COUNTING AS A MINIMAL PROBE OF LANGUAGE MODEL RELIABILITY

Tianxiang Dai
Department of Electrical Engineering
Stanford University
Stanford, CA 94305
txdai@stanford.edu

Jonathan A. Fan*
Department of Electrical Engineering
Stanford University
Stanford, CA 94305
jonfan@stanford.edu

ABSTRACT

Large language models perform strongly on benchmarks in mathematical reasoning, coding, and document analysis,[1–4] suggesting their general ability to follow instructions. However, it is unclear whether this success is due to general logical competence, the repeated application of learned procedures, or pattern matching that merely mimics rule execution.[5–7] We investigate these mechanics by introducing Stable Counting Capacity, an assay in which we ask models to count repeated symbols until failure. This assay removes knowledge dependencies, semantics, and ambiguity from model evaluation, avoids lexical and tokenization confounds,[8] and provides a more direct measure of procedural reliability than standard knowledge-based benchmarks.[9,10] Upon evaluation across over one hundred model variants, we find that the counting capacity for all models is consistently far below advertised context limits.[4] An analysis of model behavior indicates that counting is not supported by either open-ended logic or by stable application of a learned rule, but that it is based on the utilization of a finite number of count-like internal states, analogous to counting on fingers. Once this resource is exhausted, the illusion of rule following disappears and exact execution collapses into guessing, even with additional test-time compute.[11–13] These findings reveal that while current large language models exhibit fluent performance, they do not guarantee general and reliable rule following behavior.

1 Introduction

Large language models (LLMs) exhibit impressive performance on benchmarks spanning science, mathematics, coding, and long-context understanding.[9,10,14–17] These results have encouraged the view that advancements in model scaling correspond to increasing model intelligence. However, the nature of this intelligence remains an open question, as model success can reflect open-ended logical competence, the application of learned procedures, or surface pattern matching that mimics rule-following. Understanding these operating mechanics and the ability for LLMs to follow rules over many steps is essential to understanding how robust and reliable LLMs are in executing complex tasks. Many deployed uses of LLMs, such as long-form generation, repository-scale coding, multi-step tool use, and agentic planning, require the model to keep track of constraints, commitments, variables, and intermediate results as the task unfolds, thereby requiring the model to preserve procedural states and follow rules.[18–21]

To date, verifying this underlying procedural reliability remains largely unexplored in part because there is a lack of tools for directly evaluating mechanical operation. Models are instead primarily evaluated by performance on knowledge-based benchmarks (Fig. 1a, left). While knowledge-based benchmarks are valuable because they resemble real world applications, their complexity obscures the model's mechanics, as the generation of correct answers can reflect genuine reasoning, memorized knowledge, pretraining data overlap, or task-specific heuristics.[7,13,14,17]

An alternative to knowledge-based benchmarks is *mechanical benchmarks*, which are a complementary class of evaluations that reduce dependence on factual knowledge and better correlate with procedural reliability. In a mechanical benchmark, the input is synthetic, the rule is elementary, the output is exact, and new instances can be sampled without bound (Fig. 1a, right). A mechanical assay attempts to ask a precise diagnostic question: when factual knowledge and semantic clues are removed, exactly how far can a model extend a procedure before state tracking fails? Leading efforts in mechanical assay construction include ARC-AGI-style benchmarks, which attempt to remove knowledge from the

benchmark process but which still take the form of fixed, finite tests and remain vulnerable to saturation and training data leakage.[22]

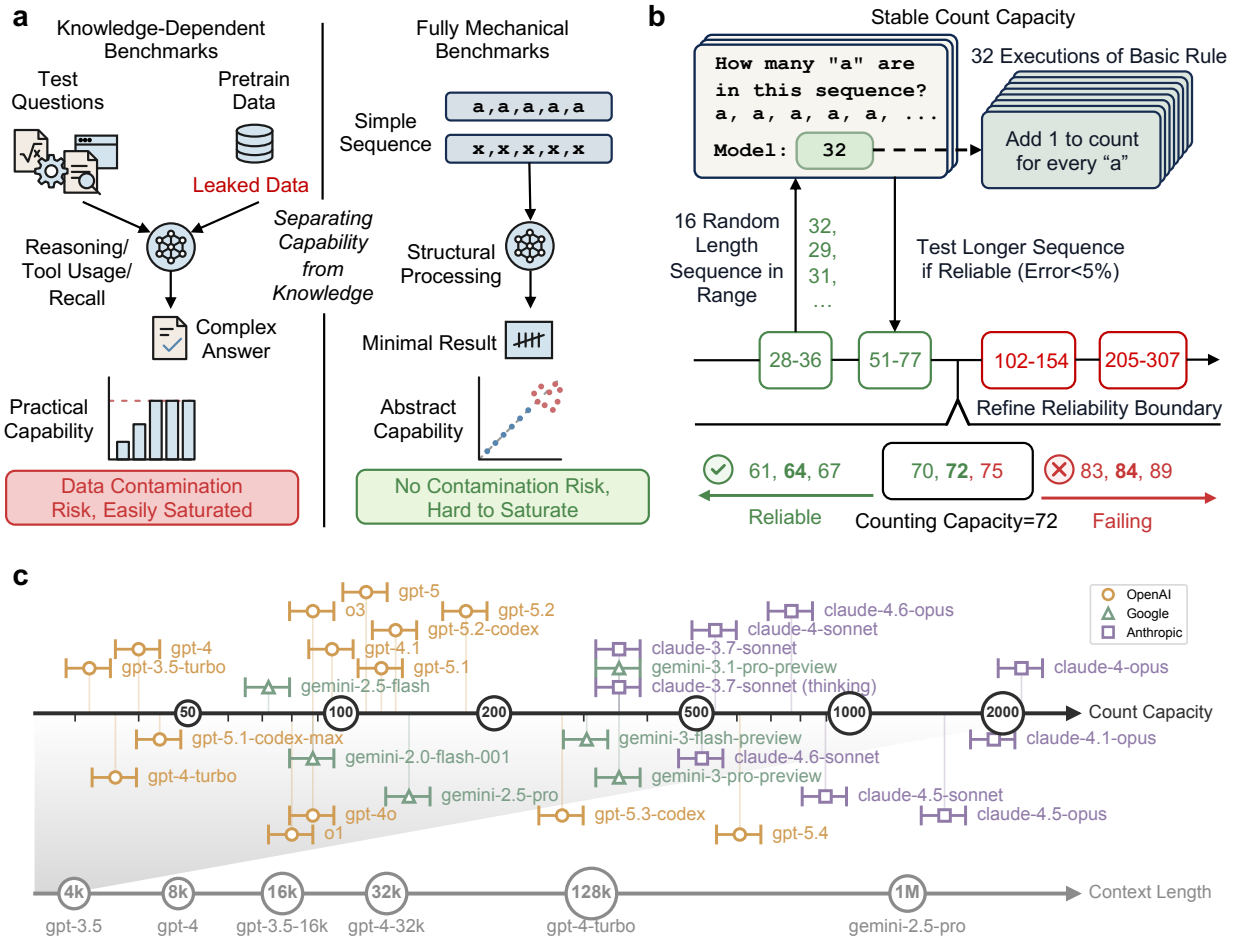

Figure 1: **Stable Counting Capacity as a fully mechanical benchmark for rule execution evaluation. a**, Classes of LLM benchmarks. Knowledge-dependent benchmarks (left) evaluate a mixture of reasoning, factual recall, and tool usage, and they can be impacted by data contamination and leaderboard saturation. Mechanical benchmarks (right) isolate structural processing by applying a simple rule to a minimal sequence without relying on semantic knowledge. **b**, The Stable Count Capacity assay. A model is asked to execute a simple counting rule over a random sequence of lengths. The test iteratively proceeds to longer sequences until the model can no longer reliably count with minimal error. **c**, Measured counting capacities across various frontier language models. Every tested model fails to reach substantial counting capacities, indicating a fundamental limitation in model procedural state maintenance.

We introduce Stable Counting Capacity (SCC), a purely mechanical assay that utilizes a minimal probe based on homogeneous sequence counting to evaluate procedural state maintenance (Fig. 1b). In this assay, the model receives a sequence of identical items and is queried to return the exact number of items as a single integer. This process is iteratively repeated with increasing sequence lengths using an adaptive randomized ladder scheme (Supplementary Note 3) until the model consistently returns an incorrect answer. We define the sequence length where failure happens as the counting capacity (CC), and this quantity specifies the precise boundary at which a model's ability to mimic procedural rule-following breaks down. With SCC, the counted unit is the item, not the token, and the prompt contains no changing symbols, semantic landmarks, or external memory aids. In addition, the main assay avoids JSON and other schemas, such that parser and tokenizer behavior is not part of the measurement.[8]

We find that all LLMs are incapable of basic counting, and all have a measurable CC. A plot of CC's for selected leading model variants is in Fig. 1c and shows values spanning a broad range, with newer models often supporting larger CCs (additional SCC benchmark data for these models are in Supplementary Table 4 and Supplementary Figs. 5

to 11). We also observe that long-context systems generally lose the ability to count far below their advertised context windows. As such, the ability to process a long prompt does not imply the ability to carry a simple rule-defined variable through that prompt. These results are an explicit indication that all models lack open-ended logical generalization and are incapable of stably executing simple learned rules.

In the following, we perform a detailed analysis of SCC to understand how it operates as a probe of LLM capability. We first investigate model dynamics during counting and find that SCC is a probe that explicitly tracks internal states within models. We further analyze these internal states as the model counts to deepen our understanding of the mechanism for rule following in LLMs. Finally, we compare SCC to other conventional evaluation benchmarks and elucidate what those benchmarks are measuring and their fundamental limitations. The implications of our analysis go beyond basic counting and broadly extend to more typical knowledge and logic LLM tasks.

2 Model dynamics when counting fails

To evaluate what exactly the SCC assay is tracking, we first evaluate the dynamics of model behavior near the counting failure point and observe that models exhibit perfect accuracy within a stable regime before suffering a sudden structural collapse. In a representative SCC run, `claude-sonnet-4-6` follows the diagonal across the stable region and then suddenly produces incorrect values near the CC boundary (Fig. 2a). A higher-resolution view shows that after failure, the outputs are not small local deviations but that the model often jumps to salient numbers such as `500`, `1000` or `2000`, even when the true count is far away. This behavior at the point of counting failure is general across all model families. When predictions are normalized by the CC of each model and overlaid across the full evaluation set, the stable region remains tightly concentrated near the diagonal, whereas the post-boundary region is characterized by unpredictable, large errors (Fig. 2b). This universal model behavior is contrary to that of the production of smoothly growing numerical errors expected from a continuous approximation, and it provides evidence that all models use a finite internal state to perform counting.

When attempting to count sequences with lengths greater than CC, the inaccurate model outputs cluster around a restricted set of preferred integers, with multiples of 10 or 100 appearing especially often (Fig. 2c). These preferred values mark out horizontal bands across a wide range of true counts (Fig. 2d). Instruction following is also significantly weakened when the model loses count. Across evaluated trials, 5% of outputs (501 of 9797) did not contain a valid single number response, producing instead blank outputs, prompt echoes, code formatting artifacts, and spurious reasoning traces. Such failures indicate that depletion of the procedural state can disrupt not only numerical accuracy, but also the control needed to maintain the requested response format (Supplementary Note 4; Supplementary Tables 1 to 3).

Swapping the counted character type or delimiter shifts the CC for several models, sometimes with little change in relative input token count (Fig. 2e). This sensitivity indicates there does not exist a fully abstract procedural state shared across all equivalent symbols. Models instead appear to use a learned state that depends on the trajectory induced by the specific character and delimiter.

Model limitations in counting tasks extend beyond one-dimensional tallying. In our evaluation of a hierarchical nested-depth tracking assay, models were asked to count records in which a key token matched the deepest token inside a structured path while ignoring distractors. The task still relies on a strict rule and an unambiguous integer output, but it requires maintaining a richer structural state than a simple increment. The resulting failure dynamics were identical to those of basic counting: even the highest-performing model reached a bounded stability of 416 true matches before structural collapse (Supplementary Note 6; Supplementary Table 5 and Supplementary Fig. 12). Thus, the finite capacity observed in simple counting is not a special case of repeated symbols,[23] but points to a broader, generalized difficulty in preserving exact procedural states.

A natural question that arises is whether models can increase their CC or even eliminate the presence of a CC through increased generation length or test-time computation. We evaluate total token usage when performing SCC up to the stable CC boundary for all evaluated models. The results are plotted in Fig. 3a and show an empirical efficiency frontier for stable counting that is approximately two consumed tokens per true count.

The plot also reveals that many models utilize tokens in a sub-optimal fashion, with reasoning-optimized models often consuming many more total tokens than non-reasoning systems without necessarily achieving larger CC values. Matched base-versus-reasoning comparisons (Fig. 3b) further enhance this point. Reasoning model variants often spend several-fold more hidden or output tokens while producing small or even negative changes in CC. These results suggest that while additional test-time computation improves many semantic tasks by providing structural scaffolding,[11–13,24] it cannot reliably reconstruct a tally lost during context processing. This indicates the existence of a finite, model-specific

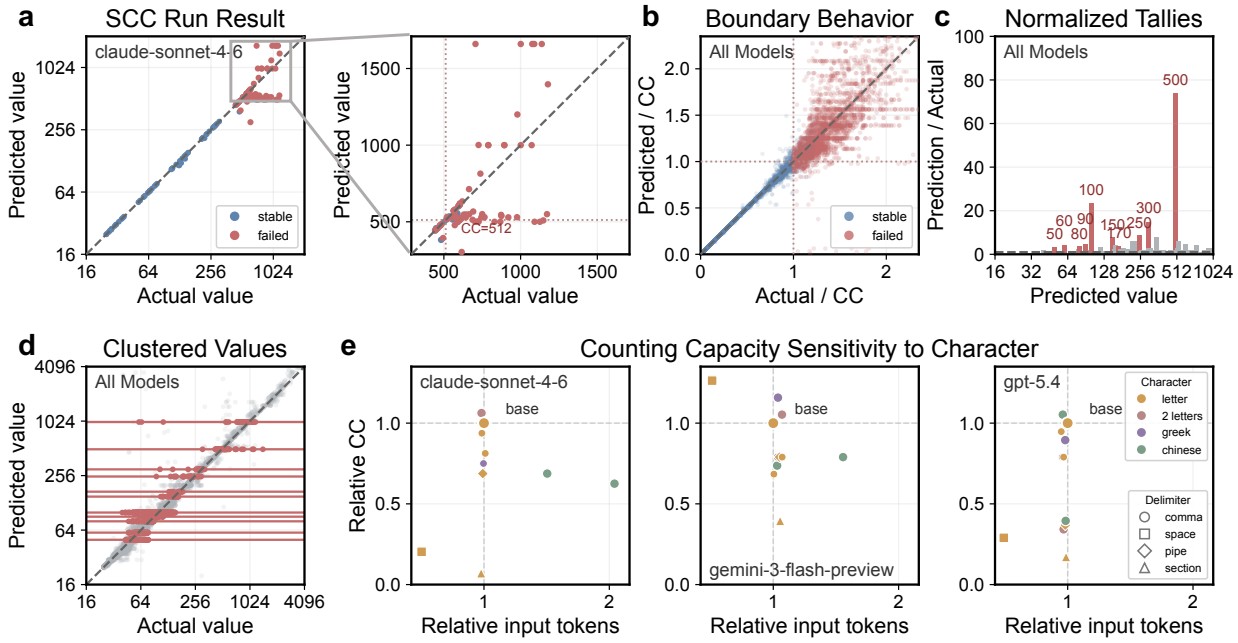

Figure 2: **Model behavior at the point of counting failure. a**, The tracking behavior of a representative model during a counting run. The model predicts the exact count perfectly before abruptly failing and defaulting to highly specific rounded numbers. **b**, A high resolution overlay of boundary behavior across all models. The transition from perfect rule execution to chaotic output is sudden, showing no controlled or gradual degradation. **c**, A histogram of normalized predictions after tracking fails across all models. Outputs heavily cluster around discrete attractors, with 500 being the most frequent guess. **d**, An overlay of actual versus predicted values for all models. The clustered predictions form distinct horizontal bands, demonstrating that models make wild guesses far from the target value after losing track of the count. **e**, The impact of varying the repeated character and delimiter on counting capacity. Syntax variations cause significant performance shifts even when the input token count remains unchanged.

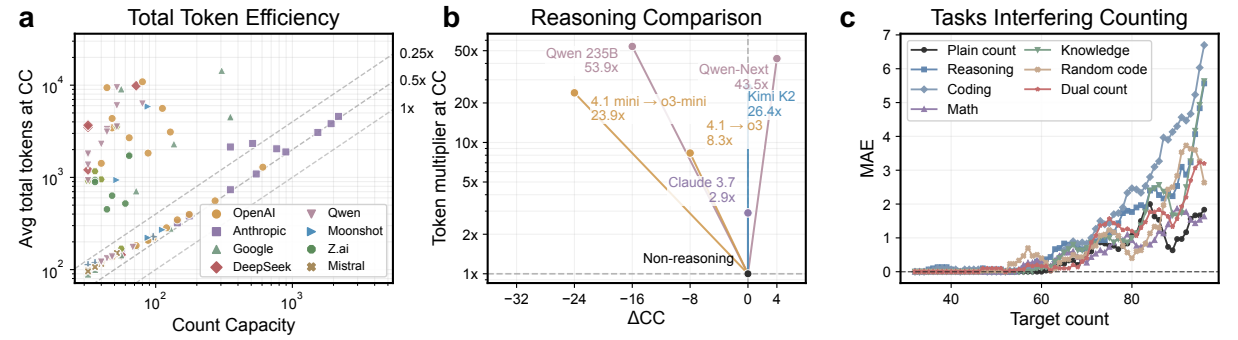

Figure 3: **Impact of token consumption and test-time compute on procedural state maintenance. a**, Average total token consumption evaluated at the CC boundary. Higher token expenditure does not guarantee a greater counting capacity. **b**, A matched comparison between base non-reasoning models and their reasoning variants. Reasoning models consume dramatically more tokens during inference, but they show negligible improvements in exact procedural execution. **c**, Error curves for matched dual task experiments. Reasoning and coding subtasks increase counting error relative to plain counting and length-matched controls, indicating that complex tasks compete for the same limited internal tracking resources.

procedural state that is consumed during counting that is independent of how many tokens the model is allowed to generate.

We additionally evaluate whether the procedural states probed by SCC are specifically used for counting or if they support more general tasks, and we set up matched dual-task experiments where `gpt-5.4-mini` simultaneously counts a marker sequence and answers a benchmark-like questions for reasoning (BBH), coding (CRUXEval-O), math (MATH-500), and knowledge (MMLU-Pro). We compare these trials with plain counting, length-matched irrelevant-code controls, and a secondary-count control. The coupling of counting with reasoning and coding tasks severely disrupts the model's counting accuracy, driving up the error rate greater than any of the control tasks (Fig. 3c). Interestingly, asking the model to keep track of a second independent count causes less interference. This contrast demonstrates that complex problem-solving and basic procedural tracking compete for the exact same limited internal resource, tied more to complexity than actual token counts. Details of the task measured are in Supplementary Note 9.

3 Probing bounded state trajectories within models

The behavioral results above suggest that successful counting is supported by specific bounded internal states. We further probe this hypothesis using Gemma 3 27B-it, a dense open-weight transformer with a standard architecture and available Gemmascope 2 sparse autoencoder features.[25,26] This model reproduces the qualitative SCC behavior of counting correctly through a stable range, with the first error occurring at 27 items followed by an abrupt collapse to repeated preferred outputs such as `60` and later `100` (Fig. 4a). The open weight setting allows us to inspect residual stream activations at the final start-of-turn token immediately before generation and at repeated token positions throughout the prompt.

We observe that a linearly readable count-related coordinate emerges during successful counting. We fit one-dimensional residual stream projections from successful runs and evaluate them across target counts at layers 16, 31, 40 and 53. The projected coordinate tracks the true count with a precise linear relationship throughout the successful regime (Fig. 4b). However, the linear structure disappears at the same point where behavior fails. The collapse of the latent state therefore predicts the collapse of counting.

Teacher-forced logit analysis shows that the failure is not merely a decoding accident. We measure the separation between the correct count token and competing tokens while forcing the correct answer format. Within the stable regime, the model strongly prefers the exact answer. When counting near and beyond the CC, the correct logit margin decays sharply and can become negative (Fig. 4c). After collapse, the model often no longer recognizes the correct integer as the preferred continuation, even when evaluated under the correct answer prefix.

To determine the extent to which the latent space is localized, we utilize sparse autoencoder analysis. The Gemmascope 2 features most correlated with count are structured and non-monotonic, as opposed to single accumulators (Fig. 4d).[26,27] Our further analysis of perturbations further show that the state is syntax sensitive. Changing the repeated character or delimiter preserves some coarse progress directions across layers (Supplementary Note 10), but reorganizes the supporting feature coalition. The model does not appear to implement one abstract counter shared cleanly across all surface forms. It instead assembles related but distinct trajectories depending on the input syntax.

We next use activation patching to test whether these internal trajectories causally control the output. The concept is based on the patching of activations from donor runs with different counts, and a schematic of the concept is shown in Fig. 4e for a base prompt with a count of 10 items. Two interventions are compared. In final token patching, we replace only the start-of-turn token immediately before generation. In sequence patching, we replace the repeated token states across the prompt. As the donor and recipient lengths differ, donor states were linearly interpolated to the recipient sequence length before patching. This interpolation is an important control because it preserves a coarse linear trajectory while disrupting exact nonlinear token-by-token correspondence.

We find that the causal pattern is layer specific. Final token patching affects the model only in late layers, around layer 51 of 62 total layers, whereas full sequence patching affects the output strongly in middle layers, around layer 31 (Fig. 4f,g). The sequence intervention is also stronger. These results suggest that the model first constructs a per-token progress trajectory in intermediate layers and later transfers count information to the final prompt state before decoding. Other patching and perturbation attempts, including interventions that tried to rescue failed sequences by clamping scalar progress coordinates (Supplementary Note 10), did not reliably recover failed counts, indicating that the causal representation is richer than a single scalar direction.

Together, these results support the mechanistic picture summarized in Fig. 4h. The model appears to assign repeated items to a finite trajectory of count-like internal states. While the states remain organized, decoding can produce the exact count. Once the states are exhausted or disrupted, information about the rule-defined count is no longer available in a decoder-usable form, and the model falls back to plausible numerical guesses. Similar progress related coordinates and steering effects were observed in Qwen 3.5 35B-A3B, a structurally distinct mixture-of-experts model

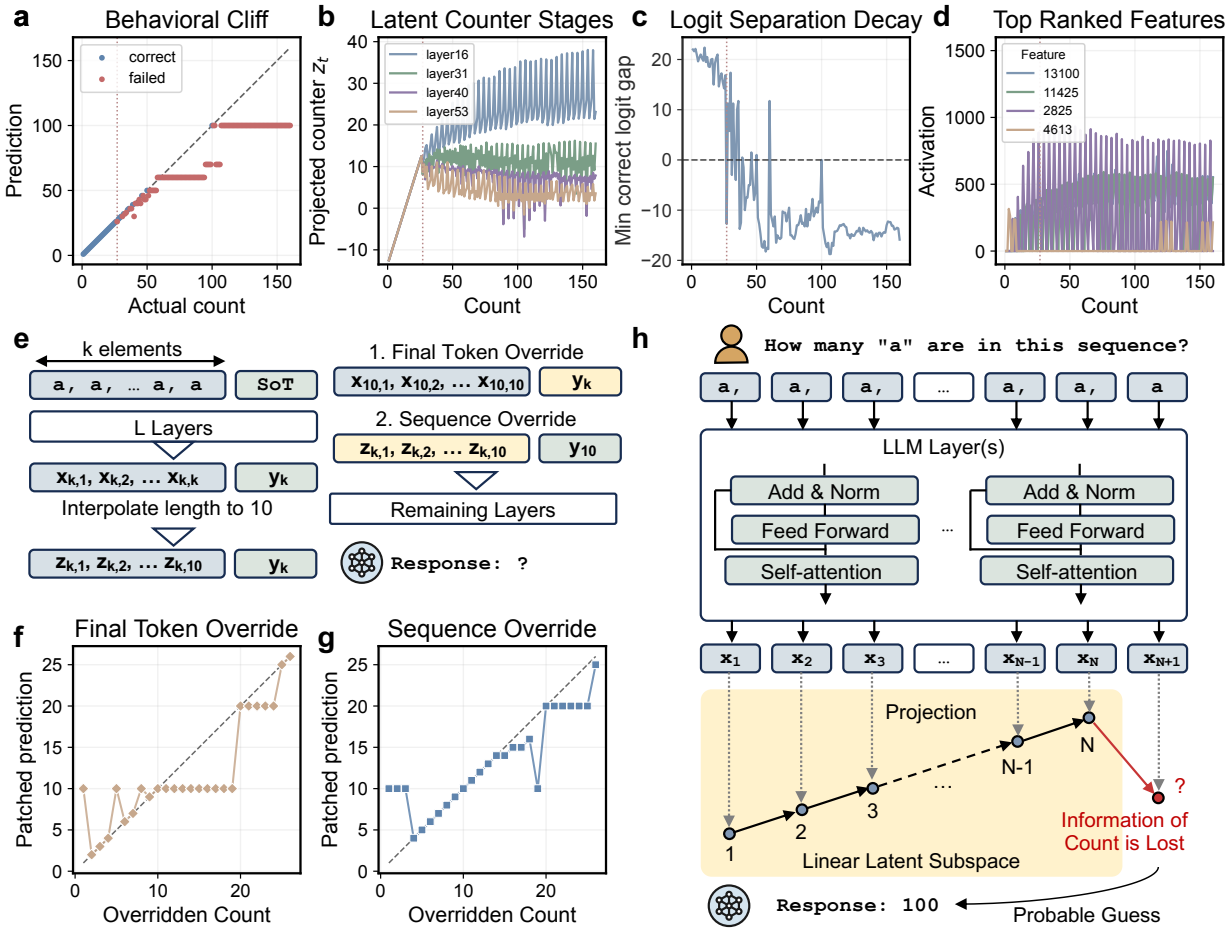

Figure 4: **Internal model probes reveal a finite count-like state. a**, Counting behavior for a dense open weight model at the CC threshold. **b**, The internal latent state projected from the final token preceding generation. A linear direction tracks the count perfectly across layers until the point of failure, after which the organized state completely disappears. **c**, The minimum correct logit gap evaluated through teacher forcing. Following internal state collapse, the model loses the ability to recognize the correct answer entirely. **d**, Activation profiles for top ranked sparse features. The absence of a single dominant tracking feature suggests this behavior is a complex coordination buried deep within the model. **e**, Schematic of activation patching design for isolating causal relations. We take the internal outputs from the same layer of difference sequence lengths and interpolate to match lengths before feeding the mixture into later layers. **f**, The effect of overriding only the final token. The model count is successfully altered only when the intervention is applied at very late layers (53). **g**, The effect of overriding the full sequence except for the final token. Patching the sequence alters the count only by patching in the middle (layer 31) and produces a stronger effect. **h**, Schematic of the rule mimicking process within LLMs. Models assign discrete linear states to track sequence elements on a per token basis. Once this finite capacity is exhausted, the exact state collapses and the model defaults to a probable guess.

(Supplementary Note 11 and Supplementary Fig. 18). The phenomenon is therefore not limited to the specific dense transformer used for mechanistic inspection.

4 Comparing SCC with standard benchmarks

Having established CC as a metric that explicitly captures the presence of finite, trajectory-bound internal states required for exact rule execution, we assess the extent to which standard AI evaluations also capture this criteria. To perform this analysis, we correlate CC with model performance on knowledge-intensive question answering (GPQA Diamond), complex coding (SWE-bench Verified), and abstract fluid intelligence (ARC-AGI-2) benchmarks for the frontier models

shown in Fig. 1c. The results are summarized in Fig. 5 and generally demonstrate that the correlations between CC and the benchmark scores are weak to moderate, indicating that conventional leaderboards are largely blind to fundamental procedural reliability (additional details are in Supplementary Figs. 2 to 4). In other words, models with higher benchmark performance do not necessarily preserve an exact procedural state over longer horizons.[9,10,22]

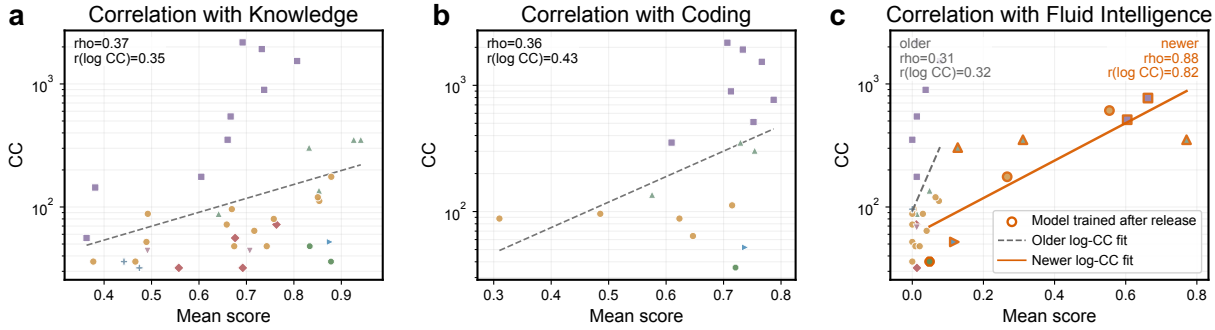

Figure 5: **Standard knowledge and reasoning benchmarks obscure fundamental limits in procedural reliability a-c**, Correlation between counting capacity and conventional benchmarks based on (a) factual knowledge, using the GPQA Diamond benchmark; (b) coding capability, using the SWE-bench Verified dataset; and (c) fluid intelligence, using the ARC-AGI-2 benchmark. For (c), models trained after the public release of the ARC-AGI-2 dataset are highlighted in orange and are strongly linearly correlated with CC.

The comparison of SCC with ARC-AGI-2 is particularly instructive, as ARC-AGI-2 attempts to probe model mechanics by reducing factual dependence through abstract transformations characterized by fixed, finite task distributions.[22] The plot of CC versus ARC-AGI-2 score (Fig. 5c) delineates the performance of models released before and after the ARC-AGI-2 benchmark, and they show qualitatively different behavior before and after benchmark release. Models trained prior to ARC-AGI-2 release display a wide CC range, reflecting a wide variation in capacity for model procedural state maintenance, but they all generally score poorly on ARC-AGI-2 tasks. Models trained after ARC-AGI-2 release have significantly improved performance on ARC-AGI-2 tasks, while yielding a nearly linear correlation between $\log(\mathrm{CC})$ and ARC-AGI-2 score. This interpretation is consistent with the ARC-AGI-3 report, which explicitly treats public-set evaluation and benchmark-specific preparation as threats to validity and withholds public-set leaderboard scores for this reason.[28]

This shift in correlation reveals several key dynamics. First, it shows that models can effectively adapt to fixed benchmarks. In the case of ARC-AGI-2, older models scored poorly simply because they were unfamiliar with the abstract task format, which is distinct from natural text. Newer models that underwent sufficient training with the knowledge of ARC-AGI-2 became familiar with the abstract task format, yielding significantly improved benchmark scores. The strong correlation between ARC-AGI-2 scoring and CC for these models indicate that their performance is now governed by fundamental mechanical limits in procedural state preservation. The mechanistic evidence explains this dynamic: these models have not developed a superior, generalizable ability to maintain procedural states. Instead, they have effectively adapted to the benchmark's specific formats. This demonstrates that fixed benchmarks allow models to achieve higher scores through task familiarity, masking the fact that their fundamental capacity for exact rule execution remains severely bounded. This blind spot necessitates minimal, non-semantic assays to uncover true operational limits.

5 Discussion

LLMs are distinguished by three capacities that are routinely conflated: accessing long contexts, solving benchmark tasks, and executing procedures reliably. SCC provides a minimal, direct assay for the third, serving as the basis for testing reliable rule execution within LLMs. It asks a minimal question: can a model apply a simple procedural update, preserve the resulting state within its context, and return the correct final value? Our findings demonstrate that the answer is broadly negative, even for systems that excel on complex reasoning, coding, and long-context benchmarks. This does not suggest that standard knowledge-dependent benchmarks are uninformative. Rather, it exposes a critical blind spot. High scores on these aggregate tasks do not, by themselves, establish underlying procedural reliability.

The observed model behavior results with counting is consistent with finite rule-like patterning. Within a supported regime, the model follows internal trajectories that produce exact answers. Beyond that regime, it continues to emit

plausible numerical outputs without preserving the rule-defined states. Thus, the model can appear to execute a procedure while no longer carrying the variable that the procedure requires.

The mechanistic findings help explain why prompting and additional test-time computation do not reliably solve the problem. During successful counting, the model constructs distributed state trajectories that support exact outputs. Near the CC boundary, those trajectories cease to be decoder-usable. Because this foundational procedural state is lost rather than simply obscured, longer generation cannot reliably reconstruct it. Therefore, while Chain-of-Thought style generation can improve semantic reasoning by providing structural scaffolding, it fails to guarantee rule execution over long horizons.[11–13,24]

The structural causes of these operational limits remain to be fully resolved. Sparse expert routing, finite-precision computation, positional encoding decay, attention sparsification, and the absence of explicit recurrence in transformers could all contribute.[29–31] Previous theoretical analyses have established that transformer architectures face inherent limits on generalized counting and formal-language generalization,[5,6,32,33] yet the failure of even basic counting in deployed systems is not yet explained by any single mechanism. Targeted ablations are needed to identify the dominant causes across models.

The conclusions from our analysis have practical consequences. Autonomous systems used for coding, tool use, planning, or decision support must maintain task states across extended interactions, including constraints, intermediate variables, commitments, and action preconditions.[18–21] Our results reveal that such state maintenance is locally reliable yet globally brittle. The broader implication is that deployed LLMs can have outputs that closely match a requested procedure for thousands of steps, masking the fact that their underlying computational mechanics are bounded and highly vulnerable to sudden, silent failure. To improve model reliability, it is insufficient to solely increase model size, perform extra training, and use more generated tokens. Rather, major improvements will require architectural support for persistent variables, explicit intermediate states, external memory, recurrence, and verifiable execution traces.[34–37]

Several limitations of our metric and study should be noted. Homogeneous counting is intentionally artificial and does not measure the full range of procedures required in natural tasks. Proprietary model evaluations depend on externally served systems whose preprocessing, hidden prompts, and reasoning-token accounting are not fully observable. Mechanistic analyses are restricted to open-weight models and should be interpreted as evidence for representative mechanisms rather than proof that every model fails in exactly the same way. Nevertheless, the behavioral dissociation across model families indicates that reliable rule execution should be measured directly rather than inferred from benchmark performance, nominal context length, or inference-time computation.

Methods

Model set and evaluation scope

We evaluated 126 language model variants, including proprietary systems and open-weight architectures. The full model catalogue, parameter metadata, context-window metadata and evaluation records are provided in Supplementary Note 5 and Supplementary Table 4. The set included instruction-tuned and reasoning-augmented variants spanning a broad range of parameter counts and nominal context lengths. This range allowed us to test whether SCC reflects a general capability axis rather than an idiosyncrasy of a particular tokenizer, serving stack or model family.

All evaluations were conducted with tool use disabled. For externally served models, we recorded the model identifier, evaluation date, decoding configuration, token-usage metadata, parsed response, target count and success or failure classification. These records were retained to account for possible changes in served model behavior over time. The complete proprietary-model SCC evaluation cost approximately US$200 in API spend, excluding local open-weight mechanistic analyses.

Prompt construction and response parsing

Counting prompts were generated from homogeneous item sequences. The baseline stimulus consisted of repeated lowercase characters separated by a comma and a space. Exact prompt templates, generation settings and retry logic are provided in Supplementary Note 2. The target value was the number of repeated items, not the number of tokens. We verified tokenization lengths for all evaluated tokenizers to identify compression anomalies and quantify the relationship between item count and input-token count.

Models were instructed to return the final count as a single integer. We did not enforce this restriction by constraining the decoder with a numerical grammar or a small `max_tokens` limit. This allowed natural failure modes, including prompt echoing, blank outputs, formatting artifacts and spurious reasoning traces, to be observed. Generated responses were parsed for the last valid integer and compared with the exact target item count. Outputs without a parsable integer

were classified as failures. For each trial, we recorded the target count, generated response, parsed prediction, binary correctness, absolute error, input-token count, output-token count and model metadata.

Adaptive estimation of Stable Counting Capacity

SCC was quantified with an adaptive randomized ladder. The base center sequence length was initialized at $L = 32$. For each length iteration, we sampled $K = 16$ independent discrete target counts x_k by drawing $y_k \sim U(0.8L, 1.2L)$ and rounding to the nearest integer, $x_k = \mathrm{round}(y_k)$. Given model predictions $\hat{x}_k$, we computed

$$\mathrm{nMAE}(L) = \frac{1}{K} \sum_{k=1}^{K} \frac{|\hat{x}_k - x_k|}{L}.$$

Tiers with $\mathrm{nMAE}(L) < 0.05$ were classified as stable and triggered upward expansion of L. Tiers above the threshold initiated binary refinement between the highest verified stable tier and the lowest failed tier. If a model failed at the minimal initial length $L = 32$, binary refinement was bypassed and its SCC was recorded categorically as < 32, represented as 0 in aggregate plots.

This design makes coarse magnitude estimation insufficient. Because the target count varies within each tier, a model that has collapsed to a common number or an approximate length estimate cannot consistently satisfy the error threshold. The randomized ladder bounds the false-positive rate for guessing-based strategies at approximately 0.025% (Supplementary Note 3). SCC therefore estimates a stable operating limit of the largest length regime over which the model can reliably preserve the exact tally required by the rule.

Hierarchical rule tracking assay

To test whether the limitation extends beyond one-dimensional counting, we designed a hierarchical rule-tracking assay (Supplementary Note 6). Models were presented with syntactically structured records. Each record contained a `KEY` token, a deeply nested `PATH` field using alternating bracket types and a `SIDE` field containing random distractor tokens.

A record was counted as a valid match if and only if the `KEY` token exactly matched the deepest token inside the `PATH` field. Models were asked to compute the total number of valid matches. This task requires applying a simple equality rule across many records while ignoring distractors and maintaining a cumulative count. Stability was quantified using the same adaptive randomized ladder used for homogeneous counting. To prevent total record number from serving as a proxy for the answer, each prompt included both valid records and negative distractors, with distractor number scaled relative to the target match count.

Cross-benchmark alignment and paired analyses

To compare SCC with standard task performance, we cross-referenced model results against public benchmark metrics, including GPQA Diamond, ARC-AGI-2, SWE-Bench Verified and OTIS Mock AIME.[9,10,22,38,39] Correlation coefficients were calculated across intersecting model subsets for 47 public benchmarks (Supplementary Figs. 2 to 4).

We also performed matched-pair analyses comparing base models with their reasoning-augmented counterparts. For each pair, we extracted SCC boundaries and average total token consumption at the stable boundary. We plotted token-expenditure multipliers against the change in capability (ΔSCC) in Fig. 5e to evaluate whether additional generation-time computation expanded stable state preservation.

Matched dual task counting controls

To measure interference between exact state tracking and complex reasoning, we constructed matched dual-task prompts in which a marker counting task was paired with a benchmark-style question. Target counts ranged from 32 to 96. Using `gpt-5.4-mini`, we evaluated a primary condition requiring the model to output a JSON object containing both the benchmark answer and the primary sequence count. Benchmark subtasks were drawn from BBH, CRUXEval-O, MATH-500 and MMLU-Pro, excluding mathematics and computer-science categories where specified.[40–43]

To isolate reasoning load from raw sequence length, we estimated the token length of each sampled benchmark prompt and synthesized matched-length distractors. Controls included irrelevant code snippets and a secondary independent counting task using a different marker, such as `b`. Six independent trials were sampled per count and category. Responses were parsed for the required JSON count field, and count errors were aggregated to quantify how additional task demands affected counting accuracy.

Dense sweeps and motif perturbations

To map the transition near collapse, we conducted dense actual-count sweeps around the SCC boundary identified by the randomized ladder. Parsed predictions were aggregated to quantify the transition from exact outputs to failed outputs. Failed-output attractors were identified by aggregating incorrect predicted values across the sweep (Fig. 2d,e).

We also varied the repeated character and delimiter syntax. Relative SCC was compared with tokenizer-specific compression rates to distinguish effects of raw token volume from syntax-dependent changes in the learned counting trajectory (Fig. 2f).

Residual stream projections and targeted causal interventions

Mechanistic analyses were performed primarily on Gemma 3 27B-it, for which full activation caching was feasible.[25,26] For each counting prompt, we cached residual-stream activations at repeated-token positions and at the final assistant-prefix token immediately preceding generation. To test whether sequence position was linearly readable, we fit one-dimensional linear projections to residual-stream activations from successful counting regimes and evaluated how these projections generalized across sequence lengths and layers (Fig. 4b). We performed the same projection analysis on Qwen 3.5 35B-A3B to test whether similar count-related coordinates appeared in a structurally distinct mixture-of-experts model.

To connect latent topology with output generation, we computed teacher-forced minimum logit margins for exact correct-answer sequences. The logit-gap curve served as a continuous measure of whether the decoder preferred the correct count at the point of generation (Fig. 4c).[44,45]

For latent manipulation and donor patching within the stable regime, we evaluated both full-sequence token replacement and targeted final-token substitution (Fig. 4i–l). For a base prompt of count 10, we extracted successful donor activations corresponding to target counts 1 through 26. For sequence-token interventions, hidden states of repeated sequence tokens from the donor were resampled by linear interpolation to match the length of the base sequence, then used to replace the corresponding residual states. For final-token interventions, patching was restricted to the assistant-prefix token immediately before decoding. Testing these interventions across layers isolated a shift in causal count representation from intermediate sequence tokens, such as layer 31, to the final response prefix, such as layer 53.

To test whether the linearly decodable progress coordinate was sufficient for state maintenance, we performed targeted counter-projection clamping on Gemma 3 27B-it during the forward pass using PyTorch hooks. We first fit a one-dimensional progress geometry from successful sequence counts within the stable regime, extracting the center, unit direction and linear fit parameters. For failed target counts beyond the stability boundary, we intercepted the residual stream at the final prompt token at selected intervention layers. We calculated the scalar projection of the hidden state onto the learned counter direction after subtracting the center, then computed the target projection implied by the stable linear fit. The hidden state was shifted only along the counter direction by the difference between the target and current projections, leaving orthogonal components unchanged. We evaluated 40 distinct trials, recording whether exact greedy generation was rescued, how teacher-forced minimum correct-logit margins changed and how residual projection errors evolved across downstream layers.

Sparse autoencoders and feature coalitions

Sparse autoencoder analyses used Gemmascope 2 to decompose residual representations into sparse feature dictionaries (Fig. 4d).[26,27] Features were ranked by Pearson correlation with count within verified successful sequence bounds. We recalculated feature rankings under motif variants, computed Jaccard overlap among top-ranked feature sets and measured rank displacement of baseline top features under syntax perturbations (Fig. 4e–h).

Data Availability

All extracted sparse autoencoder (SAE) feature activation datasets, raw inference logs for all 126 evaluated model variants, including API token-usage metadata and parsed outputs, and dual-task measurements are available at GitHub (`https://github.com/txdai/Counting-as-a-minimal-probe-of-LM-reliability`).

Code Availability

All prompt-generation scripts, adaptive randomized ladder evaluation code and analysis code for extracting internal activations via Gemmascope 2 and reproducing the Gemma 3 and Qwen 3.5 mechanistic analyses are available at GitHub (`https://github.com/txdai/Counting-as-a-minimal-probe-of-LM-reliability`).

References

[1] Trinh, T. H., Wu, Y., Le, Q. V., He, H. & Luong, T. Solving olympiad geometry without human demonstrations. *Nature* **625**, 476–482 (2024).

[2] He, C. *et al.* OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems (2024). URL `https://aclanthology.org/2024.acl-long.211/`.

[3] Chen, M. *et al.* Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374* (2021).

[4] Bai, Y. *et al.* LongBench: A bilingual, multitask benchmark for long context understanding (2024). URL `https://aclanthology.org/2024.acl-long.172/`.

[5] Yehudai, G. *et al.* When can transformers count to n? (2024). URL `https://openreview.net/forum?id=WULjblaCoc`.

[6] Delétang, G. *et al.* Neural networks and the chomsky hierarchy (2023). URL `https://openreview.net/forum?id=WbxHAzkeQcn`.

[7] Chen, S. *et al.* Benchmarking large language models under data contamination: A survey from static to dynamic evaluation (2025). URL `https://aclanthology.org/2025.emnlp-main.511/`.

[8] Cosma, A., Ruseti, S., Radoi, E. & Dascalu, M. The strawberry problem: Emergence of character-level understanding in tokenized language models (2025).

[9] Rein, D. *et al.* GPQA: A graduate-level google-proof Q&A benchmark (2023). URL `https://arxiv.org/abs/2311.12022`. arXiv:2311.12022.

[10] Jimenez, C. E. *et al.* SWE-bench: Can language models resolve real-world GitHub issues? (2024). URL `https://proceedings.iclr.cc/paper_files/paper/2024/hash/edac78c3e300629acfe6cbe9ca88fb84-Abstract-Conference.html`.

[11] Wei, J. *et al.* Chain-of-thought prompting elicits reasoning in large language models (2022). URL `https://proceedings.neurips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html`.

[12] Kojima, T., Gu, S. S., Reid, M., Matsuo, Y. & Iwasawa, Y. Large language models are zero-shot reasoners (2022). URL `https://papers.nips.cc/paper_files/paper/2022/hash/8bb0d291acd4acf06ef112099c16f326-Abstract-Conference.html`.

[13] Snell, C., Lee, J., Xu, K. & Kumar, A. Scaling LLM test-time compute optimally can be more effective than scaling model parameters (2024). URL `https://arxiv.org/abs/2408.03314`. arXiv:2408.03314.

[14] Liang, P. *et al.* Holistic evaluation of language models. *Transactions on Machine Learning Research* (2023). URL `https://openreview.net/forum?id=iO4LZibEqW`.

[15] Hendrycks, D. *et al.* Measuring massive multitask language understanding (2021). URL `https://openreview.net/forum?id=d7KBjmI3GmQ`.

[16] Srivastava, A. *et al.* Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research* (2023). URL `https://openreview.net/forum?id=uyTL5Bvosj`.

[17] White, C. *et al.* Livebench: A challenging, contamination-free LLM benchmark (2024). URL `https://arxiv.org/abs/2406.19314`. arXiv:2406.19314.

[18] Hai, N. L., Nguyen, D. M. & Bui, N. D. Q. REPOEXEC: Evaluate code generation with a repository-level executable benchmark (2024). URL `https://arxiv.org/abs/2406.11927`. arXiv:2406.11927.

[19] Wu, Y., Hee, M. S., Hu, Z. & Lee, R. K.-W. Longgenbench: Benchmarking long-form generation in long context LLMs (2024). URL `https://arxiv.org/abs/2409.02076`. arXiv:2409.02076.

[20] Yao, S. *et al.* ReAct: Synergizing reasoning and acting in language models (2023). URL `https://openreview.net/forum?id=WE_vluYUL-X`.

[21] Schick, T. *et al.* Toolformer: Language models can teach themselves to use tools (2023). URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html.

[22] Chollet, F., Knoop, M., Kamradt, G., Landers, B. & Pinkard, H. ARC-AGI-2: A new challenge for frontier AI reasoning systems (2025). URL https://arxiv.org/abs/2505.11831. arXiv:2505.11831.

[23] Barbero, F. *et al.* Interpreting the repeated token phenomenon in large language models (2024).

[24] DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948* (2025).

[25] Gemma Team. Gemma 3 technical report (2025). URL https://arxiv.org/abs/2503.19786. arXiv:2503.19786.

[26] McDougall, C. *et al.* Gemma scope 2 - technical paper (2025). URL https://storage.googleapis.com/deepmind-media/DeepMind.com/Blog/gemma-scope-2-helping-the-ai-safety-community-deepen-understanding-of-complex-language-model-behav Gemma_Scope_2_Technical_Paper.pdf. Google technical paper.

[27] Gao, L. *et al.* Scaling and evaluating sparse autoencoders (2025). URL https://openreview.net/forum?id=tcsZt9ZNKD.

[28] ARC Prize Foundation. ARC-AGI-3: A new challenge for frontier agentic intelligence (2026). arXiv:2603.24621.

[29] Shazeer, N. *et al.* Outrageously large neural networks: The sparsely-gated mixture-of-experts layer (2017). URL https://openreview.net/forum?id=B1ckMDqlg.

[30] Dettmers, T., Pagnoni, A., Holtzman, A. & Zettlemoyer, L. QLoRA: Efficient finetuning of quantized LLMs (2023). URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1feb87871436031bdc0f2beaa62a049b-Abstract-Conference.html.

[31] Lin, J. *et al.* AWQ: Activation-aware weight quantization for LLM compression and acceleration (2024). URL https://proceedings.mlsys.org/paper_files/paper/2024/hash/42a452cbafa9dd64e9ba4aa95cc1ef21-Abstract-Conference.html.

[32] Hahn, M. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics* **8**, 156–171 (2020). URL https://aclanthology.org/2020.tacl-1.11/.

[33] Strobl, L., Merrill, W., Weiss, G., Chiang, D. & Angluin, D. What formal languages can transformers express? a survey. *Transactions of the Association for Computational Linguistics* **12** (2024). URL https://aclanthology.org/2024.tacl-1.30/.

[34] Dai, Z. *et al.* Transformer-XL: Attentive language models beyond a fixed-length context (2019). URL https://aclanthology.org/P19-1285/.

[35] Bulatov, A., Kuratov, Y. & Burtsev, M. S. Recurrent memory transformer (2022). URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/47e288629a6996a17ce50b90a056a0e1-Abstract-Conference.html.

[36] Wu, Y., Rabe, M. N., Hutchins, D. & Szegedy, C. Memorizing transformers (2022). URL https://openreview.net/forum?id=TrjbxzRcnf-.

[37] Borgeaud, S. *et al.* Improving language models by retrieving from trillions of tokens (2022). URL https://proceedings.mlr.press/v162/borgeaud22a.html.

[38] SWE-bench. SWE-bench leaderboard and verified subset (2026). URL https://www.swebench.com/. Official benchmark website.

[39] Epoch AI. OTIS mock AIME 2024–2025 (2025). URL https://epoch.ai/benchmarks/otis-mock-aime-2024-2025. Official benchmark description page.

[40] Suzgun, M. *et al.* Challenging BIG-bench tasks and whether chain-of-thought can solve them (2023). URL https://aclanthology.org/2023.findings-acl.824/.

[41] Gu, A. *et al.* CRUXEval: A benchmark for code reasoning, understanding and execution (2024). URL https://proceedings.mlr.press/v235/gu24d.html.

[42] Lightman, H. *et al.* Let's verify step by step (2024). URL https://openreview.net/forum?id=v8L0pN6EOi.

[43] Wang, Y. *et al.* MMLU-pro: A more robust and challenging multi-task language understanding benchmark (2024). URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/ad236edc564f3e3156e1b2feafb99a24-Abstract-Datasets_and_Benchmarks_Track.html.

[44] Elhage, N. *et al.* A mathematical framework for transformer circuits (2021). URL https://transformer-circuits.pub/2021/framework/index.html. Transformer Circuits thread.

[45] Shai, A. S., Marzen, S. E., Teixeira, L., Oldenziel, A. G. & Riechers, P. M. Transformers represent belief state geometry in their residual stream (2024). URL https://arxiv.org/abs/2405.15943. arXiv:2405.15943.

Acknowledgements

This work was funded by the National Science Foundation under Award Number 2103301 and The Packard Foundation under grant number 2016-65132.

Author Contributions

T.D. conceived the study, designed the mechanical assays and evaluation pipeline, performed the large-scale model evaluations, carried out the correlation analyses and mechanistic investigations, and wrote the manuscript. J.A.F. supervised the project and contributed to interpretation and editing of the manuscript.

Competing Interests

The authors declare no competing interests.

Correspondence and requests for materials

Correspondence and requests for materials should be addressed to J.A.F. (jonfan@stanford.edu).