ADAPTIVE VIEW HARNESS: LEARNING TO SELECT TARGET-PRESERVING VIEWS FOR LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Different input views have been shown to affect the practical performance of large language models (LLMs). We aim to leverage historical multi-view behavior to identify a more suitable input form for a new query before inference. To this end, we propose Adaptive View Harness (AVH), which selects a target-preserving view for a new query using only pre-call information while preserving single-call inference. AVH first uses Identity (ID) as the reference and defines the expected switching gain of each candidate view in terms of conditional accuracy change, estimating its relative utility from historical multi-view outcomes using calibrated heterogeneous predictors. It then performs instance-level view selection based on the estimated relative utility, while development-level Conservative Activation disables adaptive selection when the available development evidence is insufficient and retains ID instead. We evaluate AVH on BBH and MMLU-Pro with Qwen2.5-7B-Instruct, Hunyuan-A13B-Instruct, Ling-mini-2.0, and GLM-4.5-Air. Across the eight model–benchmark settings, AVH yields non-negative point estimates relative to ID and achieves a maximum accuracy gain of $+3.42$ pp.

1 INTRODUCTION

In recent years, LLMs have continued to make progress on complex tasks such as mathematical reasoning (Cao et al., 2026) and code generation (Wang et al., 2026). Scaling model size and training data, together with advances in post-training techniques, has expanded model capabilities, while prompt design, inference strategies, tool use, and compute allocation at inference time have also become important ways to improve performance. Prompt optimization (Zhou et al., 2023) and inference-time reasoning strategies such as self-consistency (Wang et al., 2023) show that substantial performance gains can still be achieved by appropriately organizing inputs and the reasoning process without changing model parameters.

As LLMs move from standalone model capabilities toward system-level applications, execution environments outside the model increasingly take on the task of organizing inference. A harness can organize contextual information, tool interfaces, and execution workflows (Zhou et al., 2026; Tang et al., 2026), and has increasingly become an important system layer shaping model behavior in practice. Recent work has also begun to treat the harness itself as a configurable and optimizable component (Kim et al., 2026), examining the relationship between harness design and model capabilities. Input organization is a key factor affecting model behavior. Prior studies have shown sensitivity to prompt formulation (Mizrahi et al., 2024; Sclar et al., 2024), example ordering (Lu et al., 2022), and option ordering or presentation (Pezeshkpour & Hruschka, 2024; Sclar et al., 2024). These results show that input-form changes can alter model outputs even when the underlying problem semantics and answer target remain unchanged, creating potentially exploitable variation across target-preserving input views.

Existing multi-view research mainly falls into two categories: input-sensitivity analysis and multi-input inference. Input-sensitivity studies examine how prompt wording, option order, layout, and context organization affect model behavior (Sclar et al., 2024; Chatterjee et al., 2024; Mizrahi et al., 2024). Their primary goal is to characterize behavioral variation across different input representa-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

tions. Multi-input inference instead obtains multiple candidate responses through different prompts or reasoning paths and then aggregates them to improve the final prediction, as in multi-prompt inference (Arora et al., 2023) and self-consistency (Wang et al., 2023). For practical deployment, however, we seek to turn multi-view behavior accumulated from historical queries into a reusable harness capability. Given a new query, the harness should determine an input view using only information available before target-model inference and then invoke the target model once with the selected view.

Even when a harness has accumulated historical multi-view behavior, prior input-sensitivity studies show that semantics-preserving input variations can induce substantial performance fluctuations (Chatterjee et al., 2024; Sclar et al., 2024), with their effects varying across instances, tasks, and target models. Such studies reveal behavioral differences across input forms, but provide limited guidance on which view is more suitable for a new query. Meanwhile, adaptive prompt selection and routing methods have begun to choose prompts according to the input, for example by learning input–prompt compatibility (Liu et al., 2024; Gerych et al., 2024) or predicting response quality under different model–prompt configurations for instance-level selection (Sun et al., 2024). For target-preserving multi-view inputs, however, a switch may either correct an initially wrong answer or disrupt an originally correct one.

Motivated by these observations, we aim to turn historical multi-view behavior into a reusable harness capability that selects a more suitable input form for a new query with a single target-model call. To this end, we propose Adaptive View Harness (AVH), a harness for adaptive multi-view selection. First, AVH learns the relationship between query features and the correctness of each view from historical multi-view responses, and estimates the relative utility of candidate views with Identity as the reference. Second, AVH uses development data to determine whether adaptive selection should be enabled for the current target model and benchmark, retaining Identity when the development evidence does not support deployment. Finally, for a new query, AVH selects the candidate view with the highest predicted relative utility using pre-call information. When adaptive selection is enabled, AVH switches only if the predicted gain exceeds the frozen threshold and retains Identity otherwise. Our contributions are summarized as follows:

- We propose Adaptive View Harness (AVH), which turns historical multi-view behavior into an input-selection capability for new queries, enabling adaptive view use with a single target-LLM call.
- We formulate view selection through Identity-relative utility estimation and combine instance-level switching with development-level activation that retains Identity when selection evidence is insufficient.
- We evaluate AVH with four target LLMs on BBH and MMLU-Pro. Across eight model–benchmark settings, AVH yields non-negative point estimates relative to Identity, with gains of up to $+3.42$ pp.

2 Related Work

Inference-time optimization improves task performance by organizing inputs, decoding, and feedback without updating target-model parameters. Chain-of-Thought (Wei et al., 2022), Self-Consistency (Wang et al., 2023), and Reflexion (Shinn et al., 2023) improve inference through intermediate reasoning, multi-path aggregation, and iterative feedback. Prompt optimization methods such as APE (Zhou et al., 2023), OPRO (Yang et al., 2024), and PromptBreeder (Fernando et al., 2024) automate prompt search through instruction generation, natural-language optimization, and evolutionary search. At the system level, DSPy (Khattab et al., 2024) represents LM pipelines as programmable modules and optimizes prompt and demonstration configurations according to task metrics. Together, these studies demonstrate the value of optimizing inference processes outside the target model.

A fixed LLM can exhibit different behavior under semantically equivalent input representations. Prior work has identified sensitivity to few-shot example ordering (Lu et al., 2022), prompt formatting (Sclar et al., 2024), instruction paraphrases (Mizrahi et al., 2024), and intent-preserving prompt variations (Chatterjee et al., 2024). ProSA (Zhuo et al., 2024) studies instance-level prompt variation across models and datasets. In multiple-choice tasks, option ordering and option-label tokens can

systematically affect predictions (Pezeshkpour & Hruschka, 2024; Wei et al., 2024). AMA (Arora et al., 2023) improves prediction by aggregating outputs from multiple prompts, while PromptEval (Polo et al., 2024) estimates the performance distribution of large prompt sets under limited evaluation budgets. This line of work establishes that semantically equivalent input forms can produce complementary model behavior across instances.

Adaptive prompt selection conditions input configuration on the current query. Gerych et al. (Gerych et al., 2024) predict the quality of model–prompt pairs for instance-level selection. Adaptive Prompt Routing (Liu et al., 2024) selects candidate prompts according to the input and target style, while Prompt-OIRL (Sun et al., 2024) learns query-dependent prompt preferences from offline prompting demonstrations. A related line of work studies routing across different target models. FrugalGPT (Chen et al., 2024) uses adaptive cascades to balance quality and cost, RouteLLM (Ong et al., 2025) learns routing policies from preference data, and LLM-Blender (Jiang et al., 2023) exploits cross-model complementarity through response ranking and fusion.

AVH studies view selection within a fixed target model and a fixed task target. Historical outcomes from multiple target-preserving views are used to learn query-level switching utility before target inference. The selected view is then executed with one target-model call. This setting connects input-sensitivity analysis with adaptive selection and turns previously observed multi-view behavior into a reusable inference-time capability.

3 METHOD

3.1 PROBLEM FORMULATION AND OVERALL FRAMEWORK

Given an input query x and a fixed target model M, let $\mathcal{V} = \{V_1, V_2, \ldots, V_K\}$ denote a set of target-preserving views, where V_1 is the original input, referred to as Identity (ID). Let $c(x)$ denote the target answer to query x. Each view satisfies $c(V_v(x)) = c(x)$ and is evaluated by the same evaluator; that is, the input presentation may change while the evaluation target remains unchanged. For a historical query x_i, let $y_i^v \in \{0, 1\}$ indicate whether model M answers correctly under $V_v(x_i)$, and let $\mathcal{D}$ denote the dataset of historical queries and their multi-view behavior. For a new input x, the harness determines the view index $h(x) \in \{1, \ldots, K\}$ before invoking the target model, and only $V_{h(x)}(x)$ is sent to the target model. Our goal is to leverage the multi-view behavior accumulated in $\mathcal{D}$ to select a more suitable input form while retaining a single target-model call.

As shown in Fig. 1, Adaptive View Harness consists of offline learning from historical multi-view behavior and online view selection. Offline, AVH probes historical queries under multiple target-preserving views, learns view-correctness patterns from the resulting outcomes, and derives candidate utilities relative to ID together with the deployment decision rule. Online, AVH uses only information available before the target-model call to estimate candidate-view utility and determine whether to use an alternative view or retain ID. Once the input form is determined, only the selected view is sent to the fixed target model for final inference.

3.2 IDENTITY-RELATIVE VIEW UTILITY ESTIMATION

AVH focuses on the expected switching gain of an alternative view relative to ID. If view selection is directly formulated as best-view classification, a unique label often has to be imposed when multiple views are simultaneously correct, simultaneously incorrect, or yield the same outcome. More importantly, such a label does not directly characterize whether switching away from ID is beneficial. We therefore use ID as the reference and formulate view selection as relative utility estimation, so that the estimate corresponds directly to the expected accuracy change induced by switching.

Let $Y_v \in \{0, 1\}$ denote the correctness random variable of the target LLM under V_v, where the historical label y_i^v is the observed realization of Y_v for query x_i, and V_1 denotes ID. For query x, switching from ID to V_v changes instance-level accuracy by $Y_v - Y_1$. Given the pre-call information $\phi(x)$, we define the theoretical relative utility of candidate view V_v as

$$g_v(x) = \mathbb{E}[Y_v - Y_1 \mid \phi(x)] = P(Y_v = 1 \mid \phi(x)) - P(Y_1 = 1 \mid \phi(x)). \tag{1}$$

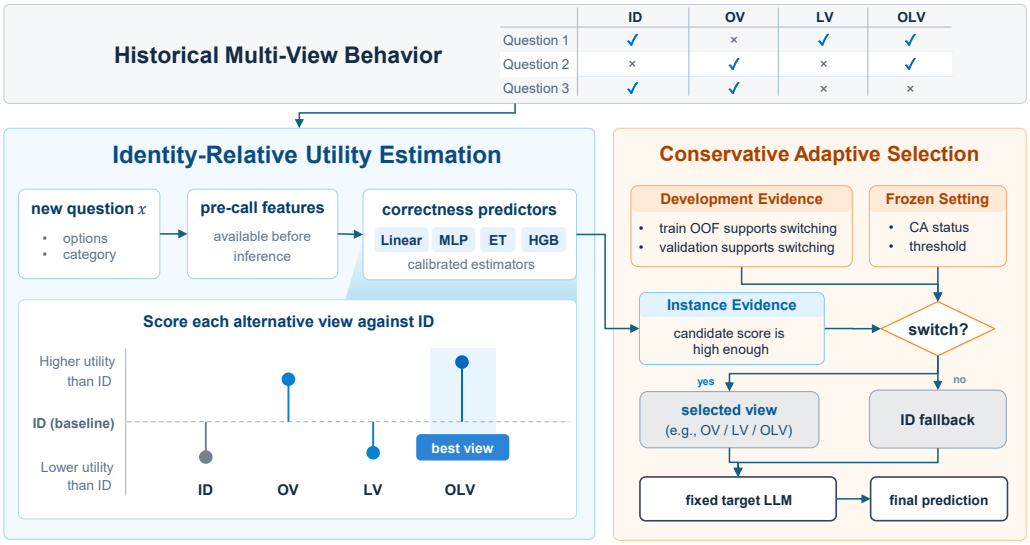

Figure 1: Overview of Adaptive View Harness (AVH). Historical multi-view behavior is used to learn Identity-relative view utilities and development-level deployment decisions. For a new query, AVH selects a view using only pre-call information and invokes the fixed target model once.

This definition directly connects relative utility to the accuracy change caused by switching. When the candidate view corrects an error made under ID, $Y_v - Y_1 = 1$; when switching turns an originally correct ID result into an incorrect one, $Y_v - Y_1 = -1$; otherwise, the difference is 0. Relative utility therefore admits an expected Rescue–Harm interpretation as the difference between the conditional probabilities of Rescue and Harm. AVH does not predict Rescue and Harm separately, but estimates their net effect through view-correctness probabilities.

The true Y_v is unobservable at online decision time, and evaluating multiple views before selection would violate the single-call constraint. We therefore use historical multi-view outcomes to learn the relationship between pre-call information and view correctness. For query x, $\phi(x)$ contains input-side information available before target inference, including the question, options, and any available category information, while excluding the target response, gold answer, and observed view outcomes for the current query. Let $p_{m,v}(x)$ denote the calibrated conditional correctness probability assigned to V_v by predictor m. The corresponding predictor-level utility estimate is $\hat{g}_{m,v}(x) = p_{m,v}(x) - p_{m,1}(x)$. A positive value favors switching to V_v, while its magnitude reflects the predicted gain relative to ID.

To reduce dependence on a single estimator, we use four calibrated predictor families: Linear, MLP, Extra Trees, and HistGradientBoosting (HGB). Calibration places their outputs on a more comparable probability scale, making probability differences and cross-estimator aggregation more meaningful. The final utility estimate for candidate view V_v is

$$\hat{g}_v(x) = \frac{1}{J} \sum_{j=1}^{J} \left[p_{j,v}(x) - p_{j,1}(x) \right], \qquad v > 1. \tag{2}$$

All predictor families receive equal weight. For each query, AVH therefore obtains $\{\hat{g}_2(x), \ldots, \hat{g}_K(x)\}$, representing the estimated gain of each candidate view relative to ID.

3.3 Conservative Adaptive View Selection

Relative utility estimates the preference of an alternative view over ID, while the switching decision also requires sufficient evidence. AVH makes this decision at two scales. Instance-level selection determines whether the predicted gain for the current query is large enough to trigger switching,

while Conservative Activation (CA) determines whether adaptive selection is enabled for the current model–benchmark setting using development evidence.

For query x, we first select the alternative view with the highest estimated relative utility, $v^* = \arg\max_{v>1} \hat{g}_v(x)$, with ties broken according to a predefined view order. To avoid switching on marginal utility differences, AVH switches from ID only when $\hat{g}_{v^*}(x) > \tau$. The threshold τ controls instance-level selectivity, with larger values retaining more queries on ID.

CA operates at the model–benchmark level while keeping the instance-level utility estimator fixed. Different target LLMs and benchmarks can respond differently to view transformations, so AVH uses development data to determine whether the adaptive policy is enabled. For a development split $\mathcal{S} \in \{\mathcal{D}_{\mathrm{OOF}}, \mathcal{D}_{\mathrm{Val}}\}$ and threshold τ, let $R_{\mathcal{S}}(\tau)$ and $H_{\mathcal{S}}(\tau)$ denote the numbers of Rescue and Harm cases, respectively, and define $\mathrm{Net}_{\mathcal{S}}(\tau) = R_{\mathcal{S}}(\tau) - H_{\mathcal{S}}(\tau)$. Each Rescue adds one correct prediction relative to ID and each Harm removes one, giving

$$\Delta\mathrm{Acc}_{\mathcal{S}}(\tau) = \frac{R_{\mathcal{S}}(\tau) - H_{\mathcal{S}}(\tau)}{|\mathcal{S}|} = \frac{\mathrm{Net}_{\mathcal{S}}(\tau)}{|\mathcal{S}|}. \tag{3}$$

AVH evaluates candidate thresholds separately on Train OOF and Validation. Train OOF uses out-of-fold base predictions, so the prediction for each training instance is produced by a fold model fitted without that instance. Validation is used for threshold selection. A threshold is eligible when it achieves positive Net on both splits:

$$\mathcal{T}^+ = \{\tau \in \mathcal{T} \mid \mathrm{Net}_{\mathrm{OOF}}(\tau) > 0,\ \mathrm{Net}_{\mathrm{Val}}(\tau) > 0\}. \tag{4}$$

If $\mathcal{T}^+$ is empty, CA is disabled and the current model–benchmark setting retains ID for all queries. Otherwise, CA is enabled and AVH selects τ^* by maximizing Validation Net over $\mathcal{T}^+$. Ties are resolved by lower Validation Harm, then lower Validation switch rate, and finally the larger threshold. CA controls the deployment of the frozen instance-level selector at the model–benchmark level.

Let $e \in \{0, 1\}$ denote the CA activation state. The final view decision is

$$h(x) = \begin{cases} v^*, & e = 1 \text{ and } \hat{g}_{v^*}(x) > \tau^*, \\ 1, & \text{otherwise.} \end{cases} \tag{5}$$

The estimated relative utility and τ^* determine instance-level switching, while e controls activation for the current setting. Queries retain ID when either condition is unmet. The selected $V_{h(x)}(x)$ is then sent to the target LLM, preserving a single target-model call.

4 Experiments

4.1 Experimental Setup

We conduct experiments from three perspectives: view complementarity, the realized gains of AVH, and the mechanism underlying adaptive selection. Specifically, we address the following three questions: (1) Do different target-preserving views exhibit exploitable complementarity at the instance level? (2) Can AVH translate this complementarity into realized performance gains with only one target-LLM call per instance? (3) How does AVH exploit view complementarity, and why does its selection behavior differ across models and benchmarks?

We evaluate AVH on two multiple-choice reasoning benchmarks, BIG-Bench Hard (BBH) and MMLU-Pro. For BBH, we use 11 reasoning tasks with 1052/263/1314 instances for Train/Validation/Test; MMLU-Pro uses 2500/500/2000 instances, respectively. The target models are Qwen2.5-7B-Instruct, Hunyuan-A13B-Instruct, Ling-mini-2.0, and GLM-4.5-Air.

All experiments use four fixed target-preserving views: Identity (ID), Order (OV), Layout (LV), and Order+Layout (OLV). OV rearranges the options according to the fixed priority C, A, D, B while preserving option content, label-to-text mapping, and the gold target; LV changes only option formatting; OLV applies both transformations.

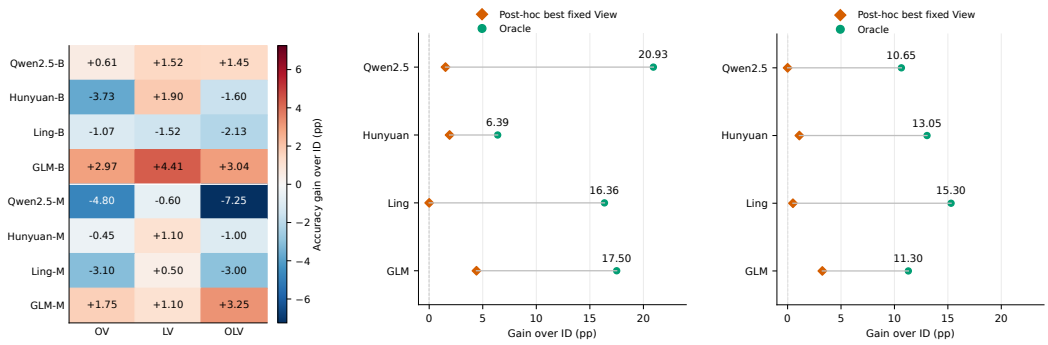

Figure 2: Fixed-view effects and Oracle headroom across model–benchmark settings.

We compare AVH with fixed-view strategies, single-predictor variants, and a Conservative Activation ablation. The fixed-view strategies include ID, OV, LV, OLV, and Development-Selected Fixed View (DS-FV), where DS-FV selects one globally fixed view using Validation. The single-predictor variants are Linear View Selection (Lin-VS), MLP View Selection (MLP-VS), Extra-Trees View Selection (ET-VS), and Histogram-Gradient-Boosting View Selection (HGB-VS). AVH without Conservative Activation (AVH w/o CA) retains the same utility estimator and instance-level switching rule while removing the model–benchmark-level CA fallback. Oracle View Selection (Oracle) uses instance-level ground-truth outcomes and serves only as a diagnostic upper bound on multi-view complementarity.

We report accuracy and gain over ID as the primary metrics, together with Switch Rate, Rescue, Harm, and $\mathrm{Net} = \mathrm{Rescue} - \mathrm{Harm}$ for switching behavior. For the main AVH results, we report 95% confidence intervals obtained by paired instance-level bootstrap. Implementation details and hyperparameter settings are provided in the Appendix.

4.2 View Complementarity and Oracle Headroom

Different target-preserving views exhibit complementarity at the instance level, but using any single fixed view cannot fully exploit these differences. Fixed-view gains vary across models and benchmarks, while Oracle reveals substantially larger instance-level performance headroom across all settings. The value of multiple views comes from heterogeneous responses to input forms across instances, with no fixed representation consistently superior.

We first compare the fixed-use performance of OV, LV, and OLV relative to ID. As shown in Figure 2a, the same view can have opposite effects across different settings. For example, LV yields an approximately $+1.90$ pp gain on Hunyuan/BBH, while OV decreases accuracy by approximately 3.73 pp under the same setting. The effect of a view therefore varies with the target model and task distribution. A fixed input form mainly exploits dataset-level average preference and cannot adapt to instance-level differences.

Figures 2b–c compare the post-hoc best fixed view with Oracle headroom on BBH and MMLU-Pro. The post-hoc best fixed view achieves an average gain of approximately 1.59 pp across the eight main settings, while Oracle headroom reaches 13.94 pp. The post-hoc best fixed view selects the fixed view with the highest accuracy using Test outcomes and is used only as a diagnostic of dataset-level fixed-view headroom. The large gap indicates that gains from multiple views are distributed across different instances. Fixed selection reflects dataset-level preference, while Oracle measures the additional headroom arising from instance-level heterogeneity.

Instance-level statistics exhibit the same pattern. In Figure 3, *rescuable* instances are those for which ID is incorrect and at least one alternative view is correct. *At-risk* instances are those for which ID is correct and at least one alternative view is incorrect. Both types account for non-negligible proportions in multiple settings. View transformations can therefore create both potential gains and switching risks. The same transformation may correct some instances while disrupting predictions

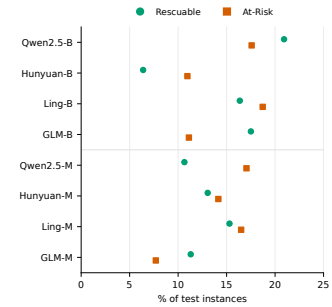

Figure 3: Rescuable and at-risk instances across model–benchmark settings.

Table 1: Main results across model–benchmark settings.

Benchmark	Model	ID	DS-FV	AVH (Δ ID)	Oracle
BBH	Qwen2.5	41.32	41.32	43.30 (+1.98)	62.25
BBH	Hunyuan	90.03	91.93	91.10 (+1.07)	96.42
BBH	Ling	45.66	44.60	46.80 (+1.14)	62.02
BBH	GLM	47.95	52.36	51.37 (+3.42)	65.45
MMLU-Pro	Qwen2.5	40.70	40.70	40.70† (+0.00)	51.35
MMLU-Pro	Hunyuan	59.20	59.20	59.20† (+0.00)	72.25
MMLU-Pro	Ling	39.70	39.70	39.70† (+0.00)	55.00
MMLU-Pro	GLM	41.45	43.20	44.65 (+3.20)	52.75

Values are Test accuracy (%). Parentheses report AVH gains over ID in percentage points. DS-FV is selected on Validation; Oracle is diagnostic only. † denotes CA-triggered ID fallback.

that were originally correct for others. These effects partially offset each other in overall accuracy, keeping fixed-view gains limited while leaving larger complementarity at the instance level.

4.3 Effectiveness of Adaptive View Harness

Overall, AVH converts part of the multi-view complementarity into realized performance gains under the single-call setting. Across the eight main model–benchmark settings, AVH achieves positive point estimates in five settings and retains ID in the remaining three. These results indicate that part of the instance-level complementarity revealed by Oracle can be realized using only pre-call information.

Table 1 summarizes the main results, while Figure 4a–b reports the accuracy gains of AVH over ID together with their 95% confidence intervals. AVH obtains positive point estimates for all four target models on BBH and for GLM on MMLU-Pro. For example, Qwen2.5/BBH improves over ID by 1.98 pp, with a 95% confidence interval of $[0.84, 3.12]$. Among the five positive settings, four have confidence intervals with lower bounds above zero, while the interval for Ling/BBH crosses zero. In the remaining three settings, CA retains ID without executing adaptive switching.

The comparison with DS-FV separates dataset-level view preference from instance-level selectability. DS-FV selects one fixed view using Validation, while AVH allows the preferred view to vary across queries. On Qwen2.5/BBH, DS-FV retains ID, whereas AVH achieves a $+1.98$ pp gain, showing that useful query-level preferences can remain even when no alternative view has a reliable global advantage. In contrast, DS-FV performs better on Hunyuan/BBH and GLM/BBH, where a fixed view already provides a strong dataset-level gain. The comparison therefore distinguishes two sources of improvement: global preference for a fixed representation and view preferences that can be identified from pre-call information at the query level.

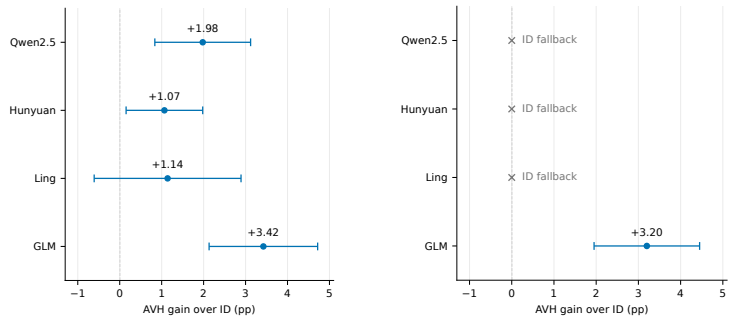

Figure 4: AVH accuracy gains over ID with 95% confidence intervals.

4.4 Understanding Adaptive View Selection

Overall, AVH gains are not monotonically related to the frequency of view switching. Effective selection requires distinguishing beneficial switches from harmful ones using pre-call information. Oracle headroom characterizes the available multi-view complementarity, while realized AVH gain reflects the portion identifiable before target inference. Conservative Activation determines whether adaptive selection is deployed for a given model–benchmark setting.

We first decompose the realized switching gain into Rescue and Harm. For a fixed test set containing N evaluation instances, the accuracy change of AVH relative to ID can be written as $\Delta\mathrm{Acc} = (\mathrm{Rescue} - \mathrm{Harm})/N$. Figure 5a relates Switch Rate to the resulting Rescue–Harm balance. Ling/BBH produces 79 Rescue cases together with 64 Harm cases, resulting in a net gain of $+1.14$ pp. In contrast, Qwen2.5/BBH achieves a $+1.98$ pp gain with fewer switches. Larger gains arise when Rescue exceeds Harm by a larger margin.

The coexistence of Rescue and Harm motivates development-level screening. Figure 5b presents the evidence associated with the frozen threshold candidates, where the x-axis reports the smaller of the Train OOF and Validation Net rates. A positive value indicates positive Net on both development partitions and satisfies the CA eligibility condition. The final threshold is selected from eligible candidates using Validation Net and the predefined tie-breaking rule, while the x-axis visualizes the corresponding development evidence. The three MMLU-Pro settings that fall back to ID have no selected threshold, reflecting insufficient development evidence for deploying adaptive selection.

The gap between Oracle headroom and realized AVH gain separates complementarity from pre-call selectability. As shown in Figure 5c, Hunyuan/MMLU-Pro has 13.05 pp of Oracle headroom, yet AVH retains ID. In contrast, GLM/MMLU-Pro converts part of its 11.30 pp Oracle headroom into a realized gain of $+3.20$ pp. Oracle measures the gains available from observed multi-view outcomes, while AVH identifies useful alternatives from information available before target inference. Realized performance therefore reflects the portion of multi-view complementarity that can be identified from pre-call information.

The CA ablation characterizes its role as a setting-level deployment control. Removing CA leaves the five enabled settings unchanged because their instance-level selectors are identical. Among the three fallback settings, Qwen2.5/MMLU-Pro changes from 0 to -0.20 pp, while Hunyuan/MMLU-Pro and Ling/MMLU-Pro remain at 0. CA acts by retaining ID when the development evidence is insufficient for deploying adaptive selection.

Single-predictor variants also exhibit substantial variation across models and benchmarks. As shown in Figure 6, no estimator consistently dominates across all settings, and different predictors perform best under different model–benchmark combinations. The ensemble provides a fixed aggregation of these heterogeneous utility estimates across all settings.

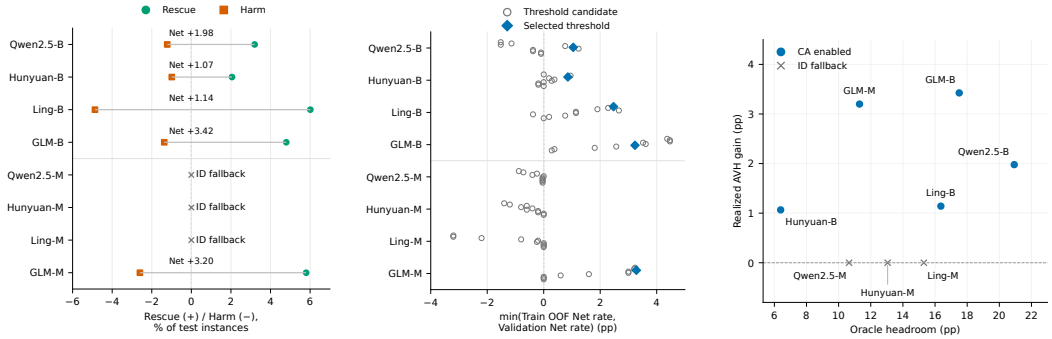

Figure 5: Behavioral analysis of Adaptive View Selection.

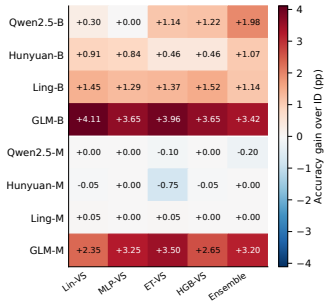

Figure 6: Accuracy gains of predictor variants and their ensemble over ID.

5 Conclusion

We introduced Adaptive View Harness (AVH), which learns from historical multi-view behavior to select a target-preserving input view before inference while preserving a single target-model call. Experiments on BBH and MMLU-Pro with four target LLMs yield positive gains in five of eight model–benchmark settings, with a maximum improvement of $+3.42$ pp, while retaining Identity in the remaining three. The results indicate that pre-call view selection can turn multi-view complementarity into practical performance gains when beneficial switches can be identified from input-side information.

AI use statement

No generative AI tools were involved at any stage of this research, including conceptualization, methodology, experimentation, data analysis, or interpretation of results. The authors independently performed all aspects of this work and bear sole responsibility for its content.

References

Simran Arora, Avanika Narayan, Mayee F. Chen, Laurel J. Orr, Neel Guha, Kush Bhatia, Ines Chami, and Christopher Ré. Ask me anything: A simple strategy for prompting language models. In *International Conference on Learning Representations*, 2023.

Bowen Cao, Dongdong Zhang, Yixia Li, Junpeng Liu, Shijue Huang, Chufan Shi, Hongyuan Lu, Yaokang Wu, Guanhua Chen, Wai Lam, and Furu Wei. From abstract to contextual: What LLMs still cannot do in mathematics. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 24794–24816, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/2a6fc6774932e7835349089b0287ed2c-Paper-Conference.pdf.

Anwoy Chatterjee, H S V N S Kowndinya Renduchintala, Sumit Bhatia, and Tanmoy Chakraborty. POSIX: A prompt sensitivity index for large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 14550–14565, 2024.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024.

Chrisantha Fernando, Dylan Sunil Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 13481–13544. PMLR, 2024.

Walter Gerych, Yara Rizk, Vatche Isahagian, Vinod Muthusamy, Evelyn Duesterwald, and Praveen Venkateswaran. Who knows the answer? finding the best model and prompt for each query using confidence-based search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 18065–18072, 2024.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. LLM-blender: Ensembling large language models with pairwise ranking and generative fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14165–14178, 2023.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into state-of-the-art pipelines. In *International Conference on Learning Representations*, 2024.

Kyungmin Kim, Youngbin Choi, Seoyeon Lee, Suhyeon Jun, Dongwoo Kim, and Sangdon Park. The interplay of harness design and post-training in LLM agents. In *Findings of the Association for Computational Linguistics: EMNLP 2026*, 2026.

Jiahang Lin, Shichun Liu, Chengjun Pan, Lizhi Lin, Shihan Dou, Zhiheng Xi, Xuanjing Huang, Hang Yan, Zhenhua Han, Tao Gui, and Yu-Gang Jiang. Agentic harness engineering: Observability-driven automatic evolution of coding-agent harnesses, 2026.

Qingyi Liu, Jinghui Qin, Wenxuan Ye, Hao Mou, Yuxuan He, and Keze Wang. Adaptive prompt routing for arbitrary text style transfer with pre-trained language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 18689–18697, 2024.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8086–8098, 2022.

Moran Mizrahi, Guy Kaplan, Dan Malkin, Rotem Dror, Dafna Shahaf, and Gabriel Stanovsky. State of what art? a call for multi-prompt LLM evaluation. *Transactions of the Association for Computational Linguistics*, 12:933–949, 2024.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs from preference data. In *International Conference on Learning Representations*, 2025.

Pouya Pezeshkpour and Estevam Hruschka. Large language models sensitivity to the order of options in multiple-choice questions. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 2006–2017, 2024.

Felipe Maia Polo, Ronald Xu, Lucas Weber, Mírian Silva, Onkar Bhardwaj, Leshem Choshen, Allysson Flavio Melo de Oliveira, Yuekai Sun, and Mikhail Yurochkin. Efficient multi-prompt evaluation of LLMs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 22483–22512, 2024.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. In *International Conference on Learning Representations*, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652, 2023.

Hao Sun, Alihan Hüyük, and Mihaela van der Schaar. Query-dependent prompt evaluation and optimization with offline inverse RL. In *International Conference on Learning Representations*, 2024.

Xinyu Tang, Han Peng, Guoxin Chen, Yuze Shi, Zitao Su, Peiyu Liu, Wayne Xin Zhao, Yawen Li, and Zhe Xue. Agent systems with harness engineering: A systematic survey. *Frontiers of Computer Science*, 2026. doi: 10.1007/s11704-026-61123-6.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023.

Zihan Wang, Jiaze Chen, Zhicheng Liu, Haojie Pan, Markus Mak, Yidi Du, Geonsik Moon, Aaron Tua, Kunshuo Peng, Jiayi Lu, Boqian Zou, Chenyang Ran, Guang Tian, Shoutai Zhu, Yeheng Duan, Zhenghui Kang, Zhenxing Lin, Shangshu Li, Qiang Luo, Qingshen Long, Zhiyong Chen, Yihan Xiao, Yurong Wu, Daoguang Zan, Hongyan Li, Mingxuan Wang, and Ming Ding. AetherCode: Evaluating LLMs' ability to win in premier programming competitions. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 140986–141008, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/e4884979ad359b8057161beb08f2bb9b-Paper-Conference.pdf.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022.

Sheng-Lun Wei, Cheng-Kuang Wu, Hen-Hsen Huang, and Hsin-Hsi Chen. Unveiling selection biases: Exploring order and token sensitivity in large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 5598–5621, 2024.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations*, 2024.

Chenyu Zhou, Huacan Chai, Wenteng Chen, Zihan Guo, Rong Shan, Yuanyi Song, Tianyi Xu, Yingxuan Yang, Aofan Yu, Weiming Zhang, Congming Zheng, Jiachen Zhu, Zeyu Zheng, Zhuosheng Zhang, Xingyu Lou, Changwang Zhang, Zhihui Fu, Jun Wang, Weiwen Liu, Jianghao Lin, and Weinan Zhang. Externalization in LLM agents: A unified review of memory, skills, protocols and harness engineering, 2026.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *International Conference on Learning Representations*, 2023.

Jingming Zhuo, Songyang Zhang, Xinyu Fang, Haodong Duan, Dahua Lin, and Kai Chen. ProSA: Assessing and understanding the prompt sensitivity of LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 1950–1976, 2024.

A Appendix. Target-Preserving View Construction and Examples

This appendix describes the construction of the four target-preserving views and provides concrete examples. Throughout the paper, a *target-preserving view* changes the surface presentation of the input while preserving the task semantics, option contents, and gold evaluation target. These views expose view-sensitive model behavior under an unchanged decision target.

A.1 VIEW CONSTRUCTION

Let a multiple-choice input be denoted by $x = (q, \mathcal{O})$, where q is the question text and $\mathcal{O} = \{(l_j, t_j)\}_{j=1}^{n}$ is the set of answer options. Here $l_j \in \{A, B, C, \ldots\}$ is the option label and t_j is the corresponding option text. We construct four fixed views:

- **Identity** (**ID**, V_1). The original prompt is used with its original option order and layout.
- **Order View** (**OV**, V_2). The option list is reordered according to a fixed priority order `C, A, D, B`. Any remaining labels retain their original relative order. The option texts and label-to-text mapping are preserved.
- **Layout View** (**LV**, V_3). The option order is kept unchanged, while the rendering format is altered from the standard `A. ...` style to a bracketed form such as `[A] ...`, with extra line breaks between options.
- **Order+Layout View** (**OLV**, V_4). OV is first applied to reorder the options, and LV is then applied to change the presentation format.

All four views use the same instruction prefix and answer format requirement, with the model instructed to return only the option label. View differences are restricted to option ordering and layout.

The gold target is held fixed across all views. Option labels are not renamed after reordering, so if the correct answer is label `A` in ID, it remains `A` in OV, LV, and OLV. Each transformation therefore presents the same decision problem under a different surface form while preserving its evaluation target. Table 2 summarizes the four views.

Table 2: Definitions of the four target-preserving views.

View	Operation	Example form
ID (V_1)	original order, original layout	`A. ...`
OV (V_2)	reordered options only	`C. / A. / D. / B.`
LV (V_3)	layout change only	`[A] ...`
OLV (V_4)	reordered options + layout change	`[C] / [A] / [D] / [B]`

A.2 PROMPT EXAMPLE

We next show a concrete example from BBH. The example is taken from the `date_understanding` task. The original question is:

```
Jane and John married on Jan 2, 1958. Today is their golden wedding
anniversary. What is the date a month ago in MM/DD/YYYY?
```

The gold answer is `A`. The four prompts given to the target model are as follows.

Identity (ID).

```
Answer the following multiple-choice question.
Return only the option label: A, B, C, D, E, or F.

Jane and John married on Jan 2, 1958. Today is their golden wedding
anniversary. What is the date a month ago in MM/DD/YYYY?

A. 12/02/2007
B. 12/03/2007
C. 01/02/2007
D. 11/18/2007
E. 11/04/2007
F. 12/02/1928
```

```
Answer:
```

Order View (OV).

```
Answer the following multiple-choice question.
Return only the option label: A, B, C, D, E, or F.

Jane and John married on Jan 2, 1958. Today is their golden wedding
anniversary. What is the date a month ago in MM/DD/YYYY?

C. 01/02/2007
A. 12/02/2007
D. 11/18/2007
B. 12/03/2007
E. 11/04/2007
F. 12/02/1928

Answer:
```

Layout View (LV).

```
Answer the following multiple-choice question.
Return only the option label: A, B, C, D, E, or F.

Jane and John married on Jan 2, 1958. Today is their golden wedding
anniversary. What is the date a month ago in MM/DD/YYYY?

[A] 12/02/2007
[B] 12/03/2007
[C] 01/02/2007
[D] 11/18/2007
[E] 11/04/2007
[F] 12/02/1928

Answer:
```

Order+Layout View (OLV).

```
Answer the following multiple-choice question.
Return only the option label: A, B, C, D, E, or F.

Jane and John married on Jan 2, 1958. Today is their golden wedding
anniversary. What is the date a month ago in MM/DD/YYYY?

[C] 01/02/2007
[A] 12/02/2007
[D] 11/18/2007
[B] 12/03/2007
[E] 11/04/2007
[F] 12/02/1928

Answer:
```

This example illustrates the scope of the transformation. The question text, the answer candidates, and the correct label remain fixed. Only the presentation of the options changes.

A.3 Illustrative Behavioral Cases

We next provide several illustrative cases showing how the same instance can lead to different model outcomes under different views.

Case 1: Rescue. For the `date_understanding` example above, Qwen2.5-7B-Instruct answers `B` under ID, which is incorrect, but answers `A` under OV and OLV, which is correct. This is a typical *Rescue* case, where switching away from ID corrects the original prediction.

Case 2: Harm. On the same example, Hunyuan-A13B-Instruct answers `A` under ID, which is correct, but answers `C` under OLV, which is incorrect. This is a typical *Harm* case, where switching to another view turns a correct ID prediction into an incorrect one.

Case 3: No useful alternative. Some instances offer no beneficial alternative view. For example, in a `geometric_shapes` sample, Qwen2.5-7B-Instruct predicts `E` under all four views, while the gold answer is `J`. In such cases, none of the available views yields a correct prediction.

These cases illustrate three forms of view-sensitive behavior. Some instances are rescuable, some incur risk under switching, and some remain unchanged across the available views. A useful harness therefore needs to judge whether switching is beneficial for the current query instead of assuming that one fixed alternative view is globally preferable.

A.4 Practical Remarks

We make several implementation choices to keep the view transformations controlled and comparable.

First, the four views are fixed before any model-specific evaluation. We do not tune the transformations separately for different target LLMs. This keeps the candidate space identical across all experiments.

Second, all views use the same answer parser and the same correctness rule. A response is counted as correct only if its parsed label matches the gold label. Parse failures are treated as incorrect predictions.

Third, the online deployment setting always uses a single target-model call. The historical multi-view responses are collected offline and used only for training and validating the selector. During inference on a new query, AVH chooses one view and submits only that view to the target LLM.

Finally, the examples in this appendix are intended to make the transformations concrete. They are not meant to imply that one view is generally better than another. The empirical results in the main paper show that view utility depends strongly on the model, the benchmark, and the instance.

B Appendix. Implementation and Hyperparameter Detailss

This appendix provides the implementation details omitted from the main paper, including data splits, predictor configuration, ensemble construction, threshold selection, Conservative Activation, and evaluation settings. All selection rules and hyperparameters reported below are fixed before Test evaluation.

B.1 Data Splits and Evaluation Setting

We use fixed Train, Validation, and Test partitions for both benchmarks. BBH contains 11 multiple-choice reasoning tasks in our evaluation subset, with 1052 instances for Train, 263 for Validation, and 1314 for Test. MMLU-Pro contains 2500/500/2000 instances for Train/Validation/Test, respectively. Table 3 summarizes the splits.

The four target-preserving views are ID, OV, LV, and OLV, as described in Appendix A. Historical multi-view responses are used during development, whereas online inference on a new query evaluates only the selected view with the fixed target LLM. AVH therefore requires exactly one target-model call per test instance.

Table 3: Dataset splits used in the AVH experiments.

Benchmark	Train	Validation	Test
BBH	1052 (40%)	263 (10%)	1314 (50%)
MMLU-Pro	2500 (50%)	500 (10%)	2000 (40%)

For output evaluation, a parsed option label is compared with the gold label. Parse failures are counted as incorrect predictions. We do not perform parse-based regeneration or retry.

B.2 Pre-Call Representation and Correctness Predictors

AVH estimates view correctness using information available before target-model inference. The selector uses input-side information derived from the question, answer options, and task/category information. No response generated by the target LLM for the current test query, observed correctness outcome, or gold answer is included in the selector input.

We use four heterogeneous predictor families:

- a linear predictor,
- a multilayer perceptron (MLP),
- Extra Trees,
- Histogram Gradient Boosting (HGB).

Each predictor estimates the correctness probability for each view. The predictor outputs are calibrated before being used as probabilities in the relative-utility computation. The same four predictor families are used for all model–benchmark settings; AVH does not select a different predictor family using Test outcomes.

The base Train predictions used for development are obtained out of fold. Calibration is fitted from the Train OOF predictions and labels and is then frozen. Consequently, Train OOF statistics used by Conservative Activation should be interpreted as development evidence rather than as an independent held-out evaluation.

B.3 Ensemble Utility Estimation

Let $p_{m,v}(x)$ denote the calibrated correctness probability predicted by estimator m for view V_v. The predictor-level utility of an alternative view relative to ID is

$$g_{m,v}(x) = p_{m,v}(x) - p_{m,1}(x), \qquad v > 1, \tag{6}$$

The four predictors are combined using equal weights. With $M = 4$, the ensemble utility is

$$g_v(x) = \frac{1}{4} \sum_{m=1}^{4} g_{m,v}(x), \tag{7}$$

No predictor receives a model-specific ensemble weight. The equal-weight design avoids introducing another development-stage weighting procedure and keeps the same aggregation rule across target models and benchmarks.

For each query, AVH chooses the alternative view with the largest ensemble utility,

$$v^* = \arg\max_{v>1} g_v(x). \tag{8}$$

Ties are resolved using the fixed ordering

$$V_2 \succ V_3 \succ V_4, \tag{9}$$

corresponding to OV, LV, and OLV. This rule is deterministic and is not selected from Test performance.

B.4 Instance-Level Switching Threshold

AVH does not switch whenever an alternative view merely has a positive estimated utility. A candidate is used only when its estimated utility exceeds a frozen threshold τ:

$$g_{v^*}(x) > \tau. \tag{10}$$

The comparison is strict; an instance with $g_{v^*}(x) = \tau$ remains on ID.

The threshold grid is fixed as

$$\mathcal{T} = \{0,\ 0.01,\ 0.025,\ 0.05,\ 0.075,\ 0.10,\ 0.15,\ 0.20,\ 0.25,\ 0.30\}. \tag{11}$$

No threshold outside this grid is introduced after inspecting Test results.

B.5 Development Selection and Conservative Activation

For a candidate threshold τ, we evaluate the corresponding switching policy separately on Train OOF and Validation. Rescue denotes an instance for which ID is incorrect and the selected alternative view is correct. Harm denotes an instance for which ID is correct and the selected alternative view is incorrect. We use

$$\mathrm{Net}(\tau) = \mathrm{Rescue}(\tau) - \mathrm{Harm}(\tau). \tag{12}$$

as the development criterion because it has the same direction as the empirical accuracy change relative to ID.

A threshold is eligible only when

$$\mathrm{Net}_{\mathrm{OOF}}(\tau) > 0 \quad \text{and} \quad \mathrm{Net}_{\mathrm{Val}}(\tau) > 0. \tag{13}$$

Among eligible thresholds, we apply the following fixed ordering:

1. larger Validation Net,
2. fewer Validation Harm cases,
3. fewer Validation switches,
4. larger τ for an exact remaining tie.

The first threshold under this lexicographic ordering is frozen as τ^*.

Conservative Activation operates at the model–benchmark level. If at least one eligible threshold exists, adaptive selection is enabled and the selected τ^* is used for test-time instance-level switching. If no candidate threshold has positive Net on both development partitions, adaptive selection is disabled and all instances in that model–benchmark setting retain ID.

This distinction separates two decisions in AVH. The threshold controls whether an individual query has enough predicted utility to switch, whereas CA determines whether the adaptive policy is deployed for the corresponding model–benchmark setting.

Table 4: Frozen AVH thresholds and Conservative Activation states.

Benchmark	Model	τ^*	CA
BBH	Qwen2.5	0.075	Enabled
BBH	Hunyuan	0.025	Enabled
BBH	Ling	0.000	Enabled
BBH	GLM	0.100	Enabled
MMLU-Pro	Qwen2.5	–	Disabled
MMLU-Pro	Hunyuan	–	Disabled
MMLU-Pro	Ling	–	Disabled
MMLU-Pro	GLM	0.025	Enabled

Table 5: Implementation and hyperparameter configuration of AVH.

Component	Configuration
Target-preserving views	ID, OV, LV, OLV
Predictor families	Linear, MLP, Extra Trees, HGB
Predictor input	Question, options, and available task/category information
Current-query target response	Not used by the selector
Predictor output	Calibrated per-view correctness probability
Ensemble	Equal-weight mean over four predictor families
Ensemble weight	0.25 per predictor family
Candidate views	OV, LV, OLV
Candidate tie-breaking	OV $\succ$ LV $\succ$ OLV
Threshold grid	$\{0, .01, .025, .05, .075, .10, .15, .20, .25, .30\}$
Instance switching	$g_{v^*}(x) > \tau^*$
Threshold eligibility	Train OOF Net > 0 and Validation Net > 0
Threshold selection	Max Validation Net, min Harm, min switches, larger τ
CA fallback	Use ID for the entire setting if no eligible threshold exists
Online inference	One target-model call per instance
Parse failure	Counted as incorrect; no regeneration or retry
Primary metric	Accuracy and accuracy gain over ID
Switching diagnostics	Switch Rate, Rescue, Harm, and Net
Confidence interval	Paired instance-level bootstrap
Bootstrap repetitions	10,000
Bootstrap interval	Percentile 95% CI
Bootstrap seed	20260922
Multiplicity correction	None

B.6 Frozen Thresholds

Table 4 reports the thresholds frozen by the development procedure for the four target models used in the main paper. A dash denotes a setting in which no threshold satisfies the CA eligibility criterion and adaptive selection is therefore disabled.

A threshold of 0 for Ling/BBH still uses the strict switching condition $g_{v^*}(x) > 0$. A disabled setting is distinct from using $\tau = 0$: when CA is disabled, no adaptive switching is performed and ID is used for every test instance.

B.7 Configuration Summary

Table 5 summarizes the configuration shared across the main experiments.

B.8 Confidence Intervals

For the main AVH comparison with ID, we estimate uncertainty using paired instance-level bootstrap. Each bootstrap replicate samples N test instances with replacement and evaluates AVH and ID on the same resampled instances. The reported statistic is the paired accuracy difference in percentage points. We use 10,000 bootstrap replicates and report the 2.5th and 97.5th percentiles as the 95% confidence interval.

The bootstrap procedure evaluates the frozen policy and does not repeat predictor fitting, calibration, threshold selection, or CA selection within each replicate. No multiplicity correction is applied across model–benchmark settings. For a setting in which CA disables adaptive selection, AVH and ID are identical by construction, yielding a point difference of zero and a degenerate paired interval of $[0, 0]$.

B.9 Frozen Evaluation Protocol

All predictor checkpoints, calibration objects, preprocessing components, threshold candidates, tie-breaking rules, and activation decisions are frozen before the reported Test evaluation. Test outcomes are not used to select a predictor family, ensemble weight, threshold, or CA state. In particular, Oracle and the post-hoc best fixed view are diagnostic quantities and do not participate in AVH selection.

The online selector never queries the target LLM before choosing a view. After AVH selects either V_{v^*} or ID, only that single transformed input is sent to the target model. Historical multi-view probing is therefore an offline development cost rather than an additional target-model inference cost for a new query.

C Appendix. Full AVH Algorithm

This appendix summarizes the complete AVH procedure from development to online inference. The development stage learns the view-utility estimators and freezes the deployment policy, while the online stage selects one view before invoking the target LLM.

Development stage.

1. Construct the four target-preserving views for each development instance and collect their historical correctness outcomes.
2. Train the Linear, MLP, Extra Trees, and HGB correctness predictors using the Train split.
3. Obtain out-of-fold predictions on Train and calibrate the predictor outputs.
4. Average the four calibrated predictors with equal weights to estimate the correctness of each view.
5. Use ID as the reference and compute the estimated gain of each alternative view.
6. For each candidate threshold, apply the instance-level switching rule on Train OOF and Validation.
7. Retain thresholds with positive Net on both development partitions.
8. Select the threshold with the highest Validation Net, using Harm, switch count, and threshold value for tie-breaking.
9. If no threshold is eligible, disable adaptive selection and retain ID for the entire model–benchmark setting.
10. Freeze the predictors, calibration, selected threshold, and Conservative Activation state.

Online inference.

1. For a new query, compute the pre-call features without querying the target LLM.
2. Estimate the correctness of all candidate views and identify the alternative with the largest estimated gain over ID.

Table 6: Detailed AVH performance on the eight main model–benchmark settings.

Benchmark	Model	N	ID	AVH	Δ ID	95% CI	Oracle
BBH	Qwen2.5	1314	41.32	43.30	+1.98	[0.84, 3.12]	62.25
BBH	Hunyuan	1314	90.03	91.10	+1.07	[0.15, 1.98]	96.42
BBH	Ling	1314	45.66	46.80	+1.14	[-0.61, 2.89]	62.02
BBH	GLM	1314	47.95	51.37	+3.42	[2.13, 4.72]	65.45
MMLU-Pro	Qwen2.5	2000	40.70	40.70	+0.00	[0.00, 0.00]	51.35
MMLU-Pro	Hunyuan	2000	59.20	59.20	+0.00	[0.00, 0.00]	72.25
MMLU-Pro	Ling	2000	39.70	39.70	+0.00	[0.00, 0.00]	55.00
MMLU-Pro	GLM	2000	41.45	44.65	+3.20	[1.95, 4.45]	52.75

Accuracy values are percentages; Δ ID and confidence intervals are percentage points.

Table 7: Switching behavior of AVH on the eight main settings.

Benchmark	Model	Switch Rate	Rescue	Harm	Net	CA
BBH	Qwen2.5	18.95	42	16	26	Enabled
BBH	Hunyuan	25.19	27	13	14	Enabled
BBH	Ling	54.79	79	64	15	Enabled
BBH	GLM	33.18	63	18	45	Enabled
MMLU-Pro	Qwen2.5	0.00	0	0	0	Disabled
MMLU-Pro	Hunyuan	0.00	0	0	0	Disabled
MMLU-Pro	Ling	0.00	0	0	0	Disabled
MMLU-Pro	GLM	62.50	116	52	64	Enabled

Switch Rate is reported in percent. Net is defined as Rescue minus Harm. Disabled settings retain ID for every Test instance.

3. If Conservative Activation is disabled, retain ID; otherwise, switch only when the estimated gain exceeds the frozen threshold.
4. Construct the selected view and invoke the target LLM exactly once.
5. Parse the returned option label as the final prediction.

D Appendix. Complete Experimental Results

This appendix provides the complete numerical results corresponding to the main experimental analyses. We first report the full AVH performance and switching statistics, followed by the detailed fixed-view results underlying the view-complementarity analysis.

D.1 Detailed AVH Results

Table 6 reports the Test accuracy of ID and AVH, the corresponding gain over ID, paired 95% confidence intervals, and Oracle accuracy. Table 7 reports the associated switching behavior.

Across BBH, AVH improves the point estimate for all four target models, with gains ranging from $+1.07$ to $+3.42$ pp. On MMLU-Pro, adaptive selection is enabled for GLM and yields a $+3.20$ pp gain, while the other three settings retain ID. Four of the five positive-gain settings have confidence intervals with lower bounds above zero; the interval for Ling/BBH crosses zero.

The switching statistics illustrate why Switch Rate alone is insufficient to characterize the effectiveness of adaptive selection. Ling/BBH switches on 54.79% of Test instances but obtains a Net of only 15 because Rescue and Harm frequently coexist. Qwen2.5/BBH switches on only 18.95% of instances yet obtains a larger accuracy gain because its 42 Rescue cases are accompanied by only 16 Harm cases. GLM/MMLU-Pro exhibits the largest number of both Rescue and Harm cases, but the positive difference between them produces a Net of 64 and a $+3.20$ pp accuracy gain.

Table 8: Test accuracy of fixed-view strategies and diagnostic upper bounds.

Benchmark	Model	ID	OV	LV	OLV	DS-FV	Post-hoc Best	Oracle
BBH	Qwen2.5	41.32	41.93	42.85	42.77	41.32	42.85	62.25
BBH	Hunyuan	90.03	86.30	91.93	88.43	91.93	91.93	96.42
BBH	Ling	45.66	44.60	44.14	43.53	44.60	45.66	62.02
BBH	GLM	47.95	50.91	52.36	50.99	52.36	52.36	65.45
MMLU-Pro	Qwen2.5	40.70	35.90	40.10	33.45	40.70	40.70	51.35
MMLU-Pro	Hunyuan	59.20	58.75	60.30	58.20	59.20	60.30	72.25
MMLU-Pro	Ling	39.70	36.60	40.20	36.70	39.70	40.20	55.00
MMLU-Pro	GLM	41.45	43.20	42.55	44.70	43.20	44.70	52.75

All values are Test accuracy (%). DS-FV is selected using Validation; Post-hoc Best and Oracle use Test outcomes only for diagnostic analysis.

D.2 Fixed-View Results

Table 8 provides the complete fixed-view results underlying the analysis in Section 4.2. OV, LV, and OLV apply one alternative view uniformly to the entire Test set. DS-FV instead selects one globally fixed view using Validation. For diagnostic comparison, Post-hoc Best reports the highest Test accuracy among ID, OV, LV, and OLV after observing Test outcomes; it is not a deployable baseline.

The fixed-view results show that no alternative view is uniformly beneficial across models and benchmarks. For example, LV improves Hunyuan/BBH from 90.03% to 91.93%, whereas OV reduces it to 86.30%. On MMLU-Pro, OLV substantially reduces Qwen2.5 accuracy but provides the strongest fixed-view result for GLM. These differences support the instance-dependent treatment of view utility adopted by AVH.

The gap between Post-hoc Best and Oracle further shows that selecting one view for an entire dataset leaves substantial instance-level headroom unused. Averaged across the eight main settings, the post-hoc best fixed view improves over ID by approximately 1.59 pp, whereas the average Oracle headroom is 13.94 pp. This gap motivates selecting views at the query level rather than assuming a globally preferred input representation.

E Appendix. Additional Ablation Analysis

This appendix provides additional analysis of Conservative Activation and the heterogeneous predictor ensemble.

E.1 Conservative Activation

Conservative Activation operates only at the model–benchmark level and does not modify the underlying instance-level utility estimator. Table 9 compares AVH with and without this fallback mechanism.

For all enabled settings, CA leaves the instance-level selection policy unchanged. When development evidence does not satisfy the activation criterion, CA instead retains ID. This behavior is consistent with its intended role as a conservative deployment control.

E.2 Predictor Variants

AVH combines Linear, MLP, Extra Trees, and HGB correctness estimators using equal weights. As analyzed in Section 4.4, the relative performance of individual predictors varies across model–benchmark settings. No single predictor family consistently dominates the others, supporting the use of heterogeneous estimators rather than committing the selector to one predictor family. The ensemble uses the same fixed aggregation rule across all experiments.

Table 9: Effect of Conservative Activation on accuracy gain over ID.

Benchmark	Model	w/o CA	AVH	CA
BBH	Qwen2.5	+1.98	+1.98	Enabled
BBH	Hunyuan	+1.07	+1.07	Enabled
BBH	Ling	+1.14	+1.14	Enabled
BBH	GLM	+3.42	+3.42	Enabled
MMLU-Pro	Qwen2.5	-0.20	+0.00	Disabled
MMLU-Pro	Hunyuan	+0.00	+0.00	Disabled
MMLU-Pro	Ling	+0.00	+0.00	Disabled
MMLU-Pro	GLM	+3.20	+3.20	Enabled

Values are accuracy changes relative to ID in percentage points.

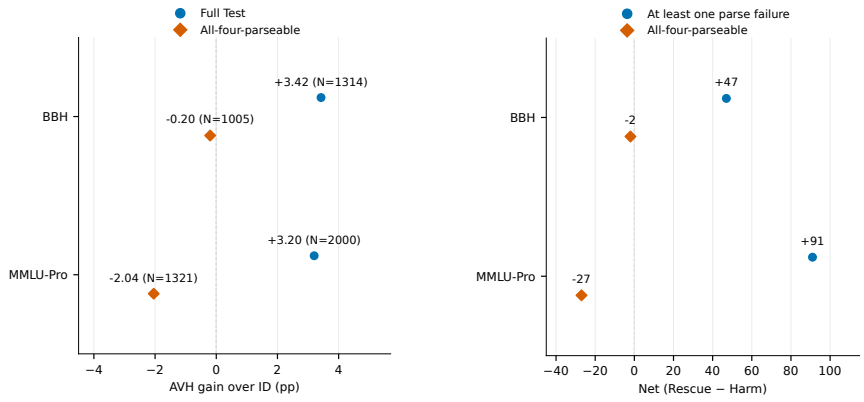

Figure 7: Parseability sensitivity analysis for GLM.

F Appendix. Parseability Sensitivity Analysis

The main evaluation treats parse failures as incorrect predictions and therefore measures end-to-end prediction performance under the frozen direct-answer protocol. Since target-preserving views can affect both answer selection and output formatting, we additionally examine GLM on the subset of instances for which all four view responses are parseable.

Figure 7 compares the complete Test evaluation with the all-four-parseable subset. The results indicate that part of the observed benefit is associated with view-dependent output-format behavior. This is consistent with the end-to-end objective of AVH, which evaluates the final parsed prediction produced under the selected target-preserving view.

The all-four-parseable subset is constructed from the observed responses and is used as a sensitivity analysis of output parseability. Predictor fitting, threshold selection, and Conservative Activation follow the original development protocol and remain unchanged.