PROMPT OPTIMIZATION AS A PROBE FOR DATA-EFFICIENT MULTIMODAL GRPO

Anonymous authors
Paper under double-blind review

ABSTRACT

Group Relative Policy Optimization (GRPO) can substantially improve multimodal reasoning, but its reliance on multiple rollouts per training problem makes data-efficient training important. Existing data-selection methods typically estimate sample utility from signals observed under a fixed prompt, providing only one view of the current policy's behavior. We introduce **PACE** (**P**rompt-Induced **A**nswer **C**hange **E**stimation), which repurposes prompt optimization as a controlled behavioral probe for GRPO data selection. PACE keeps the initial policy fixed and compares answer correctness under the original and optimized prompts. The resulting bidirectional transitions—unlocks from incorrect to correct and regressions from correct to incorrect—define a prompt-sensitive boundary that captures how the policy responds across elicitation conditions. PACE aggregates these transitions within semantic clusters, retains regions where optimized prompting has a positive aggregate effect, and applies balanced selection to construct a compact training subset. The optimized prompt is used only for probing and selection; downstream GRPO training and evaluation use the original prompt. Experiments using MM-Eureka for training and seven multimodal mathematical reasoning benchmarks for evaluation show that PACE selects only $5{,}000$ examples, approximately 9.1% of the full training set. On Qwen2.5-VL-7B-Instruct, PACE achieves an average accuracy of 51.1, compared with 50.7 for Full-Data GRPO trained on $54{,}931$ examples, while outperforming fixed-budget, larger-budget, and online selection baselines. PACE remains effective across Qwen2.5-VL-3B and InternVL3-2B. Further analyses show that the prompt-sensitive boundary captures information complementary to conventional fixed-prompt signals, that both transition directions provide useful training signals, and that prompt-sensitive filtering improves selection strategies. These results demonstrate that prompt-induced behavioral responsiveness provides an effective signal for data-efficient multimodal GRPO.

1 INTRODUCTION

Training-data selection is a central problem in efficient model training, particularly when each example is costly to optimize. Group Relative Policy Optimization (GRPO) has emerged as an effective approach for mathematical (Shao et al., 2024) and multimodal reasoning (Meng et al., 2025), but requires multiple response rollouts and reward evaluations per training problem. This cost has motivated data-selection methods that concentrate training on smaller subsets with higher estimated utility. Existing methods estimate sample utility using a variety of signals, including model learning trajectories (Li et al., 2025), sample learnability and gradient alignment (Li et al., 2026b), policy-gradient alignment with a trusted validation set (Yang et al., 2026), MCTS-based solving difficulty (Wang et al., 2025), multimodal–textual discrepancy and attention confidence (Li et al., 2026a), and online sample-level difficulty estimates (Hu et al., 2025). However, none estimates sample utility by varying prompts for a fixed model and measuring correctness changes on the same example.

This omission may matter because a model's response depends not only on the input and its parameters, but also on how the task is elicited. Chain-of-thought prompting, for example, can elicit reasoning behaviors that are not produced under standard prompting (Wei et al., 2022). Even minor or meaning-preserving changes in prompt wording and structure can alter model outputs and measured performance (Chatterjee et al., 2024; Zhuo et al., 2024;

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.

Ismithdeen et al., 2025). Consequently, correctness or difficulty measured under a single prompt may provide only a partial view of an example's training value for the current policy. As illustrated in Figure 1, the same policy may fail on an example under one prompt while solving it correctly under another. We hypothesize that examples whose correctness changes across prompts are those that the current policy is capable of solving, but cannot solve consistently under different prompts. If an incorrect answer becomes correct under another instruction, the policy can produce the correct response, but it does not do so consistently. If a correct answer becomes incorrect, the apparent success is fragile to changes in elicitation. In both cases, the example may be particularly informative for GRPO because the correct response is not yet reliably produced across elicitation conditions.

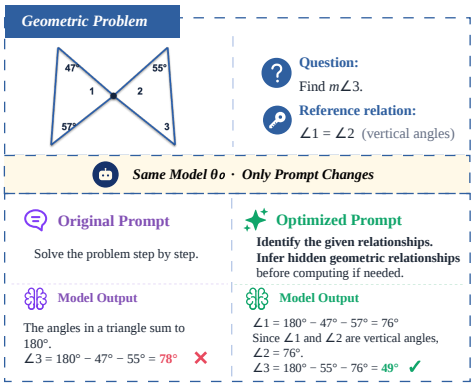


Figure 1: **Motivating example of prompt-induced change.** For the same problem and frozen policy θ_0, changing only the prompt can elicit a different reasoning path, turning an incorrect response into a correct one.

To test this hypothesis, we introduce **PACE** (**P**rompt-Induced **A**nswer **C**hange **E**stimation), an offline data-selection framework that repurposes prompt optimization as a controlled behavioral probe. Prompt optimizers typically search for instructions that improve task performance (Pryzant et al., 2023; Yang et al., 2024; Agrawal et al., 2026). PACE instead uses the optimized prompt only to construct an alternative elicitation condition. It evaluates the same frozen policy on the same examples under the original and optimized prompts while keeping the policy parameters and correctness evaluator fixed. An *unlock* occurs when an answer changes from incorrect to correct, whereas a *regression* occurs when it changes from correct to incorrect. Their union forms the *prompt-sensitive boundary*, capturing examples whose correctness is responsive to elicitation. PACE then aggregates prompt-induced correctness changes within semantic clusters to retain broadly helpful regions and construct a fixed-budget training subset. Importantly, the optimized prompt is used only for offline probing and data selection; GRPO training and evaluation use the original prompt.

We evaluate PACE on seven multimodal mathematical reasoning benchmarks using Qwen2.5-VL (Bai et al., 2025) and InternVL3 (Zhu et al., 2025) models. On Qwen2.5-VL-7B-Instruct, PACE selects only $5{,}000$ examples from a training set of $54{,}931$, or approximately 9.1% of the available data. It achieves an average accuracy of 51.1, compared with 50.7 for GRPO trained on the full original training set, while outperforming fixed-budget baselines and methods that use larger or adaptively determined training budgets. PACE improves over the Base model by 2.7 points, and yields gains on all seven benchmarks, with consistent improvements across model scales and architectures. The prompt-sensitive boundary has only weak Spearman correlations with conventional fixed-prompt signals, which select largely different examples from PACE. These results suggest that prompt sensitivity captures information complementary to standard difficulty and confidence measures. Moreover, both unlock-only and regression-only training improve over the Base model, while their union provides the strongest downstream performance. Together, these findings support prompt-induced correctness change as a distinct and useful signal for GRPO data selection.

Our contributions are threefold:

- To the best of our knowledge, we are the first to use prompt-induced correctness change as an offline signal for GRPO data selection. This signal targets examples that the current policy can solve under some prompts but not consistently across different instructions.
- We operationalize this signal through PACE, which compares a frozen policy under the original and optimized prompts, identifies bidirectional correctness transitions, and constructs a compact training subset using cluster-level helpfulness estimation and fixed-budget selection.
- With only $5{,}000$ of $54{,}931$ training examples, PACE matches Full-Data GRPO and outperforms competing baselines across model scales and architectures. Further analyses show

that its signal is complementary to fixed-prompt measures and can improve existing selectors as a candidate filter.

2 Related Work

Prior work on supervised instruction tuning has shown that carefully selected subsets can retain or even exceed full-data performance through LLM-based quality filtering (Chen et al., 2024; Liu et al., 2024a), joint consideration of data quality, complexity, and diversity (Liu et al., 2024b; Bukharin et al., 2024), or gradient-based influence estimation (Xia et al., 2024; Dai et al., 2025). More recent work has studied data selection for text-based reasoning RL. LIMR (Li et al., 2025) estimates sample utility based on its alignment with model learning trajectories, while LearnAlign (Li et al., 2026b) combines sample learnability with gradient alignment to identify useful and representative reasoning problems. GradAlign (Yang et al., 2026) further selects training problems whose policy gradients align with those of a trusted validation set. Recent work has also explored data selection for multimodal reasoning RL. ThinkLite-VL (Wang et al., 2025) uses MCTS solving effort as a model-dependent difficulty signal, RAP (Li et al., 2026a) selects examples using multimodal–textual prediction discrepancy and attention confidence, and VADE (Hu et al., 2025) dynamically prioritizes examples through online difficulty estimation. Across these approaches, sample utility is estimated from behavior observed under a fixed instruction or from signals collected as training evolves. These methods do not deliberately change the elicitation condition while holding the policy fixed. Automatic prompt optimization, in contrast, searches over instructions but typically treats the optimized prompt as the final artifact for improving task performance (Yang et al., 2024; Agrawal et al., 2026). PACE takes a different view: it uses prompt optimization as a controlled behavioral intervention and converts the resulting bidirectional correctness transitions into a signal for constructing a compact, fixed training subset before GRPO.

3 Problem Formulation

Let $\mathcal{D} = \{z_i\}_{i=1}^{N}$ denote a multimodal reasoning training dataset, where each example is defined as $z_i = (x_i, y_i)$. Each input x_i consists of a visual input v_i and its associated textual question q_i, and y_i is the corresponding ground-truth answer. Given a data budget $B \ll N$, our goal is to select a subset $\mathcal{S} \subseteq \mathcal{D}$ with $|\mathcal{S}| = B$ that maximizes downstream generalization performance after GRPO fine-tuning. Let θ_0 be the initial policy model and P_0 the original training prompt. We write $\theta_{\mathcal{S}} = \mathrm{GRPO}(\theta_0, \mathcal{S}; P_0)$ for the policy obtained by training on $\mathcal{S}$ with GRPO. The ideal budgeted data-selection objective is

$$\mathcal{S}^\star = \arg \max_{\substack{\mathcal{S} \subseteq \mathcal{D} \\ |\mathcal{S}| = B}} \mathcal{J}(\theta_{\mathcal{S}}), \tag{1}$$

where $\mathcal{J}$ measures generalization on evaluation tasks, which are not accessed during data selection.

4 PACE

To derive an offline signal for GRPO data selection, we propose PACE (**P**rompt-Induced **A**nswer **C**hange **E**stimation). PACE first optimizes the original training prompt P_0 on a compact development set while keeping the initial policy θ_0 fixed, yielding an optimized prompt $P^\star$. Whereas standard prompt optimization uses the optimized prompt to directly improve task performance, PACE uses $P^\star$ only as a controlled probe of the fixed policy's behavior.

PACE evaluates the same initial policy θ_0 on the same training examples under P_0 and $P^\star$. The policy parameters, examples, correctness evaluator, and deterministic decoding procedure remain fixed, while only the elicitation condition is changed. Specifically, we use greedy decoding under both P_0 and $P^\star$. This controlled comparison reveals examples for which the policy's correctness is responsive to prompting. PACE then aggregates these prompt-induced correctness transitions within semantic clusters and uses the resulting cluster-level signal to construct a balanced subset under the data budget B. The selected subset is used for GRPO fine-tuning with the original prompt P_0; the optimized prompt is not used during downstream training or evaluation. Figure 2 provides an overview of the PACE pipeline. The complete PACE algorithm is provided in Appendix G.

4.1 Semantic Partition and Development-Set Construction

PACE requires a semantic partition to provide distribution-aware units for constructing the development set: representative examples can be allocated according to the prevalence of different semantic regions. We encode each textual question q_i using a pretrained Sentence-BERT encoder $\phi(\cdot)$ and apply K-means clustering to the resulting embeddings, partitioning the training set $\mathcal{D}$ into K disjoint semantic clusters $\{\mathcal{C}_k\}_{k=1}^{K}$. Based on this partition, we construct a compact development set by combining cluster-proportional sampling for representative coverage with farthest-point augmentation for semantic diversity. Given a development-set budget of M, we allocate $M_{\text{rep}} = \lfloor M/2 \rfloor$ examples to representative sampling and the remaining $M_{\text{div}} = M - M_{\text{rep}}$ examples to diversity augmentation. This fixed split balances representative coverage and semantic diversity without additional tuning.

Representative coverage. We allocate the representative budget approximately in proportion to cluster size. Specifically, the quota for cluster $\mathcal{C}_k$ is

$$m_k \approx M_{\text{rep}} \frac{|\mathcal{C}_k|}{|\mathcal{D}|}, \qquad \sum_{k=1}^{K} m_k = M_{\text{rep}}, \tag{2}$$

where the quotas are rounded to integers while preserving the representative budget. From each cluster $\mathcal{C}_k$, we select the m_k examples closest to its centroid. The union of these selected examples forms $\mathcal{D}_{\text{rep}}$, which provides representative coverage of the major semantic regions in the training set.

Diversity augmentation. To broaden the semantic coverage beyond the representative subset, we first identify examples located in relatively sparse regions of the embedding space. For each example in $\mathcal{D} \setminus \mathcal{D}_{\text{rep}}$, we compute the mean cosine distance between its question embedding and those of its L nearest neighbors. We retain the Q examples with the largest mean distances to form the candidate set $\mathcal{D}_{\text{cand}}$. We initialize $\mathcal{D}_{\text{div}}^{(2)}$ with the pair of examples in $\mathcal{D}_{\text{cand}}$ having the largest pairwise distance. For each subsequent iteration $t = 3, \ldots, M_{\text{div}}$, we select

$$z_t^{\text{div}} = \underset{z_i \in \mathcal{D}_{\text{cand}} \setminus \mathcal{D}_{\text{div}}^{(t-1)}}{\arg\max} \; \min_{z_j \in \mathcal{D}_{\text{div}}^{(t-1)}} d\left(\phi(q_i), \phi(q_j)\right), \tag{3}$$

and update the selected set as $\mathcal{D}_{\text{div}}^{(t)} = \mathcal{D}_{\text{div}}^{(t-1)} \cup \{z_t^{\text{div}}\}$. Here, $d(\cdot, \cdot)$ denotes cosine distance between ℓ_2-normalized question embeddings. At each iteration, this criterion selects the candidate whose nearest previously selected example is farthest away, encouraging coverage of semantic regions.

After selecting M_{div} examples, we set $\mathcal{D}_{\text{div}} = \mathcal{D}_{\text{div}}^{(M_{\text{div}})}$ and construct the final development set as $\mathcal{D}_{\text{dev}} = \mathcal{D}_{\text{rep}} \cup \mathcal{D}_{\text{div}}$, where $|\mathcal{D}_{\text{dev}}| = M$. Representative sampling captures common question patterns, while farthest-point augmentation broadens the semantic coverage of the development set.

4.2 Prompt Optimization and Prompt-Sensitive boundary

For each example $z_i = (x_i, y_i)$, let $f_{\theta_0}(x_i; P)$ denote the response generated by the frozen initial policy θ_0 under prompt P. We define its correctness as $c_P(z_i) = \mathbb{I}[f_{\theta_0}(x_i; P)$ is correct with respect to $y_i]$, using the same correctness evaluator for all prompts. Starting from the original prompt P_0, we apply a prompt optimization procedure $\mathcal{A}_{\text{PO}}$ that searches for a prompt with higher development-set accuracy while keeping the policy parameters θ_0 fixed:

$$P^\star = \mathcal{A}_{\text{PO}}\left(P_0; \theta_0, \mathcal{D}_{\text{dev}}\right). \tag{4}$$

The prompt optimizer changes only the prompting condition; the model, development examples, and correctness evaluator remain fixed throughout the optimization process. PACE does not use $P^\star$ as the downstream training or evaluation prompt. Instead, P_0 and $P^\star$ define two controlled prompting conditions for probing the behavior of the same initial policy. Differences in correctness between these conditions reveal examples for which the model's behavior is sensitive to how the task is elicited. We then measure the prompt-induced change in answer correctness for each $z_i \in \mathcal{D}$ as

$$\Delta(z_i) = c_{P^\star}(z_i) - c_{P_0}(z_i) \in \{-1, 0, +1\}. \tag{5}$$

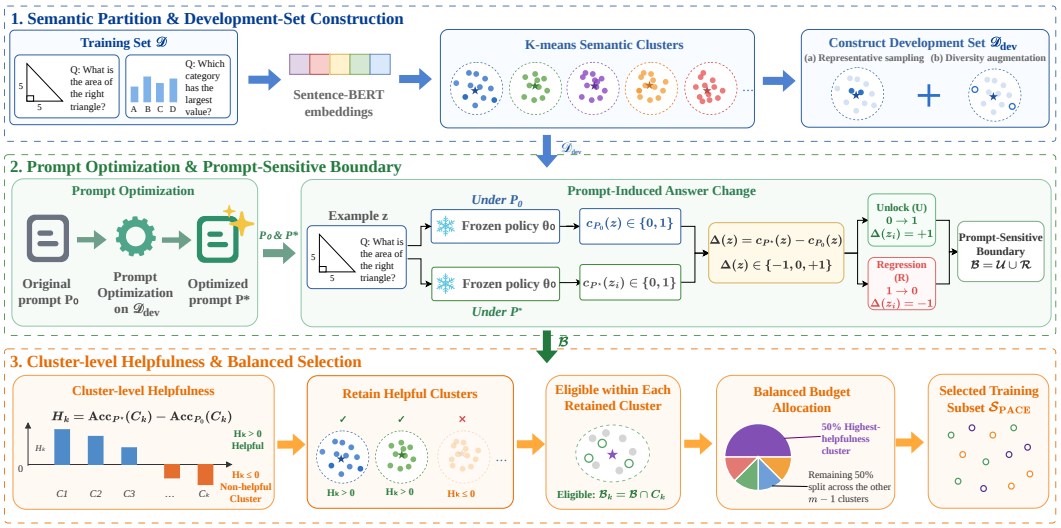


Figure 2: Overview of PACE. The optimized prompt probes a frozen policy to identify prompt-sensitive examples, which are then selected using cluster-level helpfulness and balanced allocation.

A nonzero value indicates that answer correctness changes between the two prompting conditions. The two transition directions are

$$\mathcal{U} = \{z_i \in \mathcal{D} : c_{P_0}(z_i) = 0,\ c_{P^\star}(z_i) = 1\},$$
$$\mathcal{R} = \{z_i \in \mathcal{D} : c_{P_0}(z_i) = 1,\ c_{P^\star}(z_i) = 0\}. \tag{6}$$

Unlocks $\mathcal{U}$ are examples for which $P^\star$ elicits a correct answer that is not produced under P_0. Regressions $\mathcal{R}$ are examples that are answered correctly under P_0 but incorrectly under $P^\star$.

Although regressions have a negative prompt-induced change, $\Delta(z_i) = -1$, the sign of an individual transition does not directly indicate its downstream training utility. Both unlocks and regressions capture behavioral responsiveness: the correctness of the same frozen policy changes when only the elicitation condition is modified. We therefore retain both transition directions and define their union as the prompt-sensitive boundary:

$$\mathcal{B} = \mathcal{U} \cup \mathcal{R} = \{z_i \in \mathcal{D} : |\Delta(z_i)| = 1\}. \tag{7}$$

4.3 Helpfulness-Guided Balanced Selection

A sample-level correctness transition indicates that an individual example is sensitive to the prompting condition, but it does not reveal whether the optimized prompt improves answer correctness more broadly among semantically similar examples. We therefore introduce *prompt-induced semantic helpfulness*, which aggregates prompt-induced changes in answer correctness within each semantic cluster. For each cluster $\mathcal{C}_k$, we define its prompt-induced semantic helpfulness as

$$H_k = \frac{1}{|\mathcal{C}_k|} \sum_{z_i \in \mathcal{C}_k} \Delta(z_i) = \mathrm{Acc}_{P^\star}(\mathcal{C}_k) - \mathrm{Acc}_{P_0}(\mathcal{C}_k), \tag{8}$$

where $\mathrm{Acc}_P(\mathcal{C}_k) = |\mathcal{C}_k|^{-1} \sum_{z_i \in \mathcal{C}_k} c_P(z_i)$ denotes the average correctness on cluster $\mathcal{C}_k$ under prompt P. Under this definition, unlocks contribute $+1$, regressions contribute -1, and examples whose correctness remains unchanged contribute 0. Thus, H_k measures the net change in answer correctness induced by the change from P_0 to $P^\star$ within cluster $\mathcal{C}_k$.

We use H_k to determine which clusters are retained. Specifically, we retain clusters with positive helpfulness, $\mathcal{K}_{\mathrm{ret}} = \{k \mid H_k > 0\}$. Within each retained cluster $\mathcal{C}_k$, only prompt-sensitive examples $\mathcal{B}_k = \mathcal{B} \cap \mathcal{C}_k$ are eligible for selection. We next allocate the data budget B across the retained clusters. We assign a larger share of the budget to the cluster with the highest helpfulness, while reserving the remaining budget for the other retained clusters. We first identify this cluster as $k_{\max} = \arg\max_{k \in \mathcal{K}_{\mathrm{ret}}} H_k$.

Let q_k^{tar} denote the target quota for retained cluster k. To balance concentration on the most helpful semantic region with coverage of the remaining retained regions, we allocate half of the selection budget to the cluster with the highest helpfulness: $q_{k_{\max}}^{\mathrm{tar}} = \lfloor B/2 \rfloor$. We divide the remaining budget as evenly as possible among the other retained clusters, assigning any leftover slots randomly. This fixed allocation is used in all experiments without tuning. Within each retained cluster, we select the prompt-sensitive examples closest to the cluster centroid until its target quota is reached. Combining the selected examples yields the final training subset $\mathcal{S}_{\mathrm{PACE}}$ with $|\mathcal{S}_{\mathrm{PACE}}| = B$.

5 Experimental Setup

Research Questions We organize our evaluation around five key questions: **RQ1.** How does PACE compare with GRPO trained on the full dataset and existing data-selection methods in terms of performance and data efficiency? **RQ2.** Does PACE remain effective across model scales and architectures? **RQ3.** Which components of PACE contribute to its effectiveness and data efficiency? **RQ4.** What does the prompt-sensitive boundary capture beyond difficulty, and how do unlocks and regressions contribute to downstream GRPO training? **RQ5.** Can the prompt-sensitive boundary serve as a reusable filter for other data-selection methods to yield additional performance gains?

Models and Baselines Our main experiments use Qwen2.5-VL-7B-Instruct (Bai et al., 2025). We additionally evaluate Qwen2.5-VL-3B-Instruct and InternVL3-2B (Zhu et al., 2025) to examine transfer across model scales and architectures. We compare PACE with the following baselines. **Base** denotes the initial policy without GRPO fine-tuning. **Random** uniformly samples examples from the training dataset. **Pass Rate** (Bae et al., 2026) ranks examples by their distance to a pass rate of 0.5 estimated from four fixed-prompt rollouts and selects the closest examples. **LIMR** (Li et al., 2025) selects examples according to estimated learning progress. **ThinkLite-VL** (Wang et al., 2025) uses MCTS search effort as its selection signal, while **VADE** (Hu et al., 2025) performs online selection as the policy evolves. **Full-Data GRPO** trains on all $54{,}931$ examples and serves as the high-data reference. The number of training examples used by each method is reported in the corresponding result tables. For RQ5, we additionally consider four heterogeneous selection strategies: k-Center (Sener & Savarese, 2017), Facility Location (Wei et al., 2015), entropy (Settles & Craven, 2008), and Random selection.

Datasets We perform data selection and GRPO training on MM-Eureka (Meng et al., 2025), which contains $54{,}931$ multimodal mathematical reasoning examples. We evaluate on seven multimodal mathematical reasoning benchmarks: MathVista (Lu et al., 2024), MathVerse (Zhang et al., 2024), WeMath (Qiao et al., 2025), MathVision (Wang et al., 2024), MM-Math (Sun et al., 2024), Dyna-Math (Zou et al., 2025), and PGPS9K (Zhang et al., 2023). To prevent data leakage, we ensure that there is no example-level overlap between the training set and any of the evaluation sets.

Configuration We instantiate PACE with GEPA and use greedy decoding for all probing runs so that correctness transitions reflect changes in the prompting condition rather than stochastic decoding noise. All controlled comparisons use the same initial checkpoint, training prompt, reward functions, and GRPO configuration. Full implementation and training details are provided in Appendix A.

Evaluation For multiple-choice questions, we use a deterministic parser to extract the predicted option, followed by case normalization and the removal of irrelevant whitespace and punctuation before exact matching. For open-ended questions, we use Claude Sonnet 4.5 as a fixed equivalence judge to determine whether the prediction is semantically equivalent to the ground-truth answer. We report per-benchmark accuracy and the average across the seven benchmarks. We additionally report *Gain/1K*, defined as the improvement in average accuracy over Base per 1,000 GRPO training examples:

$$\mathrm{Gain/1K} = \frac{\mathrm{AVG}(\theta) - \mathrm{AVG}(\theta_0)}{|\mathcal{S}|/1000}. \tag{9}$$

To assess uncertainty in pairwise comparisons, we perform paired bootstrap resampling within each benchmark for 10,000 repetitions. For each repetition, we recompute the difference in average accuracy across the seven benchmarks.

Table 1: Results on Qwen2.5-VL-7B-Instruct. The first block compares baselines at a fixed 5K budget. The second reports LIMR and ThinkLite-VL under their original budgets, larger-budget Random, online VADE, and Full-Data GRPO. PACE also uses 5K. Gain/1K denotes the average improvement over Base per 1K samples. Bold indicates the best result in each column.

Method	Samples	MathVista	MathVerse	WeMath	MathVision	MM-Math	DynaMath	PGPS9K	Avg.	Gain/1K
Base	–	69.8	47.1	55.2	24.8	35.2	56.2	50.5	48.4	–
Fixed 5K Baselines										
Random	5K	67.9	49.1	57.3	27.1	37.7	56.7	50.7	49.5	0.22
Pass Rate	5K	69.7	49.5	56.0	26.3	38.2	57.5	52.1	49.9	0.30
LIMR	5K	68.3	49.9	56.7	26.4	37.8	57.8	50.4	49.6	0.24
ThinkLite-VL	5K	69.4	49.6	57.0	27.2	**38.8**	56.8	49.4	49.7	0.27
Original, Larger, or Adaptive Budgets										
LIMR	8.4K	68.9	49.6	57.2	**27.6**	**38.8**	57.8	49.3	49.9	0.18
Random	20K	69.0	50.1	56.8	27.0	38.2	57.2	**53.6**	50.3	0.10
ThinkLite-VL	21K	70.1	50.2	56.2	26.1	37.2	57.1	51.4	49.8	0.06
VADE	Online	68.8	50.4	56.8	27.0	38.6	56.1	52.6	50.0	–
Full-Data GRPO	54.9K	70.0	**52.0**	57.5	27.3	38.3	**58.2**	52.0	50.7	0.04
PACE (Ours)	5K	**71.9**	51.3	**57.8**	26.8	**38.8**	57.2	**53.6**	**51.1**	**0.53**

6 Results and Analysis

6.1 RQ1: Effectiveness and Data Efficiency

Table 1 compares PACE with data-selection baselines and Full-Data GRPO on Qwen2.5-VL-7B-Instruct. The first block compares methods under the same 5K data budget, while the second block reports methods under their original, larger, or adaptive selection settings. **1. PACE achieves the best performance under the fixed 5K budget while retaining full-data performance.** PACE reaches an average accuracy of 51.1 using $5{,}000$ examples, compared with 50.7 for Full-Data GRPO trained on all $54{,}931$ examples. We therefore interpret PACE as retaining full-data performance with substantially less training data, rather than as strictly outperforming Full-Data GRPO. PACE also improves the Base model by 2.7 points and improves performance on all seven benchmarks. **2. PACE remains more data-efficient than larger- and adaptive-budget baselines.** We additionally evaluate LIMR and ThinkLite-VL under their original selection settings. LIMR selects 8.4K examples, while ThinkLite-VL selects approximately 21K examples. PACE outperforms LIMR by 1.2 points and also exceeds ThinkLite-VL despite using substantially fewer training examples. It further outperforms Random trained on 20K examples and the online VADE baseline. These results show that prompt-induced correctness changes provide a more data-efficient selection signal than the existing alternatives. Pairwise bootstrap confidence intervals and runtime comparisons are reported in Appendices B and C.

6.2 RQ2: Effectiveness Across Model Scales and Architectures

We examine whether PACE remains effective on a smaller model from the same family and on a model with a different architecture. **1. PACE retains its effectiveness at a smaller model scale.** As shown in Table 2, PACE achieves the highest average accuracy among all methods using a 5K training budget on Qwen2.5-VL-3B-Instruct. It improves the Base model by 1.6 points, reaching 41.8, outperforms Random, Pass Rate, and the online VADE baseline, and slightly exceeds Full-Data GRPO at 41.5. This result shows that the effectiveness and data efficiency of PACE extend from Qwen2.5-VL-7B to the smaller 3B model. **2. PACE transfers effectively across model architectures.** On InternVL3-2B, PACE improves the Base model from 35.2 to 36.7 and again outperforms Random and Pass Rate under the same 5K budget. It remains within 0.2 points of Full-Data GRPO despite using fewer than one-tenth of the training examples. Together with the Qwen2.5-VL results, this finding demonstrates that PACE generalizes across model scales and architectures while retaining strong data efficiency.

Table 2: Results across model scales and architectures. We compare fixed-5K baselines with online VADE and Full-Data GRPO. PACE uses 5K. Gain/1K denotes the average improvement over Base per 1K samples. Bold indicates the best result within each model block.

Method	Samples	MathVista	MathVerse	WeMath	MathVision	MM-Math	DynaMath	PGPS9K	Avg.	Gain/1K
Qwen2.5-VL-3B-Instruct										
Base	–	62.7	35.9	50.5	21.7	29.6	45.9	35.3	40.2	–
Random	5K	62.2	38.0	50.5	22.5	31.5	47.7	36.1	41.2	0.20
Pass Rate	5K	62.1	38.5	50.7	**23.3**	31.3	47.5	35.9	41.3	0.22
VADE	Online	61.5	39.0	50.2	22.6	31.6	**48.3**	35.7	41.3	–
Full-Data GRPO	54.9K	61.2	38.9	**51.3**	22.7	**31.8**	47.9	**36.6**	41.5	0.02
PACE (Ours)	5K	**63.5**	**39.9**	50.5	23.0	31.4	**48.3**	36.3	**41.8**	**0.32**
InternVL3-2B										
Base	–	50.7	31.8	43.3	19.6	22.5	39.1	39.3	35.2	–
Random	5K	55.5	32.8	43.6	20.1	22.3	40.6	39.2	36.3	0.22
Pass Rate	5K	53.6	32.8	43.1	20.0	23.6	**41.4**	**39.5**	36.3	0.22
VADE	Online	55.5	32.9	43.2	20.1	23.3	41.1	39.2	36.5	–
Full-Data GRPO	54.9K	**57.8**	**34.7**	42.2	20.1	**24.7**	39.9	38.9	**36.9**	0.03
PACE (Ours)	5K	56.2	33.6	**43.7**	**20.4**	23.3	40.3	**39.5**	36.7	**0.30**

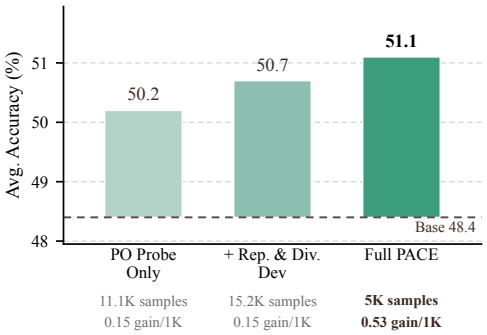


Figure 3: Ablation of PACE components on Qwen2.5-VL-7B-Instruct. Representative-and-diverse development-set construction improves selection, and helpfulness-guided balanced selection performs best at the 5K budget.

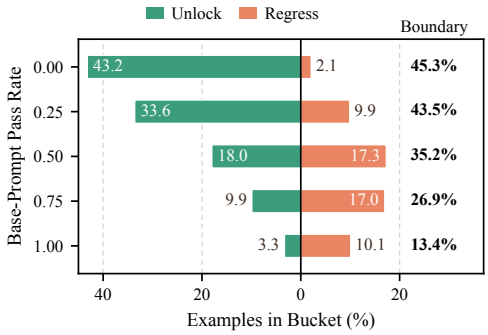


Figure 4: Relationship between base-prompt pass rate and the prompt-sensitive boundary. Prompt-sensitive examples span the pass-rate range, with unlocks dominating at lower rates and regressions at higher rates.

6.3 RQ3: Contribution of PACE Components

We examine the contributions of prompt optimization, representative-and-diverse development-set construction, and helpfulness-guided balanced selection. **1. Representative-and-diverse development-set construction improves the quality of the selected data.** As shown in Figure 3, replacing the random development set with representative-and-diverse construction increases average accuracy from 50.2 to 50.7. Although the resulting prompt-sensitive boundary grows from 11,139 to 15,230 examples, the improvement indicates that broader semantic coverage produces a more useful behavioral probe. **2. Helpfulness-guided balanced selection achieves the strongest performance and data efficiency.** Full PACE further improves average accuracy to 51.1 while reducing the training subset to the target budget of 5,000 examples, achieving the highest Gain/1K of 0.53. This shows that cluster-level helpfulness filtering and balanced selection concentrate the training budget on more useful semantic regions. Complete per-benchmark results are reported in Appendix D.

6.4 RQ4: Understanding the Prompt-Sensitive Boundary

We examine whether the prompt-sensitive boundary merely reflects conventional fixed-prompt signals and whether both transition directions provide useful signals for downstream GRPO training. **1. The prompt-sensitive boundary is only weakly related to conventional fixed-prompt sig-**

Table 3: Downstream GRPO performance using different portions of the prompt-sensitive boundary.

Training Data	Set Size	Avg.	Gain
Base	–	48.4	–
Unlock-only	11,835	49.8	+1.4
Regression-only	3,395	49.7	+1.3
Full boundary	15,230	**50.7**	**+2.3**

Table 4: Effect of prompt-sensitive candidate filtering across selection strategies on Qwen2.5-VL-7B-Instruct.

Selector	Full Set	PO-Filtered	Δ
k-Center	50.0	50.6	+0.6
Facility Location	49.6	49.8	+0.2
Entropy	50.1	50.3	+0.2
Random	49.5	50.3	+0.8
Mean	49.8	50.3	+0.5

nals. As shown in Figure 4, prompt sensitivity is more common among harder examples, with the boundary rate decreasing from 45.3% at pass rate 0 to 13.4% at pass rate 1. However, boundary membership is only weakly negatively correlated with pass rate ($\rho = -0.237$), and all eight alternative signals have $|\rho| < 0.25$. More importantly, subsets selected using these signals show limited agreement with the prompt-sensitive boundary: overlap fold ranges from 0.97 to 1.26, while Cohen's κ ranges from -0.018 to 0.146. Prompt-sensitive examples also remain among reliably solved examples, with 6.9% of the boundary coming from examples with pass rate 1. These results show that conventional fixed-prompt signals provide only limited information about prompt sensitivity and cannot substitute for directly probing correctness changes across prompts. Additional analyses are reported in Appendix E. **2. Both transition directions are useful, while the full boundary provides the strongest training signal.** As shown in Table 3, unlock-only and regression-only training achieve average accuracies of 49.8 and 49.7, respectively, both improving over the Base model at 48.4. This similarity is notable because the regression set is approximately $3.5\times$ smaller than the unlock set. However, neither direction alone matches training on the full boundary, which achieves an average accuracy of 50.7. These results show that regressions are not merely undesirable outcomes of prompt optimization and that the direction of an individual correctness transition alone does not determine its training utility. Instead, responsiveness to the prompt intervention provides useful information for data selection, while combining both transition directions yields the strongest downstream performance.

6.5 RQ5: Prompt-Sensitive Filtering Across Selection Strategies

We examine whether prompt-sensitive boundary remains useful independently of PACE's helpfulness-guided balanced selection. For each strategy, we compare selection from the full training set with selection from the prompt-sensitive boundary, while keeping the 5K training budget and GRPO configuration fixed. **Prompt-sensitive filtering improves all four selection strategies.** As shown in Table 4, restricting the candidate set to the prompt-sensitive boundary improves every tested strategy. The gains range from $+0.2$ to $+0.8$ points, with an average improvement of $+0.5$ points. The improvement holds for coverage-based, uncertainty-based, and random selection, indicating that prompt-induced correctness change can serve as a general candidate-filtering signal beyond PACE's specific selection procedure.

7 Conclusion

We introduced PACE, a data-selection framework that uses prompt optimization as a controlled behavioral probe for data-efficient multimodal GRPO. By comparing a frozen policy under the original and optimized prompts, PACE identifies prompt-sensitive examples and constructs a compact training subset using cluster-level helpfulness and balanced selection. Across seven multimodal mathematical reasoning benchmarks, PACE uses only 5K examples (9.1% of the full training set), outperforms fixed-budget baselines, and retains Full-Data GRPO performance on Qwen2.5-VL-7B while remaining effective across model scales and architectures. Further analyses show that the prompt-sensitive boundary captures information complementary to conventional fixed-prompt signals, that both unlocks and regressions provide useful training signals, and that prompt-sensitive filtering improves multiple selection strategies. These results establish prompt-induced correctness change as an effective signal for data-efficient multimodal GRPO.

AI USE STATEMENT

We made limited use of large language models—specifically ChatGPT and Claude—for language polishing, routine coding assistance, and experiment execution. All reported experimental results were obtained through actual program execution and independently verified by the authors. LLMs were not used to fabricate results or independently formulate research claims. All AI-assisted text, code, and experimental outputs were reviewed by the authors, who remain responsible for the content.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alex Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. Gepa: Reflective prompt evolution can outperform reinforcement learning. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 8479–8565, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/0e9e708b6f48e14fd0ac29e167413f76-Paper-Conference.pdf.

Sanghwan Bae, Jiwoo Hong, Min Young Lee, Hanbyul Kim, JeongYeon Nam, and Donghyun Kwak. Online difficulty filtering for reasoning oriented reinforcement learning. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 700–719, 2026.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025. URL https://arxiv.org/abs/2502.13923.

Alexander Bukharin, Shiyang Li, Zhengyang Wang, Jingfeng Yang, Bing Yin, Xian Li, Chao Zhang, Tuo Zhao, and Haoming Jiang. Data diversity matters for robust instruction tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 3411–3425, 2024.

Anwoy Chatterjee, HSVNS 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.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, and Hongxia Jin. Alpagasus: Training a better alpaca with fewer data. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 34767–34797, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/9543942c237ded1b39b1fd37259ff88e-Paper-Conference.pdf.

Qirun Dai, Dylan Zhang, Jiaqi W. Ma, and Hao Peng. Improving influence-based instruction tuning data selection for balanced learning of diverse capabilities. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 7079–7102, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.373. URL https://aclanthology.org/2025.findings-emnlp.373/.

Zengjie Hu, Jiantao Qiu, Tianyi Bai, Haojin Yang, Binhang Yuan, Qi Jing, Conghui He, and Wentao Zhang. Vade: Variance-aware dynamic sampling via online sample-level difficulty estimation for multimodal rl, 2025. URL https://arxiv.org/abs/2511.18902.

Mohamed Insaf Ismithdeen, Muhammad Uzair Khattak, and Salman Khan. Promptception: How sensitive are large multimodal models to prompts? In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational*

Linguistics: EMNLP 2025, pp. 23950–23985, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.1302. URL https://aclanthology.org/2025.findings-emnlp.1302/.

Shenshen Li, Xing Xu, Kaiyuan Deng, Lei Wang, Heng Tao Shen, and Fumin Shen. Truth in the few: High-value data selection for efficient multi-modal reasoning, 2026a. URL https://arxiv.org/abs/2506.04755.

Shipeng Li, Zhiqin Yang, Shikun Li, Xiaobo Xia, Hengyu Liu, Xinghua Zhang, Gaode Chen, Dong Fang, Ying Tai, and Zhe Peng. LearnAlign: Data selection for LLM reinforcement learning with improved gradient alignment. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 40417–40434, San Diego, California, United States, July 2026b. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.2009. URL https://aclanthology.org/2026.findings-acl.2009/.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. Limr: Less is more for rl scaling, 2025. URL https://arxiv.org/abs/2502.11886.

Liangxin Liu, Xuebo Liu, Derek F Wong, Dongfang Li, Ziyi Wang, Baotian Hu, and Min Zhang. Selectit: Selective instruction tuning for llms via uncertainty-aware self-reflection. *Advances in Neural Information Processing Systems*, 37:97800–97825, 2024a.

Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 22353–22373, 2024b. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/6091f2bb355e960600f62566ac0e2862-Paper-Conference.pdf.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 23439–23554, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/663bce02a0050c4a11f1eb8a7f1429d3-Paper-Conference.pdf.

Fanqing Meng, Lingxiao Du, Zongkai Liu, Zhixiang Zhou, Quanfeng Lu, Daocheng Fu, Tiancheng Han, Botian Shi, Wenhai Wang, Junjun He, Kaipeng Zhang, Ping Luo, Yu Qiao, Qiaosheng Zhang, and Wenqi Shao. Mm-eureka: Exploring the frontiers of multimodal reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2503.07365*, 2025.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 7957–7968, 2023.

Runqi Qiao, Qiuna Tan, Guanting Dong, Minhui Wu, Chong Sun, Xiaoshuai Song, Jiapeng Wang, Zhuoma GongQue, Shanglin Lei, YiFan Zhang, Zhe Wei, Miaoxuan Zhang, Runfeng Qiao, Xiao Zong, Yida Xu, Peiqing Yang, Zhimin Bao, Muxi Diao, Chen Li, and Honggang Zhang. We-math: Does your large multimodal model achieve human-like mathematical reasoning? In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 20023–20070, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.983. URL https://aclanthology.org/2025.acl-long.983/.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.

Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *proceedings of the 2008 conference on empirical methods in natural language processing*, pp. 1070–1079, 2008.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Kai Sun, Yushi Bai, Ji Qi, Lei Hou, and Juanzi Li. Mm-math: Advancing multimodal math evaluation with process evaluation and fine-grained classification. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 1358–1375, 2024.

Ke Wang, Junting Pan, Weikang Shi, Zimu Lu, Houxing Ren, Aojun Zhou, Mingjie Zhan, and Hongsheng Li. Measuring multimodal mathematical reasoning with math-vision dataset. *Advances in Neural Information Processing Systems*, 37:95095–95169, 2024.

Xiyao Wang, Zhengyuan Yang, Chao Feng, Hongjin Lu, Linjie Li, Chung-Ching Lin, Kevin Lin, Furong Huang, and Lijuan Wang. Sota with less: Mcts-guided sample selection for data-efficient visual reasoning self-improvement, 2025. URL https://arxiv.org/abs/2504.07934.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Kai Wei, Rishabh Iyer, and Jeff Bilmes. Submodularity in data subset selection and active learning. In *International conference on machine learning*, pp. 1954–1963. PMLR, 2015.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. Less: Selecting influential data for targeted instruction tuning. *arXiv preprint arXiv:2402.04333*, 2024.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 12028–12068, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/3339f19c5fcee3ad74502947a32be9e6-Paper-Conference.pdf.

Ningyuan Yang, Weihua Du, Weiwei Sun, Sean Welleck, and Yiming Yang. Gradalign: Gradient-aligned data selection for llm reinforcement learning. *arXiv preprint arXiv:2602.21492*, 2026.

Ming-Liang Zhang, Fei yin, and Cheng-Lin Liu. A multi-modal neural geometric solver with textual clauses parsed from diagram. In Edith Elkind (ed.), *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pp. 3374–3382. International Joint Conferences on Artificial Intelligence Organization, 8 2023. doi: 10.24963/ijcai.2023/376. URL https://doi.org/10.24963/ijcai.2023/376. Main Track.

Renrui Zhang, Dongzhi Jiang, Yichi Zhang, Haokun Lin, Ziyu Guo, Pengshuo Qiu, Aojun Zhou, Pan Lu, Kai-Wei Chang, Yu Qiao, et al. Mathverse: Does your multi-modal llm truly see the diagrams in visual math problems? In *European Conference on Computer Vision*, pp. 169–186. Springer, 2024.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, Zhangwei Gao, Erfei Cui, Xuehui Wang, Yue Cao, Yangzhou Liu, Xingguang Wei, Hongjie Zhang, Haomin Wang, Weiye Xu, Hao Li, Jiahao Wang, Nianchen Deng, Songze Li, Yinan He, Tan Jiang, Jiapeng Luo, Yi Wang, Conghui He, Botian Shi, Xingcheng Zhang, Wenqi Shao, Junjun He, Yingtong Xiong, Wenwen Qu, Peng Sun, Penglong Jiao, Han Lv, Lijun Wu, Kaipeng Zhang, Huipeng Deng, Jiaye Ge, Kai Chen, Limin Wang, Min Dou, Lewei Lu, Xizhou Zhu, Tong Lu, Dahua Lin, Yu Qiao, Jifeng Dai, and Wenhai Wang. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models, 2025. URL https://arxiv.org/abs/2504.10479.

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.

Chengke Zou, Xingang Guo, Rui Yang, Junyu Zhang, Bin Hu, and Huan Zhang. Dynamath: A dynamic visual benchmark for evaluating mathematical reasoning robustness of vision language models. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.),

International Conference on Learning Representations, volume 2025, pp. 48337–48383, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/78b248ea6f627431bba5029d92be8a3d-Paper-Conference.pdf.

A Implementation Details

PACE Configuration. We instantiate the prompt optimizer in PACE with GEPA using a 150-example development set constructed from the training dataset with $K = 15$ semantic clusters. For diversity augmentation, we compute the mean distance to the 10 nearest neighbors and retain the $Q = 5{,}000$ examples with the largest values as the candidate set. The optimized prompt is used only for offline probing and data selection; all subsequent GRPO training and evaluation use the original prompt. For all PACE probing runs, we use greedy decoding to ensure deterministic generation. Consequently, correctness transitions between P_0 and $P^\star$ are induced by changes in the prompting condition rather than stochastic decoding noise.

GRPO Training. We train all methods using EasyR1 with the AdamW optimizer and a learning rate of 1.0×10^{-6}. We use a rollout batch size of 512 prompts, sample four rollouts per prompt, and set the global update batch size to 128. All data-selection methods are trained for 60 optimization steps, while Full-Data GRPO is trained for one epoch over the complete dataset, corresponding to approximately 107 steps. All controlled comparisons use the same initial checkpoint, training prompt, reward functions, and GRPO configuration. Experiments are conducted on eight NVIDIA A100-40GB GPUs.

B Pairwise Statistical Analysis

We assess uncertainty in the main pairwise comparisons on Qwen2.5-VL-7B-Instruct. For confidence intervals, we perform 10,000 paired bootstrap resamples within each benchmark. At each repetition, we resample evaluation examples with replacement, compute the accuracy difference between PACE and the comparison method for each benchmark, and then take the equal-weight average across the seven benchmarks. We report percentile-based 95% confidence intervals.

Table 5: Pairwise statistical comparisons of PACE on Qwen2.5-VL-7B-Instruct. Observed differences are macro-average accuracy differences computed from unrounded results. Confidence intervals are obtained from 10,000 benchmark-stratified paired bootstrap resamples with seed 42.

Comparison	Difference	95% CI
PACE vs. Random 5K	$+1.58$	$[+0.82, +2.10]$
PACE vs. Pass Rate 5K	$+1.16$	$[+0.56, +1.82]$
PACE vs. LIMR 5K	$+1.46$	$[+0.80, +2.10]$
PACE vs. ThinkLite-VL 5K	$+1.33$	$[+0.64, +2.01]$
PACE vs. Random 20K	$+0.77$	$[+0.29, +1.35]$
PACE vs. ThinkLite-VL 21K	$+1.30$	$[+0.69, +1.91]$
PACE vs. Full-Data GRPO	$+0.34$	$[-0.21, +0.98]$

PACE shows positive differences over all competing data-selection methods, with 95% confidence intervals excluding zero. In contrast, the comparison with Full-Data GRPO yields a small observed difference of $+0.34$ points with a confidence interval spanning zero. This result supports our interpretation that PACE retains comparable full-data performance while using substantially fewer training examples.

C Runtime Analysis

End-to-end runtime. On Qwen2.5-VL-7B-Instruct, PACE completes data selection and GRPO training in approximately 10 hours, compared with 76 hours for LIMR, 74 hours for ThinkLite-VL, and 14 hours for Full-Data GRPO.

D Detailed Ablation Results

Table 6 reports the complete per-benchmark results for the PACE ablation. The variants are cumulative: the first variant uses only prompt-induced answer changes obtained from a randomly con-

structed development set, the second replaces the random development set with the representative-and-diverse construction, and the final variant additionally applies cluster-level helpfulness and balanced selection.

Table 6: Complete per-benchmark ablation results on Qwen2.5-VL-7B-Instruct under the unified evaluation protocol. All benchmark accuracies and averages are reported to one decimal place. Gain/1K is computed using unrounded results.

Variant	MathVista	MathVerse	WeMath	MathVision	MM-Math	DynaMath	PGPS9K	Avg.	Gain/1K
Base	69.8	47.1	55.2	24.8	35.2	56.2	50.5	48.4	–
PO probe only	68.8	51.2	57.4	26.3	38.0	56.9	52.4	50.2	0.15
+ Rep.-div. dev	70.7	51.4	**58.7**	**27.0**	37.2	**57.8**	51.9	50.7	0.15
Full PACE	**71.9**	**51.3**	57.8	26.8	**38.8**	57.2	**53.6**	**51.1**	**0.53**

E Additional Analysis of Signals

We extend the pass-rate analysis in Section 6.4 to a broader set of statistics that do not use the optimized prompt P^*. In addition to pass rate, we consider self-consistency, output entropy, logit spread, mean log-probability, confidence, log perplexity, and attention dispersion.

How each statistic is computed and used for selection. For each example, we obtain $R = 4$ rollouts under the base prompt at temperature 1.0 and top-$p = 1.0$, matching the rollout group size used in training. For token-level statistics, we approximate each next-token distribution using the top-20 log-probabilities and renormalize it before computing the corresponding statistic. To compare each statistic with the PACE boundary, we construct a size-matched subset using the direction corresponding to greater difficulty or uncertainty.

- **Pass rate**: the fraction of the R rollouts whose extracted answer is judged correct, taking one of five values $\{0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\}$. *Selection: closest to* 0.5, prioritizing examples with mixed success across rollouts.
- **Self-consistency**: the size of the largest group of rollouts sharing the same extracted answer, divided by R; unparsable rollouts are excluded from the vote. *Selection: lowest*, prioritizing examples for which sampled answers disagree most.
- **Output entropy**: the Shannon entropy of the next-token distribution, averaged over generated tokens and then over rollouts. *Selection: highest*, prioritizing examples with greater token-level uncertainty.
- **Logit spread**: the difference between the log-probabilities of the most and second-most likely tokens, averaged over generated tokens and rollouts. *Selection: lowest*, prioritizing examples with smaller margins between competing token predictions.
- **Mean log-probability**: the total log-probability of each rollout divided by its token count, averaged over rollouts. *Selection: lowest*, prioritizing generations to which the model assigns lower likelihood.
- **Confidence**: the model's self-reported confidence on a 1–5 scale, obtained by requesting a confidence rating and greedily decoding the response. *Selection: lowest*, prioritizing examples the model rates as least confident.
- **Log perplexity**: the mean token-level negative log-likelihood of the reference answer under teacher forcing, scoring only the final-answer portion. *Selection: highest*, prioritizing examples for which the reference answer receives lower model likelihood.
- **Attention dispersion**: the entropy of the last layer's self-attention distribution at the final prefix position, averaged over attention heads. *Selection: highest*, prioritizing examples with more dispersed attention over the input context.

For each statistic, we select the same number of examples as in the PACE boundary, restricted to examples for which that statistic is available. The PACE boundary consists of examples whose correctness changes between the base prompt and P^*. We then compare the selected subsets to test whether conventional signals can recover the prompt-sensitive examples identified by PACE.

Correlation with boundary membership. We compute the Spearman rank correlation between each statistic and binary boundary membership. As shown in Table 7, all correlations are weak, with absolute values below 0.25. Pass rate has the strongest association with boundary membership ($\rho = -0.237$). Among the remaining statistics, log perplexity has $\rho = +0.198$, while all others have $|\rho| \leq 0.102$. These results show that no individual difficulty, consistency, uncertainty, or confidence statistic provides a close surrogate for the PACE boundary. Prompt-induced correctness transitions therefore capture information that is not well represented by statistics obtained from a single elicitation condition.

Table 7: Spearman rank correlation between the PACE prompt-sensitive boundary and alternative statistics that do not use the optimized prompt P^*.

Signal	Spearman ρ
Pass rate	-0.237
Self-consistency	-0.102
Output entropy	$+0.026$
Logit spread	-0.001
Mean log-probability	-0.026
Confidence	$+0.011$
Log perplexity	$+0.198$
Attention dispersion	-0.011

Agreement between selected subsets. Correlation alone does not directly measure whether two signals identify the same training examples. Using the selection rules defined above, we therefore construct a size-matched subset for each statistic and compare its membership with the PACE prompt-sensitive boundary. We measure subset agreement using overlap fold and Cohen's κ. Overlap fold normalizes the observed intersection by the intersection expected from random subsets of the same size, so a value of 1 indicates chance-level overlap. Cohen's κ measures chance-adjusted membership agreement, where 0 indicates chance-level agreement.

As shown in Figures 5 and 6, the PACE boundary has limited agreement with every alternative signal. Its overlap fold ranges from 0.97 to 1.26, while Cohen's κ ranges from -0.018 to 0.146. In contrast, several alternative signals strongly agree with one another. For example, output entropy and mean log-probability reach an overlap fold of 2.49 and $\kappa = 0.84$.

Together with the correlation results, these findings show that several alternative signals identify similar examples to one another, whereas none closely recovers the prompt-sensitive examples identified by PACE.

E.1 Pass-Rate-Stratified Transition Counts

Table 8 reports the numbers of unlocks and regressions across base-prompt pass-rate bins. Since pass rate is estimated from four rollouts, it takes one of five values: 0, 0.25, 0.50, 0.75, or 1. The composition of the prompt-sensitive boundary changes systematically with base-prompt pass rate. At pass rate 0, unlocks dominate, accounting for 95.0% of boundary examples. As pass rate increases, the fraction of regressions grows steadily. At pass rate 1, the pattern reverses: regressions account for 74.9% of boundary examples, while unlocks account for 25.1%. This pattern shows that the two transition directions capture different regions of model behavior. Low-pass-rate boundary examples are dominated by cases whose correctness improves under the optimized prompt, whereas high-pass-rate boundary examples are dominated by cases whose correctness is disrupted by the prompt change.

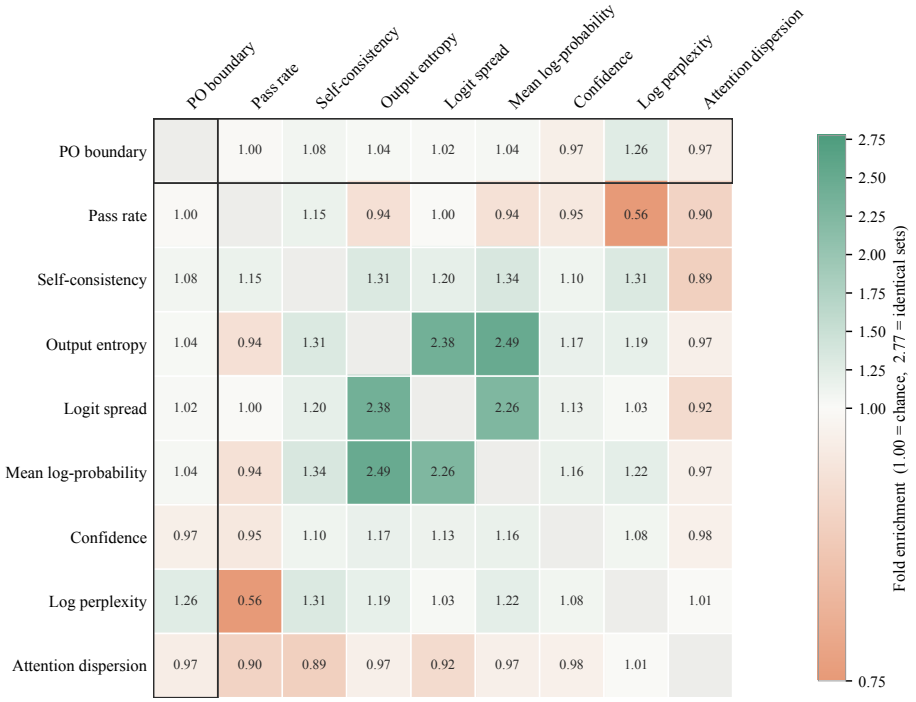


Figure 5: Pairwise overlap between the PACE prompt-sensitive boundary and size-matched subsets selected using alternative signals, normalized by the overlap expected from random selection. A value of 1 indicates random-level overlap. The PACE boundary shows limited overlap with all alternative signals, while several alternative signals strongly overlap with one another.

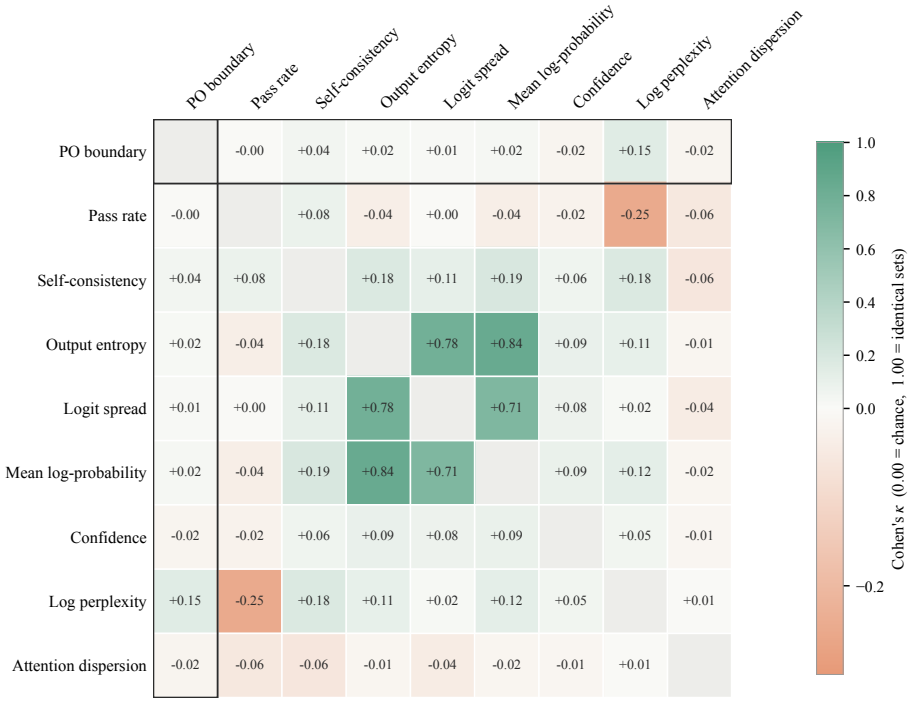


Figure 6: Pairwise agreement between the PACE prompt-sensitive boundary and size-matched subsets selected using alternative signals, measured by Cohen's κ. A value of 0 indicates chance-level membership agreement. The PACE boundary shows low agreement with all alternative signals, while several alternative signals identify highly similar subsets.

Table 8: Pass-rate-stratified counts and composition of prompt-sensitive transitions.

Pass rate	Unlock	Regress	Unlock within boundary	Regress within boundary
0	8,295	439	95.0%	5.0%
0.25	2,026	605	77.0%	23.0%
0.50	771	742	51.0%	49.0%
0.75	476	813	36.9%	63.1%
1	267	796	25.1%	74.9%
Total	**11,835**	**3,395**	–	–

F SELECTION-BUDGET SENSITIVITY

We examine the sensitivity of PACE to the selection budget B on Qwen2.5-VL-7B-Instruct. The PACE variants use the same data-selection pipeline and GRPO configuration, differing only in the number of selected training examples. Full-Data GRPO is included as a high-data reference.

Table 9: Effect of the selection budget B on Qwen2.5-VL-7B-Instruct. Average accuracy is computed across the seven evaluation benchmarks. Full-Data GRPO uses all $54{,}931$ training examples.

Training data	Avg.
PACE ($B = 2.5$K)	49.8
PACE ($B = 5$K)	**51.1**
PACE ($B = 10$K)	50.6
PACE ($B = 15$K)	50.8
Full-Data GRPO (54.9K)	50.7

As shown in Table 9, PACE performs best with a 5K selection budget, reaching an average accuracy of 51.1. Reducing the budget to 2.5K lowers the average accuracy to 49.8, indicating that an overly small subset may not provide sufficient training coverage. Increasing the budget to 10K or 15K does not yield further improvements, with average accuracies of 50.6 and 50.8, respectively. These results suggest that performance does not improve monotonically with the number of selected examples. Within the evaluated range, the 5K budget provides the strongest performance while using only approximately 9.1% of the full training pool.

G PACE ALGORITHM

Algorithm 1 PACE: Prompt-Induced Answer Change Estimation

Require: Training set $\mathcal{D}$; initial policy f_{θ_0}; original prompt P_0; development budget M; selection budget B; number of clusters K; neighborhood size L; candidate-pool size Q

Ensure: Selected subset $\mathcal{S}_{\mathrm{PACE}}$

Compute the normalized question embedding $\phi(q_i)$ for each $z_i \in \mathcal{D}$

Apply K-means to partition $\mathcal{D}$ into clusters $\{\mathcal{C}_1, \ldots, \mathcal{C}_K\}$ with centroids $\{\mu_1, \ldots, \mu_K\}$

$M_{\mathrm{rep}} \leftarrow \lfloor M/2 \rfloor$

$M_{\mathrm{div}} \leftarrow M - M_{\mathrm{rep}}$

Construct $\mathcal{D}_{\mathrm{rep}}$ using cluster-proportional quotas and examples closest to each cluster centroid

Construct $\mathcal{D}_{\mathrm{cand}}$ by retaining the Q examples in $\mathcal{D} \setminus \mathcal{D}_{\mathrm{rep}}$ with the largest mean distances to their L nearest neighbors

Construct $\mathcal{D}_{\mathrm{div}}$ from $\mathcal{D}_{\mathrm{cand}}$ using farthest-point selection initialized with the farthest pair, until M_{div} examples are selected

$\mathcal{D}_{\mathrm{dev}} \leftarrow \mathcal{D}_{\mathrm{rep}} \cup \mathcal{D}_{\mathrm{div}}$

$P^\star \leftarrow \mathcal{A}_{\mathrm{PO}}(P_0; \theta_0, \mathcal{D}_{\mathrm{dev}})$

for all $z_i \in \mathcal{D}$ **do**

    $\Delta(z_i) \leftarrow c_{P^\star}(z_i) - c_{P_0}(z_i)$

end for

$\mathcal{B} \leftarrow \{z_i \in \mathcal{D} : |\Delta(z_i)| = 1\}$

for $k = 1, \ldots, K$ **do**

    $H_k \leftarrow \dfrac{1}{|\mathcal{C}_k|} \sum_{z_i \in \mathcal{C}_k} \Delta(z_i)$

    $\mathcal{B}_k \leftarrow \mathcal{B} \cap \mathcal{C}_k$

end for

$\mathcal{K}_{\mathrm{ret}} \leftarrow \{k : H_k > 0\}$

$k_{\max} \leftarrow \arg\max_{k \in \mathcal{K}_{\mathrm{ret}}} H_k$

$q_{k_{\max}}^{\mathrm{tar}} \leftarrow \lfloor B/2 \rfloor$

Divide the remaining half of the budget as evenly as possible across $\mathcal{K}_{\mathrm{ret}} \setminus \{k_{\max}\}$ and assign any leftover slots randomly

Form $\mathcal{S}_{\mathrm{PACE}}$ by selecting from each $\mathcal{B}_k$, for $k \in \mathcal{K}_{\mathrm{ret}}$, the examples closest to centroid μ_k according to their target quotas

return $\mathcal{S}_{\mathrm{PACE}}$