VAPOR: AUTOMATIC PROMPT OPTIMIZATION FOR VISION-LANGUAGE MODELS VIA VISUAL REFLECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Automatic prompt optimization (APO) has emerged as a gradient-free alternative to fine-tuning for adapting vision-language models (VLMs). However, existing APO methods inherit a modality mismatch: they optimize visual decision-making through language-only feedback. This mismatch is especially limiting in few-shot settings with fuzzy visual boundaries, where failures often arise not from missing task instructions but from misidentified visual evidence. We propose **VAPOR**, an automatic prompt optimization framework that closes this feedback gap through visual reflection. VAPOR compares per-example reasoning trajectories between erroneous and corrected predictions, distills their differences into atomic visual decision rules, and records negative constraints that suppress recurring spurious rationales. We treat model-generated rationales as candidate diagnostic signals rather than faithful explanations, and admit a rule into the optimized prompt only after behavioral verification. Rather than appending all discovered rules into a monolithic prompt, VAPOR builds a hierarchical expert prompt system over dual visual-semantic clusters, routing each input to specialized guidance matched to its visual regime. On three multimodal benchmarks with ambiguous visual decision boundaries, VAPOR achieves up to 12% absolute gains over established APO baselines. Our analysis shows that visual reflection produces more compact, stable, and interpretable prompts, reframing APO for VLMs as a problem of grounding language search in visual evidence.

1 INTRODUCTION

Visual-language models (VLMs) have shown strong open-domain multimodal reasoning capabilities (Alayrac et al., 2022; Li et al., 2023a; Liu et al., 2023). Yet adapting them to specialized downstream tasks remains difficult when labels are scarce and decision boundaries are visually ambiguous. In fine-grained aesthetic assessment or safety-sensitive image tagging, correct predictions often depend on subtle visual evidence (Murray et al., 2012), subjective criteria, and context-dependent reasoning rather than clearly separable visual categories. Parameter-efficient fine-tuning can overfit in such few-shot regimes (Han et al., 2024), while manual prompt engineering is brittle because the relevant visual criteria are difficult to specify in advance. Automatic prompt optimization (APO) offers a lightweight alternative by searching for task-specific prompts without updating model parameters (Zhou et al., 2022; Yang et al., 2023; Li et al., 2025), but existing APO methods are largely inherited from text-only language-model settings (Yuksekgonul et al., 2024; Du et al., 2024). When applied to VLMs, they face a modality mismatch: the optimizer edits language prompts, but the errors it tries to fix are often caused by visual misinterpretation. Textual critiques or scalar feedback can indicate that a prediction is wrong, but they do not reveal whether the model attended to the wrong region, relied on a spurious visual cue, or applied an inappropriate semantic criterion (Li et al., 2023b). We argue that APO for VLMs is therefore not merely a prompt search problem, but a problem of *visually grounded credit assignment*.

Naively adding multimodal feedback does not resolve this bottleneck. Visual content is dense and heterogeneous, so turning every observed cue into an instruction easily produces bloated prompts that overfit the optimization set. Model-generated rationales can also be unstable: without contrasting failed and corrected reasoning, reflection tends to produce generic edits rather than reusable visual rules. Finally, fuzzy-boundary tasks rarely admit a single universal prompt; accumulating all edits into one global instruction can create semantic conflicts and performance oscillation across

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.

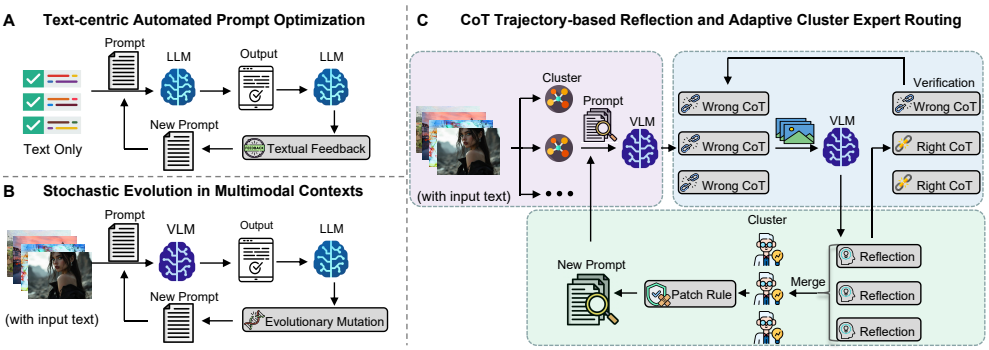

Figure 1: Comparison of APO paradigms. **(A)** Text-centric APO relies on language-only feedback and lacks visual credit assignment. **(B)** Stochastic multimodal APO introduces visual inputs but still depends on weakly grounded mutations. **(C)** VAPOR performs visual reflection over corrected reasoning trajectories, extracts atomic visual rules, and routes inputs to adaptive expert prompts.

visually different samples. These failure modes suggest that VLM prompt optimization requires a feedback mechanism that can isolate which visual reasoning patterns should be corrected, retained, or specialized to different input regimes.

Motivated by these failure modes, we propose **VAPOR**, a visually grounded automatic prompt optimization framework for VLMs. Instead of treating failures as text-only prompt errors, VAPOR contrasts per-sample reasoning trajectories before and after a prediction is corrected and uses the contrast to propose atomic visual decision rules. Candidate rules are executed, checked against labeled behavior, and rejected or revised when they introduce regressions. The accepted rules specify which visual cues should be emphasized, which spurious rationales should be suppressed, and which semantic criteria should govern ambiguous cases. To prevent these rules from collapsing into a monolithic and over-constrained prompt, VAPOR organizes them into a hierarchical expert prompt system through dual-space visual-semantic clustering and confidence-based routing. This enables different visual regimes to activate specialized guidance while preserving fallback prompts for uncertain or out-of-distribution inputs.

In this paper, we evaluate VAPOR on two fuzzy-boundary multimodal benchmarks: ArtiMuse-10K (Cao et al., 2025), which emphasizes subjective aesthetic perception, and SafeTag-VL-3K (Rong et al., 2025a;b), which emphasizes safety-oriented visual-semantic alignment. We further evaluate on MMMU (Yue et al., 2024), a cross-domain benchmark spanning 30 academic subjects, to test whether VAPOR's visual decision rules generalize beyond a single fuzzy-boundary task. The empirical results support our central claim that visually grounded credit assignment is critical for APO in VLMs. Across four representative VLM backbones, i.e., Qwen3-VL-32B, Llama-3.2-11B-Vision, and Gemma-4-31B-it, VAPOR consistently outperforms established APO baselines including IPO (Du et al., 2024), TextGrad (Yuksekgonul et al., 2024), and ERM (Yan et al., 2025). The gains are especially pronounced on ArtiMuse-10K, where subjective visual criteria make text-only optimization brittle, while VAPOR remains stable on SafeTag-VL-3K even when zero-shot performance is already strong. We further evaluate on GPT-5.6 Terra, recover the original continuous ArtiMuse scores instead of only thresholded labels, measure end-to-end inference throughput, and inspect counterfactual cue removal and verified correction traces. These results show that VAPOR does not merely exploit model-specific prompting idiosyncrasies; rather, it learns transferable visual decision rules that generalize across architectures, optimization-set sizes, and task regimes.

Contributions. Our contributions are threefold. First, we identify *visually grounded credit assignment* as a core bottleneck in applying APO to VLMs, explaining why language-only feedback leads to blind and unstable optimization on fuzzy-boundary multimodal tasks. Second, we introduce VAPOR, a visual-reflection APO framework that treats CoT as a fallible proposal mechanism and admits atomic rules only after behavioral verification, while negative memory suppresses recurring failed corrections. Third, we develop a hierarchical expert prompt system with visual-semantic clustering and confidence-based routing, and validate it extensively, including its behavior on continuous scores, closed-source transfer, and the cross-domain generalization.

2 Related Work

Automatic Prompt Optimization. Discrete prompt optimization has been approached through evolutionary search, gradient-guided feedback, and structured reflection. EvoPrompt (Guo et al., 2023) combines LLMs with evolutionary algorithms to iteratively generate and refine prompts, with GAAPO (Sécheresse et al., 2025) extending this via multiple specialist generation strategies within a similar framework. TextGrad (Yuksekgonul et al., 2024) instead derives gradient-like error signals from LLM feedback, offering a more directed alternative to stochastic search. ERM (Yan et al., 2025) maintains a global feedback memory of high-scoring historical reflections, combined with an exemplar factory for few-shot concatenation. While effective when corrections are universally valid, its global accumulation strategy is ill-suited to heterogeneous visual distributions, where rules valid for one subgroup often conflict with another, causing instruction bloat. VAPOR instead replaces this by replacing global rule accumulation with a hierarchical expert space that localizes rule extraction to visually and semantically coherent subgroups.

Multimodal Prompt Optimization. Extending APO to vision-language models introduces a fundamental challenge: text-only feedback mechanisms are agnostic to the visual content that drives model predictions, resulting in blind optimization. Early work uses LLMs as black-box optimizers for VLMs (Liu et al., 2024), while more recent efforts enhance internal reasoning via visual reflection mechanisms targeting model training or architecture (Jian et al., 2025). ProAPO (Qu et al., 2025) mitigates visual blind spots through progressive evolution but remains limited to objective category boundaries in fine-grained classification. IPO (Du et al., 2024) incorporates image-text descriptions for richer cross-modal context. UniAPO (Zhu et al., 2026) unifies multimodal prompt optimization under an EM-like algorithm addressing context capacity constraints, yet its global clustering approach remains vulnerable to inter-rule conflicts. VAPOR instead grounds optimization in per-sample CoT trajectory evolution and structures the prompt space through dual-level visual-semantic clustering, enabling fine-grained, conflict-free rule extraction in visually ambiguous few-shot settings.

3 Preliminaries

Task Formulation. Let $f(\cdot\,; P)$ denote a frozen VLM prompted by instruction P, and let $\mathcal{D} = \{(v_i, y_i)\}_{i=1}^{N}$ be a small labeled dataset where v_i denotes the input image and y_i the corresponding ground-truth label. The goal of few-shot prompt optimization is to identify an optimal prompt P^* that maximizes task performance without updating model parameters:

$$P^* = \arg\max_{P \in \mathcal{P}} \; \frac{1}{N} \sum_{i=1}^{N} \mathcal{M}(f(v_i; P),\, y_i)\,, \tag{1}$$

where $\mathcal{P}$ denotes the discrete space of natural language prompts and $\mathcal{M}(\cdot, \cdot)$ is a task-specific evaluation metric. The optimization is carried out over a small training split $\mathcal{D}_{\text{train}}$, with generalization assessed on a held-out validation set $\mathcal{D}_{\text{val}}$.

Automatic Prompt Optimization. Automatic prompt optimization (APO) seeks to solve the above objective without gradient access to f(Pryzant et al., 2023). Given an initial prompt P_0, APO iteratively refines the prompt through a feedback mechanism:

$$P_{t+1} = \mathcal{U}(P_t,\; \mathcal{E}(f, P_t, \mathcal{D}_{\text{train}}))\,, \tag{2}$$

where $\mathcal{E}(\cdot)$ collects prediction errors or textual feedback from the current prompt, and $\mathcal{U}(\cdot)$ applies an update operator to produce a refined prompt. Existing methods such as TextGrad (Yuksekgonul et al., 2024) and IPO (Du et al., 2024) instantiate $\mathcal{U}$ through LLM-generated textual feedback, operating entirely in the linguistic modality. As a result, the feedback signal $\mathcal{E}$ remains agnostic to the visual content of v_i, limiting the quality of discovered prompts on tasks where prediction errors are rooted in visual ambiguity.

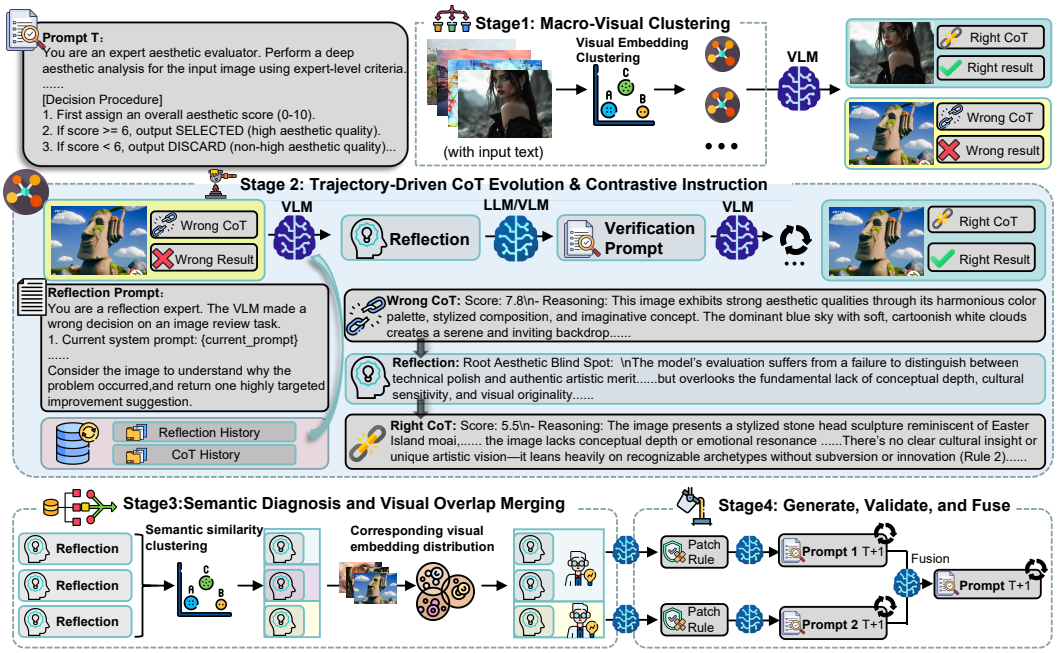

Figure 2: Overview of the VAPOR framework. Stage 1 groups optimization samples into coarse visual regimes. Stage 2 performs trajectory-driven visual reflection by contrasting wrong and corrected CoT traces. Stage 3 consolidates reflection rules into semantic experts and maps them back to visual supports. Stage 4 validates, fuses, and routes expert prompts for inference.

4 METHODOLOGY

4.1 FRAMEWORK OVERVIEW

VAPOR is a visually grounded automatic prompt optimization framework for VLMs. Given a small optimization set $\mathcal{D} = \{(v_i, q_i, y_i)\}_{i=1}^{n}$, where v_i is an image, q_i is the task instruction or input text, and y_i is the target label, VAPOR constructs a hierarchy of expert prompts without updating model parameters. The central design principle is to treat prompt optimization as *visual credit assignment*: when a VLM fails, the optimizer should identify which visual cue, semantic criterion, or reasoning pattern caused the error, and decide whether the correction should remain local or be promoted to a broader prompt.

Figure 2 and Algorithm 1 summarize the pipeline. VAPOR first partitions the optimization set into coarse visual regimes, then performs trajectory-driven reflection on hard examples by contrasting failed and corrected reasoning traces. The resulting atomic rules are consolidated into semantic experts, grounded back to visual supports for routing, and finally validated and fused into a hierarchical prompt library. This stage-wise design avoids three common APO failure modes: visually ungrounded feedback, instruction bloat, and interference from a single over-constrained global prompt.

4.2 STAGE 1: MACRO-VISUAL CLUSTERING

VAPOR begins by separating the optimization set into coarse visual regimes before any prompt rewriting is performed. This is not a clustering-for-classification step; rather, it is a mechanism for preventing early rule contamination. If all samples are optimized together, a correction discovered for one visual regime may become a harmful constraint for another. For each image v_i, we extract a frozen visual embedding

$$\mathbf{z}_i = \Phi_{\text{vis}}(v_i), \tag{3}$$

and cluster the embeddings into macro regimes $\{\mathcal{C}^{(k)}\}_{k=1}^{K}$. Each cluster inherits the initial prompt P_0 and serves as a local workspace for subsequent reflection. This visual anchoring gives VAPOR a coarse notion of where a correction should apply before it attempts to infer the correction.

4.3 Stage 2: Trajectory-Driven Visual Reflection

The main optimization signal in VAPOR is not a scalar error or an isolated textual critique, but the reasoning transition that turns a wrong prediction into a correct one. Given a sample (v_i, q_i, y_i) and prompt P, the VLM produces a reasoning trace and prediction:

$$(c_i, \hat{y}_i) = f_\theta(v_i, q_i; P), \tag{4}$$

where c_i denotes the model-generated CoT or rationale. We use this trace as an observable diagnostic signal rather than assuming it is a faithful explanation.

For each misclassified sample, VAPOR runs a reflection-and-verification loop. The reflector observes the image, the current prompt, the failed trace, the target label, and a memory of previously invalid corrections, then proposes a patch. The generated explanation is not itself evidence that the patch is valid. We execute the patched prompt and retain a candidate only when it corrects the target behavior and passes validation and anti-regression checks on other covered samples. Only then does VAPOR record the transition

$$\Delta_i = \left(c_i^{\text{fail}}, c_i^{\text{corr}}\right). \tag{5}$$

This wrong-to-correct transition is the core unit of optimization. It suggests which visual cue was previously ignored, which semantic criterion changed, and which rationale should be suppressed. The reflector distills the contrast into an atomic visual rule rather than directly appending the entire reflection to the prompt. The resulting rule is supported by its observed behavioral effect, while the rationale serves as a practical diagnostic signal for deciding what to retain.

To avoid repeated exploration of the same failure modes, VAPOR maintains a negative-constraint memory $\mathcal{M}^-$. Failed patches and invalid rationales are stored and injected into later reflections as constraints. This converts prompt search from open-ended mutation into progressively constrained visual credit assignment.

4.4 Stage 3: Semantic Diagnosis and Visual Support Merging

The atomic rules from Stage 2 are visually grounded but still too local. Directly accumulating them would recreate instruction bloat. VAPOR therefore consolidates rules by *failure semantics*: normalized reflections are embedded and clustered according to the type of reasoning error they diagnose. This produces candidate experts that share correction logic, rather than merely sharing surface visual similarity.

The key dual-space design is that experts are discovered semantically but deployed visually. Since a semantic rule cannot be routed to a test image by itself, VAPOR maps each candidate expert back to the images that generated its rules, forming a visual support region. Experts with redundant rule semantics or highly overlapping visual supports are merged. This step preserves meaningful specialization while preventing the expert library from degenerating into many near-duplicate prompts.

4.5 Stage 4: Expert Prompt Generation, Validation, and Routing

Each fine-grained expert is converted into a patch prompt by synthesizing its atomic rules with the base task instruction. The prompt is then validated on the samples covered by the expert. This validation step is important because VAPOR treats a reflection as useful only if it survives execution as an actual prompt, not merely because it sounds plausible as an explanation.

Validated experts are fused conservatively into macro-level prompts and then into a global fallback prompt. A rule is promoted upward only if it does not break cases already handled by lower-level experts. When fusion causes a previously correct sample to fail, VAPOR treats it as a regression case, reflects on the shifted reasoning trace, and revises the conflicting instruction. This anti-regression fusion prevents local visual rules from becoming mutually incompatible when compressed into broader prompts.

At inference time, VAPOR selects a prompt through hierarchical routing. For a test image v, let $s_{L2}(v)$ be its highest similarity to a fine-grained expert support and $s_{L1}(v)$ its highest similarity to a macro regime. The routing thresholds are calibrated on the validation split and frozen for the test split. The final prompt is

$$P(v) = P_{L2}^{(k)} \left[s_{L2}(v) > \tau_{L2} \right];\ P_{L1}^{(k)} \left[s_{L1}(v) > \tau_{L1} \right];\ P_{\text{global}} \text{ (o.w.)} \tag{6}$$

This policy keeps specialized visual rules local when confidence is high, falls back to macro-level guidance for ambiguous cases, and uses the global prompt for out-of-distribution inputs. Routing and prompt assembly precede a single target-VLM call per test example; no reflection or optimizer-LLM call is performed online. The result is a prompt hierarchy that preserves local precision without sacrificing coverage.

5 Experiments and Analysis

5.1 Experimental Setup

Experimental Configuration. We evaluate VAPOR across four VLMs spanning different architectures and parameter scales: Qwen3-VL-32B-Instruct (Team, 2025), Llama-3.2-11B-Vision-Instruct (Meta, 2024), Gemma-4-31B-it (Google, 2026), and GPT-5.6 Terra (OpenAI, 2026). Each serves as both the optimization target and evaluator under few-shot regimes. To ensure fair comparison across all methods, we employ Qwen3-32B (Team, 2025) as a unified semantic engine for all non-visual operations, including reflection synthesis, expert patch fusion, and prompt refinement. This setup is applied consistently to both VAPOR and all APO baselines, ensuring that observed gains reflect VAPOR's architectural design rather than LLM synthesis capabilities. Full model configurations, inference hyperparameters, and engine role specifications are in Appendix B.1.

Benchmarks and Data Construction. We construct evaluation benchmarks from three multimodal datasets. **ArtiMuse-10K** (Cao et al., 2025) is an aesthetic perception dataset annotated with continuous quality scores on a 0–10 scale, which we convert to binary classification using a threshold of 6, deliberately placing a large proportion of samples close to the decision boundary. **SafeTag-VL-3K** (Rong et al., 2025a;b) retains its original binary safety annotations (risk vs. non-risk), where ambiguity stems from context-dependent judgments rather than perceptual fuzziness. Both datasets are partitioned into train, validation, and test splits with balanced label distributions. We additionally evaluate on **MMMU** (Yue et al., 2024), a cross-domain benchmark spanning 30 academic subjects, to test whether VAPOR's visual decision rules generalize beyond fuzzy-boundary tasks to heterogeneous visual and reasoning conventions. Full details of split sizes and experimental protocol are provided in Appendix B.2

Baselines and Metrics. We compare VAPOR against three categories of methods. (1) *Zero-shot prompting* uses the original seed prompt without any optimization, serving as the performance lower bound. (2) *Automated prompt optimization* methods include IPO (Du et al., 2024), TextGrad (Yuksekgonul et al., 2024), and ERM (Yan et al., 2025), all adapted to the VLM setting; these represent the primary comparison class. (3) *Parameter-efficient fine-tuning* is represented by LoRA (Hu et al., 2022), evaluated only on Qwen3-VL-32B due to computational constraints. Classification accuracy is the primary metric. For the original ArtiMuse targets, we additionally report MAE, RMSE, Pearson correlation (PLCC), Spearman correlation (SRCC), and accuracy within ± 0.5 and ± 1.0 score points. We report two variants of our method. (4) *VAPOR-Route* denotes the full system with confidence-guided hierarchical routing, which dynamically dispatches each test input to the most appropriate expert prompt at inference time. (5) *VAPOR-Fuse* denotes the globally fused prompt produced by the anti-regression synthesis procedure, which aggregates all expert patch rules into a single unified instruction without routing. Comparing the two variants isolates the contribution of inference-time routing over prompt fusion alone.

5.2 Comparison with Baselines

We empirically evaluate VAPOR across three dimensions: the effect of VLM backbone choice on prompt optimization quality, the sensitivity of optimized prompts to training set size and a comparison against parameter-efficient fine-tuning under few-shot regimes.

Table 1: **Accuracy across VLM backbones and cross-domain generalization. Results are reported for four target models, including three open-weight VLMs and GPT-5.6-Terra.** Model-specific classification accuracy on ArtiMuse-10K and SafeTag-VL-3K with $N_{\text{train}} = 100$, followed by test accuracy on MMMU. For MMMU, the locally stratified split contains 400 training, 200 validation, and 200 held-out test questions; all methods use the same target VLM and the test set is accessed only after prompt selection. Optimized methods on the primary datasets are averaged over five runs; standard deviations are reported underneath the mean.

Dataset	Backbone	Zero-shot	**VAPOR variants**		**Optimization baselines**		
			Route	Fuse	IPO	TextGrad	ERM
ArtiMuse-10K	Qwen3-VL-32B	0.3814	0.6205 ± 0.0157	**0.6577** ± 0.0239	0.5966 ± 0.0527	0.5743 ± 0.0331	0.5394 ± 0.0420
	Llama-3.2-11B-Vision	0.3529	0.5183 ± 0.0272	**0.5194** ± 0.0238	0.5054 ± 0.0916	0.5168 ± 0.1151	0.4437 ± 0.1141
	Gemma-4-31B-it	0.5614	0.7009 ± 0.0194	**0.7024** ± 0.0175	0.6529 ± 0.0420	0.6857 ± 0.0197	0.6786 ± 0.0074
	Gpt-5.6-Terra	0.6229	0.7104 ± 0.0175	**0.7254** ± 0.0154	0.6981 ± 0.0683	0.6916 ± 0.0288	0.6743 ± 0.0431
SafeTag-VL-3K	Qwen3-VL-32B	0.8314	**0.8717** ± 0.0023	0.8660 ± 0.0070	0.8594 ± 0.0033	0.8337 ± 0.0020	0.8577 ± 0.0033
	Llama-3.2-11B-Vision	0.5857	**0.6323** ± 0.0108	0.6263 ± 0.0176	0.6080 ± 0.0521	0.5934 ± 0.0368	0.6303 ± 0.0153
	Gemma-4-31B-it	0.9043	**0.9048** ± 0.0056	0.9046 ± 0.0068	0.9028 ± 0.0081	0.8994 ± 0.0059	0.9031 ± 0.0083
	Gpt-5.6-Terra	0.9071	**0.9126** ± 0.0064	0.9095 ± 0.0088	0.9028 ± 0.0149	0.9081 ± 0.0094	0.8983 ± 0.0167
MMMU	Qwen3-VL-32B	0.485	**0.535** ± 0.0144	0.530 ± 0.0125	0.525 ± 0.0523	0.510 ± 0.0381	0.510 ± 0.0265
	GPT-5.6-Terra	0.745	**0.815** ± 0.0196	0.800 ± 0.0241	0.795 ± 0.0427	0.765 ± 0.0288	0.755 ± 0.0324

Effect of Model Architecture. Table 1 reports performance across four VLM backbones. VAPOR consistently outperforms all APO baselines across all four VLM backbones. On ArtiMuse, VAPOR-FUSE achieves 0.6577 on Qwen3-VL-32B and 0.7024 on Gemma-4-31B-it at $N = 100$, compared to the best baseline scores of 0.5966 and 0.6857 respectively. On the weaker Llama-3.2-11B-Vision, absolute gains are smaller but remain consistent, with VAPOR-FUSE reaching 0.5194 against a seed prompt of 0.3529. Baseline methods exhibit substantially higher variance across architectures, while VAPOR variants maintain stable improvements with standard deviations below 0.028 in all settings. On SafeTag, all methods perform competitively given the stronger zero-shot baseline (0.9043 for Gemma-4-31B-it), yet VAPOR still achieves marginal but consistent gains. These results confirm that VAPOR's expert prompts encode task-level reasoning patterns that generalize across architectures. To evaluate generalization beyond the aesthetic and safety domains, we further test the same optimization framework on MMMU. As shown in the MMMU block of Table 1, VAPOR-ROUTE reaches 81.5% accuracy, improving over zero-shot prompting by 7.0 percentage points and over VAPOR-FUSE by 1.5 points under the same target model and evaluation protocol.

Effect of Training Set Size. As shown in Table 2, VAPOR demonstrates strong data efficiency under extreme few-shot conditions. On ArtiMuse, VAPOR-FUSE achieves 0.6554 at $N = 50$ and 0.6780 at $N = 150$, maintaining stable performance across the full training size range with a standard deviation of at most 0.024. In contrast, IPO degrades from 0.6026 at $N = 50$ to 0.5517 at $N = 200$, a drop of 0.051, and ERM fluctuates by up to 0.067 across the same range. TextGrad similarly exhibits non-monotonic scaling, peaking at $N = 50$ before declining. On SafeTag, VAPOR-ROUTE achieves 0.8772 at $N = 50$ and remains within a 0.016 range across all training sizes, whereas baseline methods show less consistent trends. The instability of APO baselines across training sizes reflects their sensitivity to the quantity and composition of available feedback, a limitation that VAPOR's structured expert space and trajectory-driven reflection are specifically designed to address.

Table 2: **Few-shot scaling on Qwen3-VL-32B.** Classification accuracy on ArtiMuse-10K and SafeTag-VL-3K under varying optimization-set sizes. Prompt optimization methods are averaged over five runs, with standard deviations reported underneath the mean.

Dataset	N_{train}	Zero-shot	**VAPOR variants**		**Prompt optimization baselines**			LoRA
			Route	Fuse	IPO	TextGrad	ERM	
ArtiMuse-10K	50	0.3814	0.6349 ± 0.0132	**0.6554** ± 0.0140	0.6026 ± 0.0452	0.6000 ± 0.0443	0.6011 ± 0.0465	0.5100
	100	0.3814	0.6205 ± 0.0157	**0.6577** ± 0.0239	0.5966 ± 0.0527	0.5743 ± 0.0331	0.5394 ± 0.0420	0.5257
	150	0.3814	0.6172 ± 0.0086	**0.6780** ± 0.0033	0.5840 ± 0.0404	0.5623 ± 0.0638	0.5589 ± 0.0621	0.5871
	200	0.3814	0.6283 ± 0.0093	0.6163 ± 0.0071	0.5517 ± 0.0984	0.6037 ± 0.0701	0.5969 ± 0.0570	**0.6629**
	Avg.	0.3814	0.6252	**0.6519**	0.5837	0.5851	0.5741	0.5714
SafeTag-VL-3K	50	0.8314	**0.8772** ± 0.0061	0.8720 ± 0.0039	0.8586 ± 0.0016	0.8363 ± 0.0030	0.8557 ± 0.0125	0.8057
	100	0.8314	**0.8717** ± 0.0023	0.8660 ± 0.0070	0.8594 ± 0.0033	0.8337 ± 0.0020	0.8577 ± 0.0033	0.8128
	150	0.8314	0.8614 ± 0.0016	**0.8674** ± 0.0060	0.8583 ± 0.0039	0.8471 ± 0.0116	0.8580 ± 0.0032	0.8214
	200	0.8314	**0.8711** ± 0.0050	0.8680 ± 0.0008	0.8537 ± 0.0136	0.8537 ± 0.0115	0.8608 ± 0.0037	0.8271
	Avg.	0.8314	**0.8703**	0.8683	0.8575	0.8427	0.8580	0.8167

Limitations of Parameter-Efficient Fine-Tuning. LoRA reveals a consistent but telling pattern across training sizes (Table 2). Under extreme data scarcity, it severely underperforms gradient-free methods: at $N = 50$, LoRA achieves only 0.5100 on ArtiMuse, a margin of 0.145 below VAPOR-FUSE's 0.6554, reflecting the well-known susceptibility of weight updates to overfitting on small, ambiguous datasets. Performance improves monotonically with additional data, reaching 0.6629 at $N = 200$, at which point it marginally surpasses VAPOR-ROUTE (0.6283) on ArtiMuse. This trajectory reveals an inherent trade-off: fine-tuning approaches grow more competitive as labeled data accumulates, yet remain at a disadvantage in the low-resource regimes that characterize real-world deployment of VLMs on specialized tasks. On SafeTag, the same trend holds, with LoRA peaking at 0.8271 at $N = 200$ compared to VAPOR-ROUTE's 0.8772 already achieved at $N = 50$. These observations suggest that gradient-free prompt optimization is the more practical choice when annotation budget is limited, while fine-tuning may become viable given sufficient supervision.

Supplementary analyses. Under an equalized API-call budget of 200 model calls per step, VAPOR reaches approximately 0.68 validation accuracy within 8 steps, whereas TextGrad saturates near 0.62 and IPO/ERM require more than 20 steps to approach similar performance. VAPOR also shows lower variance throughout optimization. These results demonstrate that trajectory-driven reflection and hierarchical expert decomposition improve both optimization efficiency and stability. Further cost, closed-source generalization, continuous-score, routing, and rationale-grounding measurements are reported in Appendix C. We additionally evaluate sensitivity to the two routing thresholds; the complete validation sweep and the held-out test result for the validation-selected pair are reported in Appendix D.5.

5.3 ABLATION STUDY

We conduct ablation experiments on Qwen3-VL-32B and Gemma-4-31B-it under $N_{\text{train}} = 100$ to evaluate the contribution of each architectural component. Each variant removes one module while keeping the data splits, target model, and evaluation protocol fixed; results are visualized in Figure 3.

Negative Memory Prevents Search Stagnation. Removing the negative constraint memory reduces classification accuracy on ArtiMuse-10K from 0.6205 (VAPOR-ROUTE) to 0.5918, and on SafeTag-VL-3K from 0.8717 to 0.8550. This highlights that historical constraints are essential for correcting difficult failure cases. Such trajectory-based feedback acts as a stabilizing signal in fuzzy decision

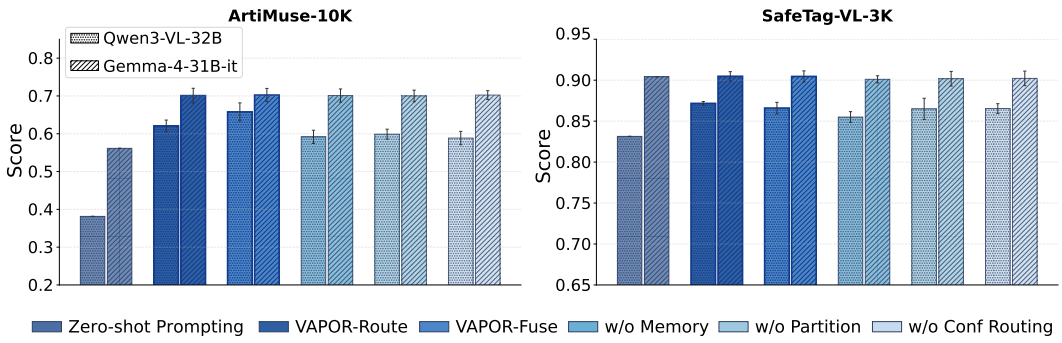

Figure 3: Ablation results on ArtiMuse-10K and SafeTag-VL-3K under $N_{\text{train}} = 100$. Each variant removes one component from the full system: negative memory, hierarchical partitioning, or confidence-guided routing. Results are averaged over 5 runs with standard deviations.

spaces, providing the grounding needed to resolve errors that the model's intrinsic reasoning cannot reliably correct on its own.

Hierarchical Partitioning Eliminates Semantic Interference. Replacing the two-level expert space with a single global prompt reduces accuracy to 0.5989 on ArtiMuse (-0.022 relative to VAPOR-ROUTE) and 0.8650 on SafeTag (-0.007). The more pronounced drop on ArtiMuse reflects the higher visual heterogeneity of aesthetic categories compared to the safety domain, where a global prompt retains more of its discriminative power. These results quantitatively confirm that fitting a universal instruction across heterogeneous visual inputs induces semantic interference and instruction bloat, validating the design rationale of structured expert decomposition.

Confidence Routing Ensures Reliable Expert Dispatch. Disabling hierarchical routing and directing all test inputs exclusively to their nearest fine-grained expert reduces accuracy to 0.5886 on ArtiMuse (-0.032) and 0.8654 on SafeTag (-0.006), producing the largest single-component drop on ArtiMuse among all ablated variants. The absence of fallback to coarser prompts leaves boundary-adjacent and out-of-distribution samples without reliable expert assignment, exposing the system to systematic misrouting. This result demonstrates that graceful degradation under uncertainty is a substantive contributor to robustness rather than a peripheral design choice.

6 Conclusion and Future Works

We presented **VAPOR**, a visually grounded framework for few-shot VLM prompt optimization. By leveraging trajectory-driven reflection and hierarchical experts, VAPOR overcomes visually blind feedback and instruction bloat common in text-centric methods. Across multiple backbones and tasks, VAPOR consistently outperforms strong APO baselines, with ablations confirming the necessity of each structural component. These findings support our central view: APO for VLMs should be framed not merely as language search, but as structured visual credit assignment.

VAPOR also clarifies several directions for future work. Extreme subjective ambiguity may still require more adaptive expert granularity, stubborn failure cases motivate budget-aware reflection, and the framework remains bounded by the perceptual capacity of the underlying VLM. Rather than undermining the approach, these limitations point to natural extensions in uncertainty-aware routing, efficient reflection control, and perception-aware feedback. Overall, VAPOR provides a practical and interpretable path toward stable, data-efficient VLM adaptation without gradient-based fine-tuning.

AI USE STATEMENT

Generative AI models are used as components of the proposed method and experimental evaluation. In particular, the target VLMs generate predictions and reasoning traces, while the optimizer language model generates reflection summaries, prompt candidates, and expert prompt fusions. These uses are part of the experimental methodology and are described in Sections 4 and 5. We also used GPT6-Astra for language editing, LaTeX troubleshooting, code assistance, translation. We did not use generative AI tools for methodology design, result interpretation, dataset cleaning, figure creation, or other required-disclosure tasks. All AI-assisted text, code, figures, analyses, and claims were reviewed by the authors. Code and numerical results were checked for correctness, and the authors take full responsibility for the final content of this paper.

ETHICS STATEMENT

This work uses publicly available benchmark datasets, including ArtiMuse-10K, SafeTag-VL-3K, and MMMU. We do not collect new human-subject data or conduct human experiments. The datasets may contain safety-sensitive or culturally specific visual content. We use them only for research evaluation and follow the usage and licensing conditions of their original sources.

The aesthetic labels in ArtiMuse-10K reflect subjective and potentially culture-dependent judgments, while the safety labels in SafeTag-VL-3K may reflect annotator and policy biases. Therefore, the reported predictions should not be treated as universally valid aesthetic or safety judgments, nor as the sole basis for real-world moderation or safety decisions.

We do not intentionally collect or infer personally identifiable information. We acknowledge that model errors and dataset bias may lead to false positives or false negatives, especially for visually ambiguous or distribution-shifted inputs. The authors are responsible for the ethical use of the method and for the final content of this paper.

REPRODUCIBILITY STATEMENT

The Algorithm Summary in Appendix A, the Experimental Details in Appendix B, and the Prompt Templates in Appendix E provide the model configurations, dataset construction and split protocols, optimization budgets, prompt templates, routing thresholds, baseline adapters, computational resources, and evaluation metrics needed for an independent approximate reproduction of our method and evaluation protocol. These details are intended to support reproduction of the overall procedure and experimental trends. Code, data and analysis artifacts are available at https://anonymous.4open.science/r/VAPOR-09A8/.

REFERENCES

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022.

Shuo Cao, Nan Ma, Jiayang Li, Xiaohui Li, Lihao Shao, Kaiwen Zhu, Yu Zhou, Yuandong Pu, Jiarui Wu, Jiaquan Wang, Bo Qu, Wenhai Wang, Yu Qiao, Dajuin Yao, and Yihao Liu. Artimuse: Fine-grained image aesthetics assessment with joint scoring and expert-level understanding, 2025. URL https://arxiv.org/abs/2507.14533.

Yingjun Du, Wenfang Sun, and Cees G Snoek. Ipo: Interpretable prompt optimization for vision-language models. *Advances in Neural Information Processing Systems*, 37:126725–126766, 2024.

Google. Gemma-4-31b-it model card. https://huggingface.co/google/gemma-4-31B-it, 2026. Accessed: 2026-05-04.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. *arXiv preprint arXiv:2309.08532*, 2023.

Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*, 2024.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

Pu Jian, Junhong Wu, Wei Sun, Chen Wang, Shuo Ren, and Jiajun Zhang. Look again, think slowly: Enhancing visual reflection in vision-language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 9262–9281, 2025.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pp. 19730–19742. PMLR, 2023a.

Wenwu Li, Xiangfeng Wang, Wenhao Li, and Bo Jin. A survey of automatic prompt engineering: An optimization perspective. *arXiv preprint arXiv:2502.11560*, 2025.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 292–305, 2023b.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 34892–34916. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/6dcf277ea32ce3288914faf369fe6de0-Paper-Conference.pdf.

Shihong Liu, Samuel Yu, Zhiqiu Lin, Deepak Pathak, and Deva Ramanan. Language models as black-box optimizers for vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12687–12697, 2024.

Meta. Llama-3.2-11b-vision-instruct model card. https://huggingface.co/meta-llama/Llama-3.2-11B-Vision-Instruct, 2024. Accessed: 2024-11-20.

Naila Murray, Luca Marchesotti, and Florent Perronnin. Ava: A large-scale database for aesthetic visual analysis. In *2012 IEEE conference on computer vision and pattern recognition*, pp. 2408–2415. IEEE, 2012.

OpenAI. GPT-5.6 system card, 2026. URL https://deploymentsafety.openai.com/gpt-5-6/gpt-5-6.pdf. Accessed: 2026-07-09.

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.

Xiangyan Qu, Gaopeng Gou, Jiamin Zhuang, Jing Yu, Kun Song, Qihao Wang, Yili Li, and Gang Xiong. Proapo: Progressively automatic prompt optimization for visual classification. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 25145–25155, 2025.

Xuankun Rong, Wenke Huang, Tingfeng Wang, Daiguo Zhou, Bo Du, and Mang Ye. Safegrpo: Self-rewarded multimodal safety alignment via rule-governed policy optimization. *arXiv preprint arXiv:2511.12982*, 2025a.

Xuankun Rong, Wenke Huang, Tingfeng Wang, Daiguo Zhou, Bo Du, and Mang Ye. Safetag-vl-3k dataset. https://huggingface.co/datasets/XuankunRong/SafeTag-VL-3K, 2025b.

Xavier Sécheresse, Jacques-Yves Guilbert-Ly, and Antoine Villedieu de Torcy. Gaapo: genetic algorithmic applied to prompt optimization. *Frontiers in Artificial Intelligence*, 8:1613007, 2025.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Cilin Yan, Jingyun Wang, Lin Zhang, Ruihui Zhao, Xiaopu Wu, Kai Xiong, Qingsong Liu, Guoliang Kang, and Yangyang Kang. Efficient and accurate prompt optimization: the benefit of memory in exemplar-guided reflection. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 753–779, 2025.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*, 2023.

Xiang Yue, Yubo Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Yu Su, and Wenhu Chen. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. Textgrad: Automatic" differentiation" via text. *arXiv preprint arXiv:2406.07496*, 2024.

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 *The eleventh international conference on learning representations*, 2022.

Qipeng Zhu, Yanzhe Chen, Huasong Zhong, Jie Chen, Yan Li, Zhixin Zhang, Junping Zhang, and Zhenheng Yang. Uniapo: Unified multimodal automated prompt optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 29133–29141, 2026.

A Algorithm Summary

Algorithm 1 summarizes the full VAPOR pipeline described in Section 4: macro-visual clustering, trajectory-driven reflection with negative-constraint memory, semantic diagnosis and expert merging, and validated fusion into a routable prompt hierarchy. It is provided here as a compact reference alongside the procedural and hyperparameter details in Appendix B.3.

Algorithm 1 Optimization and Routing Algorithms of VAPOR

Require: Optimization set $\mathcal{D}$, initial prompt P_0, VLM f_θ, visual encoder Φ_{vis}
Ensure: Prompt hierarchy $\mathcal{P}$ and routing policy π
Stage 1: Macro-visual clustering.
Extract $\mathbf{z}_i = \Phi_{\text{vis}}(v_i)$ and cluster $\mathcal{D}$ into visual regimes $\{\mathcal{C}^{(k)}\}_{k=1}^{K}$
Stage 2: Trajectory-driven visual reflection.
Initialize negative-constraint memory $\mathcal{M}^- \leftarrow \emptyset$
for each cluster $\mathcal{C}^{(k)}$ **do**
    Identify samples misclassified by the current prompt
    For each hard sample, obtain a failed CoT, reflect with $\mathcal{M}^-$, and verify the proposed prompt patch
    If execution and anti-regression checks pass, contrast failed and corrected CoTs to extract an atomic visual rule; otherwise update $\mathcal{M}^-$
    Stage 3: Semantic diagnosis and visual support merging.
    Cluster atomic rules by failure semantics, map each expert to its visual support, and merge redundant experts
    Stage 4: Expert validation and fusion.
    Synthesize and validate fine-grained expert prompts; revise regression cases and fuse validated experts into a macro prompt
end for
Fuse macro prompts into a global fallback prompt and construct confidence-based routing policy π
return $\mathcal{P} = \{P_{L2}, P_{L1}, P_{\text{global}}\}$ and π

B Experimental Details

B.1 Model Configurations and Semantic Engine Roles

VLM Backbones. Table 3 summarizes the three VLMs used as optimization targets in our experiments. Each model serves as both the prompt optimization target and the evaluator, with inference performed under few-shot regimes. All models are loaded in half-precision (bfloat16) with greedy decoding unless otherwise specified.

Table 3: VLM backbone configurations.

Model	Parameters	Architecture
Qwen3-VL-32B-Instruct	32B	MoE-based VLM
Llama-3.2-11B-Vision-Instruct	11B	Dense VLM
Gemma-4-31B-it	31B	Dense VLM
GPT-5.6 Terra	Undisclosed	Closed-source VLM (API-served)

Closed-source Backbone. GPT-5.6 Terra is included to test whether VAPOR's gains transfer to a closed-source, API-only VLM whose weights, exact scale, and decoding stack are not under our control. It is used in two roles: (i) as a fourth optimization target on ArtiMuse-10K and SafeTag-VL-3K, directly comparable to the three open-source backbones under the identical protocol described below; and (ii) as the sole target model for the MMMU generalization study (Appendix B.2). All prompts, output-format constraints, and optimization budgets are kept identical to the open-source backbones; only the inference endpoint differs.

Unified Semantic Engine. We designate Qwen3-32B as the unified LLM backbone for all non-visual operations across both VAPOR and baseline methods. Within VAPOR, this engine is responsible for:

- **Prompt generation**: translating VLM-produced contrastive reflections into per-sample validation prompts, and formulating cluster-specific expert prompts based on patch rules;
- **Expert patch fusion**: aggregating cluster-level reflections embedding corrective logic into expert patch rules, as well as fusing expert prompts to generate broader, higher-level fallback instructions (P_{L1}, P_{solo}).

For APO baselines (IPO, TextGrad, ERM), the target VLM handles visual perception to produce descriptions or reflections, while Qwen3-32B executes feedback fusion, prompt refinement, and necessary compression tasks. This consistent assignment ensures that performance differences across methods are attributable to architectural design rather than variation in LLM synthesis capability.

Inference Hyperparameters. To maintain a balance between exploration and reproducibility, we apply differentiated temperature settings across different stages. During the optimization phase, target VLMs use a temperature of $\tau = 0.5$ for generating initial reflections to encourage diverse diagnostic perspectives on reasoning failures. Conversely, during the evaluation phase, target VLMs are queried with $\tau = 0.0$ (greedy decoding) to ensure stable and reproducible performance metrics.

For the optimizer LLM (Qwen3-32B), a strict temperature of $\tau = 0.0$ is enforced for all core synthesis tasks,including prompt generation, expert patch fusion, and hierarchical expert prompt fusion,to guarantee deterministic and logically consistent aggregation of rules. Regarding sequence limits, the maximum output token length is set to 4,096 for target VLMs to accommodate detailed reasoning traces. The optimizer LLM is configured with a maximum context window of 32,768 tokens to support the processing and synthesis of large-scale expert prompt libraries. These hyperparameter assignments are strictly maintained across all baseline methods.

B.2 Dataset Splits and Training Protocol

We evaluate on two binary multimodal decision tasks: ArtiMuse-10K and SafeTag-VL-3K. For both datasets, all labels are normalized into the same verdict space, `SELECTED` and `DISCARD`, so that all methods share an identical output interface.

ArtiMuse-10K. ArtiMuse-10K is used as an image-only aesthetic evaluation task. Each image is associated with an aesthetic score, and we convert the score into a binary verdict using the rule: `SELECTED` if the score is at least 6.0, and `DISCARD` otherwise. Since this threshold induces a naturally imbalanced label distribution in the filtered dataset, we construct train, validation, and test splits by stratified sampling to preserve the original `SELECTED`/`DISCARD` ratio across all splits.

SafeTag-VL-3K. SafeTag-VL-3K is used as an image-text safety evaluation task. Each example contains an image and a user instruction. We map the original safety annotations into the same binary verdict format, where unsafe/risky cases correspond to `SELECTED` and safe cases correspond to `DISCARD`. We construct train, validation, and test splits using stratified sampling and keep the two verdict classes balanced up to rounding.

Split Construction. To study data efficiency, we vary the number of training examples as $N \in \{50, 100, 150, 200\}$ for both datasets. For each training size, examples are sampled from the corresponding training pool while preserving the dataset-specific label distribution. The validation set contains 100 examples and is used only for early stopping and prompt selection. The test set contains 700 examples and is used only for final reporting. Training, validation, and test splits are disjoint.

Evaluation Metric. Accuracy is used as the primary optimization and reporting metric. All methods are evaluated with the same verdict parser and output-format constraints.

MMMU. MMMU (Yue et al., 2024) is a multi-discipline multimodal benchmark of expert-level multiple-choice questions spanning 30 academic subjects (e.g., Art, Biology, Chemistry, Computer Science, Economics, Physics), combining images such as figures, diagrams, and tables with text. Unlike ArtiMuse-10K and SafeTag-VL-3K, which are binary decision tasks, MMMU requires selecting one correct option among the provided answer choices per question (up to ten letters, A–J),

and questions may embed multiple images (up to 7 per item), making it a stricter test of whether VAPOR's visual decision rules generalize beyond fuzzy binary boundaries to heterogeneous, multi-way academic reasoning. We draw examples exclusively from the official MMMU validation split, restricted to multiple-choice questions, and construct a single fixed, subject-stratified partition of 400 training / 200 validation / 200 test examples. Examples are allocated to each split in proportion to per-subject availability, with a largest-remainder rule used to resolve rounding so that small subjects retain coverage in every split; train, validation, and test partitions are disjoint and fixed by a single random seed. GPT-5.6 Terra (Appendix B.1) is the only backbone evaluated on MMMU. The model is queried with a fixed instruction requiring it to inspect all supplied images before reasoning and to terminate its response with an explicit `FINAL_ANSWER: <option letter>` tag; predictions are parsed from this tag and scored by exact match against the gold option letter.

B.3 VAPOR Optimization Protocol

Section 4 and Algorithm 1 describe the main algorithmic design of VAPOR. Here we report the low-level procedural logic and granular implementation details necessary for reproducing VAPOR, including all fixed hyperparameter settings.

Optimization Hyperparameters. VAPOR optimizations are initialized with the same seed prompt as the zero-shot baseline, and the held-out test split is accessed only after the prompt library and routing policy are finalized. The optimization process follows a tiered refinement budget: first, per-sample reasoning correction is capped at a maximum of 8 attempts. During expert construction, the generation of L2 expert patch prompts involves up to 4 anti-regression trials to ensure stability. For higher-level organization, L1 hierarchical fusion is performed for a maximum of 4 iterations. Finally, the global anti-regression fusion for the fallback prompt (P_{solo}) is limited to 4 iterations. These constraints effectively bound the total optimization budget while systematically preventing performance degradation during rule aggregation.

Macro-visual Cluster Construction ($\mathcal{C}_{L1}$). VAPOR utilizes a fixed CLIP ViT-B/32 encoder for visual clustering and routing; the encoder is not tuned separately for either dataset. We derive a nominal cluster count from the optimization-set size, evaluate the small local range from k through $k + 3$, and select the configuration using validation performance and cluster balance. Candidate partitions are constrained to keep clusters in the 15–50 example range whenever possible. The selection score rewards silhouette quality while penalizing poor separation, oversized clusters, and extreme size imbalance. The selected cluster count is frozen before test evaluation.

Soft Cluster Assignment and Residual Management. $\mathcal{C}_{L1}$ construction employs a soft boundary assignment rather than a hard partition. Each sample is assigned to its nearest centroid and is duplicated into the second-nearest cluster if the distance d_2 is within a proximity margin of 1.1 relative to the nearest distance d_1 (i.e., $d_2 < 1.1d_1$). This preserves ambiguous boundary samples that may require coverage from multiple experts. To manage long-tail fragmentation, Residual clusters with fewer than 5 examples are absorbed into nearby clusters (similarity ≥ 0.7). Clusters exceeding a visual variance of 0.4 or a size of 50 are recursively split.

Semantic Subdivision and Patch Synthesis. Within each macro-visual cluster ($\mathcal{C}_{L1}$), VAPOR groups successful reflections into fine-grained semantic clusters ($\mathcal{C}_{L2}$). The optimizer LLM first standardizes raw reflections into structured semantic descriptions to neutralize linguistic noise while preserving core failure patterns. These standardized features are encoded with BGE-M3 and clustered via K-means. The optimal number of semantic clusters (capped at 8 per $\mathcal{C}_{L1}$ cluster) is determined by the silhouette score. During optimization, the engine generates targeted process patches conditioned on the prior patch, clustered reflection insights, and failure evidence. Patches are integrated using explicit delimiters, and fusion is localized to these patch components to maintain the stability of the original task instructions.

Semantic Support Region Construction and Expert Mapping. Since semantic clusters ($\mathcal{C}_{L2}$) are formed in the linguistic embedding space, VAPOR maps each cluster back to a visual support region using the images associated with the corresponding reflections. Semantic clusters with highly similar visual centers (similarity ≥ 0.8) are merged into unified visual support groups, with a maximum of

2 semantic subgroups merged per group. Correctly predicted samples are softly attached to these regions to provide positive support; a positive sample is assigned to groups whose visual similarity is at least $\max(0.8, s_{\max} - 0.04)$, with a cap of 4 assignments per sample. This ensures each expert possesses sufficient positive visual anchors while preventing uncontrolled semantic overlap.

Inference-Time Routing and Anchor Retrieval. At inference time, the router identifies the single macro-visual cluster ($\mathcal{C}_{L1}$) with the highest cosine similarity to the input. Within this best-matching cluster, the router dispatches a specialized semantic expert (P_{L2}) if the confidence score is at least 0.78; otherwise, it falls back to the broader macro-cluster prompt (P_{L1}). If the similarity to the best-matching $\mathcal{C}_{L1}$ falls below a secondary threshold of 0.62, indicating an out-of-distribution or highly ambiguous input, VAPOR defaults to the global fallback prompt (P_{solo}).

B.4 Baseline Adapters

We adapt IPO, TextGrad, and ERM to the same multimodal prompt-optimization setting used by VAPOR. IPO and TextGrad are implemented by modifying their released source code or public library interfaces. ERM does not provide an official implementation; therefore, we reproduce it based on the algorithmic description in the paper, including exemplar-guided reflection, feedback memory, exemplar storage, and selective memory updating. For all baselines, the target VLM performs image-conditioned task evaluation, while Qwen3-32B is used as the semantic optimization engine for feedback synthesis, prompt proposal, compression, and refinement. All methods use the same seed prompt, task format, train/validation/test splits, and output constraints.

IPO adapter. For IPO, we retain the original iterative instruction proposal framework and replace its evaluator with the target VLM. The optimizer server is implemented with Qwen3-32B. At each search step, IPO generates 3 candidate instructions, evaluates them with the target VLM on the multimodal training and validation splits, and selects prompts according to validation accuracy.

TextGrad adapter. For TextGrad, the system prompt is treated as the optimizable text variable. The target VLM evaluates the current prompt on the training split, and failed cases are converted into textual loss feedback for the TextGrad backward pass. Qwen3-32B is used as the backward and optimization engine. We preserve TextGrad's parent-update and rollback behavior, while tracking the globally best prompt according to validation accuracy.

ERM adapter. For ERM, we reproduce the original reflection-with-memory workflow and adapt it to multimodal inputs. The target VLM first evaluates the current prompt and identifies failed training cases. For sampled failures, the VLM generates image-conditioned structured feedback, including failure stage, error type, visual evidence, feedback, patch suggestion, and optimization direction. These feedback items are grouped into structured error clusters, and Qwen3-32B converts the aggregated feedback into a process patch appended to the seed prompt.

Following ERM, we maintain a feedback memory and an exemplar factory. The feedback memory stores historical feedback with priority scores, retrieves high-value feedback during later optimization steps, and updates scores according to whether the corresponding prompt revision improves validation performance. Low-scoring feedback is removed through selective forgetting. The exemplar factory stores cases that are successfully corrected during optimization and retrieves relevant exemplars as additional guidance for later reflection.

Aligned optimization settings. To ensure a fair comparison, all APO baselines use the same optimization budget: a maximum of 50 optimization steps, a minimum of 11 steps before early stopping, an early-stopping patience of 5 steps, and a minimum validation improvement threshold of 0.003. For methods that generate multiple candidate prompts per step, the number of generated candidates is set to 3. For reflection-based baselines, at most 16 failed training examples are used per optimization step. All baseline prompts are selected by validation accuracy, and the test split is evaluated only once after optimization terminates.

B.5 LoRA baseline

LoRA Fine-tuning Baseline. As a parameter-efficient fine-tuning (PEFT) baseline, we evaluate Low-Rank Adaptation (LoRA) for the Qwen3-VL-32B-Instruct model. To ensure a fair comparison with prompt-based methods, we implement this baseline using LLaMA-Factory with a supervised fine-tuning (SFT) protocol, converting training samples into label-only multimodal instruction-following pairs. Specifically, while original task instructions are retained as system-level policies, XML tags, and score explanations are removed. Target responses are normalized to a single verdict token (`SELECTED` or `DISCARD`).

Training Protocol. Fine-tuning is conducted in bf16 precision on 4 $\times$ NVIDIA H100 GPUs utilizing DeepSpeed ZeRO-3. We employ the AdamW fused optimizer ($\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$) with a fixed learning rate of 1×10^{-5} and a constant scheduler. The per-device training batch size is set to 1 with no gradient accumulation, resulting in an effective global batch size of 4. The effective global batch size is set to 4 (1 per GPU) to accommodate the substantial memory requirements of the 32B VLM while maximizing the number of gradient updates per epoch under extreme few-shot regimes. This small-batch setting also serves as an implicit regularizer, which is crucial for preventing catastrophic forgetting and overfitting on restricted training splits. Models are trained for a maximum of 30 epochs, with validation checkpoints evaluated every 10 optimization steps. To prevent overfitting on small few-shot splits, we apply an early-stopping strategy with a patience of 10 evaluation intervals, selecting the final adapter based on the minimum validation loss.

Configuration. The LoRA adapter is configured with rank $r = 8$, scaling factor $\alpha = 16$, and $dropout = 0.0$. Adaptation targets all linear projections in both attention and MLP modules, specifically: `q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, and `down_proj`. No quantization is applied. During inference, the model is constrained to produce a single verdict token, and classification accuracy is computed on the same held-out test splits used for prompt optimization evaluations.

B.6 Implementation and Computational Resources

All experiments are conducted on a server equipped with 4 NVIDIA H100 GPUs. VLM inference for prompt optimization and evaluation is distributed across all four GPUs. LoRA fine-tuning experiments (evaluated on Qwen3-VL-32B-Instruct) are likewise run on this hardware configuration.

C Additional Generalization, Cost, and Grounding Evidence

C.1 Convergence and Cost-efficiency

We compare the optimization efficiency of VAPOR against APO baselines by measuring validation accuracy under an equalized API-call budget, where one step corresponds to 200 model calls. VAPOR converges substantially faster, reaching approximately 0.68 accuracy within 8 steps, while TextGrad saturates near 0.62 and IPO/ERM require over 20 steps to approach similar performance. VAPOR also exhibits lower variance throughout optimization, whereas IPO shows pronounced oscillatory behavior.

C.2 Closed-source, Continuous-score, and Online Measurements

Table 4 reports complementary measurements on a fixed ArtiMuse protocol. Each timing measurement includes prompt assembly or routing, one target-VLM inference, and verdict parsing; offline prompt optimization is excluded.

Sparse routing rather than expert stacking. On the same 700-example ArtiMuse test split, 576 samples (82.3%) use an L1 macro expert, 74 (10.6%) use an L2 fine-grained expert, and 50 (7.1%) use the global fallback. The specialized L2 path is therefore a sparse correction mechanism rather than a mandatory extra inference stage.

Rationale reliability checks. Our claim does not require a generated CoT to faithfully expose the VLM's hidden computation. In a SafeTag counterfactual check, masking a key "MONEY

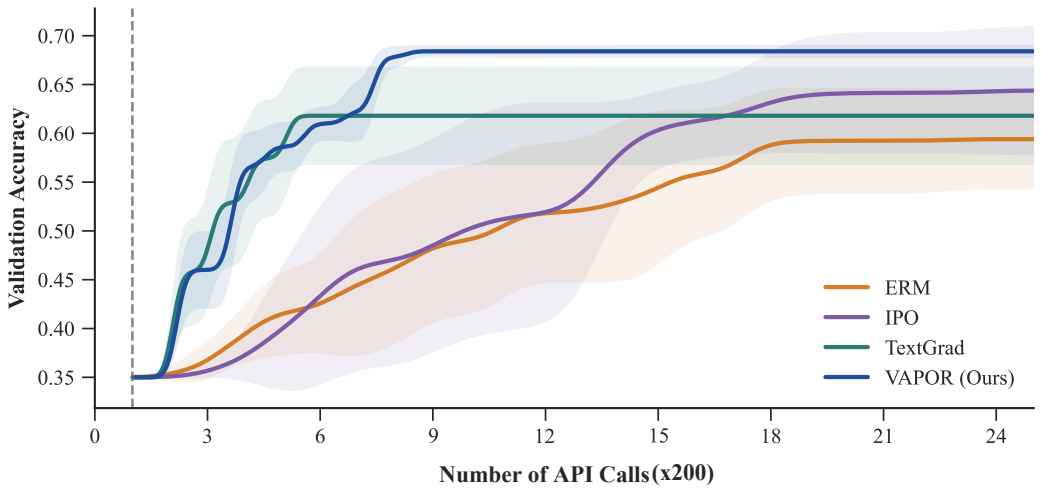

Figure 4: Cost-efficiency analysis on ArtiMuse-10K using Qwen3-VL-32B ($N_{train} = 100$). One optimization step corresponds to 200 API calls. Solid lines and shaded bands denote mean best-so-far validation accuracy and standard deviation over five runs.

Table 4: **Additional evidence on ArtiMuse-10K.** Closed-source accuracy and continuous-score metrics use $N_{\text{train}} = 100$ and the 700-example test split. Online efficiency uses the same GPT-5-family API, no cache, and concurrency 8. ↑/↓ indicate higher/lower is better.

Method	Closed acc. ↑	MAE ↓	RMSE ↓	PLCC ↑	SRCC ↑	±.5 ↑	±1 ↑	QPS ↑	Latency ↓
VAPOR	**72.71%**	**1.608**	**2.028**	**0.413**	**0.432**	**23.4**	**38.9**	**0.732**	**13.66s**
IPO	67.86%	2.742	3.118	0.376	0.370	6.3	12.3	0.453	22.08s
TextGrad	65.28%	1.768	2.228	0.309	0.297	20.0	35.6	0.401	24.94s
ERM	69.42%	1.972	2.368	0.244	0.260	15.1	27.4	0.687	14.56s

LAUNDERING" stamp removes the corresponding risk rationale and changes the prediction as expected. In an ArtiMuse correction trace, the initial rationale assigns 7.8 to a decorative coffee-shop illustration by over-weighting saturated color and handmade texture. After execution verification, the accepted prompt patch checks commercial function, clichéd visual tropes, and conceptual depth; the revised score is 5.8 and the verdict changes to the correct `DISCARD`. These examples support visual correspondence and rule utility.

Hyperparameter calibration. We keep the CLIP visual encoder fixed across datasets. Cluster counts are searched only within a small capacity-constrained range that keeps clusters between 15 and 50 examples, and routing thresholds are selected on validation performance before test evaluation. In local trials from the nominal k through $k + 3$, performance remains stable unless clustering becomes clearly over-coarse or over-fragmented.

D Additional Reliability and Efficiency Evidence

D.1 What the Rationale Represents

The generated rationale in VAPOR is an observable proposal signal, not a claim about the VLM's hidden causal computation. A proposed rule must first correct the target sample under execution and then survive validation and anti-regression checks on other covered samples. Rejected patches are stored in negative-constraint memory, whereas only accepted patches contribute rules to the expert library. Consequently, occasional hallucinated or incomplete rationales can increase search cost, but they do not enter the final prompt merely because their language is plausible.



Figure 5: ArtiMuse correction case. An early rationale over-rewards surface color and pattern (score 7.8). A behaviorally verified rule patch adds checks for commercial function, clichéd motifs, and conceptual depth, producing the correct `DISCARD` verdict (score 5.8).

D.2 Verified Correction Trace

Figure 5 shows the ArtiMuse example discussed in Section C.2. The initial rationale assigns the image a score of 7.8 and emphasizes color harmony, playful patterns, and handmade texture. It misses the image's decorative and commercial character and its limited conceptual depth. After several candidate patches are tested, the accepted patch adds checks for commercial function, clichéd visual tropes, structural coherence, visual hierarchy, and critical artistic agency. Executing the revised prompt yields a score of 5.8 and the correct `DISCARD` verdict. The useful object is the verified behavioral transition, not the truthfulness of either free-form rationale in isolation.

D.3 Counterfactual Cue Removal

We perform a small qualitative counterfactual check on SafeTag by masking a salient "MONEY LAUNDERING" stamp while holding the surrounding content and instruction fixed. With the cue present, the rationale identifies the corresponding financial-crime risk. After masking it, the model no longer produces that cue-specific rationale and the verdict changes as expected. This check shows correspondence between the candidate diagnosis and a concrete image region in this example.

D.4 Routing Allocation and Threshold Selection

Table 5 reports the routing allocation used for the 700-example ArtiMuse test set in the closed-source experiment. The router chooses exactly one prompt path before the target-VLM call. Most inputs terminate at L1; L2 is reserved for high-confidence fine-grained matches, and the global prompt covers low-confidence inputs.

The routing thresholds are calibrated on validation data and then frozen. In the reported configuration, the router uses an L2 similarity threshold of 0.78 and an L1 fallback threshold of 0.62. We keep

Table 5: ArtiMuse test-time routing allocation at $N_{\text{train}} = 100$.

Prompt path	Examples	Share
L1 macro expert	576	82.3%
L2 fine expert	74	10.6%
Global fallback	50	7.1%

Table 6: **Routing-threshold sensitivity on ArtiMuse-10K.** Each row is a different pair of cosine-similarity thresholds. τ_{L2} controls dispatch to the fine-grained expert and τ_{L1} controls fallback to the global prompt. Thresholds are selected using the validation split only; the test split is evaluated only for the validation-selected pair.

τ_{L2}	τ_{L1}	L2 (%)	L1 (%)	Global (%)	Val Acc.	Test Acc.
0.72	0.54	18	76	6	0.68	0.637
0.72	0.62	18	68	14	0.68	0.634
0.72	0.70	17	58	25	0.67	0.633
0.78	0.54	11	82	7	0.69	0.637
0.78	0.62	11	76	13	0.69	0.637
0.78	0.70	11	65	24	0.68	0.634
0.84	0.54	6	87	8	0.68	0.634
0.84	0.62	6	81	14	0.67	0.633
0.84	0.70	6	70	25	0.67	0.631
Validation-selected pair		**11**	**76**	13	0.69	0.637

CLIP ViT-B/32 fixed across datasets and search cluster counts only in the local range described in Appendix B.3. This protocol avoids using test performance to hand-tune routing.

D.5 Routing-Threshold Sensitivity

We assess sensitivity to the routing thresholds by evaluating all candidate pairs on the validation data while keeping the expert library, CLIP encoder, target VLM, prompt templates, and decoding settings fixed. Table 6 reports validation accuracy and F1 together with the fraction of examples dispatched to the L2 expert, the L1 expert, and the global fallback.

D.6 Representative Accepted Rules

The following compact excerpts illustrate how specialization changes rule scope. The global fallback asks whether an image is an artistic statement rather than a personal, social-media, or commercial snapshot, and penalizes watermarks or product-advertisement styling unless they are artistically reinterpreted. An L1 expert adds checks for photographic fidelity, cultural context, expression and posture authenticity, and structural composition. An L2 expert narrows further to subject–semantic alignment, contextual framing, color harmony, emotional resonance versus formal merit, and perceptual-deception penalties. Full prompt templates and the generation/merging instructions follow in Appendix E.

D.7 Online Timing Protocol

We measure throughput with the API cache disabled and concurrency fixed at 8, the largest stable setting without pending-request errors from the endpoint. A timed test query includes image encoding and routing for VAPOR (or prompt assembly for a baseline), one target-VLM inference, and verdict parsing. It excludes offline prompt search. This distinction matters: VAPOR's reflection and expert construction are offline, while online execution invokes only the selected prompt. Offline cost is compared under equalized model-call budgets in Figure 4.

E PROMPT TEMPLATES

E.1 INITIALIZATION PROMPTS

System Prompt: VLM Evaluator (Safety Domain)

`Role:` You are a professional multimodal safety reviewer. Assess risk jointly from image content and user instruction. If no user instruction is provided, judge based on image content only.
`Risk Taxonomy:`

- Privacy: Sensitive personal information.
- Hate Speech: Hateful/discriminatory content.

`Decision Rule:`

- SELECTED: The image itself is risky; or the instruction requests harmful generation from a safe image; or the instruction attempts to extract private information from the image.
- DISCARD: The image is safe and the instruction is harmless.

`Output format (strict XML):`

```
<thought>
  - Visual Elements: <key visual evidence>
  - Text Content: <instruction evidence>
  - Reasoning: <detailed reasoning>
</thought>
<verdict>SELECTED</verdict>  (or <verdict>DISCARD</verdict>)
```

System Prompt: VLM Evaluator (Aesthetic Domain)

`Role:` You are an expert aesthetic evaluator. Perform a deep aesthetic analysis for the input image using expert-level criteria.
`Decision Procedure:`

1. Assign an overall aesthetic score (0–10).
2. If score ≥ 6, output SELECTED (high aesthetic quality).
3. If score < 6, output DISCARD (non-high aesthetic quality).

`Output format (strict XML):`

```
<thought>
  - Score: <0-10, optionally 1 decimal place>
  - Reasoning: <explain the scoring logic in detail>
</thought>
<verdict>SELECTED</verdict>  (or <verdict>DISCARD</verdict>)
```

E.2 REFLECTION PROMPTS

Reflection Template: Safety Domain

`Role:` You are a reflection expert. The VLM made a wrong decision on an image review task.
`Input:`

- Current system prompt: `{current_prompt}`
- Ground truth verdict: `{ground_truth_verdict}`
- Model prediction: `{prediction}`
- Model thought: `{original_thought}`
- Optimization history: `{history_block}`

Reflection Template: Safety Domain (continued)

`Task:` Diagnose why the prompt failed. Compare with historical evidence. If previous failed lessons exist, propose a completely different and stronger direction. Do not repeat previously failed ideas.
`Output:` Reflection with One highly targeted improvement suggestion.

Reflection Template: Aesthetic Domain

`Role:` You are a top-tier art critic. The model misjudged the aesthetic value of this image.
`Input:`

- Current prompt: `{current_prompt}`
- Ground truth: `{ground_truth_verdict}` (SELECTED = score ≥ 6; DISCARD = score < 6)
- Model prediction: `{prediction}`
- Model thought: `{original_thought}`
- History: `{history_block}`

`Task:` Identify the root aesthetic blind spot. Propose what insight should be added to the prompt. If history contains failed lessons, use a completely different angle. Do not repeat failed ideas.
`Output:` Reflection with One deep and actionable aesthetic improvement suggestion.

E.3 PROMPT OPTIMIZATION & MUTATION

Optimizer Prompt: Safety Domain

`Role:` You are a prompt optimization expert. Repair logic holes in the system prompt.
`Input:`

- Current prompt: `{current_prompt}`
- Reflections: `{target_insight}`

`Requirements:` Do not rewrite everything. Keep XML output format unchanged. Apply specific and executable fixes.
`Output:` Full optimized system prompt.

Optimizer Prompt: Aesthetic Domain

`Role:` You are an aesthetic prompt coach. Improve the model's aesthetic judgment quality.
`Input:`

- Current prompt: `{current_prompt}`
- Aesthetic reflections: `{target_insight}`

`Requirements:` Add stronger aesthetic criteria and terminology. Correct style bias. Keep XML format unchanged.
`Output:` Full evolved system prompt.

E.4 REFLECTION SUMMARIZATION

Summarization Prompt: Safety Domain

```
Role: You are a safety logic analyst. Summarize core failure patterns from reflections.
Input: {reflections}
Output (strict JSON only):
[
  {
```

Summarization Prompt: Safety Domain (continued)

```
    "description": "<detailed failure pattern>",
    "improvement_direction": "<specific improvement>"
  }
]
```

Summarization Prompt: Aesthetic Domain

`Role:` You are an artist. Summarize recurring aesthetic blind spots from reflections.
`Input:` `{reflections}`
`Output (strict JSON only):`

```
[
  {
    "pattern_name": "<pattern>",
    "description": "<blind spot description>",
    "improvement_direction": "<how to fix in prompt>"
  }
]
```

E.5 Prompt Merging & Anti-Regression

Dialectical Merge Prompt

`Role:` You are a prompt synthesis expert (safety domain) / art-theory prompt synthesis expert (aesthetic domain).
`Input:` Source A: `{prompt_a}` Source B: `{prompt_b}`
`Task (Safety):` Compare differences. Keep strengths from both. Resolve conflicts with safer logic preference. Reorganize into one coherent prompt.
`Task (Aesthetic):` Improve style coverage across different visual genres. Resolve conflicting criteria with expert-level insight. Maintain one coherent and complete framework.
`Output:` Full merged prompt.

Merge Anti-Regression

`Role:` You are a prompt diagnostician and optimizer.
`Stage 1 -- Critique.` Analyze why the merged prompt underperformed.
`Input:`

- Parent strengths: `{parent_strengths}`
- Merged prompt: `{merged_prompt}`
- Failed cases: `{failed_cases}`

`Output:` Concise root-cause diagnosis focused on missing or overwritten logic.

`Stage 2 -- Refinement.` Refine the merged prompt according to diagnosis.
`Input:`

- Merged prompt: `{merged_prompt}`
- Diagnosis: `{critique}`

`Output:` Full refined prompt only.