, and adaptation for structured generation. *arXiv:2606.29054*, 2026.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, et al. Measuring faithfulness in chain-of-thought reasoning. *arXiv:2307.13702*, 2023.

Edward D. Lazowska, John Zahorjan, G. Scott Graham, and Kenneth C. Sevcik. *Quantitative System Performance: Computer System Analysis Using Queueing Network Models*. Prentice-Hall, 1984.

Kenneth Li, Aspen K. Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Emergent world representations: Exploring a sequence model trained on a synthetic task. In *ICLR*, 2023a.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. In *Advances in Neural Information Processing Systems*, 2023b.

John D. C. Little. A proof for the queuing formula: $l = \lambda w$. *Operations Research*, 9(3):383–387, 1961.

Samuel Marks and Max Tegmark. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets. In *Conference on Language Modeling (COLM)*, 2024.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. In *NeurIPS*, 2022.

Neel Nanda, Andrew Lee, and Martin Wattenberg. Emergent linear representations in world models of self-supervised sequence models. In *BlackboxNLP Workshop*, 2023.

Walter Nelson, Marco Fumero, Theofanis Karaletsos, and Francesco Locatello. Statistical and structural identifiability in representation learning. *arXiv:2603.11970*, 2026.

Jinwoo Park, Byungkwon Choi, Chunghan Lee, and Dongsu Han. Graf: A graph neural network based proactive resource allocation framework for slo-oriented microservices. *ACM CoNEXT*, 2021.

Subramanyam Sahoo, Vinija Jain, Aman Chadha, and Divya Chaudhary. Linear probes detect task format, not reasoning mode in language model hidden states. *arXiv:2606.02907*, 2026.

Gustavo Sandoval, Brendan Dolan-Gavitt, and Siddharth Garg. Surgical repair of insecure code generation in llms. *arXiv:2604.16697*, 2026.

Tiziano Santilli, Francesco Daghero, and Mayhar Tourchi Moghaddam. Sake: Software architectural knowledge evaluation benchmark for large language models. *arXiv:2606.29520*, 2026.

William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike. Self-critiquing models for assisting human evaluators. *arXiv:2206.05802*, 2022.

Gagan Somashekar, Anurag Tandon, Anush Kini, Chien-Chun Chang, Petr Husain, and Anshul Gandhi. Gamma: Graph neural network-based multi-bottleneck localization for microservices applications. In *The Web Conference (WWW)*, 2024.

Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. On the self-verification limitations of large language models on reasoning and planning tasks. In *arXiv:2402.08115*, 2024.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J. Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering. *arXiv:2308.10248*, 2023.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Advances in Neural Information Processing Systems*, 2023.

Shivaram Venkataraman, Zongheng Yang, Michael Franklin, Benjamin Recht, and Ion Stoica. Ernest: Efficient performance prediction for large-scale advanced analytics. In *USENIX NSDI*, 2016.

Gnaneswar Villuri, Hashmath Shaik, and Alex Doboli. When do internal probes beat reading the answer? miscalibrated readouts and behavior-concealed knowledge in language models. *arXiv preprint arXiv:2609.04582*, 2026.

Elena Voita and Ivan Titov. Information-theoretic probing with minimum description length. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pp. 183–196, 2020.

Peter West, Ximing Lu, Nouha Dziri, Faeze Brahman, Linjie Li, Jena D. Hwang, Liwei Jiang, Jillian Fisher, Abhilasha Ravichander, Khyathi Chandu, Benjamin Newman, Pang Wei Koh, Allyson Ettinger, and Yejin Choi. The generative ai paradox: What it can create, it may not understand. In *ICLR*, 2024.

Yuexi Yang, Alyssa Wu, Ji Luo, Richeng Xuan, Zhichao Hu, Yuhong Liu, et al. Repoprobe: Benchmarking architecture-aware repository comprehension with checklists. *arXiv:2608.04783*, 2026.

Yanqi Zhang, Weizhe Hua, Zhuangzhuang Zhou, G. Edward Suh, and Christina Delimitrou. Sinan: Ml-based and qos-aware resource management for cloud microservices. In *ASPLOS*, 2021.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In *NeurIPS*, 2023.

Andy Zou, Long Phan, Sarah Chen, et al. Representation engineering: A top-down approach to ai transparency. *arXiv:2310.01405*, 2023.

A THE TOLERANCE QUANTITY AND THE PROBE

The capacity theory that supplies our labels also says *when* identifying the bottleneck should be easy. Bottleneck identification is robust to an error ϵ in the estimated relative demands exactly when ϵ is smaller than half the gap between the largest and second-largest relative demand: if the true gap is $\gamma = \tilde{d}_{(1)} - \tilde{d}_{(2)}$ and every estimate is within ϵ of its target, the estimated leader is at least $\tilde{d}_{(1)} - \epsilon$ and the estimated runner-up at most $\tilde{d}_{(2)} + \epsilon$, so the ordering survives whenever $\gamma > 2\epsilon$, and a γ below that admits an estimate that reorders them. Instances with a wide gap are therefore identifiable under sloppier estimation than instances with a narrow one. That is a statement about estimators in general; it makes a prediction about this model in particular.

The prediction holds. Pooled over the three scales this analysis was run on, with the probe layer fixed by validation data, 720 instance-scale observations give a Spearman correlation between the gap and per-instance correctness of $+0.111$ ($p = 0.003$). In a logistic model of correctness on the gap, the number of resources, and scale fixed effects, the gap carries $\beta = +0.898$ (se $= 0.315$, $p = 0.004$) and the number of resources carries $\beta = -0.392$ ($p = 10^{-4}$), so the two act separately and the

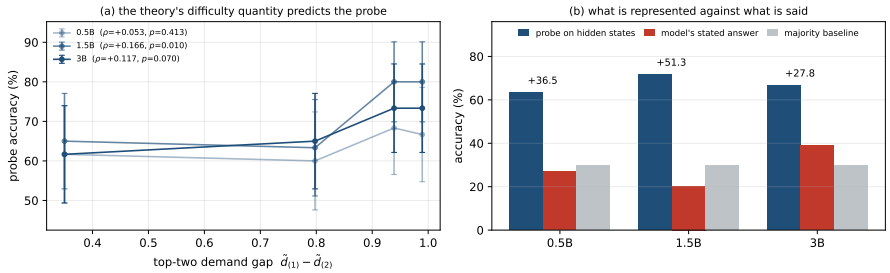


Figure 3: (a) Probe accuracy against the top-two demand gap, by quartile with binomial intervals, on the held-out 40% of instances and at the layer selected on the training split; the legend gives the per-scale Spearman correlation and its p-value. (b) What a linear read-out of the hidden states supports against what the model says, with the majority baseline, and the difference annotated. Both panels are produced by the script that writes the corresponding numbers in the text.

gap is not a proxy for how many options there are. By quartile the pooled accuracies are 0.628, 0.628, 0.739 and 0.733: a step between the lower and upper halves of the gap distribution, with no monotone trend. Per scale the effect is $\rho = +0.166$ ($p = 0.010$) at 1.5B, $+0.117$ ($p = 0.070$) at 3B and $+0.053$ at 0.5B on samples of 240; the pooled estimate is the one with the power to resolve it.

Two things follow. An analytic difficulty measure derived from the task's own theory grades the instances for a neural probe: the target is not only decodable, it is decodable more often where the theory says identification is easier, and the gap moves pooled accuracy by eleven points between the halves of its distribution. And two quantities computable from the code without running the model, the demand gap and the number of resources, together predict when a system-level read-out can be trusted.

B Which layer to read from

At the smallest scale the four targets place their read-out layers at two, six, fourteen and eighteen out of twenty-four, in increasing order of how much composition the property requires. Read as a practical instruction, this says that a probe deployed to extract a system property should not default to the last layer: for the bottleneck, the layer chosen on validation data sits in the second half of the network at every scale and the final layer is never the best. Read as an observation about the network, the ordering is monotone in composition depth, the behaviour one would expect if the property is assembled and not retrieved.

C Rendered instance

A rendered handler declares its pools, defines one helper per resource kind, and calls them in the specified order with the per-visit cost in a comment. The unindexed aggregate carries an explicit note, the runtime is identified by the pool constructor, and the admission pool is a semaphore whose size is stated. Costs vary over roughly two orders of magnitude across resource kinds, and visit counts over one.

D Surface features

The control uses the number of resources, the maximum per-visit cost and the index of the resource carrying it, the maximum visit count and its index, and the character and line counts of the rendered file. Its accuracy on the bottleneck target is close to the majority class, as intended: the target is a product of a cost and a count aggregated by the runtime, and no single one of these features determines it.