Test of Time: Rethinking Temporal Signal of Benchmark Contamination

Anonymous Authors[1]

Abstract

Post-cutoff performance decay of LLMs has been widely interpreted as a temporal signal for benchmark contamination, where public information released before the training cutoff may have been included into training corpora and inflated model performance by memorization. We critically examine this view and demonstrate that this temporal signal is highly sensitive to how benchmark questions are constructed, even if the underlying source material remains invariant. Specifically, we show that LLM-transformed questions can produce remarkably different temporal patterns compared to fill-in-the-blank (cloze) questions directly retrieved from the very same documents. We validate this effect on prior benchmarks that report clear post-cutoff decay (LiveCodeBench), and show that a simple LLM-driven transformation of the same problems can effectively remove the temporal pattern. We further provide a mechanistic understanding of this phenomenon using influence function analysis. Overall, our results suggest that post-cutoff performance decay is a sensitive contamination signal, motivating more robust contamination probes for reliable LLM evaluation.

1. Introduction

As frontier large language models are trained on increasingly larger datasets, publicly available benchmarks may be included into massive web-scale training corpora (Xu et al., 2024; Ravaut et al., 2025). This leads to the problem of **benchmark contamination**, where models may answer evaluation questions by memorizing leaked items rather than reasoning. As a result, contamination poses a fundamental threat to the reliability of evaluation: it can conflate memorization with genuine capability (Deng et al., 2024; Dong et al., 2024).

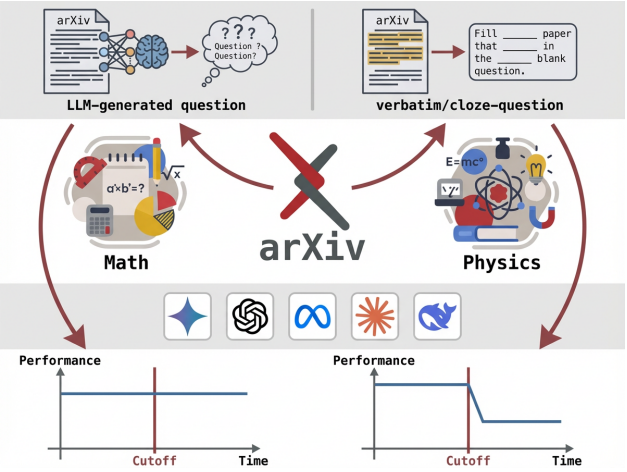


Figure 1. Overview of the temporal analysis framework: based on the same underlying source material (arXiv papers in math and physics), we compare the temporal pattern of model performance in 2 types of QAs: verbatim/cloze questions directly retrieved from public sources may demonstrate post-cutoff performance decay, whereas questions generated/transformed by LLMs do not.

To detect such contamination, a growing line of work generates variants of existing benchmarks by way of rephrasing (Golchin & Surdeanu, 2025; Xu et al., 2025), expert annotation (Zhang et al., 2024; Huang et al., 2025), or LLM-based synthesis of new questions (Kassem et al., 2025; Liang et al., 2026). Most previous work in this direction consistently reported performance degradation on variants comparing to the original benchmark, a phenomenon widely interpreted as evidence to the brittleness of LLM reasoning and potential contamination (Djiré et al., 2025).

Another widely used approach is *temporal analysis*, which compares performance on questions released before vs. after a model's training cutoff date (Li & Flanigan, 2024; Golchin & Surdeanu, 2024). It's considered as a simple yet intuitive contamination probe: if models perform worse on questions released after their cutoff dates (which we term as **post-cutoff performance decay**), this gap is frequently interpreted as evidence that pre-cutoff questions (or close variants) leaked into training data. Previous work on this line has reported such post-cutoff performance decay on questions directly retrieved from public websites across

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the FoGen Workshop at ICML 2026. Do not distribute.

multiple domains, including math, coding and reasoning at large (Jain et al., 2025; White et al., 2025; Roberts et al., 2024).

However, a recent study (Zhang et al., 2025) reported a counter-intuitive *lack* of post-cutoff decay on LLM-generated questions from mathematics arXiv papers, which is commonly present in web-scale training corpora used for frontier models. This discrepancy raises an intriguing question: why can temporal analysis show clear decay on verbatim benchmarks like LiveCodeBench (Jain et al., 2025), yet fail to show decay on LLM-synthesized questions from time-stamped sources?

As illustrated in Figure 1, we first investigated this counter-intuitive observation by validating the lack of post-cutoff performance decay at scale: specifically, we extended temporal analysis to 1,643 LLM-generated questions from 20,277 arXiv papers spanning 26 months across 8 models and 2 domains (mathematics and physics), confirming that the lack of post-cutoff decay is a consistent pattern across different model families and datasets. We hypothesize that the key factor behind this phenomenon may be the LLM generation process that transformed the base material (the same arXiv papers) enough such that models can no longer recall them from training corpora.

We put this hypothesis to the test by constructing cloze (fill-in-the-blank) questions using the same set of papers and find that post-cutoff performance decay appears even though both cloze and LLM-generated questions are based on the same source papers. Furthermore, we validated this hypothesis beyond arXiv papers by showing that LLM generation could also effectively remove post-cutoff decay on LiveCodeBench (Jain et al., 2025) (which previously reported clear post-cutoff performance decay) and a FreshQA-style dataset (Vu et al., 2024) based on Wikipedia without changing the underlying solution.

Finally, we use influence function analysis on provably contaminated training documents to provide a mechanistic understanding of our observation: when given verbatim/cloze questions simply copy-pasted from arXiv papers, models were able to identify source documents among the most influential training documents, yet LLM-generated QA based on the same set of papers were much harder to identify.

Taken together, our findings call for a careful re-examination of post-cutoff performance decay as a temporal signal for benchmark contamination, as it can be highly sensitive to different benchmark construction methodologies, among other potential confounders. We further argue that future studies should explore more robust contamination probes under various benchmark construction methods.

2. Related Work

Benchmark Contamination. Benchmark contamination refers to overlap between a model's training data and evaluation benchmarks, which can inflate scores via memorization rather than genuine reasoning (Dong et al., 2024). This overlap can take several forms, including exact inclusion, near-duplicates, and derivative content (Xu et al., 2024; Deng et al., 2024). Unfortunately, direct contamination audit is extremely difficult at scale (Cheng et al., 2025) because most frontier model developers do not publicly release their training data (*open-data*) though some of them release their model weights (*open-weight*).

Probe Contamination by LLM-Generated Evaluation. A widely used method for detecting contamination is to generate variants of existing benchmark questions and check whether model performance degrades (Golchin & Surdeanu, 2025; Xu et al., 2025) on the variant. Previous work has explored simple paraphrasing using GPT-4o (Djiré et al., 2025), perturbations using human expert annotators or stronger LLMs (Yang et al., 2023; Zhang et al., 2024; Huang et al., 2025), and even fully synthetic questions generated by LLMs (Kassem et al., 2025; Liang et al., 2026). These studies consistently reported performance degradation on the generated variants, which showcased the brittleness of LLM performance on reasoning benchmarks.

Probe Contamination by Temporal Analysis. Temporal analysis compares performance on questions released before versus after a model's announced training cutoff (Li & Flanigan, 2024; Golchin & Surdeanu, 2024). It has been used in coding benchmarks including LiveCodeBench (Jain et al., 2025) and LiveBench (White et al., 2025), as well as in mathematics benchmarks with clearly dated problem releases (Roberts et al., 2024). Beyond contamination audits, temporal splits are also used to study how quickly model knowledge goes stale and how retrieval or search augmentation can mitigate this issue (Vu et al., 2024). Across these settings, a recurring finding is that LLMs often perform worse on post-cutoff questions when the evaluation uses questions from public sources.

3. Temporal Analysis as a Measure of Contamination Detection

Temporal analysis is a widely used measure for detecting contamination: if models perform better on questions released before their knowledge cutoff than on those released after in the same dataset, the resulting gap (*post-cutoff performance decay*) is often interpreted as evidence that pre-cutoff materials may have leaked into training data.

In this section, we first validate the lack of post-cutoff performance decay on RealMath (Zhang et al., 2025) at scale.

Table 1. Examples of synthesized QA questions and corresponding CLOZE questions from the same arXiv papers.

Math — arXiv:2406.19979v2 - On quantitative convergence for stochastic processes

Synthesized QA: Given parameters $\varepsilon > 0$, $K > 0$, and a function $g : \mathbb{N} \to \mathbb{N}$, what explicit bound N in terms of ε, K, and g guarantees that for any monotone sequence (x_n) in $[-K, K]$ there exists some $n \leq N$ such that $|x_i - x_j| < \varepsilon \quad$ for all $n \leq i \leq j \leq n + g(n)$?

CLOZE Question (from abstract)

We develop a general framework for extracting highly uniform bounds on local stability for [blank1] processes in terms of information on fluctuations or crossings. This includes a large class of [blank2] : As a corollary of our main abstract result, we obtain a quantitative version of Doob's convergence theorem for [blank3] -sub- and supermartingales, but more importantly, demonstrate that our framework readily extends to more complex stochastic processes such as [blank4] , thus paving the way for future applications in stochastic optimization. Fundamental to our approach is the use of ideas from logic, particularly a careful analysis of the [blank5] structure of probabilistic statements and the introduction of a number of abstract notions that represent stochastic convergence in a quantitative manner. In this sense, our work falls under the 'proof mining' program, and indeed, our quantitative results provide new examples of the phenomenon, recently made precise by the first author and Pischke, that many proofs in probability theory are proof-theoretically tame, and amenable to the extraction of quantitative data that is both of low complexity and independent of the underlying probability space.

Physics — arXiv:2407.02415v2 - The Symplectic Schur Process

Synthesized QA: Let $i(\tau) = \left\lfloor \frac{9n}{8} + \sqrt{\frac{27n}{64}}\,\tau \right\rfloor$ and $u(\alpha) = -n + \left\lfloor \left(\frac{n}{12}\right)^{1/4} \alpha \right\rfloor$. For a fixed $k \in \mathbb{Z}_{\geq 1}$ and real parameters $\tau_1, \ldots, \tau_k$ and $\alpha_1, \ldots, \alpha_k$, define $i_\ell = i(\tau_\ell)$ and $u_\ell = u(\alpha_\ell)$. What is the limit as $n \to \infty$ of

$$\det_{1 \leq \ell, \ell' \leq k} \left[\left(\tfrac{n}{12}\right)^{1/4} \left(\delta_{i_\ell, i_{\ell'}}\, \delta_{u_\ell, u_{\ell'}} - K^{\mathrm{SSP}}(i_\ell, u_\ell; i_{\ell'}, u_{\ell'}) \right) \right] ?$$

CLOZE Question (from abstract)

We define a measure on tuples of partitions, called the [blank1] Schur process, that should be regarded as the right analogue of the Schur process of [blank2] for the Cartan type C. The weights of our measure include factors that are universal symplectic characters, as well as a novel family of "Down-Up Schur functions" that we define and for which we prove new identities of [blank3] -type. Our main structural result is that the point process corresponding to the symplectic Schur process is [blank4] and we find an explicit correlation kernel. We also present dynamics that preserve the family of symplectic Schur processes and explore an alternative sampling scheme, based on the [blank5] insertion algorithm, in a special case. Finally, we study the asymptotics of the Berele insertion process and find explicit formulas for the limit shape and fluctuations near the bulk and the edge. One of the limit regimes leads to a new kernel that resembles the symmetric Pearcey kernel.

We extend their setting to a longer 26-month time range and multiple domains (mathematics and physics), and evaluate more diverse models with distinct developers & cutoff dates to test whether the lack of post-cutoff decay is a consistent pattern. We then use the resulting evidence to motivate our hypothesis, which we subsequently validate in Section 4.

3.1. Methodology

We retrieve 20,277 arXiv papers using the arXiv API[1] with complete metadata from May 2023 to June 2025 (26 months), which is selected to cover at least 6 months before and after the cutoff dates of all evaluated models. We retrieve papers from physics and mathematics domains to validate the findings across disciplinary boundaries. For physics, we specifically filter out papers from 5 subdomains, namely General Relativity and Quantum Cosmology, Mathematical Physics, Exactly Solvable and Integrable Systems,

Table 2. Summary of evaluated models and their knowledge cutoff dates. We carefully include 4 frontier model families (2 models per family with different cutoff dates) to test whether temporal patterns are consistent across architectures and time windows.

Model	Knowledge Cutoff
DeepSeek-R1-0528	2024.07
DeepSeek-R1	2023.10
OpenAI-o4-mini	2024.06
OpenAI-o3-mini	2023.10
Gemini-2.5-Flash	2025.01
Gemini-2.0-Flash	2024.08
Llama-4-Scout	2024.08
Llama-3.3-70B	2023.12

[1] https://info.arxiv.org/help/api/index.html

Computational Physics, and Fluid Dynamics. We found these subdomains have high density of useful theorems in our analysis and also enough paper corpus per month to generate adequate QA pairs, which is a strict requirement for a temporally balanced evaluation across months.

Following the question generation protocols of RealMath (Zhang et al., 2025), we use o4-mini (OpenAI, 2025a) to generate multi-step reasoning questions from these theorems. The resulting QA pairs are further filtered to remove trivial or duplicated ones using GPT-4.1 (OpenAI, 2025b). Each resulting question has a unique deterministic ground-truth answer and requires more than five intermediate reasoning steps to solve. We implement monthly quotas to ensure roughly equal question distribution across 26 months.

We manually inspect the questions to confirm the following quality criteria: (1) clear deterministic answers, (2) at least 5 clearly separate steps in the answer, (3) unambiguous problem statements, and (4) correct derivation from source material. This process yields 1,098 papers producing suitable questions, with human quality checks removing 47 of 1,690 initial generated questions (2.8% rejection rate), resulting in 1,643 LLM-generated questions.

Evaluation Setup. We evaluate 4 frontier model families (Table 2), where each family is represented by 2 models with different cutoff dates to test consistency across architectures and time windows. We extend evaluation across two domains (mathematics and physics) to see whether patterns generalize across scientific disciplines. All models are queried via OpenRouter API[2] using default settings without any web search access. To address potential concerns about how frontier models could be equipped with hidden retrieval-augmented generation mechanisms, we include both proprietary models (OpenAI (OpenAI, 2025a), Gemini (Deepmind, 2025)) and open-weight models (DeepSeek (DeepSeek-AI, 2025), Llama (AI, 2025; Grattafiori et al., 2024)) deployed by third-party providers, which yielded consistent trends in general.

3.2. Results

We followed standard grading protocols from RealMath (Zhang et al., 2025) except using o4-mini as a stronger judge model for evaluation. To account for statistical variance in questions per month, we normalize model performance as $\text{Accuracy}_m = \frac{C_m}{Q_m}$, where C_m is correct answers in month m and Q_m is total questions in month m. The aggregate trend across models shows no systematic post-cutoff performance decay. Table 3 outlines average pre- versus post-cutoff normalized monthly performance in physics (with additional results in Appendix D.1), with full monthly trajectories across the 26-month period provided in Appendix D.2. To mitigate monthly variance, we further aggregate performance across equal-duration time windows. Specifically, we compare the aggregate performance of n months before cutoff (nB) with n months after cutoff (nA) for $n \in \{2, 3, 4, 5\}$ in Appendix D.3. Across all aggregation windows, we observe a consistent absence of post-cutoff performance decay. We further validate the statistical significance of our findings in the following paragraphs.

Statistical Analysis. We define our unit of analysis as model-aggregated performance across temporal windows, where each observation represents the mean performance difference (post-cutoff minus pre-cutoff accuracy) for one model. This yields 16 independent observations (8 models × 2 domains). While monthly observations may exhibit temporal correlation, our aggregation approach and use of multiple model families with different cutoff dates help mitigate this concern. The mean performance change across all observations was $+2.19$ percentage points (95% CI: $[+0.61, +3.78]$ *pp*). A paired t-test confirmed this improvement was statistically significant ($t(15) = 2.95,\ p = 0.010$). This overall pattern provides statistically significant evidence that decay is not universal in LLM-generated benchmarks, supporting our central thesis that temporal decay patterns are sensitive to benchmark formulation rather than solely reflecting intrinsic contamination of underlying source materials.

Error Mode Analysis. We conducted a manual inspection of 500 randomly sampled incorrect responses to reveal three major error types: misuse of theory or formulas (51%), where models apply inappropriate theoretical principles; misinterpretation of problem statements (42%), where models misunderstand requirements; and miscalculation (7%), involving computational mistakes in multi-step derivations. Theory misuse errors often involve selecting superficially relevant but contextually inappropriate formulas, for instance, an LLM applying Fubini's Theorem without verifying the required integrability condition: $\iint |f(x, y)|\, dx\, dy < \infty$.

4. Impact of benchmark construction on temporal pattern of model performance

In Section 3, we have validated that the lack of post-cutoff performance decay is indeed a consistent pattern across various frontier model families. We hypothesize that the main driver behind this phenomenon lies in the process of LLM generation from arXiv papers, which transformed the underlying material in such a way that evaluated models can no longer recognize them even if the source papers may have been included into the training corpora.

[2]OpenRouter (https://openrouter.ai) provides unified API access to a wide range of frontier models.

Table 3. Pre- versus Post-cutoff monthly accuracy in the physics domain (averaged over the full time window of evaluation). Gap (pp) denotes Post − Pre in percentage points; additional domain results are in Appendix D.1.

Model Label	Pre-cutoff (%)	Post-cutoff (%)	Gap (pp)
DeepSeek-R1	21.1	22.7	+1.6
DeepSeek-R1-0528	21.8	26.2	+4.4
Gemini-2.0-Flash	21.6	26.7	+5.1
Gemini-2.5-Flash	33.3	39.2	+5.9
Llama-3.3-70B	15.1	15.5	+0.4
Llama-4-Scout	14.3	16.8	+2.5
o3-mini	31.3	36.2	+4.9
o4-mini	36.8	40.5	+3.7

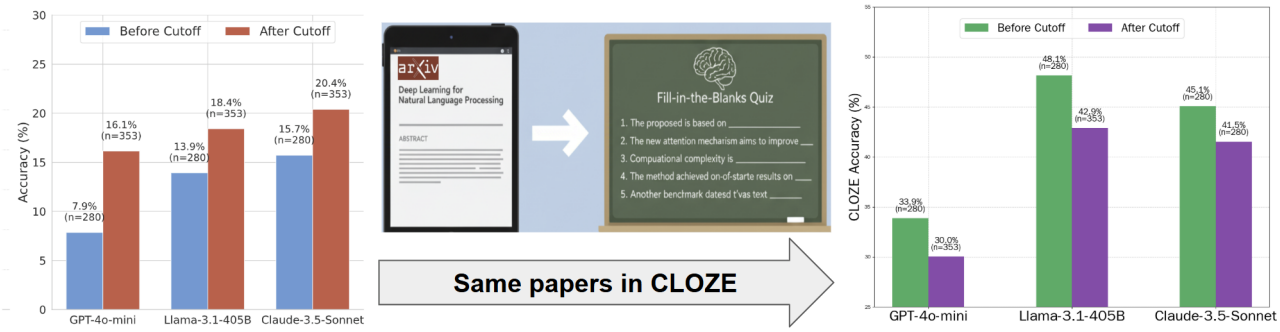


Figure 2. Temporal analysis on the same arXiv source material under two benchmark construction methods: LLM-synthesized multi-step questions (left) versus directly retrieved verbatim/cloze questions (right). Dashed vertical lines mark the evaluated models' respective knowledge cutoff dates.

4.1. Validation on cloze questions

In this section, our aim is to test this hypothesis by constructing questions based on the very same arXiv papers without LLM generation.

Methodology. Specifically, we create 5 cloze (fill-in-the-blank) questions per paper by masking semantically meaningful terms from its abstract while preserving grammatical structure (Sample questions in Table 1). Each blank represents substantive content rather than easily inferred grammatical elements from context such as "a/an". This process is performed by OpenAI-o4-mini with high reasoning effort and is validated manually. For evaluation, we employ multiple metrics: BLEU (Papineni et al., 2002), and ROUGE-L (Lin, 2004) scores and another LLM judge (with an independent o4-mini instance) with binary scoring (1 if answer matches the blanked phrase, 0 otherwise), cross-validated on 500 random instances with 94% inter-annotator agreement to human annotation.

Results. We evaluated questions from 400 papers in our dataset on 3 models (o4-mini, Gemini-2.0-Flash, Llama-4-Scout) from our study (Table 5), as well as questions from all the papers in RealMath (Zhang et al., 2025) on the models used in their work (Table 4).

In sharp contrast to the absence of decay in LLM-generated QA, we observe a predominant trend of visible post-cutoff performance decay on cloze questions across evaluated models and metrics.

Table 4. Performance decay on CLOZE questions constructed from RealMath (Zhang et al., 2025) arXiv abstracts.

Model	Pre (%)	Post (%)	Gap (pp)
(a) LLM-Judge			
GPT-4o-mini	33.86	30.03	-3.83
Llama-3.1-405B	48.14	42.89	-5.25
Claude-3.5-Sonnet	45.07	41.53	-3.54
(b) ROUGE-L			
GPT-4o-mini	39.10	35.24	-3.85
Llama-3.1-405B	49.42	44.21	-5.22
Claude-3.5-Sonnet	48.75	43.76	-4.98
(c) BLEU			
GPT-4o-mini	16.43	14.41	-2.02
Llama-3.1-405B	24.32	21.28	-3.04
Claude-3.5-Sonnet	16.55	9.95	-6.60

4.2. Validation on LiveCodeBench

To validate that LLM-driven transformation consistently acts as a confounder beyond arXiv papers, we examine LiveCodeBench (Jain et al., 2025), which has previously reported clear post-cutoff performance decay.

Table 5. Performance on CLOZE questions synthesized in this work.

Model	Pre (%)	Post (%)	Gap (pp)
(a) LLM-Judge			
o4-mini	49.70	49.00	-0.70
Gemini-2.0-Flash	40.00	37.70	-2.30
Llama-4-Scout	34.20	32.70	-1.50
(b) ROUGE-L			
o4-mini	47.93	47.68	-0.25
Gemini-2.0-Flash	41.77	39.78	-1.99
Llama-4-Scout	35.59	35.02	-0.57
(c) BLEU			
o4-mini	23.71	23.40	-0.30
Gemini-2.0-Flash	17.96	17.68	-0.28
Llama-4-Scout	15.33	15.84	+0.51

Methodology. We transform LiveCodeBench-Release-v1 (with data released between May 2023 and Mar 2024 containing 400 problems) using o4-mini with high reasoning effort to transform the problem statements while maintaining that the same algorithmic solutions apply to both transformed and original problems equally. We refer to this transformed dataset as **PerturbLiveCodeBench** following the nomenclature of previous work in this field (Huang et al., 2025). Detailed perturbation prompts and sample questions before vs. after transformation are provided in Appendix C.

Results. Following the original evaluation protocol of Jain et al. (2025), we evaluate GPT-4o and GPT-4 on PerturbLiveCodeBench as they are the same models that previously reported post-cutoff performance decay.[3] As shown in Figure 3, PerturbLiveCodeBench no longer exhibits a clear post-cutoff performance decay. This stark contrast demonstrates that LLM-based question generation systematically removes post-cutoff performance decay even on problems that clearly exhibit decay in their original form. Crucially, this validation experiment is performed in a completely different domain (coding) and benchmark construction paradigm (competitive programming), which further validates that this pattern is not unique to arXiv papers.

4.3. Validation on Wiki-based QA

We perform an additional validation experiment using a Wiki-based FreshQA-style (Vu et al., 2024) setting.

Methodology. Inspired by FreshQA-style setting, we crawl verbatim dated event text from *Wikipedia Current Events archives* and construct dated multiple-choice questions from these event lines, spread uniformly across temporal windows of pre- and post-cutoff. We use o4-mini to apply semantic transformations only to the question statement while preserving the corresponding options and final answer. We evaluate GPT-3.5-turbo, GPT-4, and GPT-4o-mini on both the original and transformed variants.

[3]Gemini-1.5 and DS-Ins-33B were deprecated by model developers at the time of experimentation.

Results. All 3 models exhibit significant post-cutoff performance decay on original dataset of the MCQs. GPT-3.5-turbo demonstrates -2.65 *pp* performance gap; GPT-4 shows a gap of -1.04 *pp*; GPT-4o-mini exhibits the largest gap of -7.59 *pp* on the original variants. We observe a stark contrast when evaluated on the LLM-transformed variants, where GPT-3.5-turbo shows a performance gap of -0.62 *pp*; GPT-4 has an astounding increase in post-cutoff performance of $+2.81$ *pp*; while GPT-4o-mini shows a change of -4.99 *pp*. This observation (Table 6) underscores the impact of benchmark construction method on temporal patterns, generally altering the pattern of post-cutoff performance decay significantly.

Table 6. Comparison of LLM performance on original versus transformed variants of Wiki-based QA. Post-cutoff performance (*Post* − *Pre*) shows significant change.

Model	Original (pp)	Transformed (pp)
GPT-3.5-turbo	-2.65	-0.62
GPT-4	-1.04	+2.81
GPT-4o-mini	-7.59	-4.99

5. Mechanistic Interpretability Analysis

In the previous sections, we focus on temporal analysis on the level of model performance by comparing accuracy score in pre- vs. post-cutoff questions. In this section, our aim is to study the underlying mechanism behind this phenomenon by studying the internal working of LLMs using mechanistic interpretability methods. Influence functions (Koh & Liang, 2017; Grosse et al., 2023) offer a powerful tool to trace and rank the most responsible data points for a given model input-output pair as shown in Figure 4. Specifically, our aim is to show that based on the same contaminated paper, LLM-transformed questions are much harder for models to trace back to their source paper than cloze questions.

5.1. Review of Influence Functions

Let $\mathcal{D}$ denote the training dataset for model $\mathcal{M}$ parameterized by θ. The influence function quantifies how a training document $z \in \mathcal{D}$ impacts the model's prediction on a test input-output pair $z_{\text{test}} = (z_p, z_c)$, where z_p is the model *input* (prompt/prefix tokens) and z_c is the model *output* (target completion tokens). Following the standard formulation (Koh & Liang, 2017), we approximate the change in loss at z_{test} if training point z were up-weighted by an infinitesimal ϵ as

$$\mathcal{I}(z, z_{\text{test}}) = -\nabla_\theta L(z_{\text{test}}, \hat{\theta})^\top H_{\hat{\theta}}^{-1} \nabla_\theta L(z, \hat{\theta}), \quad (1)$$

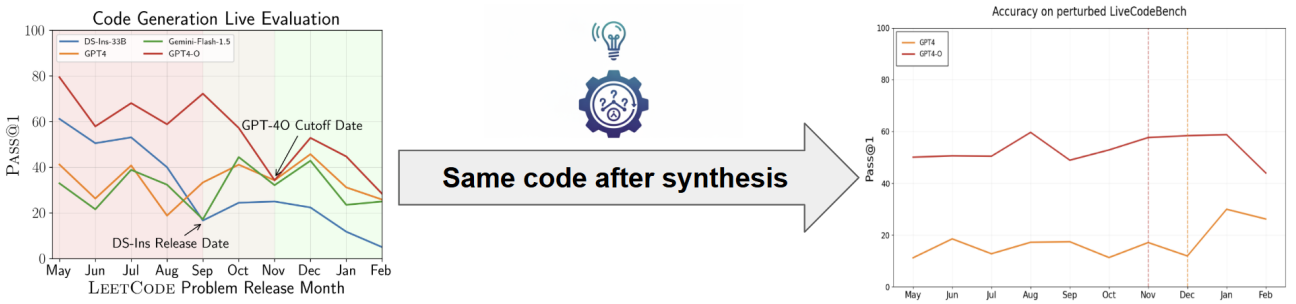


Figure 3. Validation Experiment on LiveCodeBench: temporal decay in the original LiveCodeBench (Jain et al., 2025) (left) versus the same problems after OpenAI-o4-mini transformation (right), which preserves the underlying solution code. Dashed vertical lines mark the evaluated models' respective knowledge cutoff dates.

where L is the loss function, $\hat{\theta}$ is the empirical risk minimizer, and $H_{\hat{\theta}}$ is the Hessian of the loss. Due to the high dimensionality of θ in LLMs, exact inverse-Hessian computation is intractable. In our implementation, we follow Kronfluence (Grosse et al., 2023) and define $f(\theta) = \log p(z_c \mid z_p; \theta)$, the conditional log-likelihood of the output tokens z_c given the input tokens z_p. The influence score is then approximated as

$$I_f(z) \approx -\nabla_\theta f(\theta^s)^\top (G + \lambda I)^{-1} \nabla_\theta L(z, \theta^s), \quad (2)$$

where θ^s are the pretrained weights, G is a (block-diagonal) Gauss–Newton or Fisher curvature matrix, and $\lambda > 0$ is a damping constant for numerical stability. We compute the required inverse-curvature vector products using EKFAC (George et al., 2018) as implemented in Kronfluence. Additional details are provided in Appendix B.

5.2. Experiments

Most of the mainstream open-weight models (such as DeepSeek, Llama, Qwen) release model weights but not their training data, which makes it impossible to provably tell whether a given document is part of its training data. Therefore, we make use of OLMo2-7B-Instruct (OLMo et al., 2025), one of the best-performing open-data LLMs with publicly available training dataset. We select 40 arXiv papers from the training corpora of OLMo2-7B-Instruct[4], which are provably contaminated since they form part of the training documents.

For each paper, we create both a cloze question and an LLM-generated question based on the same framework used in our previous experiments. From the full training corpus, we randomly sample 10,000 documents that includes the 40 arXiv papers we used for evaluation and rank the top 100 most influential training documents for each query. Implementation details are in Appendix B.

[4]This sample size represents the maximum processable on a single GH200 GPU within 24 hours. We use instruct model, as the base version has insufficient instruction-following capability to properly answer our questions in this setting.

Table 7. Top-K hit rates where influence function methods successfully identified the contaminated documents among the Top-K most influential training documents. The sample size for this metric is 40 papers.

Hit Rate	cloze	LLM-generated QA
Top-1	77.5%	17.5%
Top-3	100.0%	25.0%

5.3. Results

We define **Top-K Hit Rate** as the percentage of queries for which the corresponding source paper appears among the K most influential documents. For example, Top-1 Hit Rate indicates the percentage where the source paper is identified as the single most influential document.

We report the hit rate for Top-1 and Top-3 in Table 7. For cloze questions, the evaluated model was able to identify the corresponding source document as the single most influential training data point with a top-1 hit rate of 77.5%. Furthermore, the correct source document could be 100% identified within Top-3 most influential training documents, which indicated clear recall from training corpora when trying to answer these questions.

On the other hand, LLM-generated QA based on the very same papers in the training corpora yielded a significantly lower rate with Top-1 hit rate at only 17.5% and Top-3 hit rate at 25.0%. It's worth noting that the sample size for this result is very limited due to the compute-intensive nature of influence function analysis, a meaningful direction for future investigations could be scaling up this type of analysis on larger datasets to obtain more statistically significant results.

6. Discussion

Overall, we provide a complementary analysis at both the model-performance level and the level of internal mechanisms:

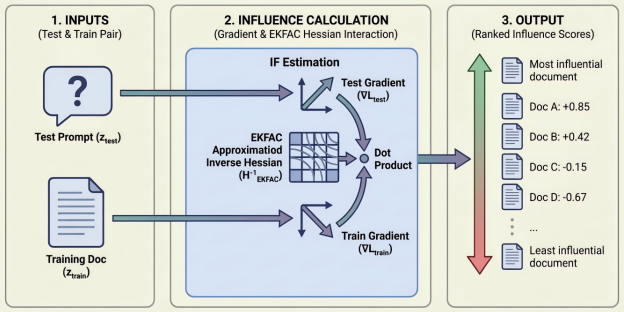


Figure 4. Overview of our influence function analysis framework.

At the performance level, we first validated the lack of post-cutoff decay across 8 frontier models over 26 months using LLM-generated questions based on arXiv papers. But when we convert these same papers to cloze questions, post-cutoff decay patterns emerge on the same models to show that the key factor lies in the LLM generation process itself rather than the underlying source material.

We further validated this hypothesis beyond arXiv papers using LiveCodeBench (Jain et al., 2025) (which has reported clear post-cutoff performance decay) and FreshQA-type (Vu et al., 2024) questions based on Wikipedia. We show that simply rephrasing these questions with LLM without changing their underlying solutions could also effectively remove post-cutoff performance decay.

At the mechanistic level, influence function analysis further confirmed our hypothesis by showing that models can identify the responsible source documents as the most influential when given cloze questions, but much less so when given LLM-generated questions.

Our key thesis centers around the fragility of post-cutoff performance decay as a temporal contamination signal. We specifically showed that simple LLM generation/transformation could effectively remove post-cutoff performance decay from questions that previously reported such decay (Jain et al., 2025). Compared to previous work that explicitly adds new knowledge after knowledge cutoff (Wu et al., 2025), we further demonstrated that even for the same underlying material, transformation alone could already effectively impact the temporal pattern of model performance without incorporating any new additional information.

7. Conclusion and Outlook

We call on the community to carefully consider the reliability of post-cutoff performance decay as a temporal signal for benchmark contamination, as such signal is highly sensitive to benchmark construction methodology, which could yield very different temporal pattern even based on the same source material.

It's worth noting that we do **not** claim that absence of performance decay means absence of contamination. Indeed, our influence function analysis reveals that LLM-generated questions from provably contaminated papers show a clear lack of post-cutoff decay. Our counter-intuitive finding affirms that post-cutoff performance decay as a temporal signal for contamination is highly sensitive to how the questions are constructed. To elucidate the practical implication of this work, we offer a few concrete recommendations for future work:

- Document detailed methodology for benchmark construction and carefully examine how the different construction methods may impact evaluation results;
- Recognize that the temporal pattern of model performance before vs. after cutoff is a highly sensitive probe that should be validated against other contamination detection methods;
- Develop more robust contamination detection methods that report model performance as more than a scalar score, while also factoring in the level of fluctuation when the same problem is transformed without changing its core.

In conclusion, we hope this work could serve as a thought-provoking piece to motivate future work on more robust contamination detection methods and more reliable benchmarking practices.

Limitations

First, our temporal analysis is based on a 26-month time window with a limited amount of papers due to budgetary concerns (processing long arXiv papers will incur considerable token usage costs). Our framework remains scalable to any arbitrary time window, where the only bottleneck factor is the number of papers available on arXiv in any user-defined time window. Similarly, influence function analysis is highly compute-intensive in nature, which is why we chose to limit our scope of experimentation to 40 papers. Lastly, we also acknowledge that the counter-intuitive rise of model performance after cutoff dates in some models remains a puzzling open question for future investigations, one potential explanation could be that the exponentially increasing number of papers on arXiv in recent years may contain much more trivial ones, subsequently making the questions considerably easier to answer. We also recognize that the variance of arXiv papers and thus the difficulty of our synthesized questions may be a confounder in our analysis, which we strive to mitigate using a large sample size and relatively long time window in the temporal analysis.

References

AI, M. Introducing LLaMA 4: Advancing multimodal intelligence, 2025. Accessed: April 18, 2026.

Cheng, Y., Chang, Y., and Wu, Y. A survey on data contamination for large language models, 2025.

Deepmind. Gemini Flash System Card, 2025.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.

Deng, C., Zhao, Y., Heng, Y., Li, Y., Cao, J., Tang, X., and Cohan, A. Unveiling the Spectrum of Data Contamination in Language Model: A Survey from Detection to Remediation. In Ku, L., Martins, A., and Srikumar, V. (eds.), *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pp. 16078–16092. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-ACL.951.

Djiré, A. E., Kaboré, A. K., Barr, E. T., Klein, J., and Bissyandé, T. F. Memorization or interpolation ? detecting llm memorization through input perturbation analysis, 2025.

Dong, Y., Jiang, X., Liu, H., Jin, Z., Gu, B., Yang, M., and Li, G. Generalization or Memorization: Data Contamination and Trustworthy Evaluation for Large Language Models. In Ku, L., Martins, A., and Srikumar, V. (eds.), *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pp. 12039–12050. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-ACL.716.

George, T., Laurent, C., Bouthillier, X., Ballas, N., and Vincent, P. Fast Approximate Natural Gradient Descent in a Kronecker Factored Eigenbasis. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 9573–9583, 2018.

Golchin, S. and Surdeanu, M. Time Travel in LLMs: Tracing Data Contamination in Large Language Models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

Golchin, S. and Surdeanu, M. Data Contamination Quiz: A Tool to Detect and Estimate Contamination in Large Language Models. *Trans. Assoc. Comput. Linguistics*, 13: 809–830, 2025. doi: 10.1162/TACL.A.20.

Grattafiori, A. et al. The llama 3 herd of models, 2024.

Grosse, R., Bae, J., Anil, C., Elhage, N., Tamkin, A., Tajdini, A., Steiner, B., Li, D., Durmus, E., Perez, E., Hubinger, E., Lukošiūtė, K., Nguyen, K., Joseph, N., McCandlish, S., Kaplan, J., and Bowman, S. R. Studying Large Language Model Generalization with Influence Functions, 2023.

Huang, K., Guo, J., Li, Z., Ji, X., Ge, J., Li, W., Guo, Y., Cai, T., Yuan, H., Wang, R., Wu, Y., Yin, M., Tang, S., Huang, Y., Jin, C., Chen, X., Zhang, C., and Wang, M. Math-perturb: Benchmarking llms' math reasoning abilities against hard perturbations, 2025.

Jain, N., Han, K., Gu, A., Li, W., Yan, F., Zhang, T., Wang, S., Solar-Lezama, A., Sen, K., and Stoica, I. LiveCodeBench: Holistic and Contamination Free Evaluation of Large Language Models for Code. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

Kassem, A. M., Mahmoud, O., Mireshghallah, N., Kim, H., Tsvetkov, Y., Choi, Y., Saad, S., and Rana, S. ALPACA AGAINST VICUNA: Using LLMs to Uncover Memorization of LLMs. In Chiruzzo, L., Ritter, A., and Wang, L. (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2025 - Volume 1: Long Papers, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, pp. 8296–8321. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.NAACL-LONG.421.

Koh, P. W. and Liang, P. Understanding Black-box Predictions via Influence Functions. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1885–1894. PMLR, 2017.

Li, C. and Flanigan, J. Task Contamination: Language Models May Not Be Few-Shot Anymore. In Wooldridge, M. J., Dy, J. G., and Natarajan, S. (eds.), *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pp. 18471–18480. AAAI Press, 2024. doi: 10.1609/AAAI.V38I16.29808.

Liang, R., Chen, J., Jia, B., Deng, B., Xie, C., Wang, Y., Jin, K., Wang, X., Zhang, L., and Wang, C. Dvd: A

robust method for detecting variant contamination in large language model evaluation, 2026.

Lin, C.-Y. ROUGE: A package for automatic evaluation of summaries. In *Text summarization branches out*, pp. 74–81, 2004.

Liu, J., Zhang, T. J., Faulkner, R., Huang, X. A., Zouhar, V., Glandorf, D., Dahlgren, I., Dagli, R., Chen, Y., Leeb, F., Truong, V. Q., Pandey, P. S., Bicker, Y., Majumder, S., Jiang, W., Qiu, Z., Chowdhury, S. P., Sachan, M., Schölkopf, B., Diab, M. T., and Jin, Z. PaperMentor: A Human-Centered Multi-Agent Writing Tutor for AI Research Papers in Overleaf. In *ACL 2026 System Demonstration Track*, 2026.

OLMo, T., Walsh, P., Soldaini, L., Groeneveld, D., Lo, K., Arora, S., Bhagia, A., Gu, Y., Huang, S., Jordan, M., Lambert, N., Schwenk, D., Tafjord, O., Anderson, T., Atkinson, D., Brahman, F., Clark, C., Dasigi, P., Dziri, N., Ettinger, A., Guerquin, M., Heineman, D., Ivison, H., Koh, P. W., Liu, J., Malik, S., Merrill, W., Miranda, L. J. V., Morrison, J., Murray, T., Nam, C., Poznanski, J., Pyatkin, V., Rangapur, A., Schmitz, M., Skjonsberg, S., Wadden, D., Wilhelm, C., Wilson, M., Zettlemoyer, L., Farhadi, A., Smith, N. A., and Hajishirzi, H. 2 OLMo 2 Furious, 2025.

OpenAI. OpenAI o3 and o4-mini System Card, 2025a.

OpenAI. Introducing GPT-4.1 in the API, April 2025b. Accessed: 2026-04-18.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. Bleu: a Method for Automatic Evaluation of Machine Translation. In Isabelle, P., Charniak, E., and Lin, D. (eds.), *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135.

Ravaut, M., Ding, B., Jiao, F., Chen, H., Li, X., Zhao, R., Qin, C., Xiong, C., and Joty, S. A comprehensive survey of contamination detection methods in large language models, 2025.

Roberts, M., Thakur, H., Herlihy, C., White, C., and Dooley, S. To the cutoff... and beyond? a longitudinal perspective on LLM data contamination. In *The Twelfth International Conference on Learning Representations*, 2024.

Ruis, L., Mozes, M., Bae, J., Kamalakara, S. R., Talupuru, D., Locatelli, A., Kirk, R., Rocktäschel, T., Grefenstette, E., and Bartolo, M. Procedural Knowledge in Pretraining Drives Reasoning in Large Language Models, 2025.

Vu, T., Iyyer, M., Wang, X., Constant, N., Wei, J., Wei, J., Tar, C., Sung, Y.-H., Zhou, D., Le, Q., and Luong, T. FreshLLMs: Refreshing Large Language Models with Search Engine Augmentation. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 13697–13720, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.813.

White, C., Dooley, S., Roberts, M., Pal, A., Feuer, B., Jain, S., Shwartz-Ziv, R., Jain, N., Saifullah, K., Dey, S., Shubh-Agrawal, Sandha, S. S., Naidu, S. V., Hegde, C., LeCun, Y., Goldstein, T., Neiswanger, W., and Goldblum, M. LiveBench: A Challenging, Contamination-Limited LLM Benchmark. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.

Wu, X., Pan, L., Xie, Y., Zhou, R., Zhao, S., Ma, Y., Du, M., Mao, R., Luu, A. T., and Wang, W. Y. AntiLeakBench: Preventing Data Contamination by Automatically Constructing Benchmarks with Updated Real-World Knowledge. In Che, W., Nabende, J., Shutova, E., and Pilehvar, M. T. (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 18403–18419. Association for Computational Linguistics, 2025.

Xu, C., Guan, S., Greene, D., and Kechadi, M.-T. Benchmark data contamination of large language models: A survey, 2024.

Xu, X., Lawrence, R., Dubey, K., Pandey, A., Ueno, R., Falck, F., Nori, A. V., Sharma, R., Sharma, A., and Gonzalez, J. RE-IMAGINE: Symbolic benchmark synthesis for reasoning evaluation. In *Forty-second International Conference on Machine Learning*, 2025.

Yang, S., Chiang, W.-L., Zheng, L., Gonzalez, J., and Stoica, I. Rethinking benchmark and contamination for language models with rephrased samples, 2023.

Zhang, H., Da, J., Lee, D., Robinson, V., Wu, C., Song, W., Zhao, T., Raja, P., Zhuang, C., Slack, D., Lyu, Q., Hendryx, S., Kaplan, R., Lunati, M., and Yue, S. A Careful Examination of Large Language Model Performance on Grade School Arithmetic. In Globersons, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J. M., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

Zhang, J., Petrui, C., Nikolić, K., and Tramèr, F. Realmath: A continuous benchmark for evaluating language models on research-level mathematics. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

A. Responsible NLP Statements

Risk and AI Usage Statement Currently we don't see any ethical risk of this study. Our dataset contains no personal identifying information or offensive content and we use datasets which are all publicly accessible under the CC-BY-4.0 license for non-commercial academic research. AI assistants (such as LLMs) were used as aids for paper writing in grammatical checks and help with visualization such as creating flaticons in figures. We used PaperMentor (Liu et al., 2026) to get feedback on our structure of the paper and overall writing.

Experimental Data Statistics Our evaluation subset contains 1,643 QA pairs (856 mathematics, 787 physics) synthesized from 20,277 arXiv papers (May 2023–June 2025). Initial corpus (4,235 mathematics, 16,042 physics) was filtered to 1,098 papers (579 mathematics, 519 physics).

B. Influence Function Analysis Details

This section provides the technical details for the influence function analysis presented in the main paper. We use influence functions (Grosse et al., 2023; Ruis et al., 2025) to mechanistically investigate the relationship between data points in pretraining documents and model output on different input queries (in this case the difference of CLOZE vs. synthesized QA questions).

B.1. Methodology

For each arXiv paper in our dataset, we compute influence scores for two paired questions: (1) a retrieval-based CLOZE question testing direct recall from the abstract, and (2) a synthesis-based question by reasoning models requiring multi-step problem solving.

B.2. Preliminaries

We briefly review influence functions and the EK-FAC (Eigenvalue-corrected Kronecker-Factored Approximate Curvature) approximations before presenting our findings. Influence functions estimate how individual training points affect model predictions, and EK-FAC provides a scalable second-order approximation that makes this analysis feasible for large LLMs.

Influence functions. Influence functions estimate how upweighting or removing a training example affects model predictions (Koh & Liang, 2017). For a query $z_{\text{test}} = (z_p, z_c)$, we are interested in the influence of a candidate training sequence $z \in \mathcal{D}$ on the model's conditional log-likelihood. Here, z_p denotes the *prefix* (the model *input*/prompt tokens), and z_c denotes the *completion* (the model *output*/target continuation tokens) whose likelihood is evaluated under the

model. We define $f(\theta) = \log p(z_c \mid z_p; \theta)$. The influence is approximated as

$$I_f(z) \approx -\nabla_\theta f(\theta^s)^\top (G + \lambda I)^{-1} \nabla_\theta L(z, \theta^s), \quad (3)$$

where θ^s are the pretrained model weights, L is the training loss, G is the Gauss-Newton Hessian, and $\lambda > 0$ is a damping constant for numerical stability.

Method used: EK-FAC approximation. We use the *Eigenvalue-Corrected Kronecker-Factored Approximate Curvature (EK-FAC)* method (George et al., 2018) integrated within the Kronfluence architecture (Grosse et al., 2023) to perform influence function analysis. EK-FAC leverages the Kronecker structure of the Fisher information matrix (or Gauss-Newton Hessian) in deep networks, enabling efficient eigendecomposition and inversion. Specifically:

- The Kronecker factors A and S (capturing input and output covariances of each layer) admit tractable eigendecompositions.
- Their Kronecker product $A \otimes S$ is diagonalized via the eigenvectors of A and S, yielding a diagonal approximation Λ whose entries capture variance in the projected pseudo-gradients.
- Damping is naturally incorporated by adding λ to the eigenvalues, so IHVPs reduce to rescaling in this eigenbasis.

Let $Q_{AS} = Q_A \otimes Q_S$ denote the Kronecker eigenbasis.

Formally, EK-FAC approximates G as

$$G \approx Q_{AS} \, \Lambda \, Q_{AS}^\top, \quad (4)$$

and the damped IHVP as

$$(G + \lambda I)^{-1} v \approx Q_{AS} \, (\Lambda + \lambda I)^{-1} \, Q_{AS}^\top v \quad (5)$$

Once the eigendecomposition is computed, IHVPs can be applied efficiently for many queries, making EK-FAC well-suited to influence-function analysis in large-scale transformer LMs.

B.3. Experimental Setup

Model and Corpus We use the OLMo2-7B-Instruct model from the OLMo 2 collection for influence function analysis. The pretraining corpus consists of diverse documents spanning scientific publications, web content, and technical documentation. For computational tractability, we sample 10,000 documents evenly distributed across the corpus to approximate the Hessian matrix.

Computational Justification The Hessian matrix encodes second-order optimization information across all model parameters. While computing the exact Hessian would require processing the entire training corpus, the primary computational bottleneck lies in gradient calculations for each document. Prior work (Ruis et al., 2025) (Appendix A.2) demonstrates that influence scores remain highly correlated even when computed using a subset of documents for Hessian approximation. Our 10,000-document sample provides a tractable yet representative estimate without materially affecting the validity of our findings, while also minimizing computational resource usage for environmental considerations.

B.4. Results Interpretation

Influence Score Distribution Influence scores quantify how much each pretraining document affects the model's output for a given query. In our experiments, scores range from approximately 25-27 million for the most influential documents down to 8-16 million for the least influential among the top 100. This variation reflects differing document impact: higher scores indicate that removing or modifying the document would induce larger changes in model behavior, while lower scores correspond to comparatively smaller influence.

C. Details for Validation on LiveCodeBench

For the perturbed LiveCodeBench (Jain et al., 2025) experiment, we use o4-mini to generate semantically equivalent variations of the original coding problems while preserving algorithmic complexity, enforcing (i) *algorithmic equivalence* (the same algorithmic approach and computational complexity as the original), (ii) *consistent transformation* throughout problem statements and test cases (e.g., `abc` → `XYZ`, `acb` → `XZY`), and (iii) *test case validity*, where expected outputs are updated to reflect the input transformations; this perturbation approach enables controlled experiments distinguishing genuine problem-solving abilities from memorization of specific problem formulations. We detail the prompts below:

```
You are tasked with creating a variation of the
    following programming problem.

The variation should:
1. Keep the exact same algorithmic approach and
    complexity
2. Change variable names, function names, and
    context
   (e.g., if it uses 'abc', use something like 'XYZ
       ')
3. Modify specific values in test cases consistently
     with the context change
4. Maintain the same difficulty level and logic

Original Problem:
{problem_text}

Original Test Examples:
```

```
{test_examples}

Provide the perturbed problem AND perturbed test
    cases in the following JSON format:
{
    "problem_statement": "...",
    "test_cases": [
        {"input": "...", "output": "...", "testtype
            ": "stdin"}
    ]
}

Make sure to perturb ALL test values consistently.
    If the original uses 'abc', 'acb', 'bac',
etc., and you change to 'XYZ', then use 'XYZ', 'XZY
    ', 'YXZ', etc. correspondingly.
```

C.1. PerturbLiveCodeBench Examples

We demonstrate representative examples of original problems and their transformed variants generated using the perturbation prompt described in Section C. The goal of these transformations is to preserve the underlying algorithmic structure and solution strategy while altering surface-level characteristics such as variable names, semantic framing, and symbolic representations.

Table 8. Examples of original vs. transformed problems from LiveCodeBench.

Example A
Original:
You need to find two numbers in an array that add up to a target sum. Return the indices of the two numbers.
Transformed:
You are given a list of integer weights. Identify two weights whose combined total matches a specified target value. Return the positions of the two weights.
Example B
Original:
Given two sorted arrays nums1 and nums2, find the median of the two sorted arrays.
Transformed:
You are given two sorted sequences dataX and dataY. Determine the median of the merged sorted collection.
Example C
Original:
Given a string containing only '(' and ')', find the length of the longest valid parentheses substring.
Transformed:
Given a sequence containing only '[' and ']', find the length of the longest valid bracket substring.

These examples illustrate three common categories of perturbations: (i) lexical substitution (e.g., `array` $\rightarrow$ `list`, `numbers` $\rightarrow$ `weights`), (ii) variable renaming, and (iii) symbol substitution (e.g., parentheses replaced with brackets). Despite these changes, the core computational tasks correspond to well-known problems such as two-sum, median of two sorted arrays, and longest valid parentheses.

Therefore, we emphasize that these transformations are intentionally lightweight and do not introduce additional reasoning complexity. As such, they provide a controlled setting for our validation experiment without conflating it with task difficulty.

D. Additional Visualizations

This section provides supplementary visualizations that contextualize the temporal analysis results in the main paper. We include (i) pre/post-cutoff averages, (ii) full month-by-month trajectories, and (iii) aggregated n-month windows to reduce variance from uneven monthly sample sizes.

D.1. Mean Accuracy Trends

The summary statistics below report normalized accuracy averaged over all pre-cutoff months versus all post-cutoff months for each model (with the sign convention Gap $=$ Post $-$ Pre). The accompanying plots visualize the same comparison, highlighting that the observed differences are small and not systematically negative after the cutoff.

Table 9 offers a compact view of these averages in the mathematics domain. Figure 5 complements it by showing the mean pre- versus post-cutoff accuracy for both domains.

D.2. Monthly Performance Trends

While the pre/post averages collapse time into two bins, the following plot shows the full month-by-month trajectories from May 2023 to June 2025. Dashed vertical lines indicate each model's reported cutoff month; visually, performance does not exhibit a consistent downward shift immediately after these boundaries.

D.3. n-Month Aggregated Results

To further mitigate month-level noise, we aggregate accuracy over fixed-width windows around the cutoff. For each model and each $n \in \{2, 3, 4, 5\}$, we compute accuracy on the n months immediately before the cutoff (denoted nB) and the n months immediately after the cutoff (denoted nA). This view tests whether conclusions are sensitive to the exact choice of temporal window.

Overall, the aggregated windows lead to the same qualitative conclusion as the monthly plots: across models and domains, post-cutoff performance does not consistently drop relative to pre-cutoff performance, and in several cases it slightly increases.

Table 9. Pre- versus post-cutoff accuracy in the Math domain. Difference $(Gap\ (pp))$ is reported as increase in post-cutoff accuracy as compared to pre-cutoff accuracy.

Model	Pre-cutoff (%)	Post-cutoff (%)	Gap (pp)
Deepseek-R1	32.8%	35.0%	+2.2
Deepseek-R1-0528	33.0%	35.8%	+2.8
Gemini-2.0-Flash	27.9%	29.4%	+1.5
Gemini-2.5-Flash	39.3%	42.4%	+3.1
Llama-3.3-70B	24.4%	19.2%	-5.2
Llama-4-Scout	19.2%	23.3%	+4.1
o3-mini	49.1%	45.6%	-3.5
o4-mini	48.8%	50.4%	+1.6

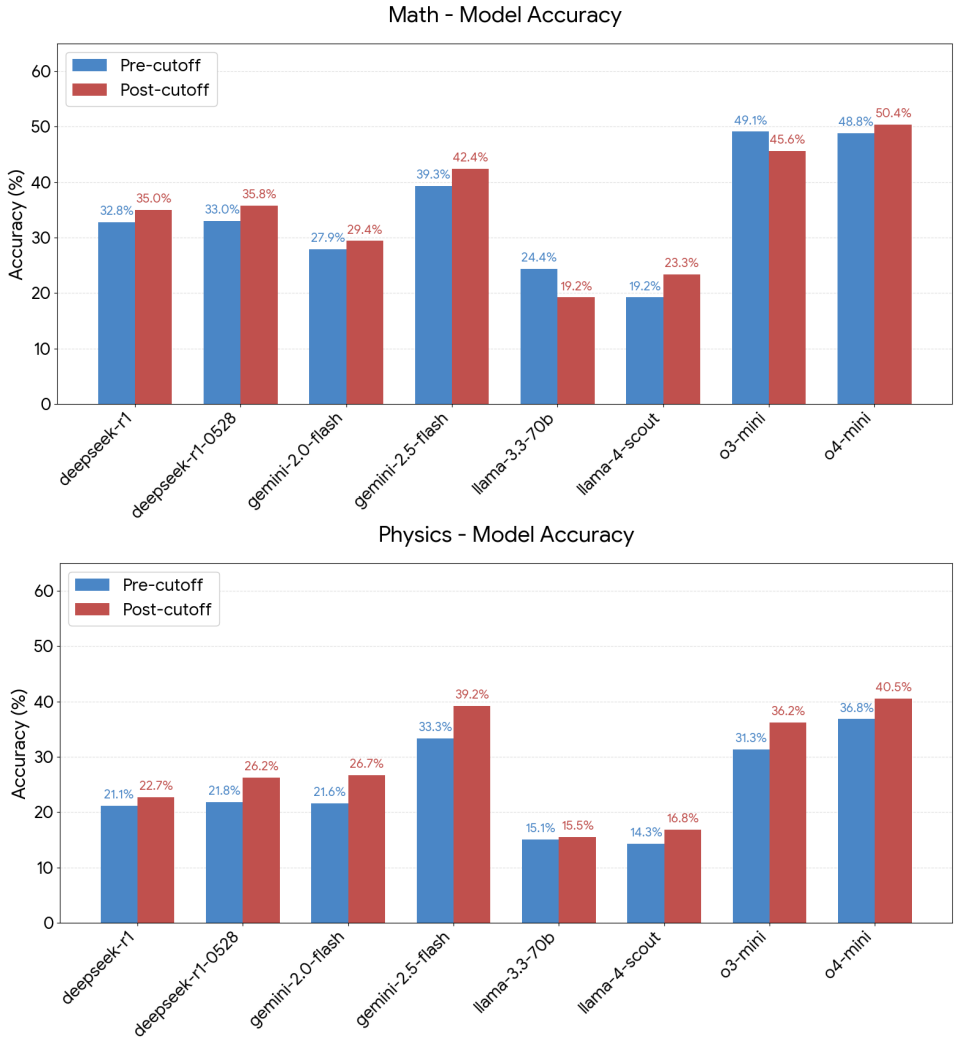


Figure 5. Mean accuracy trends before (blue) versus after (red) knowledge cutoff dates for the Mathematics and Physics domain.

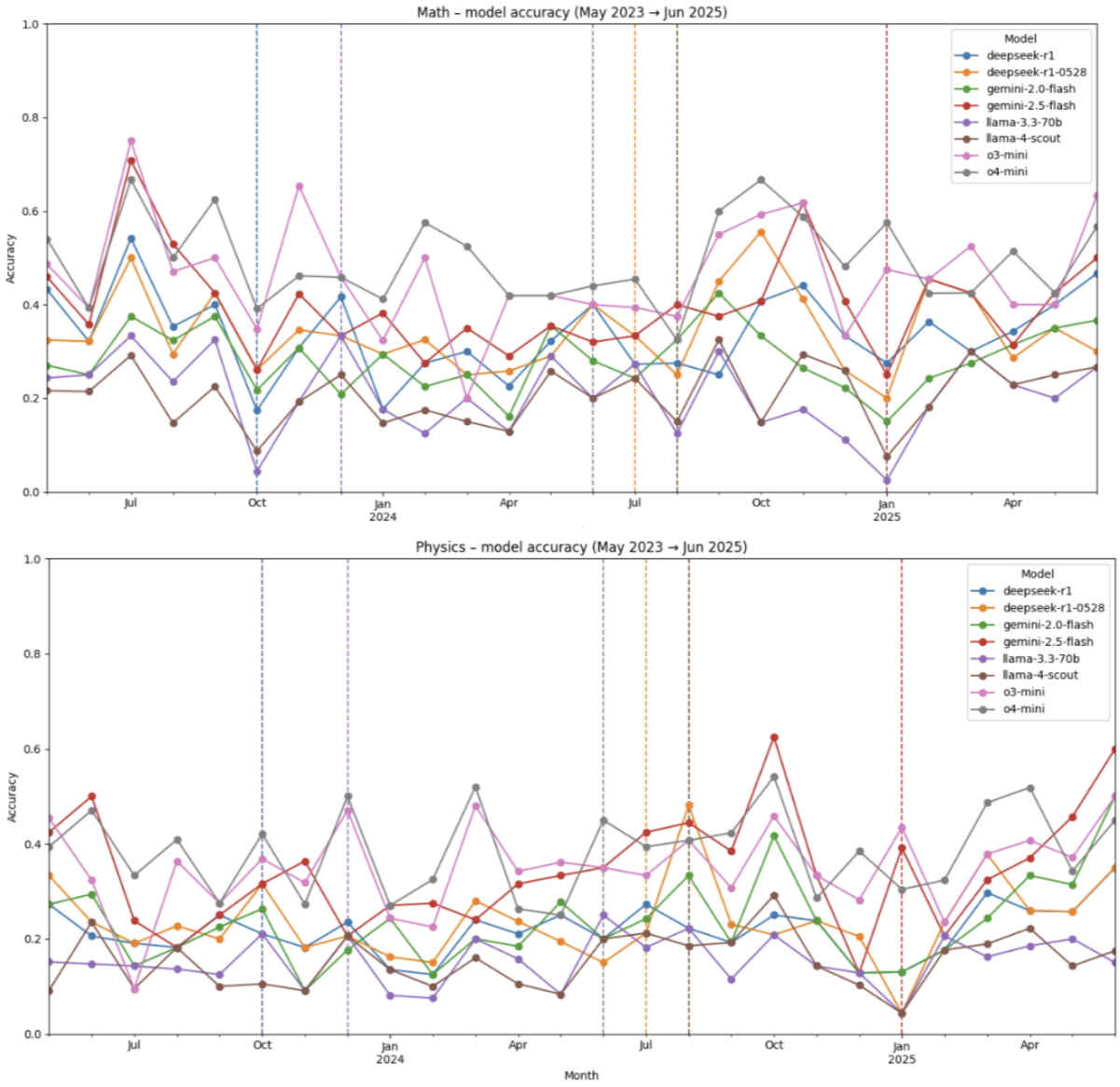
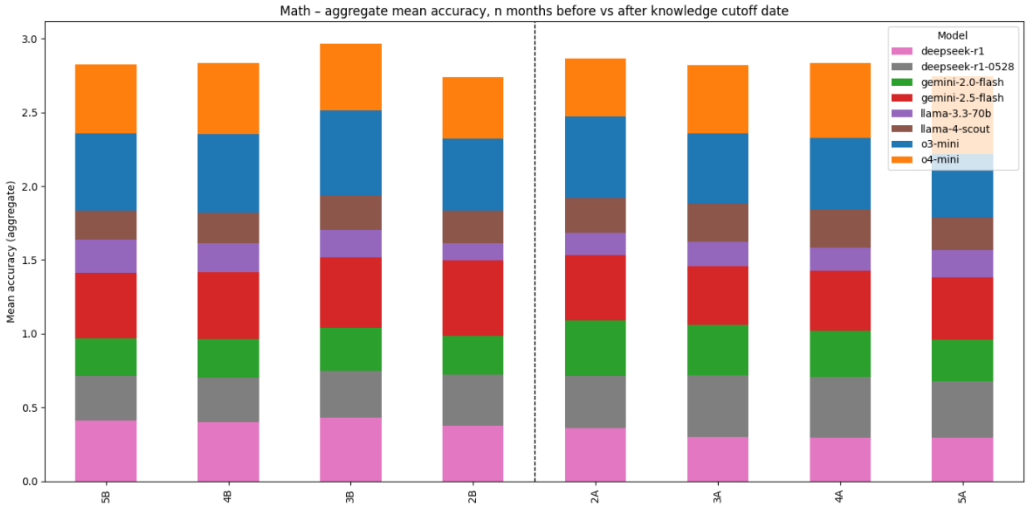
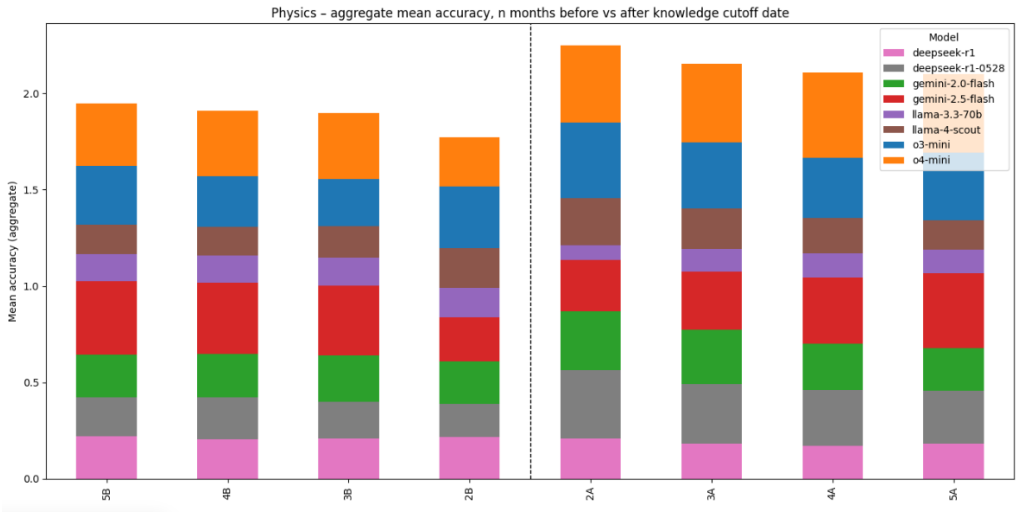


Figure 6. Model performance on synthesis-based questions by reasoning models across 26 months (May 2023 to June 2025) for mathematics (top) and physics (bottom) domains, with knowledge cutoff dates marked by dashed lines. Performance remains stable across all models' cutoff boundaries, with some models showing slight improvements post-cutoff.



(a) Mathematics: Aggregated time windows



(b) Physics: Aggregated time windows

Figure 7. Mathematics and Physics aggregated performance across multiple time windows (nB marks n months before, nA marks n months after cutoff). Each color represents a different model in question whereas each bar represents one time window for aggregation.