FORECASTING FUTURE BEHAVIOR AS A LEARNING TASK

Anonymous authors
Paper under double-blind review

ABSTRACT

Trust in an AI system is often anchored by explanations of how it works, which one then uses to forecast its behavior on new inputs. For large reasoning models (LRMs), this conventional route is particularly difficult to follow: explanation methods for single token generations do not naturally generalize to long trajectories, and the trajectories themselves are often not faithful when read as natural language. We propose an alternative that bypasses the explanation step: treat behavior forecasting as a learnable task and train *Behavior Forecasters* that operates on a single reasoning trajectory to make the same forecasts one would typically seek from an explanation. The forecaster's training data is obtained by querying the LRM with no human annotation, and its inference is done in a single forward pass. We instantiate this approach on two tasks: how likely the LRM is to repeat its answer on re-runs, and how removing parts of the input changes its answer. We evaluate this approach on both tasks across three diverse reasoning datasets and find that trained Behavior Forecasters are more accurate than GPT-5.4 and Claude Opus 4.6 reading the same trajectories as naive readers, at a small fraction of their inference cost. We find that fine-tuning the backbone end-to-end and initializing it from the target LRM are each necessary for strong performance. These results show that the reasoning trajectory carries information about the LRM's future behavior that goes beyond what naive reading conveys.

1 INTRODUCTION

Relying on an AI system requires anticipating how it will behave on new inputs (Lee & See, 2004; Hoff & Bashir, 2015; Jacovi et al., 2021; Zhou et al., 2026). Conventionally, this anticipation is anchored by explanations of the model's computation, widely treated as a device for forecasting[1] how it will behave on new inputs or in new scenarios (Hempel & Oppenheim, 1948; Douglas, 2009; Jacovi et al., 2021; Doshi-Velez & Kim, 2017; Lipton, 2018). This view is reflected in the way researchers in machine learning operationalize and evaluate explanations by the accuracy of the forecasts they enable (Miller, 2019). The most basic granularity at which such forecasts can drive deployment decisions is the individual input, for example how consistent the model's answer is across reruns or how it would change under counterfactual variants of the input (Ribeiro et al., 2016; Lundberg & Lee, 2017; Sundararajan et al., 2017; Alvarez-Melis & Jaakkola, 2018; Barredo Arrieta et al., 2020). These per-input behavior forecasts are the focus of this paper (Section 2.1).

Explanations that enable such forecasts are particularly difficult to obtain for large reasoning models (LRMs) that emit a long sequence of reasoning tokens before producing a final answer (e.g., O1 (OpenAI, 2024), R1 (DeepSeek-AI et al., 2025)) (Section 2.2). Methods developed for a single forward pass tell us about the generation of one token, and these methods do not naturally extend to the long reasoning trajectories these models emit. The reasoning tokens themselves, despite their natural-language appearance, are often not a faithful account of the computation that produced the answer (see Section 2.2.1 for supporting literature), so *naive reading* of the trajectory may lead to wrong forecasts. This leaves two unsatisfying options for behavior forecasting on LRMs: read the tokens naively and accept an unreliable picture, or disregard them. We see this as motivation to study behavior forecasting as a learnable task in its own right.

[1]We use *forecasting* rather than *predicting* to distinguish between the action of stating something about a model's future behavior and the operation of a neural network producing its output.

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.

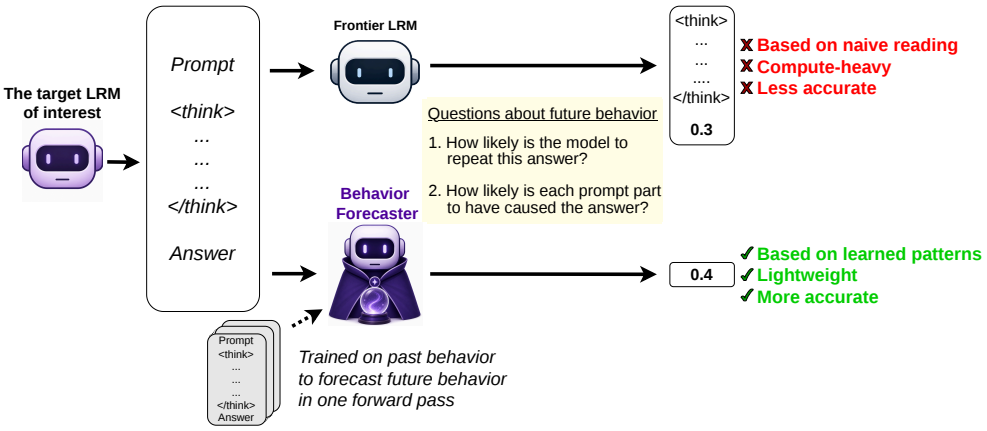

Figure 1: Behavior forecasting from a single reasoning trajectory. Given one observed trajectory of the target LRM (prompt, reasoning, answer), a trained Behavior Forecaster predicts a property of the LRM's future behavior in one forward pass: how likely the LRM is to repeat the answer (rerun consistency), or how likely each prompt part is to have caused the answer (counterfactual sensitivity). The Behavior Forecaster is trained on many such trajectories with behavioral labels obtained by querying the LRM. A frontier LLM reading the same trajectory naively relies on the text being faithful, does not scale as it requires a lot more compute, and yields less accurate predictions.

In this paper, we propose an alternative to making such forecasts via explanations: train an external model to forecast future behavior directly from a single reasoning trajectory (Section 3). We call this trained model a *Behavior Forecaster*: it relies on the tokens in the reasoning trajectory to carry information about the underlying computation, but does not require this information to be recoverable by reading the tokens as text (Levy et al., 2025b). We treat behavior forecasting as a supervised learning problem in which the training data is generated at scale without human annotation by querying the LRM itself, producing ground-truth behavioral labels.

Given one reasoning trajectory of the target LRM on the input of interest, a Behavior Forecaster predicts in one forward pass a statistic of the LRM's future behavior; we instantiate it on two such statistics: *rerun consistency*, computed over reruns on the same input, and *counterfactual sensitivity*, computed over perturbations of the input. We initialize the forecaster from the target LRM, attach a task-specific prediction head, and train them jointly.

We show that the Behavior Forecaster successfully learns both tasks across three reasoning datasets (Section 4). The trained Behavior Forecaster is more accurate than GPT-5.4 and Claude Opus 4.6 reading the same trajectory naively, while consuming less than $1/10{,}000$ their estimated compute. We also show that forecasters trained on two datasets transfer to a held-out third dataset with at most 600 fine-tuning steps, on both tasks, and that within a single dataset family they generalize to held-out variants without any fine-tuning. The counterfactual-sensitivity forecaster also transfers to predicting whether the LRM relied on a user-provided hint, which is a common way to evaluate the faithfulness of the reasoning text (Turpin et al., 2023; Chen et al., 2025; Chua & Evans, 2025; Marioriyad et al., 2025).

We then ablate the forecaster to identify which architectural and training choices matter (Section 5). Training only the prediction head, initializing from a randomized backbone, or removing the reasoning tokens from the input all reduce performance, and the best arrangement of prompt, reasoning, and answer in the input depends on the task.

More broadly, these findings suggest that the reasoning trajectory carries information about the LRM's future behavior beyond what naive reading conveys, inviting us to treat reasoning trajectories as data with learnable patterns rather than as natural language to be read.

2 BEHAVIOR FORECASTING FOR LRMS

2.1 FORECASTING BEHAVIOR FROM A SINGLE TRAJECTORY

We focus on forecasting properties of a model's behavior on a given input, such as how consistent the model's answer is across reruns or how sensitive its answer is to counterfactual perturbations of the input, from a single observed run (Ribeiro et al., 2016; Lundberg & Lee, 2017; Sundararajan et al., 2017). Working at this granularity is useful in practice for two reasons. First, per-input signals can directly drive per-query decisions such as abstention, flagging, or routing to human review, which are sometimes required from deployed AI systems (European Parliament and Council of the European Union, 2024). And second, larger behavioral testing, debugging, and auditing pipelines can aggregate these local measurements across inputs to characterize the system's overall behavior (Ribeiro et al., 2020; Bhatt et al., 2020). Both settings require low per-input cost, which rules out collecting multiple resamples online.

Formal setup. Let M be a stochastic LRM. On a prompt P, one execution of M produces an observed trajectory $\tau = (P, R, A)$, where R is the LRM's reasoning tokens and A is the final answer. A behavior forecasting problem specifies (i) a set of related prompts $\{P'\}$ around P (which may be P itself for reruns), and (ii) a target $b(M, P, A)$ defined as a statistic of M's answer distribution under these related prompts, conditioned on the observed answer A. The task is: given one observed trajectory $\tau \sim M(P)$, forecast $b(M, P, A)$ without running M on any related prompt P'.

We instantiate this definition on two tasks.

- **Rerun consistency:** the related prompt is P itself ($P' = P$); $b(M, P, A)$ is the probability that another run produces the same answer A.
- **Counterfactual sensitivity:** for each segment s in the set $S(P)$ of removable segments of P, the related prompt is the perturbation $P' = P_{-s}$ that removes s from P; $b(M, P, A)$ has one component per s, equal to how much removing s reduces the probability that M produces A.

In both, target values lie in $[0, 1]$ and are derived from M's induced answer distribution. The task is therefore regression against probability-derived targets.

2.2 WHY WE NEED A NEW APPROACH FOR LRMS

There are two natural approaches to behavior forecasting on an LRM, and both run into limitations that motivate a different approach: reading the reasoning trajectory as an explanation of the LRM's computation (Section 2.2.1), and applying standard behavior forecasting tools (Section 2.2.2).

2.2.1 NAIVE READING OF THE REASONING TRAJECTORY IS OFTEN UNRELIABLE

A behavior forecast asks, for a given input, how confidently the LRM arrived at its answer (rerun consistency) and which input clauses it actually used (counterfactual sensitivity).

The reasoning trajectory is the natural candidate source for these outcomes in an LRM (Korbak et al., 2025; Guan et al., 2025). If the trajectory's natural-language surface faithfully tracked the LRM's computation, recovering each outcome would reduce to a classical natural language processing task on that surface text: confidence detection on the trajectory for rerun consistency, and tracing how the trajectory's reasoning steps connect input segments to the final answer for counterfactual sensitivity. We refer to this approach as *naive reading*: interpreting the trajectory through generic English semantics, without learned familiarity with the patterns the target LRM uses in its reasoning tokens. Both human readers and other LRMs read naively in this sense. But the natural-language surface of the trajectory is often not a faithful account of the computation behind the answer.

Omission. LRMs omit factors from the natural-language reasoning that shape the final answer. Planted cues such as biased answer positions, user hints, and stereotype signals shift predictions without appearing in the natural-language reasoning (Turpin et al., 2023; Yee et al., 2024; Chua & Evans, 2025; Arcuschin et al., 2025; Marioriyad et al., 2025; Lindsey et al., 2025; Mirtaheri & Belkin, 2026; Young, 2026; Bachmann et al., 2026).

Semantic mismatch. Even the steps that do appear in the natural-language reasoning can diverge from the computation the LRM does to produce the answer. The surface natural language can be illegible to humans and AI naive readers while the LRM still arrives at correct answers (Jose, 2025), injecting changes into the natural-language reasoning often leaves the final answer unchanged (Lanham et al., 2023; Paul et al., 2024), and causal mediation analysis shows that LRMs do not use their intermediate steps in the way a human reader would predict (Levy et al., 2025a).

Chen et al. (2025) and Han et al. (2026) also show that faithfulness is not guaranteed to improve with model scale or more performant models. Naive reading is therefore unlikely to reliably deliver the outcomes we are after, motivating a learned approach to behavior forecasting.

2.2.2 Standard behavior forecasting methods are not applicable

Standard behavior forecasting methods are unlikely to be useful for forecasting LRMs behavior either. The long, stochastic trajectories LRMs produce make these methods either prohibitively expensive or blind to the trajectory itself.

Resampling methods are prohibitively expensive. Resampling underlies existing methods for forecasting future model behavior (Jones et al., 2025; Serrano et al., 2026). It is straightforward in principle but impractical in deployment. Estimating either rerun consistency from N reruns or counterfactual sensitivity from N perturbations multiplies the already high cost of one long reasoning run by N.

Single-location probes miss the trajectory. Standard attribution and uncertainty signals all explain the generation of a single fixed position in the model's output. Gradient-based input attributions like integrated gradients (Sundararajan et al., 2017) require a differentiable path from input to output, but the LRM computational graph involves discrete sampling steps that break the gradient flow; in LLMs the common substitute is the attention from the final-answer tokens back to the prompt (Chuang et al., 2024; Jain & Wallace, 2019). Token-probability uncertainty signals (Manakul et al., 2023; Farquhar et al., 2024) score the final-answer logits in the same way. None of these registers what happens inside the reasoning trajectory, where much of the LRM's computation actually unfolds.

3 Method

Section 2 showed that standard tools for behavior forecasting either break down or become prohibitively expensive on LRMs, and that naive reading of the reasoning trajectory is unlikely to faithfully answer behavior forecasting questions. We propose an alternative: we train a model to forecast behavioral properties of the LRM from a single observed reasoning trajectory. We call this trained model a *Behavior Forecaster* (or simply *forecaster*).

3.1 Behavior forecasting as a learning task

We treat behavior forecasting as a supervised learning task: from one observed reasoning trajectory, predict a target behavioral statistic of the LRM. This relies on the trajectory carrying information about the underlying computation, but does not require this information to be readable as natural-language text (Levy et al., 2025b). Framing it this way lets behavior forecasting benefit from data and compute (Sutton, 2019).

To generate training data, for each prompt P we first obtain one observed trajectory $\tau \sim M(P)$ with answer A. We then run the LRM on the related executions defined by the forecasting problem and use those executions to estimate the target statistic $b(M, P, A)$.

We train a Behavior Forecaster to map the observed trajectory to the target statistic. The expensive executions are needed only for data generation. The cost gap relative to resampling at inference is large: estimating the rerun-consistency label for one input from a fresh resample requires additional trajectories from the LRM, each up to thousands of tokens generated autoregressively, while the Behavior Forecaster needs one forward pass over the already-observed trajectory, orders of magnitude less compute and far lower latency at deployment.

3.2 TASK-SPECIFIC SUPERVISION

We estimate the answer probabilities entering $b(M, P, A)$ empirically: for each prompt P we run the LRM 10 times on P, and for counterfactual sensitivity we additionally run the LRM 10 times on each perturbed version of P, all with the LRM's recommended sampling settings (temperature 0.6, top-p 0.95). We extract a final answer from each trajectory and treat two runs as producing the same answer when their extracted answers match, rather than requiring identical output text. A sample is labeled only if at least 5 of its runs yields an extractable answer; full details are in Appendix A.

Rerun consistency. For each run on a prompt, the run-level target is the fraction of the prompt's other valid runs that produce the same extracted answer; this estimates the chance that a fresh run on the same prompt would reproduce the answer.

Counterfactual sensitivity. For each prompt P, we generate one perturbed prompt P_{-s} per segment $s \in S(P)$; the segmentation scheme is dataset-specific. We retain only prompts whose answer A appears in at least 70% of the 10 original runs, so labels reflect omission effects rather than base sampling noise. The segment-level label measures how much removing s reduces the rate at which the LRM produces A, normalized to $[0, 1]$ so a value of 1 means removing s always changes the answer. Every token in segment s shares this score; the per-segment prediction aggregates the forecaster's per-token outputs within the segment.

3.3 BEHAVIOR FORECASTER ARCHITECTURE

The Behavior Forecaster's backbone shares its architecture with the target LRM and initializes from the LRM's weights, so it starts with representations already attuned to the reasoning tokens. We use the same kind of backbone for both tasks, training a separate copy per task and pairing each with a task-specific input arrangement and head.

Counterfactual sensitivity: prompt-echo arrangement. For counterfactual sensitivity, the task is one score per segment of the prompt; we aggregate per-token outputs into a per-segment score (Appendix B.2). The natural placement for these per-token predictions is at the prompt positions. But the backbone is a standard autoregressive decoder: a prediction head at a prompt-position token attends only to earlier tokens, which excludes the reasoning R and the answer A. Without those, the head has none of the trajectory's evidence about how the LRM used each prompt segment. We therefore append a copy of the prompt after the observed trajectory and produce per-token predictions at the echoed-prompt positions. A per-token MLP head produces one prediction at each echoed-prompt position, and the per-segment score in $[0, 1]$ is obtained by aggregating these within each segment.

Rerun consistency: cross-attention pooling. For rerun consistency, the target is one score per trajectory, with no prompt-side structure to attribute to. Trajectories vary in length, so we pool the backbone's hidden states into a fixed-size representation that a simple MLP head can map to one scalar in $[0, 1]$. We pool with cross-attention over a fixed set of learned query vectors; the queries attend over all trajectory positions, so the pooled representation can draw on evidence from anywhere in the trajectory. Architectural details (number of query vectors, head sizes, etc.) are in Appendix B.3.

4 BEHAVIOR FORECASTING IS LEARNABLE

We evaluate Behavior Forecasters on both tasks across three reasoning datasets. A Behavior Forecaster trained on the pooled three datasets is more accurate than both strong naive readers and the standard single-location probes, at a small fraction of the readers' inference cost (Section 4.2); the same comparison holds on a second target LRM, Qwen3.5-2B. It also generalizes beyond its training data (Section 4.3): within a single dataset family the forecaster generalizes zero-shot to a held-out variant, and across datasets a forecaster trained on two of the three adapts to the held-out third with target-side fine-tuning. Finally, we test whether a forecaster trained for counterfactual sensitivity can initialize a new Behavior Forecaster for a related hint-sensitivity target (Section 4.4).

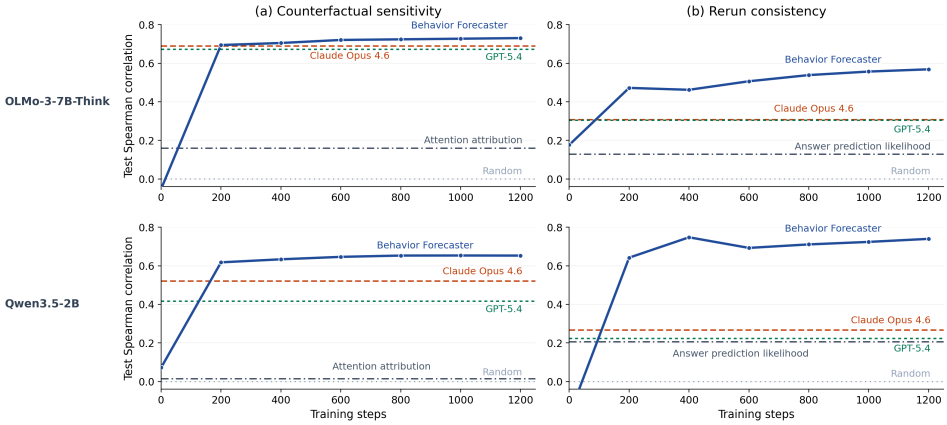

Figure 2: A trained Behavior Forecaster is more accurate than two frontier naive readers and the standard single-location probes, at a small fraction of the readers' inference cost. Rows show target LRM: OLMo-3-7B-Think (top; FEVEROUS, RuleTaker, TreeCut) and Qwen3.5-2B (bottom; FEVEROUS, RuleTaker). For Qwen3.5-2B, TreeCut is omitted because its trajectories were too long for our available generation resources. Each panel plots the forecaster's test Spearman correlation over training steps; horizontal lines mark the two frontier naive readers (GPT-5.4, Claude Opus 4.6), the task-specific single-location probe on the target LRM (attention attribution for counterfactual sensitivity, answer prediction likelihood for rerun consistency), and a random-prediction baseline. Detailed metrics, probe results, and uncertainty estimates are in Appendices C and D.

4.1 Experimental setup

Target LRMs and datasets. Our main target LRM is OLMo-3-7B-Think (Team Olmo et al., 2025), whose training data and training pipeline are fully released; this lets us build evaluation sets from data we can verify the LRM was not trained on, and lets follow-up work that depends on knowing the LRM's training data or pipeline build directly on the same target. For the main learnability comparison (Section 4.2) we additionally evaluate on Qwen3.5-2B (Qwen Team, 2026) as a second target LRM, to check that the gap against naive readers is not specific to OLMo; all other experiments use OLMo only. We use three reasoning datasets that cover substantively different tasks: mathematical word problems (TreeCut (Ouyang, 2025)), factual claims grounded in Wikipedia (FEVEROUS (Aly et al., 2021)), and synthetic logical reasoning (RuleTaker (Clark et al., 2020)). Full data construction and splits details, and per-dataset sample counts are in Appendix A.1.

Training. We initialize the Behavior Forecaster from the target LRM's weights and train it end-to-end. Training settings, hardware, architectural hyperparameters, and the transfer fine-tuning budget are in Appendix B. All results in the paper, except for the main learnability figure (Figure 2), are evaluated on a checkpoint selected by validation loss; the main figure plots the test correlation across training steps directly.

Naive reader baselines. We compare against frontier LRMs (Claude Opus 4.6 and GPT-5.4) used as strong naive readers that receive the reasoning trajectory as input and predict the target property from its surface meaning. Maslej et al. (2025) report that LRMs perform at or above human level on the kind of basic text-understanding tasks naive reading reduces to. The Behavior Forecaster and naive readers differ in their exposure to the target LRM's reasoning trajectories: the Behavior Forecaster is trained on those trajectories and so can learn the patterns the LRM uses to encode information in its reasoning tokens. For each task we tested three prompt variants on a small pilot slice and selected, per reader, the variant with the strongest pilot correlation. Full prompts and selected variants are in Appendix C.1.

Metric. We report Spearman correlation in the main text, which depends only on the rank order of predictions and is insensitive to compression, shift, or any monotone rescaling of the predicted scale. Additional results using Pearson correlation are in the appendix.

4.2 Trained Behavior Forecasters are more accurate than untrained baselines

Naive readers. The trained Behavior Forecaster is more accurate than strong naive readers (GPT-5.4 and Claude Opus 4.6) reading the same trajectories, at a small fraction of their inference cost, on both target LRMs (Figure 2). On Qwen3.5-2B the Behavior Forecaster reaches Spearman 0.653 on counterfactual sensitivity (vs 0.417 for GPT-5.4 and 0.522 for Claude Opus 4.6) and 0.740 on rerun consistency (vs 0.224 and 0.267 respectively). We also find that a same-LRM reader control, in which OLMo-3-7B-Think reads its own observed trajectory, is substantially weaker than both the trained Behavior Forecaster and the frontier naive readers (Appendix C.3).

Single-location probes. The Behavior Forecaster also outperforms the single-location probes from Section 2.2.2 applied to the target LRM: final-answer attention to candidate input segments for counterfactual sensitivity (Chuang et al., 2024; Jain & Wallace, 2019) and final-answer log-likelihood for rerun consistency (Manakul et al., 2023; Farquhar et al., 2024). Both clear the random-prediction baseline but remain too weak to be useful in practice (on the OLMo target, 0.160 and 0.129 respectively; on Qwen3.5-2B, 0.016 and 0.206), well below the Behavior Forecaster on both targets and below the naive readers (Appendix C.2).

4.3 Generalization to new data

We test how a trained Behavior Forecaster generalizes to data it was not trained on, in two settings.

Same-family check. We train a Behavior Forecaster on three subsets of the FEVEROUS dataset, holding out the numerical subset entirely, and then evaluate it directly on FEVEROUS numerical examples with no fine-tuning. The Behavior Forecaster remains ahead of both frontier naive readers on this held-out subset for both tasks (Table 1).

Transfer across datasets. We hold out one of our three datasets (FEVEROUS, RuleTaker, or TreeCut), train a source Behavior Forecaster on the remaining two, and fine-tune it on the held-out target for at most $\sim$600 steps. Transfer improves performance on every held-out target for both tasks, though rerun consistency remains weaker on TreeCut than on FEVEROUS or RuleTaker after adaptation (Table 2).

Table 1: Same-family transfer within FEVEROUS, for the OLMo-3-7B-Think Behavior Forecaster. The Behavior Forecaster is trained on the non-numerical FEVEROUS subsets and evaluated on FEVEROUS numerical without target-side fine-tuning.

	Counterfactual sensitivity		*Rerun consistency*	
System	Spearman	Pearson	Spearman	Pearson
Behavior Forecaster	**0.565**	**0.730**	**0.311**	**0.350**
GPT-5.4	0.322	0.406	0.223	0.226
Claude Opus 4.6	0.463	0.622	0.173	0.193

4.4 Transfer across behavioral targets

We also test transfer to a new behavioral target. Starting from the counterfactual-sensitivity Behavior Forecaster, we adapt to hint-sensitivity forecasting: given a single trajectory produced with an explicit answer hint, predict the probability that the target LRM's no-hint answer would differ from the observed hinted answer. The test split holds out two strong hint templates and contains 610 trajectories from 122 FEVEROUS source examples. As shown in Table 3, the transferred forecaster obtains the best rank correlation on this surface, outperforming both direct hint-sensitivity training from the target LRM and frontier naive readers that see the same hinted prompt, observed answer, and full completion.

Table 2: Transfer to held-out datasets, for the OLMo-3-7B-Think Behavior Forecaster. The *source* Behavior Forecaster is trained on the two other datasets and evaluated on the held-out target without target-side training; the *adapted* Behavior Forecaster is the fine-tuned checkpoint. Src and Adp denote source and adapted.

	Counterfactual sensitivity				*Rerun consistency*			
	Spearman		Pearson		Spearman		Pearson	
Held-out target	Src	Adp	Src	Adp	Src	Adp	Src	Adp
FEVEROUS	0.368	0.641	0.319	0.757	0.242	0.539	0.233	0.471
RuleTaker	0.446	0.661	0.692	0.962	0.005	0.438	0.238	0.854
TreeCut	0.650	0.846	0.584	0.938	0.118	0.193	0.157	0.295

Table 3: Transfer to hint-sensitivity forecasting on held-out FEVEROUS hint templates, for the OLMo-3-7B-Think Behavior Forecaster. The direct Behavior Forecaster is trained on the hint-sensitivity surface from the target LRM initialization; the transferred Behavior Forecaster initializes from a counterfactual-sensitivity model and is then adapted to the same hint-sensitivity surface. All methods are evaluated on the same 610 held-out trajectories.

System	**Spearman**	**Pearson**
Behavior Forecaster (transferred)	**0.800**	**0.817**
Behavior Forecaster (direct)	0.732	0.814
Claude Opus 4.6 reader	0.768	0.779
GPT-5.4 reader	0.668	0.680

5 Ablating the Behavior Forecaster

We now ask which parts of the Behavior Forecaster make its learning possible: the input arrangement, the pretrained initialization, and end-to-end backbone training. We vary the input arrangement (which of prompt P, reasoning R, and answer A are present and in what order), whether the backbone is initialized from the target LRM or randomly initialized with the same architecture, and whether the backbone is frozen while only the prediction head is trained. Table 4 reports the resulting test correlations.

Table 4: Behavior Forecaster variants across input arrangement, initialization, and backbone training. Test Spearman and Pearson for OLMo-backed ablations. P, R, and A denote prompt, reasoning tokens, and final answer. "OLMo init" is short for OLMo-3-7B-Think initialization. The bold row in each task is the main Behavior Forecaster used in Section 4.

Counterfactual sensitivity			*Rerun consistency*		
Variant	Spearman	Pearson	**Variant**	Spearman	Pearson
OLMo init, P–R–A–P	**0.723**	**0.925**	OLMo init, P–R–A	**0.568**	**0.649**
OLMo init, P–A–P	0.714	0.909	OLMo init, P–R–A–P	0.554	0.653
OLMo init, P–R–A	0.578	0.620	OLMo init, P–A	0.316	0.251
OLMo init, prompt only	0.567	0.602	OLMo init, prompt only	0.348	0.279
OLMo arch., random init	0.534	0.553	OLMo arch., random init	0.477	0.504
Frozen OLMo, head-only	0.245	0.251	Frozen OLMo, head-only	0.475	0.552
Random predictions	0.000	0.000	Random predictions	0.000	0.000

The useful arrangement is task-specific. Counterfactual sensitivity is strongest with the prompt-echo P–R–A–P arrangement, while rerun consistency is strongest with the plain P–R–A arrangement; swapping one task's preferred arrangement for the other's hurts performance on both. Remov-

ing the reasoning tokens hurts both tasks substantially, so the reasoning tokens themselves, not just the prompt and answer, carry the signal the Behavior Forecaster uses.

Initialization from the target LRM helps. On both tasks, initializing from OLMo-3-7B-Think is stronger than a randomly initialized backbone with the same architecture. The gap is larger for counterfactual sensitivity than for rerun consistency. This suggests that pretrained representations contribute beyond what the architecture alone provides.

An external trained model beats attaching only a head to the LRM. Attaching only a prediction head to the original LRM, i.e., freezing the OLMo backbone and training only the head, performs well below the fully fine-tuned Behavior Forecaster (a separately trained copy of the backbone with its own head). The original LRM's hidden states already contain useful signal, but adapting a separate model end-to-end on top of those states substantially improves extraction of that signal.

6 Related work

Probing internal states for current-run properties. A line of work trains probes on the internal states of language models to predict properties of the model's behavior on the current input. Such methods predict whether a generated statement is factual (Azaria & Mitchell, 2023; Burns et al., 2022), whether the model will hallucinate (Kossen et al., 2024; Kadavath et al., 2022), whether the model will refuse a request (Arditi et al., 2024), and broader behavioral attributes such as honesty or power-seeking tendencies (Zou et al., 2023); earlier probing work also recovers behaviorally relevant signals from the residual stream more broadly (Alain & Bengio, 2017; Belrose et al., 2023; Karvonen et al., 2025). A growing line of work applies the same idea inside LRM trajectories: probes on hidden states at intermediate steps predict whether the current intermediate answer is correct, supporting early exit (Zhang et al., 2025); attention probes decode the same run's eventual final answer well before it appears in the generated text (Boppana et al., 2026); and mechanistic analysis finds that in-context concepts are progressively refined across the trajectory, with steering experiments confirming that this refinement causally drives accuracy (Kharlapenko et al., 2026). All of these targets concern the current run: properties of the trajectory or its eventual answer. Our work targets the LRM's future behavior instead: rerun consistency and counterfactual sensitivity describe the distribution over future executions related to the input, not properties of the realized one.

Resampling-based analysis of LRM trajectories. Prior work estimates the causal effect of individual reasoning steps on the answer distribution by repeatedly resampling and intervening on the trajectory (Bogdan et al., 2025; Macar et al., 2025). For deployment-time per-query decisions, the per-input cost of these methods is prohibitive: dozens to hundreds of fresh trajectories, each thousands of forward passes long. We treat the resampling cost as a one-time investment to label training data, and train a Behavior Forecaster that approximates the same statistics from a single observed trajectory.

7 Conclusion

Trust in an AI system rests on the ability to forecast how it will behave, which is particularly challenging for LRMs (Section 2.2). We formulated behavior forecasting as its own learning task and introduced *Behavior Forecasters*, models that forecast behavioral properties of the target LRM from a single observed reasoning trajectory. We trained Behavior Forecasters on rerun consistency and counterfactual sensitivity, with labels generated by querying the LRM itself.

The task is learnable: a trained Behavior Forecaster outperforms strong naive readers and single-location probes, generalizes across datasets with fine-tuning and across same-family variants without it, and relies on initialization from the target LRM, end-to-end fine-tuning, and a task-specific input arrangement.

Our results show that the trajectory holds more information about the LRM's future behavior than on surface level. We hope this motivates further study of behavior forecasting as a learning task: any LRM behavioral property that can be automatically labeled is a candidate target for low-cost forecasting at deployment scale.

AI USE STATEMENT

We used generative AI to assist with coding and to polish parts of this paper. LRMs generated reasoning trajectories used for behavioral supervision (Section 3.2). The authors take responsibility for the manuscript, code, and results.

REPRODUCIBILITY STATEMENT

The accompanying repository provides code and experiment-specific instructions for reproducing the OLMo-3-7B-Think Behavior Forecaster training and evaluation procedures from LRM reasoning trajectories and their corresponding behavioral labels, using the specified model weights, data splits, and training settings. Section 3 and Appendices A and B describe the task definitions, supervision construction, and training procedures.

REFERENCES

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. 2017.

David Alvarez-Melis and Tommi S Jaakkola. On the robustness of interpretability methods. In *ICML Workshop on Human Interpretability in Machine Learning*, 2018. URL https://arxiv.org/abs/1806.08049.

Rami Aly, Zhijiang Guo, Michael Schlichtkrull, James Thorne, Andreas Vlachos, Christos Christodoulopoulos, Oana Cocarascu, and Arpit Mittal. FEVEROUS: Fact extraction and VERification over unstructured and structured information. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021. URL https://arxiv.org/abs/2106.05707.

Iván Arcuschin, Jett Janiak, Robert Krzyzanowski, Senthooran Rajamanoharan, Neel Nanda, and Arthur Conmy. Chain-of-thought reasoning in the wild is not always faithful. *arXiv preprint arXiv:2503.08679*, 2025.

Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction, 2024. URL https://arxiv.org/abs/2406.11717.

Amos Azaria and Tom Mitchell. The internal state of an llm knows when it's lying, 2023. URL https://arxiv.org/abs/2304.13734.

Gregor Bachmann, Yichen Jiang, Seyed Mohsen Moosavi Dezfooli, and Moin Nabi. The potential of cot for reasoning: A closer look at trace dynamics, 2026. URL https://arxiv.org/abs/2602.14903.

Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115, 2020.

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

Umang Bhatt, Alice Xiang, Shubham Sharma, Adrian Weller, Ankur Taly, Yunhan Jia, Joydeep Ghosh, Ruchir Puri, José M F Moura, and Peter Eckersley. Explainable machine learning in deployment. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency (FAT*)*, pp. 648–657, 2020.

Paul C. Bogdan, Uzay Macar, Neel Nanda, and Arthur Conmy. Thought anchors: Which llm reasoning steps matter?, 2025. URL https://arxiv.org/abs/2506.19143.

Siddharth Boppana, Annabel Ma, Max Loeffler, Raphael Sarfati, Eric Bigelow, Atticus Geiger, Owen Lewis, and Jack Merullo. Reasoning theater: Disentangling model beliefs from chain-of-thought, 2026. URL https://arxiv.org/abs/2603.05488.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision. *arXiv preprint arXiv:2212.03827*, 2022. URL https://arxiv.org/abs/2212.03827.

Yanda Chen, Joe Benton, Ansh Radhakrishnan, Jonathan Uesato, Carson Denison, John Schulman, Arushi Somani, Peter Hase, Misha Wagner, Fabien Roger, Vlad Mikulik, Samuel R. Bowman, Jan Leike, Jared Kaplan, and Ethan Perez. Reasoning models don't always say what they think, 2025. URL https://arxiv.org/abs/2505.05410.

James Chua and Owain Evans. Are deepseek r1 and other reasoning models more faithful?, 2025. URL https://arxiv.org/abs/2501.08156.

Yung-Sung Chuang, Linlu Qiu, Cheng-Yu Hsieh, Ranjay Krishna, Yoon Kim, and James Glass. Lookback lens: Detecting and mitigating contextual hallucinations in large language models using only attention maps, 2024. URL https://arxiv.org/abs/2407.07071. EMNLP 2024 main conference long paper.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. Transformers as soft reasoners over language. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI)*, 2020. URL https://arxiv.org/abs/2002.05867.

DeepSeek-AI, Daya Guo, Dejian Yang, and Haowei Zhang et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. URL https://arxiv.org/abs/2501.12948.

Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017. URL https://arxiv.org/abs/1702.08608.

Heather Douglas. Reintroducing prediction to explanation. *Philosophy of Science*, 76:444 – 463, 2009. doi: 10.1086/648111. URL https://api.semanticscholar.org/CorpusID:120278496.

European Parliament and Council of the European Union. Regulation (eu) 2024/1689 laying down harmonised rules on artificial intelligence (artificial intelligence act), 2024. URL https://eur-lex.europa.eu/eli/reg/2024/1689/oj. Article 14: Human oversight. Official Journal of the European Union, 12 July 2024.

Sebastian Farquhar, Jannik Kossen, Lorenz Kuhn, and Yarin Gal. Detecting hallucinations in large language models using semantic entropy. *Nature*, 630:625–630, 2024. doi: 10.1038/s41586-024-07421-0. URL https://www.nature.com/articles/s41586-024-07421-0.

Melody Y Guan, Miles Wang, Micah Carroll, Zehao Dou, Annie Y Wei, Marcus Williams, Benjamin Arnav, Joost Huizinga, Ian Kivlichan, Mia Glaese, Jakub Pachocki, and Bowen Baker. Monitoring monitorability. *arXiv preprint arXiv:2512.18311*, 2025.

Yunseok Han, Yejoon Lee, and Jaeyoung Do. RFEval: Benchmarking reasoning faithfulness under counterfactual reasoning intervention in large reasoning models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=2Gc8aj0afg.

Carl G. Hempel and Paul Oppenheim. Studies in the logic of explanation. *Philosophy of Science*, 15:135 – 175, 1948. doi: 10.1086/286983. URL https://api.semanticscholar.org/CorpusID:16924146.

Kevin Anthony Hoff and Masooda Bashir. Trust in automation: Integrating empirical evidence on factors that influence trust. *Human Factors*, 57(3):407–434, 2015. URL https://doi.org/10.1177/0018720814547570.

Alon Jacovi, Ana Marasović, Tim Miller, and Yoav Goldberg. Formalizing trust in artificial intelligence: Prerequisites, causes and goals of human trust in ai. In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pp. 624–635, 2021.

Sarthak Jain and Byron C. Wallace. Attention is not explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 3543–3556, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1357. URL https://aclanthology.org/N19-1357/.

Erik Jones, Meg Tong, Jesse Mu, Mohammed Mahfoud, Jan Leike, Roger Grosse, Jared Kaplan, William Fithian, Ethan Perez, and Mrinank Sharma. Forecasting rare language model behaviors. *arXiv preprint arXiv:2502.16797*, 2025.

Arun Jose. Reasoning models sometimes output illegible chains of thought, 2025. URL https://arxiv.org/abs/2510.27338.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Adam Karvonen, James Chua, Clément Dumas, Kit Fraser-Taliente, Subhash Kantamneni, Julian Minder, Euan Ong, Arnab Sen Sharma, Daniel Wen, Owain Evans, and Samuel Marks. Activation oracles: Training and evaluating llms as general-purpose activation explainers, 2025. URL https://arxiv.org/abs/2512.15674.

Dmitrii Kharlapenko, Alessandro Stolfo, Arthur Conmy, Mrinmaya Sachan, and Zhijing Jin. Fluid representations in reasoning models, 2026. URL https://arxiv.org/abs/2602.04843.

Tomek Korbak, Mikita Balesni, Elizabeth Barnes, Yoshua Bengio, Joe Benton, Joseph Bloom, Mark Chen, Alan Cooney, Allan Dafoe, Anca Dragan, Scott Emmons, Owain Evans, David Farhi, Ryan Greenblatt, Dan Hendrycks, Marius Hobbhahn, Evan Hubinger, Geoffrey Irving, Erik Jenner, Daniel Kokotajlo, Victoria Krakovna, Shane Legg, David Lindner, David Luan, Aleksander Mądry, Julian Michael, Neel Nanda, Dave Orr, Jakub Pachocki, Ethan Perez, Mary Phuong, Fabien Roger, Joshua Saxe, Buck Shlegeris, Martín Soto, Eric Steinberger, Jasmine Wang, Wojciech Zaremba, Bowen Baker, Rohin Shah, and Vlad Mikulik. Chain of thought monitorability: A new and fragile opportunity for ai safety, 2025. URL https://arxiv.org/abs/2507.11473.

Jannik Kossen, Jiatong Han, Muhammed Razzak, Lisa Schut, Shreshth Malik, and Yarin Gal. Semantic entropy probes: Robust and cheap hallucination detection in llms, 2024. URL https://arxiv.org/abs/2406.15927.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, Saurav Kadavath, Shannon Yang, Thomas Henighan, Timothy Maxwell, Timothy Telleen-Lawton, Tristan Hume, Zac Hatfield-Dodds, Jared Kaplan, Jan Brauner, Samuel R. Bowman, and Ethan Perez. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*, 2023. URL https://arxiv.org/abs/2307.13702.

John D. Lee and Katrina A. See. Trust in automation: Designing for appropriate reliance. *Human Factors*, 46(1):50–80, 2004. URL https://doi.org/10.1518/hfes.46.1.50.30392.

Mosh Levy, Zohar Elyoseph, and Yoav Goldberg. Humans perceive wrong narratives from ai reasoning texts, 2025a. URL https://arxiv.org/abs/2508.16599.

Mosh Levy, Zohar Elyoseph, Shauli Ravfogel, and Yoav Goldberg. State over tokens: Characterizing the role of reasoning tokens. *arXiv preprint arXiv:2512.12777*, 2025b.

Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L. Turner, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. On the biology of a large language model. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/biology.html.

Zachary C Lipton. The mythos of model interpretability. *Queue*, 16(3):31–57, 2018.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Uzay Macar, Paul C. Bogdan, Senthooran Rajamanoharan, and Neel Nanda. Thought branches: Interpreting llm reasoning requires resampling, 2025. URL https://arxiv.org/abs/2510.27484.

Potsawee Manakul, Adian Liusie, and Mark Gales. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 9004–9017, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.557. URL https://aclanthology.org/2023.emnlp-main.557/.

Arash Marioriyad, Shaygan Adim, Nima Alighardashi, Mahdieh Soleymani Banghshah, and Mohammad Hossein Rohban. Unspoken hints: Accuracy without acknowledgement in llm reasoning, 2025. URL https://arxiv.org/abs/2509.26041.

Nestor Maslej, Loredana Fattorini, Raymond Perrault, Yolanda Gil, Vanessa Parli, Njenga Kariuki, Emily Capstick, Anka Reuel, Erik Brynjolfsson, John Etchemendy, Katrina Ligett, Terah Lyons, James Manyika, Juan Carlos Niebles, Yoav Shoham, Russell Wald, Tobi Walsh, Armin Hamrah, Lapo Santarlasci, Julia Betts Lotufo, Alexandra Rome, Andrew Shi, and Sukrut Oak. Artificial intelligence index report 2025, 2025. URL https://arxiv.org/abs/2504.07139.

Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019. doi: 10.1016/j.artint.2018.07.007. URL https://doi.org/10.1016/j.artint.2018.07.007.

Parsa Mirtaheri and Mikhail Belkin. Catching rationalization in the act: Detecting motivated reasoning before and after CoT via activation probing, 2026. URL https://arxiv.org/abs/2603.17199.

OpenAI. Openai o1 system card, 2024. URL https://openai.com/index/openai-o1-system-card/. Accessed: 2024.

Jialin Ouyang. Treecut: A synthetic unanswerable math word problem dataset for llm hallucination evaluation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 1073–1085, 2025.

Debjit Paul, Robert West, Antoine Bosselut, and Boi Faltings. Making reasoning matter: Measuring and improving faithfulness of chain-of-thought reasoning, 2024. URL https://arxiv.org/abs/2402.13950.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144, 2016.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4902–4912, 2020. doi: 10.18653/v1/2020.acl-main.442.

Alex Serrano, Wen Xing, David Lindner, and Erik Jenner. Frontier models can take actions at low probabilities. *arXiv preprint arXiv:2603.02202*, 2026.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 3319–3328, 2017.

Rich Sutton. The bitter lesson. http://www.incompleteideas.net/IncIdeas/BitterLesson.html, 2019.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, Pradeep Dasigi, Robert Berry, Saumya Malik, Saurabh Shah, Scott Geng, Shane Arora, Shashank Gupta, Taira Anderson, Teng Xiao, Tyler Murray, Tyler Romero, Victoria Graf, Akari Asai, Akshita Bhagia, Alexander Wettig, Alisa Liu, Aman Rangapur, Chloe Anastasiades, Costa Huang, Dustin Schwenk, Harsh Trivedi, Ian Magnusson, Jaron Lochner, Jiacheng Liu, Lester James V. Miranda, Maarten Sap, Malia Morgan, Michael Schmitz, Michal Guerquin, Michael Wilson, Regan Huff, Ronan Le Bras, Rui Xin, Rulin Shao, Sam Skjonsberg, Shannon Zejiang Shen, Shuyue Stella Li, Tucker Wilde, Valentina Pyatkin, Will Merrill, Yapei Chang, Yuling Gu, Zhiyuan Zeng, Ashish Sabharwal, Luke Zettlemoyer, Pang Wei Koh, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. Olmo 3, 2025. URL https://arxiv.org/abs/2512.13961.

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*, volume 36, 2023. URL https://arxiv.org/abs/2305.04388. Paper examining the faithfulness of chain-of-thought explanations in language models.

Evelyn Yee, Alice Li, Chenyu Tang, Yeon Ho Jung, Ramamohan Paturi, and Leon Bergen. Dissociation of faithful and unfaithful reasoning in llms. *arXiv preprint arXiv:2405.15092*, 2024.

Richard J. Young. Lie to me: How faithful is chain-of-thought reasoning in reasoning models?, 2026. URL https://arxiv.org/abs/2603.22582.

Anqi Zhang, Yulin Chen, Jane Pan, Chen Zhao, Aurojit Panda, Jinyang Li, and He He. Reasoning models know when they're right: Probing hidden states for self-verification, 2025. URL https://arxiv.org/abs/2504.05419.

Lexin Zhou, Pablo A.M. Casares, Fernando Martínez-Plumed, John Burden, Ryan Burnell, Lucy Cheke, Cèsar Ferri, Alexandru Marcoci, Behzad Mehrbakhsh, Yael Moros-Daval, Seán Ó hÉigeartaigh, Danaja Rutar, Wout Schellaert, Konstantinos Voudouris, and José Hernández-Orallo. Predictable artificial intelligence. *Artificial Intelligence*, 353:104491, 2026. ISSN 0004-3702. doi: 10.1016/j.artint.2026.104491. URL https://www.sciencedirect.com/science/article/pii/S0004370226000172.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. Representation engineering: A top-down approach to ai transparency, 2023. URL https://arxiv.org/abs/2310.01405.

Table 5: Counterfactual sensitivity sample counts for the $85/10/5$ split. FEVEROUS is treated as one dataset, with its budget distributed across its four subsets.

Dataset	Selected samples	Train	Val	Test
FEVEROUS	2,000	1,700	200	100
RuleTaker	1,628	1,384	163	81
TreeCut	2,000	1,700	200	100
Total	5,628	4,784	563	281

A Data and label construction

A.1 Datasets and splits

The paper-facing experiments use three reasoning datasets: FEVEROUS (Aly et al., 2021), RuleTaker (Clark et al., 2020), and TreeCut (Ouyang, 2025). FEVEROUS includes multiple subsets; we take its four largest labeled subsets and combine them into a single dataset for splitting and balancing.

We balance by dataset so that no single dataset dominates training and so that headline metrics are not driven by the largest available pool. For counterfactual sensitivity, the per-dataset cap is $2{,}000$ original samples where available; RuleTaker contributes its full filtered pool of $1{,}628$ samples. For rerun consistency, each dataset contributes $1{,}500$ original samples. The combined FEVEROUS budget is split across its four subsets in proportion to the available pool of each subset, which preserves the natural relative size of each subset within the FEVEROUS budget.

Trajectory length. We cap the length of the target LRM's generation (reasoning tokens plus final answer) at sampling time. We chose this cap to be long enough that the great majority of trajectories complete naturally and short enough that each item fits within the per-item memory budget of our training hardware. We use $8{,}000$ tokens. Across the paper-facing datasets, between 98.6% and 99.8% of generated runs reach a final answer within the cap.

Splits. The main learnability runs use a sample-level $95/0/5$ train/test split with seed 42. Runs that require validation-based checkpoint selection use an $85/10/5$ train/val/test split derived from the same held-out test surface.

Counterfactual sensitivity counts. For each original sample we sample the target LRM 10 times on the original prompt and 10 times on each perturbed prompt. A sample is kept only if it has at least 5 valid runs of the original prompt and an original-answer consistency of at least 0.7, so its labels are built on a sufficiently stable original answer. Each kept sample carries multiple segment-level labels, one per kept removable segment.

Table 5 reports the per-dataset sample counts.

Metric aggregation. For counterfactual sensitivity, each retained segment label is one evaluation item. We compute Spearman and Pearson correlations over the flattened vector of segment-level predictions and labels, rather than first averaging predictions within each original sample. Consequently, samples with more retained segments contribute more evaluation items to the reported correlation. Table 6 reports the corresponding held-out sample counts and segment-level evaluation entries.

Rerun consistency counts. For each original sample we sample the target LRM 10 times on the prompt. We require at least 5 valid runs for a sample to be labeled, so each label is built from a sufficient sample size. At training and evaluation time, the Behavior Forecaster consumes 5 observed runs per sample as separate inputs. This separates the runs used to estimate each label (up to 10, post-filter) from the runs actually fed to the Behavior Forecaster (5, the inference-time constant we hold fixed across samples).

Table 6: Evaluation units for counterfactual sensitivity correlations. Each segment entry contributes one prediction–label pair to Spearman and Pearson.

Dataset	Test samples	Segment entries
FEVEROUS entity	4	70
FEVEROUS multi-hop	39	1,105
FEVEROUS numerical	19	780
FEVEROUS tables/text	38	1,200
RuleTaker	81	11,740
TreeCut	100	1,425
Total	281	16,320

Table 7: Rerun consistency per-dataset sample counts in the $85/10/5$ split.

Dataset	Train	Val	Test	Total
FEVEROUS	1,275	150	75	1,500
RuleTaker	1,275	150	75	1,500
TreeCut	1,275	150	75	1,500
Total	3,825	450	225	4,500

Table 7 reports per-dataset sample counts.

A.2 Answer extraction

The label-construction rules in the rest of this appendix treat two runs as producing the same answer when their dataset-specific extracted values are equal, not when their full output strings match. We apply the parser to the LRM's post-`</think>` answer span only. For each dataset we apply a short ordered list of regular expressions and use the first match; if none matches, the run is marked extraction-failed and excluded from the labeled run set.

FEVEROUS (multiple choice). The three FEVEROUS verdicts (SUPPORTS, REFUTES, NOT ENOUGH INFO) are presented as a multiple-choice question with letters A–C. The parser looks, in order, for: (i) a boxed letter, e.g. `\boxed{A}`; (ii) an explicit "`(final )?answer/choice (is|:) X`" phrasing; (iii) a single letter on its own line; (iv) a trailing letter at end of text. The extracted value is the matched letter.

RuleTaker (true/false). The parser looks, in order, for: (i) an explicit "`label: true|false`" phrasing; (ii) the last occurrence of a standalone `true` or `false` token in the answer span. The extracted value is the matched token, lowercased.

TreeCut (numeric or unknown). The parser first checks for an explicit UNKNOWN answer using "`(final )?answer (is|:) unknown`" or a standalone `unknown` at the end of the answer span. Otherwise it looks for, in order: (i) a boxed numeric value, e.g. `\boxed{16.5}`; (ii) an "`answer/result (is|:)` n" phrasing; (iii) a number at end of text; and as a last resort the last numeric literal anywhere in the answer span. Two numeric values are treated as the same when they are exactly equal under `float` comparison; UNKNOWN is its own equivalence class.

A.3 Counterfactual sensitivity label construction

For each original prompt P we define a dataset-specific set of removable segments $S(P)$ and create one perturbed prompt P_{-s} for each $s \in S(P)$. A run is valid if it has no generation failure, has an extractable answer, reaches the closing `</think>` tag, and has non-empty post-`</think>` answer text. For each kept prompt, we identify the answer A produced by at least 0.7 of the 10 valid original runs; prompts without such an answer are dropped.

Table 8: Omission units used to construct counterfactual sensitivity perturbations.

Dataset	Removed unit
FEVEROUS	One evidence bullet
RuleTaker	One statement
TreeCut	One evidence paragraph

Let c_{orig} be the fraction of valid original runs producing A, and c_{-s} the fraction of valid perturbed runs on P_{-s} producing A. The segment label is

$$\text{AF}(s) = \text{clip}\left(\frac{c_{\text{orig}} - c_{-s}}{c_{\text{orig}}},\, 0,\, 1\right).$$

A perturbed instance is kept only if it has at least 5 valid runs. A sample can survive with only a subset of its segment labels; missing segments contribute no supervision during training. This per-segment retention rule is intentional: it lets us keep a sample's well-supported segments without forcing every segment to meet the run-count threshold.

Omission units. Table 8 lists the dataset-specific unit removed to form one perturbed instance.

A.4 Rerun consistency label construction

A run is valid if it has no generation failure and has an extractable answer. If run i has extracted answer a_i, and c_i of the n valid runs on the same input have extracted answer a_i, the run-level target is

$$\ell_{\text{cons}}(i) = \frac{c_i - 1}{n - 1}.$$

This estimates the likelihood that another run on the same input will produce the same answer as run i.

If any of a sample's runs hits the trajectory length cap, we discard the entire sample rather than dropping individual long runs. Discarding the whole sample preserves the exact run set its labels were computed on, so we do not need to recompute labels after the length filter.

B Behavior Forecaster training and architecture

B.1 Training settings

We initialize the Behavior Forecaster from the target LRM's weights and train it end-to-end with AdamW, weight decay 0.01, and a linear-decay schedule with 100 warmup steps and a minimum learning rate of 1×10^{-6}. We use an effective batch size of 32 and train for 2 epochs. Counterfactual sensitivity uses learning rate 1×10^{-5} and a token-level MLP head; rerun consistency uses learning rate 5×10^{-5} and a sequence-level cross-attention MLP head. All the Behavior Forecasters in the paper were trained on 4 NVIDIA H200 GPUs in under 24 hours.

B.2 Training objectives

Counterfactual sensitivity. Counterfactual sensitivity uses the dataset-defined set of removable segments $S(P)$ during training. For an observed trajectory $\tau = (P, R, A)$ in the prompt-echo arrangement, the Behavior Forecaster emits one logit z_t at each token position in the echoed prompt region. For each retained segment $s \in S(P)$, we build a binary mask $m_{s,t}$ over the echo-region positions corresponding to that segment. The Behavior Forecaster first pools logits within the segment,

$$\bar{z}_s = \frac{\sum_t m_{s,t} z_t}{\sum_t m_{s,t}}, \qquad \hat{a}_s = \sigma(\bar{z}_s).$$

The target a_s is the clipped attributable-fraction label from Section A.3. The loss is binary cross-entropy on the pooled segment logit,

$$\mathcal{L}_{\text{cf}} = \frac{1}{|\mathcal{S}_{\text{valid}}|} \sum_{s \in \mathcal{S}_{\text{valid}}} \left[-a_s \log \sigma(\bar{z}_s) - (1 - a_s) \log(1 - \sigma(\bar{z}_s)) \right].$$

Thus each retained segment contributes one loss term, even though the Behavior Forecaster produces token-level logits. This objective assumes the removable segments and prompt-alignment masks are provided by the dataset construction; the Behavior Forecaster is trained to score the supplied segments, not to discover the segment boundaries. At evaluation time we use the same pooled predictions and compute correlations over the flattened set of retained segment labels.

Rerun consistency. Rerun consistency is a trajectory-level prediction problem and has no segment masks. For each observed trajectory i, the Behavior Forecaster emits one scalar logit z_i and prediction $\hat{\ell}_i = \sigma(z_i)$. We train with mean-squared error against the continuous run-level consistency label:

$$\mathcal{L}_{\text{rerun}} = \frac{1}{N} \sum_{i=1}^{N} \left(\hat{\ell}_i - \ell_{\text{cons}}(i) \right)^2.$$

Each of the 5 observed runs per sample is treated as a separate training and evaluation item. All rerun-consistency results reported in the paper use this continuous objective.

B.3 Architectural hyperparameters and transfer fine-tuning budget

Backbone and heads. The OLMo-3-7B-Think backbone has hidden size $d = 4096$. The token-level head used for counterfactual sensitivity is a 2-layer MLP with hidden sizes $[512, 256]$ applied at each echoed-prompt position, producing one logit per token. The sequence-level head used for rerun consistency pools the backbone's hidden states with cross-attention over 12 learned query vectors of dimension d (32 attention heads, dropout 0.1), then concatenates the pooled vectors and passes them through a 2-layer MLP with hidden sizes $[1024, 256]$ and GELU activations between layers (each layer is Linear $\rightarrow$ LayerNorm $\rightarrow$ GELU $\rightarrow$ Dropout) to produce one scalar logit.

Transfer fine-tuning budget for Table 2. For each held-out target dataset, we initialize from the source Behavior Forecaster (trained on the other two datasets with the same recipe as Section 4) and fine-tune the entire model on the target's train split with the same optimizer, weight decay, schedule, effective batch size, and per-task learning rate as the main runs.

C Baselines and probes

C.1 Naive reader prompts

Choice of naive readers. We use Claude Opus 4.6 and GPT-5.4 as frontier naive readers. These readers come well-equipped for what the reading task reduces to, but they have not learned how the target LRM uses its reasoning tokens. In practice, researchers already use frontier LLMs as readers of reasoning trajectories, deploying them to judge whether a target LRM's trajectory tracks its underlying computation (Chen et al., 2025; Young, 2026). These readers are larger than the Behavior Forecaster and use their own extended reasoning, making them substantially more expensive to run; any margin a Behavior Forecaster shows against them is therefore a conservative measure of the trained-versus-naive gap.

Inputs and prompt selection. Each reader is given three pieces of information per evaluation example: the original input the target LRM saw, the target LRM's observed final answer for that run, and the target LRM's full completion text for that run. The reader returns a continuous score in $[0, 1]$ and is not trained or calibrated on the task.

For each task we tested three prompt variants on a 100-sample pilot slice of the evaluation set and selected, per reader model, the variant with the highest pilot Spearman correlation. This per-reader selection is deliberate: it gives each reader its strongest configuration before we compare against the trained Behavior Forecaster.

Counterfactual sensitivity prompts. All three variants ask the reader to score, for each candidate segment, how strongly removing that segment would shift the target LRM's final answer away from the observed answer. The variants differ only in framing wording. GPT-5.4 used variant v2 (Box C.1); Claude Opus 4.6 used variant v3 (Box C.1). Variant v1 frames the score as "how strongly removing the segment would reduce the chance of the same final answer" and is included in the released prompt set.

Counterfactual sensitivity, variant v3 (Claude Opus 4.6)

```
Below is a problem and one observed run produced by an AI model.  The
problem is partitioned into candidate segments.
You will see the original problem, the full completion from that run,
the observed final answer, and the candidate segments.
Your task is to estimate, for each candidate segment, how strongly
that segment supports the observed answer:  if you removed the
segment, how likely is the model to move away from the observed
answer?
Technical details:  target model and sampling settings
(temperature=0.6, top_p=0.95) are provided in the prompt.  Score must
be clipped to [0,1].  Answer 0 if removing the segment would still
leave the observed answer more likely than a different answer.  Answer
1 if removing the segment would make a different answer more likely
than the observed answer.
Return one floating-point score for each segment as JSON:
{"scores":{"1":...,"2":...}}, with exactly one entry per candidate
segment.
Original problem, observed final answer, observed full completion,
and the candidate segment list are interpolated into the prompt as
separate fields.
```

Counterfactual sensitivity, variant v2 (GPT-5.4)

```
Below is one observed run from an AI model on a problem.  The problem
text has been divided into candidate segments.
You will see the original problem, the model's observed full
completion, the observed final answer, and the candidate segments.
For each candidate segment, imagine removing only that segment from
the problem.  Predict how much that removal would reduce the chance
that the model gives the same final answer as in the observed run.
Technical details and score range match variant v3.  Answer 0 if,
after removing the segment, the same final answer is more likely than
a different final answer; answer 1 if a different final answer is more
likely.
Return one floating-point score per segment as JSON, with exactly one
entry per candidate segment.
```

Rerun consistency prompts. All three variants ask the reader to forecast how often the observed final answer would recur on reruns, differing only in how the score is framed. GPT-5.4 and Claude Opus 4.6 both used variant (a), the direct stability prompt (Box C.1). Variant (b) reframes the target as "the probability that another run on the same input would produce the same final answer" and variant (c) as "the expected fraction of many reruns whose final answer would match the observed final answer"; both are included in the released prompt set.

Rerun consistency, variant (a) (both readers)

```
You are forecasting how stable a reasoning model's final answer is
under rerunning.
You are given the exact input the model saw, the model's observed
final answer on this run, and the model's full completion on this run.
Task:  predict the stability score of this observed answer on a scale
from 0 to 1.  0 means this answer would be unique among repeated runs
on the same input; 1 means essentially all repeated runs would give
```

Table 9: OLMo-3-7B-Think test Spearman and Pearson correlation for the trained Behavior Forecaster and two frontier naive readers on each task.

	Counterfactual sensitivity		**Rerun consistency**	
	Spearman	Pearson	Spearman	Pearson
Behavior Forecaster	**0.731**	**0.931**	**0.571**	**0.667**
GPT-5.4 (naive)	0.672	0.890	0.305	0.359
Claude Opus 4.6 (naive)	0.690	0.888	0.308	0.537

Table 10: Per-dataset test Spearman/Pearson correlations for the OLMo-3-7B-Think comparison in Figure 2. Behavior Forecaster values are computed from the last saved checkpoints: step 1668 for counterfactual sensitivity and step 1336 for rerun consistency. For counterfactual sensitivity, the paper-facing plotted checkpoint is step 1600; the last saved checkpoint differs only in the fourth decimal place in the pooled metric. FEVEROUS aggregates the four FEVEROUS subsets, matching the three-dataset framing used in the main text.

Task	**Dataset**	n	**Behavior Forecaster**	**Claude Opus 4.6**	**GPT-5.4**	**Single-location probe**
Counterfactual sensitivity	FEVEROUS	3,155	0.660/0.775	0.574/0.578	0.621/0.674	0.209/0.116
	RuleTaker	11,740	0.668/0.967	0.649/0.961	0.601/0.949	−0.008/0.017
	TreeCut	1,425	0.840/0.946	0.775/0.865	0.804/0.887	0.258/0.186
Rerun consistency	FEVEROUS	375	0.580/0.517	0.607/0.475	0.537/0.370	0.495/0.489
	RuleTaker	375	0.627/0.887	0.069/0.692	0.245/0.380	0.057/ − 0.012
	TreeCut	375	0.140/0.455	−0.043/0.365	0.058/0.248	0.098/0.095

```
this same answer; intermediate values mean this answer would recur on
some but not all reruns.
Predict answer stability under rerunning, not correctness.  Base your
prediction only on the observed input and this observed run.  Return
only a single decimal number between 0 and 1, with no explanation.
```

Behavior Forecaster and naive reader test metrics. Table 9 reports the OLMo-3-7B-Think test Spearman and Pearson correlations summarized by the top row of Figure 2. Behavior Forecaster numbers are the test results used for the paper-facing comparison.

Table 10 breaks the OLMo-3-7B-Think comparison down by dataset family. The Behavior Forecaster is more accurate than both naive readers in aggregate and on most per-dataset cells.

C.2 Single-location probe baselines on the target LRM

The dashed single-location probes in Figure 2 are direct adaptations of standard attribution and uncertainty signals to each task: each reads a single fixed location of the frozen target LRM and produces a per-instance score, with no further training, threshold tuning, or calibration. They are computed on the same test surfaces as the main figure.

Answer likelihood for rerun consistency. For each observed trajectory $\tau_i = (P_i, R_i, A_i)$, we teacher-force the stored prompt and completion through OLMo-3-7B-Think and score only the final-answer tokens after the first closing `</think>` tag. The reported score is the mean final-answer log probability,

$$s_i^{\text{lik}} = \frac{1}{|T(A_i)|} \sum_{t \in T(A_i)} \log p_M(x_t \mid x_{<t}),$$

where $T(A_i)$ is the set of final-answer token positions. Higher values mean lower answer perplexity and therefore higher predicted rerun consistency. This baseline follows the broad use of token probabilities as uncertainty or hallucination signals in generation baselines (Manakul et al., 2023; Farquhar et al., 2024); here we intentionally use the simplest answer-only version rather than semantic clustering or resampling.

Answer-to-input attention for counterfactual sensitivity. For each labeled input segment, we teacher-force the observed trajectory through OLMo-3-7B-Think with attention outputs enabled. We then use the final-answer tokens as queries and score a candidate segment by the last-layer attention mass assigned to that segment's prompt tokens:

$$s_j^{\text{attn}} = \frac{1}{H|T(A)|} \sum_{h=1}^{H} \sum_{t \in T(A)} \sum_{u \in T(S_j)} \alpha_{t,u}^{(L,h)}.$$

Here $T(S_j)$ is the token span of segment j, $\alpha^{(L,h)}$ is head h in the final transformer layer, and higher values mean that final-answer tokens attend more to the segment. This sits in the standard family of attention-based attribution probes for context use (Chuang et al., 2024; Jain & Wallace, 2019), applied here at the natural readout point in an LRM: from the final-answer tokens back to the prompt.

Table 11: Single-location probe baselines on the target LRM, used as dashed lines in Figure 2. Both are evaluated on the same test surfaces as the main figure.

Task	**Probe**	n	**Spearman**	**Pearson**
Counterfactual sensitivity	Answer-to-input attention mass	16,320	0.160	0.160
Rerun consistency	Final-answer likelihood	1,125	0.129	0.118

C.3 TARGET-LRM NAIVE READER CONTROL

As an additional diagnostic, we evaluate OLMo-3-7B-Think itself as a naive reader of its own observed trajectory. This baseline tests whether the target LRM can recover the future-behavior signal by prompted reading alone, without any training or calibration. In the stateless variant, one prompt contains the original input, the observed OLMo completion, the observed final answer, and the same forecasting task given to the other naive readers. In the stateful variant, we replay the original input as the user message and the observed OLMo completion as the assistant message, then append a follow-up user message asking for the forecast.

D STATISTICAL UNCERTAINTY ESTIMATES

For the main learnability comparison on OLMo-3-7B-Think, we estimate uncertainty by cluster bootstrapping over original samples, stratified by dataset family. This treats an original sample, not a flattened segment or observed run, as the resampling unit. We use $5{,}000$ bootstrap resamples for confidence intervals. For paired tests, we compare the Behavior Forecaster and each naive reader on exactly matched prediction rows, resampling the same sample clusters for both systems. We also run a one-sided paired permutation test with $10{,}000$ permutations, swapping the two systems' predictions together at the original-sample cluster level and using the directional alternative that the Behavior Forecaster has higher correlation than the naive reader. Table 13 reports Spearman uncertainty for the main metric.

For transfer, the unit of replication is the held-out target dataset, so the analysis is necessarily low-power with three targets per task. The adapted source improves over the unadapted source for all three held-out targets on both tasks. The mean adapted-minus-source Spearman gap is 0.228 with bootstrap CI $[0.196, 0.273]$ for counterfactual sensitivity and 0.268 with CI $[0.075, 0.433]$ for rerun consistency; the one-sided sign-test value is $p = 0.125$ in both cases because there are only three targets. We treat ablation rows as descriptive single-run comparisons rather than significance tests, since they do not have repeated seeds or saved paired predictions for every variant.

Table 12: Appendix-only target-LRM naive reader control. OLMo-3-7B-Think is prompted to read its own observed trajectory and predict the target behavioral property without training.

	Counterfactual sensitivity		**Rerun consistency**	
	Spearman	Pearson	Spearman	Pearson
OLMo as naive reader (stateless)	0.167	0.182	−0.025	0.019
OLMo as naive reader (stateful)	0.226	0.245	0.159	0.197

Table 13: Cluster-bootstrap uncertainty and one-sided paired permutation tests for the main Spearman results. CIs are 95% cluster-bootstrap intervals over original samples.

Task	**Reader**	**Forecaster**	**Reader**	**Paired gap**	**One-sided perm.** p
Counterfactual sensitivity	GPT-5.4	0.731 [0.701, 0.756]	0.672 [0.629, 0.713]	0.059 [0.027, 0.088]	0.011
Counterfactual sensitivity	Claude Opus 4.6	0.731 [0.701, 0.756]	0.690 [0.650, 0.727]	0.041 [0.013, 0.068]	0.037
Rerun consistency	GPT-5.4	0.571 [0.476, 0.657]	0.305 [0.208, 0.398]	0.266 [0.162, 0.361]	< 0.001
Rerun consistency	Claude Opus 4.6	0.571 [0.476, 0.657]	0.308 [0.196, 0.412]	0.263 [0.167, 0.361]	< 0.001