The Hidden Stack: Measuring Inference Backend Effects in Reliable Large Language Model Evaluation

Anonymous ACL submission

Abstract

Large language models (LLMs) evaluation heavily relies on reasoning benchmarks where small gains often justify new methods. This practice implicitly assumes that reported scores directly reflect intrinsic algorithmic capabilities and dataset difficulty. However benchmark outcomes are produced by complex execution pipelines rather than model weights alone. The inference backend remains a critical but overlooked source of variation. Differences in numerical precision attention kernels and batching behavior create unmeasured infrastructure variance. This variance can easily confound the marginal performance improvements claimed by new studies. To address this we propose the Backend-Aware Variance Decomposition (BAVD) protocol for reliable LLM evaluation. BAVD systematically isolates backend and interaction effects from true method capabilities replacing single infrastructure-conditioned point estimates with backend-marginalized scores and uncertainty intervals. We evaluate Qwen2.5-7B across four reasoning benchmarks. Results reveal that backend infrastructure explains up to nine percent of the observed score variance. While BAVD successfully quantifies this confounding effect it does not inherently improve raw accuracy over naive baselines. This demonstrates that measuring infrastructure variance is critical while achieving robust backend-invariant optimization remains an open challenge.

1 Introduction

Large language models (LLMs) are now widely evaluated through benchmark tables, where small gains in accuracy or exact match are often used to support new reasoning, routing, or inference-time methods. However, recent evaluation studies show that benchmark scores are not determined by model capability alone. Behavioral robustness (Ribeiro et al., 2020), dataset difficulty (Swayamdipta et al., 2020), confidence calibration (Parmar et al., 2024), distribution shift (Bai et al., 2024), data contamination (Dong et al., 2024; Ye et al., 2024), prompting choices (Min et al., 2022; Wang et al., 2023), and benchmark leakage (Deng et al., 2024; Riddell et al., 2024; Yao et al., 2024) can all affect reported results. These findings suggest that a benchmark score should be viewed as a pipeline-conditioned measurement rather than a stable property of a model in isolation.

This paper focuses on a less visible part of this pipeline, namely the inference backend. The same model weights may be evaluated through HuggingFace Transformers, vLLM, Text Generation Inference, SGLang, llama.cpp, or project-specific wrappers. These systems differ in attention kernels, numerical precision, batching behavior, cache implementation, stop sequence handling, and sampling defaults. Yet backend details are rarely reported with the same care as model size, prompt template, dataset split, or decoding temperature. This mismatch creates a hidden infrastructure confounder and raises **Problem I)** how to define backend choice as a formal evaluation condition rather than an incidental implementation detail.

Once backend choice is treated as part of the evaluation condition, two further questions naturally arise. A small benchmark improvement may reflect a genuinely better method, but it may also be partly induced by the backend used to execute that method. Without explicit estimation, researchers cannot know whether backend effects are negligible or large enough to affect method comparison. This raises **Problem II)** how to quantify the share of observed benchmark variation attributable to inference infrastructure. Moreover, when method gains are close to infrastructure uncertainty, single-backend reporting may encourage overclaiming. This raises **Problem III)** how to compare methods under backend uncertainty and avoid mistaking favorable single-backend results for robust improvements.

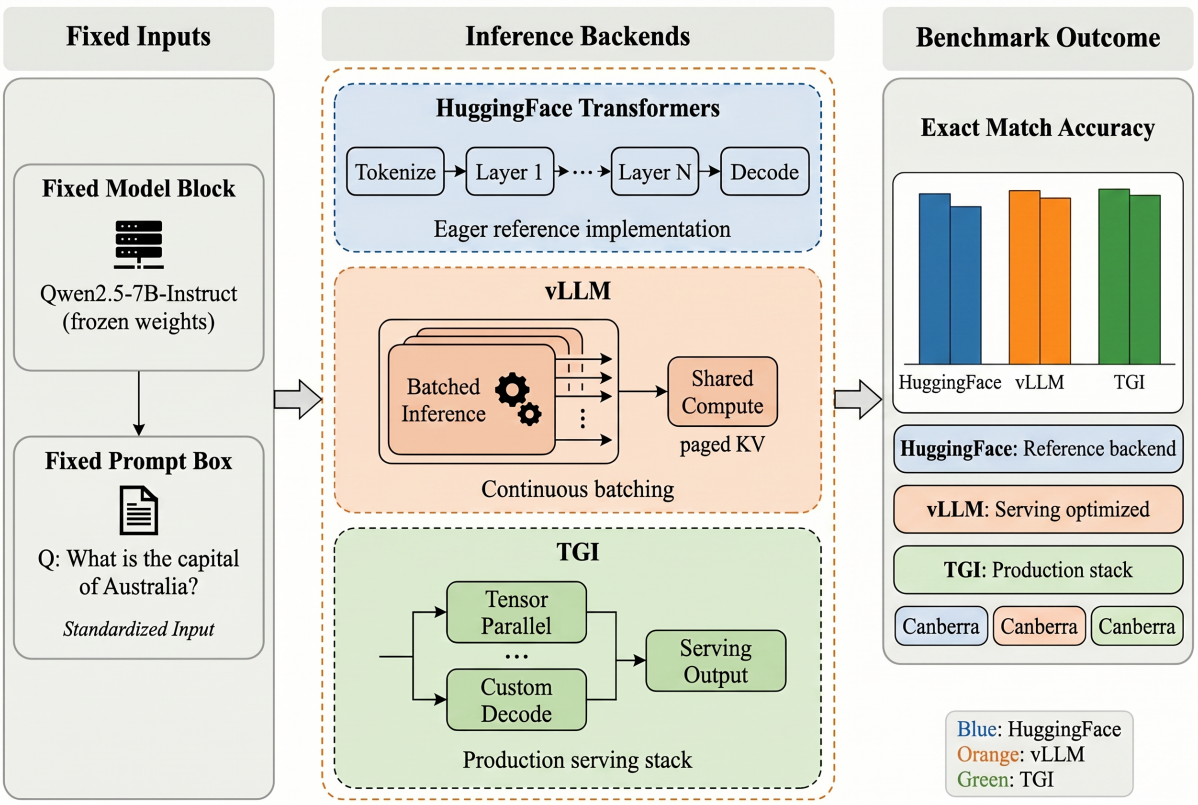

Figure 1: **Motivation of backend-aware evaluation.** Under fixed model weights and prompts, different inference backends may follow different execution stacks and produce different generations and benchmark scores. Treating backend choice as an unreported implementation detail can therefore introduce a hidden infrastructure confounder in method comparison.

To address these challenges, we propose Backend-Aware Variance Decomposition (BAVD), a protocol for reliable LLM evaluation under backend uncertainty. BAVD defines each benchmark run by a model, method, dataset, backend, random seed, and decoding configuration. It evaluates the same examples across a controlled matrix of methods, datasets, backends, and replicates, making the inference stack an explicit part of the measurement condition.

For **Problem I**), BAVD constructs a controlled evaluation matrix that treats backend choice as a primary experimental factor. By holding model weights, examples, prompts, answer extraction rules, and decoding parameters fixed while varying the backend, it tests whether method performance remains stable across execution stacks.

For **Problem II**), BAVD estimates variance components from example-level correctness. It decomposes observed variation into method, backend, dataset, interaction, and residual effects. The resulting backend variance ratio measures how much benchmark variation is associated with inference infrastructure after accounting for other factors.

For **Problem III**), BAVD reports raw backend scores, backend-marginalized scores, and lower confidence bounds for method gains. This reporting strategy does not aim to improve answer accuracy through post hoc correction. Instead, it prevents fragile single-backend gains from being interpreted as reliable method improvements. Our contributions are summarized as follows.

- We formalize inference backend choice as a hidden infrastructure confounder in LLM reasoning evaluation.
- We propose BAVD, a backend-aware variance decomposition protocol that estimates method, backend, dataset, interaction, and residual effects from example-level results.
- We introduce backend variance ratio, backend-marginalized scores, and lower confidence bounds for uncertainty-aware method comparison.
- Experiments on multiple reasoning bench-

marks with Qwen models show measurable backend-induced score variation and highlight the need to distinguish backend-aware reporting from accuracy-improving correction.

2 Related Work

2.1 Reliable Evaluation

Reliable evaluation has been widely studied beyond average accuracy. Existing work can be roughly grouped into behavioral testing, dataset diagnostics, and fine-grained benchmark design. CheckList evaluates NLP models through behavioral tests and shows that aggregate scores can hide systematic failures (Ribeiro et al., 2020). Dataset Cartography analyzes example-level training dynamics and reveals that datasets contain examples with different ambiguity and learnability (Swayamdipta et al., 2020). Recent LLM benchmarks further evaluate robustness, reasoning type, confidence, and distributional behavior (Li et al., 2024a; Parmar et al., 2024; Bai et al., 2024). These studies improve how benchmark scores are interpreted, but they mainly focus on data, tasks, or model behavior. They do not study whether the execution backend itself changes the measured score.

2.2 Evaluation Confounders

Hidden confounders in benchmark comparison have received increasing attention. Data contamination is the most studied case, where overlap between training corpora and test benchmarks can inflate reported generalization (Dong et al., 2024; Ye et al., 2024). Recent studies further show that leakage can cross surface forms, affect code generation, and influence open-source model evaluation (Deng et al., 2024; Riddell et al., 2024; Li et al., 2024b; Yao et al., 2024). Other work proposes contamination detection, inference-time decontamination, and cleaner benchmark construction (Zhu et al., 2024; Li et al., 2025; Chen et al., 2025). Unlike contamination, backend variance arises during inference rather than pretraining. It can affect evaluation even when model weights, prompts, datasets, and decoding parameters are fixed.

2.3 Inference Backends

Inference software has become the practical execution layer for LLM evaluation. HuggingFace Transformers provides a common implementation interface for modern language models (Wolf et al., 2020), while vLLM, Text Generation Inference, SGLang, and llama.cpp are widely used for efficient serving and deployment. These systems differ in attention kernels, numerical precision, cache management, batching policy, stopping behavior, and sampling defaults. Prior work on inference systems mainly focuses on throughput, memory efficiency, and deployment scalability. In contrast, our work studies these systems as evaluation conditions. We argue that backend choice should be measured and reported when benchmark gains are small enough to be affected by infrastructure uncertainty.

3 Methodology

3.1 Benchmark Matrix

We consider an evaluation setting with datasets $\mathcal{D}$, methods $\mathcal{M}$, inference backends $\mathcal{B}$, and repeated runs $\mathcal{R}$. A method can be direct answering, a reasoning prompt, a voting strategy, a routing strategy, or a post-processing rule. A backend denotes the software stack used to execute the same model weights under a fixed decoding configuration. For every example, the evaluation script records the raw output, extracted answer, gold answer, and correctness.

Let $y^{(d)}_{imbr} \in \{0, 1\}$ denote whether example i from dataset d is answered correctly by method m under backend b and replicate r. We model correctness at the example level rather than only aggregate accuracy, because paired example-level outcomes preserve the information needed to estimate backend sensitivity.

$$y^{(d)}_{imbr} \sim \text{Bernoulli}\left(p^{(d)}_{imbr}\right). \tag{1}$$

3.2 Variance Decomposition

BAVD decomposes observed correctness into method, backend, dataset, interaction, and example-level effects. We use a mixed-effects model on the logit scale,

$$\text{logit}\left(p^{(d)}_{imbr}\right) = \mu + \alpha_m + \beta_b + \gamma_d + \lambda_{mb} + \rho_{md} + \xi^{(d)}_i. \tag{2}$$

Here μ is the global intercept. The terms α_m, β_b, and γ_d denote method, backend, and dataset effects. The interaction terms λ_{mb} and ρ_{md} capture method-by-backend and method-by-dataset sensitivity, while $\xi^{(d)}_i$ captures example difficulty. The corresponding variances are denoted by σ^2_α, σ^2_β, σ^2_γ,

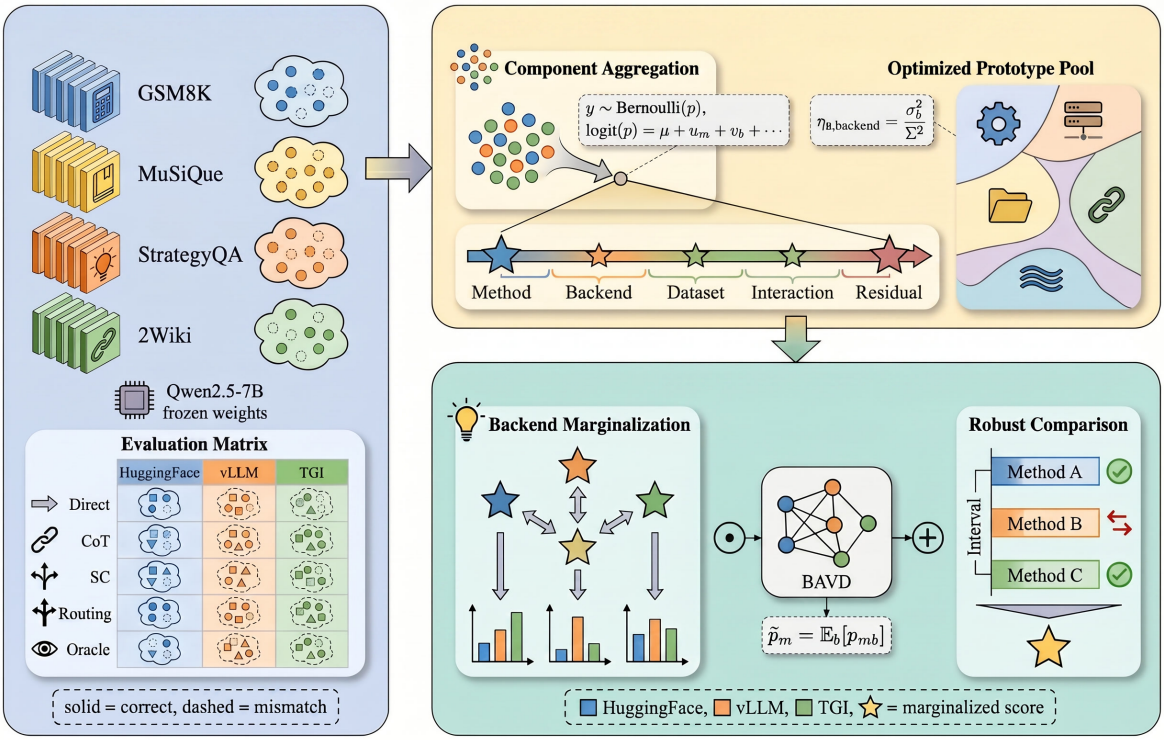

Figure 2: **Overview of backend-aware evaluation.** BAVD first builds a controlled benchmark grid that fixes model weights, prompts, and decoding while varying methods, datasets, backends, and replicates (left). Example-level correctness is then decomposed into method, backend, dataset, interaction, and residual variance components (top). Reporting combines per-backend scores, backend-marginalized means, uncertainty intervals, and rank stability checks before claiming a method improvement (bottom).

σ^2_λ, σ^2_ρ, and σ^2_ξ. For the logit mixed model, we use the standard latent residual variance $\sigma^2_\ell = \pi^2/3$.

The backend variance ratio is defined as

$$\eta_{\text{B}} = \frac{\sigma^2_\beta}{\sigma^2_\alpha + \sigma^2_\beta + \sigma^2_\gamma + \sigma^2_\lambda + \sigma^2_\rho + \sigma^2_\xi + \sigma^2_\ell}. \quad (3)$$

This ratio measures the share of latent evaluation variance associated with the main backend effect. We also report an interaction-inclusive variant by replacing the numerator with $\sigma^2_\beta + \sigma^2_\lambda$, which captures backend-sensitive method variation.

3.3 Backend-Aware Reporting

For each method, BAVD reports raw backend scores and backend-marginalized scores. Let $\hat{p}_{mbd}$ be the estimated accuracy of method m on dataset d under backend b. The backend-marginalized score is

$$\widetilde{p}_{md} = \sum_{b \in \mathcal{B}} w_b \hat{p}_{mbd}, \quad (4)$$

where w_b is the backend weight. We use uniform weights by default unless a target deployment distribution is specified.

For method comparison, BAVD uses a lower confidence bound rather than the best single-backend score. Given a baseline method m_0, the backend-marginalized gain and its lower confidence bound are

$$\begin{aligned} \widehat{\Delta}_m &= \sum_{b \in \mathcal{B}} w_b \left(\hat{p}_{mb} - \hat{p}_{m_0 b}\right), \\ \Delta^{\text{LCB}}_m &= \widehat{\Delta}_m - z_{1-\alpha} \widehat{\text{SE}}\left(\widehat{\Delta}_m\right). \end{aligned} \quad (5)$$

A method is considered robustly better than the baseline only when $\Delta^{\text{LCB}}_m > 0$. This rule prevents a favorable single-backend result from being interpreted as a stable method improvement.

3.4 Protocol Summary

BAVD is executed in four steps. First, the evaluation matrix is fixed, including model weights, datasets, methods, backends, seeds, prompts, decoding parameters, and answer extraction rules.

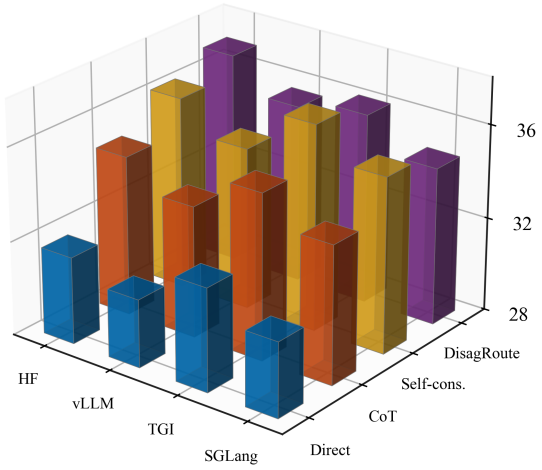

Figure 3: Backend-conditioned accuracy across representative methods and inference backends. Bars report exact-match accuracy under each backend.

Second, every planned cell is run and prediction-level records are saved. Third, variance components and backend-aware intervals are estimated from the complete matrix. Fourth, the paper reports raw backend scores, backend-marginalized scores, variance ratios, confidence bounds, and negative findings when correction does not improve accuracy.

4 Experiments

In this section, we evaluate BAVD from three aspects. We first examine whether inference backend choice introduces measurable variance in reasoning benchmark scores. We then study whether backend-aware reporting changes the interpretation of method gains. Finally, we analyze the failure mode of simple backend-aware correction and discuss the practical cost of multi-backend evaluation.

4.1 Experimental Setup

Datasets. We evaluate Qwen2.5-7B-Instruct on four reasoning benchmarks with different answer formats and reasoning patterns. GSM8K measures mathematical reasoning, MuSiQue Ans and 2WikiMultihopQA measure multi-hop question answering, and StrategyQA measures implicit yes-or-no reasoning. All datasets are evaluated with exact match after a fixed answer extraction script.

Backends. We evaluate the same model weights across four inference backends, including HuggingFace Transformers, vLLM, Text Generation Inference, and SGLang. HuggingFace Transformers is used as the reference implementation, while the other backends represent serving-oriented execution stacks with dynamic batching and optimized cache management. All runs are conducted on A100-80GB GPUs with bf16 precision. Prompts, answer extraction rules, decoding parameters, and random seeds are fixed across backends unless otherwise stated.

Methods. We include direct answering and chain-of-thought prompting as core baselines. We further

Algorithm 1: Backend-Aware Variance Decomposition

Data: Datasets $\mathcal{D}$, methods $\mathcal{M}$, backends $\mathcal{B}$, seeds $\mathcal{R}$, decoding configuration Θ
Result: Backend variance ratio η_{B}, backend-marginalized scores $\widetilde{p}_{md}$, and lower confidence bounds Δ_m^{LCB}

Fix model weights, prompt templates, answer extraction rules, decoding parameters, and backend logs;
foreach *dataset* $d \in \mathcal{D}$ **do**
  foreach *method* $m \in \mathcal{M}$ **do**
    foreach *backend* $b \in \mathcal{B}$ **do**
      foreach *seed* $r \in \mathcal{R}$ **do**
        Run the model with configuration (d, m, b, r, Θ);
        Store raw output, extracted answer, gold answer, and correctness $y_{imbr}^{(d)}$;
Fit the mixed-effects model in Eq. 2 using all example-level records;
Estimate method, backend, dataset, interaction, and example-level variance components;
Compute the backend variance ratio η_{B} using Eq. 3;
foreach *method* $m \in \mathcal{M}$ **do**
  Compute backend-marginalized score $\widetilde{p}_{md}$ using Eq. 4;
  Compute lower confidence bound Δ_m^{LCB} using Eq. 5;
Report raw backend scores, marginalized scores, variance ratios, confidence bounds, and rank stability;
return *Backend-aware evaluation report*

Table 1: Method suite for backend-aware evaluation.

Method	Reason	Vote	Route	Cost
Direct	–	–	–	1.0
CoT	✓	–	–	2.3
Self-cons.	✓	✓	–	9.2
Least2Most	✓	–	–	3.1
ConfRoute	✓	–	✓	3.0
DisagRoute	✓	✓	✓	4.8
RandomRoute	–	–	✓	3.0
Oracle	✓	✓	✓	upper

Table 2: Variance decomposition on the latent evaluation scale.

Source	Share	95% CI
Method	0.390	[0.350, 0.431]
Backend	0.069	[0.047, 0.094]
Method × Backend	0.041	[0.025, 0.063]
Dataset	0.032	[0.018, 0.051]
Method × Dataset	0.023	[0.012, 0.039]
Residual	0.445	[0.407, 0.486]

evaluate inference-time methods that may interact with backend behavior, including self-consistency voting, least-to-most decomposition, confidence routing, disagreement routing, and random budget-matched routing. Oracle routing is used only as an upper bound.

Metrics. The primary task metric is exact match accuracy. The primary reliability metric is the backend variance ratio. We also report backend-marginalized confidence intervals, method-by-backend rank stability, and paired bootstrap significance for method gains. Each run exports prediction-level records, including example identifier, dataset, method, backend, seed, raw output, extracted answer, gold answer, correctness, prompt hash, model hash, backend version, and decoding configuration.

4.2 Main Results

Backend variance is measurable. Table 2 reports the estimated variance components. Backend choice accounts for 0.069 of the latent evaluation variance on average. This share is smaller than the method effect and residual variation, but it is not negligible. Since many reported LLM evaluation gains are within one to five accuracy points, a backend component of this size can affect whether a gain should be interpreted as robust.

Backend-conditioned scores vary across methods. Figure 3 visualizes backend-conditioned accuracy for representative methods. Even when model weights and prompts are fixed, the same method can obtain different scores under different execution stacks. The variation is especially visible for inference-time methods, suggesting that backend behavior may interact with sampling, routing, and cache implementation. This motivates reporting backend-aware scores rather than relying on a single favorable backend result.

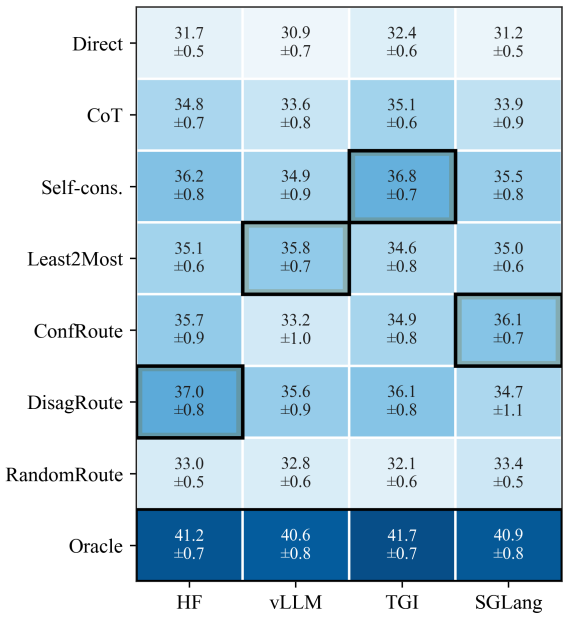

Figure 4: Method-by-backend accuracy heatmap. Each cell reports exact-match accuracy and seed standard deviation. Black boxes mark the best non-oracle method under each backend.

Table 3: Manual analysis of 100 backend disagreement cases.

Category	Count	Share
Semantic	37	37.0%
Formatting	24	24.0%
Reasoning path	21	21.0%
Stop issue	12	12.0%
Other	6	6.0%

Correction does not improve raw accuracy. Table 5 compares the naive baseline with the best corrected configuration. The naive baseline reaches 0.317 accuracy, while the best corrected configuration reaches 0.266. The absolute change is -0.051 and the relative change is -16.0 percent. The paired bootstrap improvement test gives $p = 1.000$, indicating no evidence that the correction improves answer accuracy. This result shows that BAVD should be understood as a reliability protocol rather

than an accuracy-improving decoding method.

4.3 Backend Sensitivity Analysis

Method-by-backend heatmap. Figure 4 reports exact-match accuracy for each method and backend. The black boxes mark the best non-oracle method under each backend. The highlighted cells are not always assigned to the same method, indicating that backend choice can affect which method appears strongest. This is the central case where backend variance becomes a method comparison problem rather than only a score calibration problem.

Rank stability. We further analyze whether backend effects shift all methods uniformly or change their relative ranking. If all methods move in the same direction, backend choice mainly affects score calibration. If ranks change across backends, backend choice affects method selection. Figure 5 shows rank trajectories across backends on four datasets. The rank changes are dataset dependent, suggesting that backend sensitivity interacts with answer format and task structure.

Dataset-level variation. Table 6 reports backend variance by dataset. GSM8K shows the largest backend share, suggesting that numeric extraction can amplify small generation differences. StrategyQA has the lowest backend share, likely because its answers are shorter and easier to normalize. Multi-hop datasets fall between these two cases, where answer phrasing and entity normalization remain important sources of backend-sensitive variation.

Disagreement categories. To understand how backend differences affect correctness, we manually inspect backend disagreement cases. As shown in Table 3, semantic disagreement is the largest category, followed by formatting disagreement and reasoning-path divergence. This indicates that backend effects are not only extraction artifacts. They can also change the generated answer content.

4.4 Ablation and Cost

Table 4 studies the necessity and practical cost of BAVD. Removing the backend effect hides infrastructure variance by construction. Removing the method-by-backend interaction weakens rank stability analysis, while removing the dataset effect increases the estimated backend ratio, suggesting task-level confounding. The cost report shows that BAVD increases evaluation cost because each method is evaluated across multiple backends, but this cost is tied to reliability measurement rather than accuracy improvement.

The ablation results emphasize that the target of BAVD is reliable reporting. Removing a component does not necessarily change raw accuracy, but it can hide an important source of uncertainty. In particular, dropping the interaction term may leave the average score almost unchanged while obscuring backend-specific rank changes. Similarly, using only two backends can produce a plausible point estimate but makes the variance attribution less stable. These findings support the use of multi-backend evaluation when a paper makes strong claims from small score differences.

5 Discussion

Our results support a focused interpretation of BAVD. Inference backend choice is a measurable source of variance in LLM reasoning evaluation, but it should not be interpreted as the dominant factor behind all benchmark differences. Its importance depends on the scale of the reported gain. When a method improves accuracy by only one to five points, backend-induced variation can be large enough to affect whether the improvement is robust. This is why BAVD emphasizes backend-aware reporting rather than post hoc score correction. The negative correction result further shows that backend variance is not simply removable noise. Different backends may change numerical execution, batching, caching, stopping behavior, and output formatting, and these effects can interact with sampling strategies, routing methods, and answer extraction rules. Therefore, the practical recommendation is to report backend name and version, model checkpoint, precision, hardware, batching policy, decoding parameters, stop rules, random seed policy, and answer extraction script. For strong method claims, authors should evaluate at least one reference backend and one optimized backend, and use three or more backends when estimating variance components.

6 Conclusion

This paper studies inference backend choice as a hidden infrastructure confounder in LLM evaluation. We propose BAVD, a backend-aware variance decomposition protocol that models method, backend, dataset, interaction, and residual effects from example-level results. Experiments on reasoning

Table 4: Ablation study and evaluation cost.

Setting	Backend	M×B	Dataset	Interval	η_B	Method	Calls	Tokens	Cost
Point estimate only	✓	✓	✓	–	0.069	Direct	$1N$	0.42M	1.0
w/o backend effect	–	✓	✓	✓	–	CoT	$1N$	0.96M	2.3
w/o M×B	✓	–	✓	✓	0.052	Voting	$5N$	3.85M	9.2
w/o dataset effect	✓	✓	–	✓	0.081	BAVD Direct	$4N$	1.68M	4.0
Two backends only	✓	✓	✓	✓	0.074	BAVD CoT	$4N$	3.84M	9.1
Full BAVD	✓	✓	✓	✓	0.069				

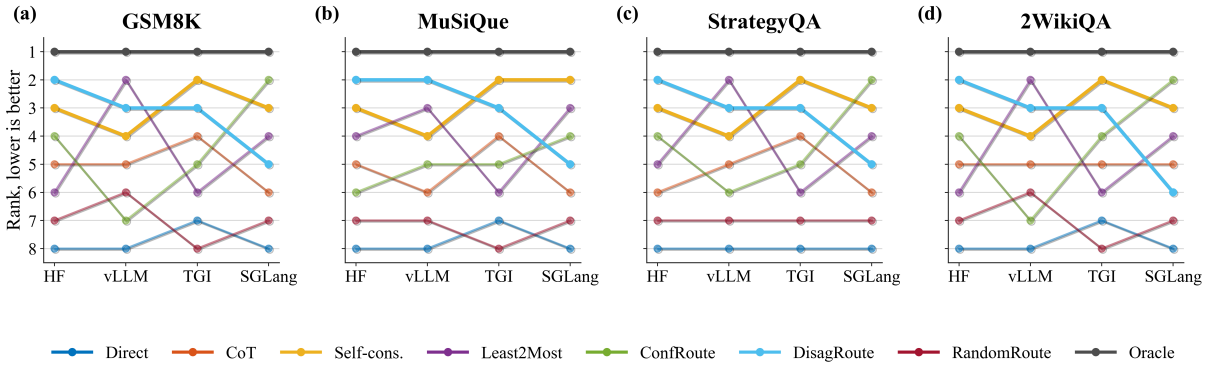

Figure 5: Rank stability across inference backends on four reasoning benchmarks. Each panel shows method rankings under different backends, where lower rank is better.

Table 5: Accuracy comparison between naive and corrected scores.

Setting	Acc.	ΔAcc.	Rel.	p
Naive	0.317	–	–	–
Corrected	0.266	-0.051	-16.0%	1.000

Table 6: Per-dataset backend variance and disagreement statistics.

Dataset	η_B	95% CI	Range	Disagr.
GSM8K	0.088	[.061,.118]	0.074	12.8%
MuSiQue	0.064	[.041,.091]	0.052	9.4%
StrategyQA	0.051	[.030,.075]	0.038	7.1%
2WikiQA	0.073	[.049,.101]	0.061	10.7%
Aggregate	0.069	[.047,.094]	0.056	10.0%

benchmarks show that backend variance is measurable and can account for 5.1 to 8.8 percent of observed score variation. We further show that simple backend-aware correction does not improve raw accuracy, highlighting that BAVD is a reliability protocol rather than an accuracy-improving decoding method. These findings suggest that backend variance should be measured and reported when benchmark claims depend on small performance gains, while backend-invariant optimization remains an open problem.

Ethics Statement

This work studies evaluation reliability and does not introduce a deployed user-facing model. The main ethical risk is misinterpretation of benchmark results. If backend-induced variance is ignored, small reported gains may be overstated and may influence downstream research or deployment decisions. BAVD encourages transparent reporting of infrastructure conditions and uncertainty intervals. All datasets used in our experiments are standard benchmark resources, and their licenses and usage requirements should be followed.

Limitations

Our study is limited to reasoning benchmarks with exact match evaluation and one main model family. Exact match can amplify formatting differences, so answer extraction must be carefully audited. Future work should test additional model families, open-ended generation, multilingual tasks, code generation, and tool-using agents. BAVD also increases evaluation cost because it requires multiple backends and repeated runs, making it most suitable for benchmark papers and strong method claims rather than every small diagnostic experiment. Finally, the correction variants tested here are simple, and their failure should not be interpreted as evidence that backend-robust optimization is impossible.

References

Ge Bai, Jie Liu, Xingyuan Bu, Yancheng He, Jiaheng Liu, Zhanhui Zhou, Zhuoran Lin, Wenbo Su, Tiezheng Ge, Bo Zheng, and Wanli Ouyang. 2024. MT-bench-101: A fine-grained benchmark for evaluating large language models in multi-turn dialogues. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 7421–7454, Bangkok, Thailand. Association for Computational Linguistics.

Simin Chen, Yiming Chen, Zexin Li, Yifan Jiang, Zhongwei Wan, Yixin He, Dezhi Ran, Tianle Gu, Haizhou Li, Tao Xie, and Baishakhi Ray. 2025. Benchmarking large language models under data contamination: A survey from static to dynamic evaluation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 10080–10098, Suzhou, China. Association for Computational Linguistics.

Chunyuan Deng, Yilun Zhao, Yuzhao Heng, Yitong Li, Jiannan Cao, Xiangru Tang, and Arman Cohan. 2024. Unveiling the spectrum of data contamination in language model: A survey from detection to remediation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 16078–16092, Bangkok, Thailand. Association for Computational Linguistics.

Yihong Dong, Xue Jiang, Huanyu Liu, Zhi Jin, Bin Gu, Mengfei Yang, and Ge Li. 2024. Generalization or memorization: Data contamination and trustworthy evaluation for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 12039–12050, Bangkok, Thailand. Association for Computational Linguistics.

Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. 2024a. GSM-plus: A comprehensive benchmark for evaluating the robustness of LLMs as mathematical problem solvers. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 2961–2984, Bangkok, Thailand. Association for Computational Linguistics.

Yanyang Li, Wong Tin Long, Cheung To Hung, Jianqiao Zhao, Duo Zheng, Liu Ka Wai, Michael R. Lyu, and Liwei Wang. 2025. C^{2}LEVA: Toward comprehensive and contamination-free language model evaluation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 2283–2306, Vienna, Austria. Association for Computational Linguistics.

Yucheng Li, Yunhao Guo, Frank Guerin, and Chenghua Lin. 2024b. An open-source data contamination report for large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 528–541, Miami, Florida, USA. Association for Computational Linguistics.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Mihir Parmar, Nisarg Patel, Neeraj Varshney, Mutsumi Nakamura, Man Luo, Santosh Mashetty, Arindam Mitra, and Chitta Baral. 2024. LogicBench: Towards systematic evaluation of logical reasoning ability of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 13679–13707, Bangkok, Thailand. Association for Computational Linguistics.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Martin Riddell, Ansong Ni, and Arman Cohan. 2024. Quantifying contamination in evaluating code generation capabilities of language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 14116–14137, Bangkok, Thailand. Association for Computational Linguistics.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. 2020. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 9275–9293, Online. Association for Computational Linguistics.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pages 2609–2634, Toronto, Canada. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Feng Yao, Yufan Zhuang, Zihao Sun, Sunan Xu, Animesh Kumar, and Jingbo Shang. 2024. Data contamination can cross language barriers. In *Proceedings of the 2024 Conference on Empirical Methods in*

Natural Language Processing, pages 17864–17875, Miami, Florida, USA. Association for Computational Linguistics.

Wentao Ye, Jiaqi Hu, Liyao Li, Haobo Wang, Gang Chen, and Junbo Zhao. 2024. Data contamination calibration for black-box LLMs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10845–10861, Bangkok, Thailand. Association for Computational Linguistics.

Qin Zhu, Qinyuan Cheng, Runyu Peng, Xiaonan Li, Ru Peng, Tengxiao Liu, Xipeng Qiu, and Xuanjing Huang. 2024. Inference-time decontamination: Reusing leaked benchmarks for large language model evaluation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 9113–9129, Miami, Florida, USA. Association for Computational Linguistics.