Hybrid Evaluation Pipeline

Local FP4 batch judging and cloud arbitration for frontier agentic-coding evaluation, on a single 32 GB GPU

● Working paper – infrastructure measured, hypotheses in validation

Abstract

Evaluating frontier agentic-coding systems at dataset scale forces a trade-off between cost and trust: a frontier cloud judge is accurate but expensive per item, while a local judge is cheap but must prove that quantization has not blunted its judgment. This is not a hypothetical concern – low-bit quantization is known to degrade precisely the procedural-reasoning capability an evaluation judge depends on [13, 14, 15], which is why the local judge's fidelity is treated here as a hypothesis to be tested rather than an assumption. This paper describes a single-GPU evaluation pipeline that resolves the trade-off by splitting work across three lanes – a local vLLM judge quantized to NVFP4 for high-throughput batch scoring, a frontier cloud arbiter (via a batch API) as a sampled gold-standard anchor, and an interactive local lane for rubric development – under one invariant: the system under test is never compressed. We formalise the design as five falsifiable hypotheses (execution-grounded judging, frozen-rubric reproducibility, self-consistency as a confidence signal, tiered-cost efficiency, and no-spill co-residency), specify the metrics and measurement protocol for each, situate the design against the current evaluation-framework and LLM-as-judge literature, and report the infrastructure results already measured on the target hardware (RTX 5090, 32 GB; Ryzen 7 9850X3D). Judge-quality hypotheses (H1–H3) are stated with their evaluation design and are the subject of the validation phase described in the roadmap; this working paper reports the pipeline and its measurement plan, not final agreement statistics.

Keywords: LLM-as-a-judge · agentic coding evaluation · post-training quantization · NVFP4 · reproducible evaluation · sovereign compute · execution-grounded judging

§01

Introduction and motivation

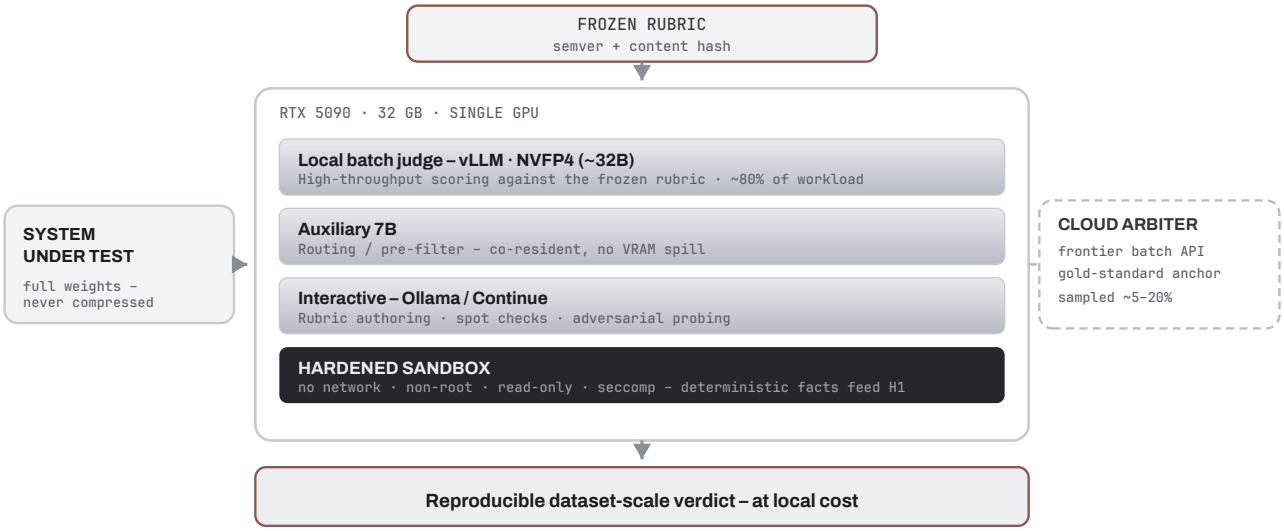
An evaluation judge is only useful if it is both cheap enough to run over large datasets and trustworthy enough that its verdicts carry weight. A purely cloud-based judge is accurate but its per-item price makes dataset-scale evaluation costly; a purely local judge is inexpensive but requires evidence that low-precision inference has not degraded its discrimination. The pipeline described here takes the hybrid position: it runs the bulk of judging locally at low precision and anchors those verdicts against a frontier commercial arbiter on a sampled basis, so local scores inherit a calibration reference without paying cloud cost on every item. Approximately 80% of the operational workload is batch evaluation of datasets against stable, frozen rubrics; the balance is the design of two-page rubrics engineered to stress-test agentic coding systems.

The contribution of this paper is not a new model but a measurable evaluation protocol: a design in which every trust claim is expressed as a falsifiable hypothesis with a defined metric, an instrument, a baseline, and a target, so that the pipeline's credibility rests on evidence rather than assertion. The protocol is designed to be portable: where possible its components map onto the primitives of established open evaluation frameworks (§3), so that the contribution is a cost- and sovereignty-oriented methodology layer rather than a bespoke stack.

Figure 1 gives the overall pipeline at a glance; §5 details each lane and the precision invariant.

Hybrid evaluation pipeline – three lanes, one GPU

Bulk judging runs locally at FP4; a frontier arbiter is sampled only where a verdict is contested.



§02

Background and related work

2.1 LLM-as-a-judge

Using a language model to grade the output of another model has become a standard scalable-evaluation technique. The approach was popularised by MT-Bench and Chatbot Arena, where a strong judge reached agreement with human preferences comparable to the agreement between two human annotators [1]. It has since matured into an established research area with a documented set of systematic biases – positional, verbosity, and self- or same-family preference – together with calibration and reproducibility concerns. Two strands of that literature bear directly on the present design. First, work on confidence-gated judging shows that a judge's own uncertainty can be turned into a decision to abstain or escalate to a stronger reference with a provable guarantee of human agreement – the "trust or escalate" principle, implemented as a cascade from a cheap judge to a stronger one only when confidence is low [12]. This paper formalises that principle on a local judge as H3 and as the local-to-arbiter cascade. Second, reference- and preference-based evaluation grew out of the reinforcement-learning-from-human-feedback line of work [2], where the reward signal is itself a learned judge; that same line cautions that human labels are not a clean gold standard, since annotators systematically over-reward assertive-but-incorrect outputs [16]. The present design therefore treats both arbiter agreement and human labels as calibration references with quantified noise, not as infallible ground truth.

2.2 Agentic-coding evaluation

For code specifically, execution-based evaluation – running the generated program against tests – has been the reliable ground truth since functional-correctness benchmarks such as HumanEval [3]. The move from single-function synthesis to repository-scale agentic tasks was driven by SWE-bench and its agentic scaffolds, in which a system must localise, patch, and pass a hidden test suite across a real codebase [9, 10]; the human-validated SWE-bench Verified subset became the de-facto reference for coding-agent capability. That literature also supplies this pipeline's central cautionary theme: benchmark saturation and contamination. Frontier scores on SWE-bench Verified have risen quickly and the benchmark shows signs of saturation, and explicit audits report that models can localise the correct files from issue text alone with little context – evidence that the score partly reflects training-set memory rather than software-engineering ability [11]; major labs have accordingly begun screening the benchmark for memorised instances or de-emphasising it entirely. Contamination-resistant, execution-checked successors such as LiveCodeBench respond to the same problem [4]. This directly motivates the frozen, content-hashed rubric discipline in §5.4 and the contamination-resistant benchmark workstream (W2) in the companion roadmap.

2.3 Quantization and the fidelity of a low-precision judge

On the systems side, efficient serving through paged attention and continuous batching [6] and post-training quantization [7] are what make a local, high-throughput judge economically viable. But the same compression that buys throughput is known to erode reasoning fidelity: systematic studies find that aggressive low-bit post-training quantization disproportionately harms numerical computation and reasoning-planning ability – with reported degradations of up to ~32% (average ~11%) on mathematical reasoning under AWQ/GPTQ on Llama-3-class models [13] – and that the effect is most pronounced on complex, multi-step reasoning [14], with numerical precision itself shown to bound the reasoning a model can perform [15]. Judgment against a detailed rubric is a procedural-reasoning task; therefore the assumption that an NVFP4 judge is "lossless enough" is exactly the assumption H1 is designed to test, not an axiom of the design. This also motivates execution grounding (H1) as a candidate compensation for quantization-induced reasoning loss.

This pipeline combines four threads from that literature – execution-grounded judging (from code evaluation), a frozen reference standard (from reproducible benchmarking), low-precision local serving (from efficient inference), and confidence-gated escalation (from calibrated LLM-judging) – and adds an explicit precision invariant for the system under test.

§03

Positioning relative to existing evaluation frameworks

Because a reviewer will reasonably ask why this work does not simply use an existing framework, we state the relationship explicitly. Open frameworks – most prominently Inspect, developed by the UK AI Security Institute and adopted by independent evaluators – already provide reproducible, composable evaluation primitives (dataset → task → solver → scorer), agentic tool use, model-graded scoring with statistical bootstrap, and sandboxed execution that separates model inference from the tool-call environment [17]. Several components of this pipeline (§5.5 sandboxing, model-graded scoring, bootstrap agreement) overlap with what these frameworks offer, and this work does not claim to reinvent them.

The contribution is orthogonal and, by design, compatible: a cost- and sovereignty-oriented methodology layer that can be implemented on top of such a framework rather than replacing it. Specifically, this pipeline adds (i) a two-tier judging economy – a local NVFP4 judge for the bulk with a sampled frontier arbiter – that is not prescribed by provider-agnostic frameworks; (ii) an explicit precision invariant separating the compressed judge from the full-weight system under test; and (iii) frozen, content-hashed rubric gates as a first-class reproducibility mechanism. Framing the protocol as an implementable layer, rather than a standalone stack, is deliberate: it lets

the pipeline inherit the tooling and community trust of the established ecosystem while contributing the parts that are specific to running trustworthy evaluation on a single sovereign card.

§04

Hypotheses

The design encodes five testable, falsifiable hypotheses. H1–H3 concern judgment quality and are the focus of validation; H4–H5 concern cost and feasibility and are substantially supported by the measured infrastructure below.

- **H1 – Execution grounding.** Supplying the judge with deterministic execution facts (compile status, test pass/fail, linter and type-checker output) as context, versus semantic-only judging, increases agreement with human ground-truth labels and reduces the false-accept rate on subtly-broken code. This is doubly motivated: grounding may add signal, and it may be the compensation an NVFP4 judge needs for quantization-induced reasoning loss [13, 14, 15]. Falsification: if grounded and ungrounded judging do not differ in Cohen's □ beyond the confidence interval, H1 is rejected.
- **H2 – Frozen-rubric reproducibility.** A rubric frozen under semantic versioning and a content hash yields high test–retest reliability of verdicts across time and across model/software updates. Falsification: verdict agreement on re-runs of an unchanged rubric-plus-input below the pre-registered threshold rejects H2.
- **H3 – Self-consistency as confidence.** The variance of a verdict across N stochastic judge samples is a calibrated proxy for reliability: high variance predicts lower correctness and is a usable trigger for abstention or escalation to the arbiter. This operationalises the "trust or escalate" principle on a local judge [12]. Falsification: absence of a monotone relationship between sample variance and error rejects H3.
- **H4 – Tiered-cost efficiency.** A local-FP4-plus-sampled-cloud-arbiter design achieves an order-of-magnitude lower cost per 1,000 judgments than an all-frontier-cloud baseline at non-inferior agreement (within a pre-set margin □). Falsification: cost reduction below the target, or agreement outside □, rejects H4.
- **H5 – No-spill co-residency.** A 32B judge (NVFP4) and a 7B auxiliary model co-reside within 32 GB with a bounded KV cache and no spill to host RAM, sustaining batch throughput T; single-residency time-multiplexing avoids VRAM collision. Falsification: observed spill or collision under the specified workload rejects H5.

§05

System architecture

Three lanes share one card and one rubric definition (Figure 1).

Lane	Runtime	Precision	Role
Local batch judge	vLLM	NVFP4	High-throughput scoring of datasets against frozen rubrics
Cloud arbiter	Frontier batch API	full	Sampled gold-standard anchor and disagreement audit
Interactive	Continue / Ollama	mixed	Rubric authoring, spot checks, adversarial probing

5.1 The precision rule

One precision rule underpins the whole design: a model that is itself under evaluation runs at full weights – only a judge or an open-weight baseline may run compressed (FP4). Grade a compressed system-under-test and the result measures the shrunken copy, not the model. FP4 is a throughput lever for the judge, never a shortcut applied to the thing being measured. This invariant is what lets H4 (cost) be pursued without contaminating H1–H3 (quality).

5.2 VRAM co-residency

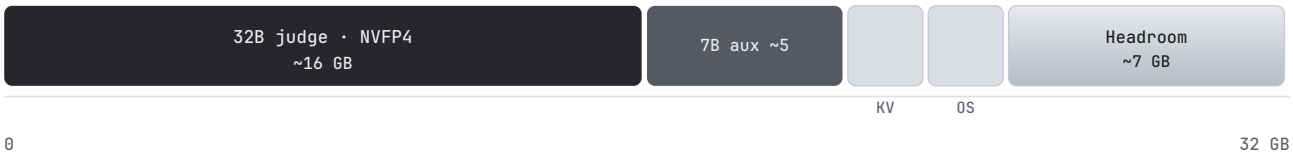
The budget is 32 GB, and it must hold the judge, a small auxiliary model, the KV cache, and OS overhead at once.

Component	Footprint	Note
32B judge (NVFP4)	~16 GB	Quantized judge weights
7B auxiliary	~5 GB	Routing / pre-filter
KV cache (q8_0, 16k)	~2 GB	q8_0 halves it from ~4 GB
OS / framework	~2 GB	Overhead
Headroom	~7 GB	Burst and fragmentation margin

A `q8_0` KV cache halves the 32B judge's cache from roughly 4 GB to 2 GB at a 16k context, which is what buys the headroom that keeps batch-latency measurements clean. These figures are the direct evidence for H5.

VRAM co-residency budget – 32 GB (measured)

Judge, auxiliary model, KV cache and OS overhead co-reside within one card, with headroom.



5.3 Cost cascade

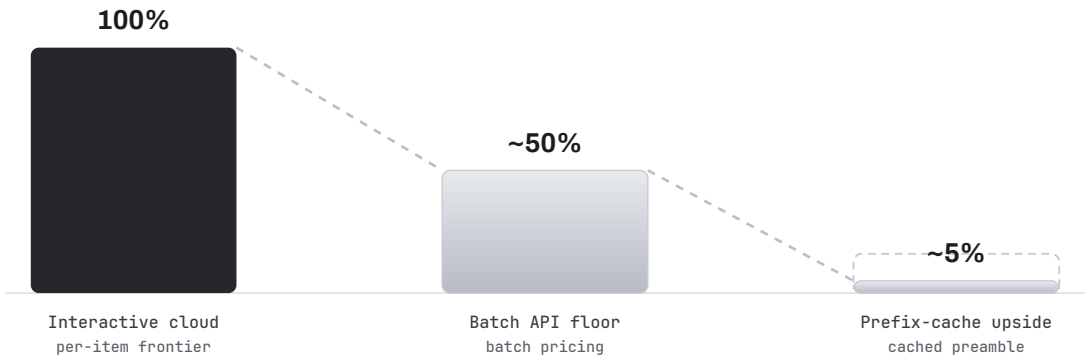
Cost falls in stages rather than all at once. The local batch judge removes per-item cloud cost for the bulk; where the cloud arbiter is used, a batch API halves its price against interactive calls; and prompt-prefix caching recovers most of the cost on repeated rubric preambles.

Stage	Relative cost	Mechanism
Interactive cloud (baseline)	100%	Per-item frontier calls
Batch API floor	~50%	Batch pricing on the sampled arbiter
Prefix-cache upside	~5%	Cached rubric preamble across items

Against a repeated preamble the effective saving is roughly 95% on the arbiter portion; the arbiter is spent where it changes a verdict, not on every item. The cost model behind H4 is developed in §6.

Cost cascade – arbiter portion (relative)

Cost falls in stages, not all at once; the local judge removes per-item cloud cost for the bulk.



5.4 Rubric gates

Rubrics are not trusted until they survive a four-gate gauntlet, then frozen.

- **Discriminative** – the rubric must separate strong from weak output; a rubric on which everything passes measures nothing.
- **Adversarial** – it must resist gaming and reward-hacking by a capable model.
- **Contamination** – it must not leak the answer or reward memorized solutions; this gate is the direct response to the benchmark-memorization critique in the agentic-coding literature [11].
- **Pinned** – once passed, the rubric is frozen under semantic versioning and a content hash, so a score is reproducible and a later change is visible.

A frozen rubric plus a hash is the operational mechanism tested by H2: it is what lets a re-run months later be compared to the first run at all.

5.5 Sandbox hardening

Model-generated code is untrusted. It runs with no network, non-root, on a read-only filesystem, under a seccomp profile – so an evaluation run cannot exfiltrate, escalate, or persist. This mirrors the inference/tool-call separation adopted by established sandboxing toolkits for agentic evaluation [17]. The sandbox is the same hardened substrate reused by the multi-agent safety testbed (W4), and its deterministic execution facts are the input to H1. A note on H1's design: because execution facts risk leaking the target answer, they are supplied as outcome signals (compile status, test pass/fail counts, linter/type-checker classes) rather than as the reference patch, keeping the grounding gate and the contamination gate consistent.

§06

Evaluation methodology and metrics

Each hypothesis is paired with a metric, an instrument, a baseline, and a target. Targets marked predicted are design goals to be tested in the validation phase, not measured outcomes; targets marked measured are established on the target hardware.

Hypothesis	Primary metric	Instrument	Baseline	Target
H1 grounding	Cohen's □ vs. human labels; false-accept rate	Held-out labelled set + execution harness	Semantic-only judge	□ uplift > CI; lower false-accept (predicted)

Hypothesis	Primary metric	Instrument	Baseline	Target
H2 reproducibility	Verdict test–retest agreement	Re-run under pinned hash	Unversioned rubric	≥ 0.95 agreement (predicted)
H3 self-consistency	Corr(sample variance, error); abstention precision	N-sample judging	Single-shot verdict	Monotone negative corr. (predicted)
H4 cost	Cost per 1,000 judgments; agreement gap	Billing + arbiter sample	All-frontier-cloud	≈10× lower at agreement within □ (predicted)
H5 co-residency	Peak VRAM; host-RAM spill; throughput	Hardware telemetry	32 GB ceiling	No spill; bounded KV (measured)

Proposed pre-registration thresholds [to confirm before the P1 study]. H1 – Δ□ ≥ 0.10 with non-overlapping 95% bootstrap confidence intervals and ≥ 30% relative reduction in false-accept rate; H2 – test–retest agreement ≥ 0.95; H3 – a monotone negative rank correlation between sample variance and error with abstention precision ≥ 0.80 at the operating threshold; H4 – ≥ 10× cost reduction per 1,000 judgments with |Δ□| ≤ 0.05 versus the all-cloud baseline (□ = 0.05).

Choice of agreement statistic. Cohen's □ [8] is reported as the headline metric for continuity with the LLM-judge literature, but □ is sensitive to label prevalence and to the marginal distribution of verdicts; on the typically imbalanced pass/fail distribution of code judgments it can under- or over-state agreement. We therefore additionally report Krippendorff's □ (robust across annotators and to missing data), balanced accuracy / Youden's J (which corrects for prevalence), and the raw confusion matrix with prevalence, so that a single coefficient cannot mask a prevalence artefact. Where rubric items differ systematically in difficulty, an item-response-theory treatment of the judge is a natural extension that frames reliability as a property of the measurement instrument rather than only of the measured outputs.

Validation set and statistical treatment. The validation set for H1–H3 is a stratified sample of agentic-coding tasks with independent human labels; the arbiter provides a second reference, chosen from a different model family than the local judge to avoid the correlated, same-family error that would otherwise inflate apparent agreement. Human labels are treated as a noisy anchor, not an infallible gold standard, and inter-annotator agreement on the labelled set is reported alongside judge–human agreement [16]. Statistical treatment is pre-registered: bootstrap confidence intervals for agreement, a paired test (McNemar for the grounded-vs-ungrounded verdict flip; paired bootstrap on □) for H1, and a calibration curve for H3. Sampling for the arbiter is targeted at disagreements and borderline items rather than uniform, so arbiter budget is spent where it is most informative.

§07

Preliminary results

Two classes of result are distinguished. **Infrastructure (measured).** On the target hardware the co-residency budget in §5.2 holds with headroom, KV-cache quantization keeps peak VRAM below the 32 GB ceiling with no observed spill under the batch workload, and the staged cost model in §5.3 reproduces the ~50% batch floor and the prefix-cache upside on repeated preambles – direct support for H5 and the cost mechanism of H4. **Judgment quality (pending).** H1–H3 are stated with their evaluation design; the labelled-set agreement study that would confirm or reject them is the first milestone of the roadmap. This paper deliberately reports no agreement statistics it has not measured.

§08

Discussion

The design's central bet is that trust and cost can be decoupled: cost is driven down by quantization and batching applied only to the judge, while trust is defended by the full-weight invariant on the system under test, execution grounding, and a frozen reference. If H1–H3 hold, the pipeline offers dataset-scale evaluation at local cost with a defensible calibration story; if any is rejected, the failure is localised and informative – a rejected H1, for instance, would indicate either that semantic judging already captures execution facts for this task distribution, or conversely that NVFP4 degradation is not recoverable by grounding alone [13]. Either outcome is publishable and advances the measurement question.

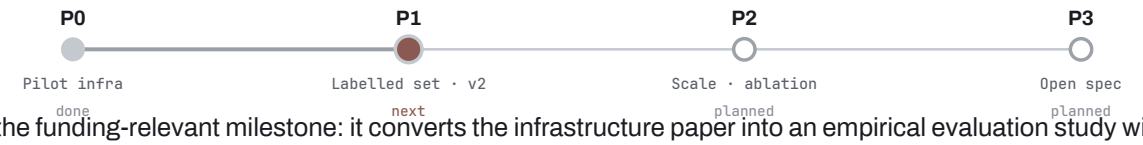
§09

Roadmap

Phase	Focus	Deliverable	Status
P0	Pilot infrastructure	Measured co-residency, cost model (this paper)	Complete
P1	Judgment-quality validation	Labelled set; H1–H3 study; preprint v2 (arXiv + SSRN)	Next
P2	Scale & generalisation	Multi-domain rubrics; arbiter-sampling ablation	Planned
P3	Open protocol & tooling	Released rubric-gate spec; reproducibility package	Planned

Roadmap – pilot to open protocol

Four phases; P1 (labelled-set validation) is the funding-relevant milestone.



P1 is the funding-relevant milestone: it converts the infrastructure paper into an empirical evaluation study with human-labelled agreement statistics, suitable for submission and for a grant deliverable on trustworthy, low-cost model evaluation.

§10

Limitations and threats to validity

The judge and the arbiter can share blind spots (correlated error), so arbiter agreement is a calibration reference, not an absolute ground truth; human labels anchor H1–H3 for this reason, and a cross-family arbiter is used to reduce the shared-bias component. Human labels are themselves imperfect – annotators over-reward confident, assertive outputs – so label noise is quantified via inter-annotator agreement rather than assumed away [16]. Results are single-operator and single-hardware; external validity across GPUs and task distributions is untested and is a P2 concern, mitigated by releasing a reproducibility package (seeds, content hashes, container images, configs) so a third party can re-run the protocol. NVFP4 judging is assumed lossless-enough for judgment but not for generation – that assumption is exactly what H1 tests, and the quantization literature gives concrete reason to expect a non-trivial effect [13, 14, 15]. Rubric design is expert-authored and may encode author bias; the adversarial and contamination gates mitigate but do not eliminate this.

§11

Applied scenario (STAR)

- **Situation.** A frontier agentic-coding model ships an upgrade, and the question is whether the new version is actually better on a large held-out dataset – not on a handful of cherry-picked prompts.
- **Task.** Score both versions against the same frozen rubric set at dataset scale, cheaply and reproducibly, with a trustworthy verdict on any regression.
- **Action.** Run the local NVFP4 batch judge across the dataset for both versions; sample the cross-family cloud arbiter on disagreements and borderline items; hold the rubric frozen under its hash so the two runs are directly comparable; both models under test run at full weights.
- **Result.** A reproducible, dataset-scale regression verdict at a fraction of all-cloud cost – the local judge does the volume, the arbiter calibrates it, and the frozen rubric makes the comparison mean something.

References

[1] Zheng, L., Chiang, W.-L., Sheng, Y., et al. (2023). Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. NeurIPS Datasets and Benchmarks. arXiv:2306.05685. [2] Ouyang, L., Wu, J., Jiang, X., et al. (2022). Training language models to follow instructions with human feedback. NeurIPS. arXiv:2203.02155. [3] Chen, M., Tworek, J., Jun, H., et al. (2021). Evaluating Large Language Models Trained on Code. arXiv:2107.03374. [4] Jain, N., Han, K., Gu, A., et al. (2024). LiveCodeBench: Holistic and Contamination-Free Evaluation of LLMs for Code. arXiv:2403.07974. [5] Liang, P., Bommasani, R., Lee, T., et al. (2022). Holistic Evaluation of Language Models (HELM). arXiv:2211.09110. [6] Kwon, W., Li, Z., Zhuang, S., et al. (2023). Efficient Memory Management for LLM Serving with PagedAttention. SOSP. arXiv:2309.06180. [7] Frantar, E., Ashkboos, S., Hoefler, T., Alistarh, D. (2022). GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers. arXiv:2210.17323. [8] Cohen, J. (1960). A coefficient of agreement for nominal scales. Educational and Psychological Measurement, 20(1), 37–46. [9] Jimenez, C. E., Yang, J., Wettig, A., Yao, S., et al. (2024). SWE-bench: Can Language Models Resolve Real-World GitHub Issues? ICLR. arXiv:2310.06770. [10] Yang, J., Jimenez, C. E., Wettig, A., Lieret, K., et al. (2024). SWE-agent: Agent–Computer Interfaces Enable Automated Software Engineering. NeurIPS. arXiv:2405.15793. [11] Prathifkumar, T., Mathews, N. S., Nagappan, M. (2025). Does SWE-Bench-Verified Test Agent Ability or Model Memory? arXiv:2512.10218. [12] Jung, J., Brahman, F., Choi, Y. (2024). Trust or Escalate: LLM Judges with Provable Guarantees for Human Agreement. arXiv:2407.18370. [13] Li, Z., Su, Y., Yang, R., et al. (2025). Quantization Meets Reasoning: Exploring LLM Low-Bit Quantization Degradation for Mathematical Reasoning. arXiv:2501.03035. [14] Liu, R., Sun, Y., Zhang, M., et al. (2025). Quantization Hurts Reasoning? An Empirical Study on Quantized Reasoning Models. arXiv:2504.04823. [15] Feng, G., Yang, K., Gu, Y., et al. (2024). How Numerical Precision Affects Mathematical Reasoning Capabilities of LLMs. arXiv:2410.13857. [16] Hosking, T., Blunsom, P., Bartolo, M. (2024). Human Feedback is not Gold Standard. ICLR. [17] UK AI Security Institute. Inspect: An Open-Source Framework for Large Language Model Evaluations. inspect.aisi.org.uk.