

Where Do Vision-Language Models Fail on Abstract Reasoning?

An Information Processing Analysis of ARC

Xuan Zhou, Xuefeng Wei, Zhi Qu, Yusuke Sakai, Hidetaka Kamigaito, Taro Watanabe

Nara Institute of Science and Technology (NAIST), Japan

{xuan.zhou.xc7, xuefeng.wei.yb1}@naist.ac.jp

{qu.zhi.pv5, sakai.yusuke.sr9, kamigaito.h, taro}@is.naist.jp

Abstract

Current vision-language models (VLMs) perform well on many multimodal benchmarks, but still struggle with ARC-style abstract reasoning, where models must infer novel grid transformations from few examples. We analyze these failures through an information-processing decomposition into encoding, transformation, and construction, comparing three controlled settings: direct image-based solving, solving from model-produced text grids, and solving from gold text grids. Across existing ARC-1 and ARC-2 together with our controllable diagnostic benchmark **V-ARAOC**, we observe benchmark-dependent diagnostic signatures: ARC-1 is highly sensitive to encoding quality, ARC-2 remains difficult even after transcription noise is removed, and V-ARAOC is highly sensitive to constrained output construction. These results reveal benchmark-dependent information-processing bottlenecks in VLM-based ARC solving, with different benchmarks exhibiting distinct dominant failure modes.

1 Introduction

Current vision-language models (VLMs) achieve strong performance on many multimodal benchmarks, but many of these tasks rely more on knowledge stored in model parameters than on the ability to solve novel problems from minimal examples. In cognitive research, the latter ability is closely related to *fluid intelligence* (Cattell, 1963). The Abstraction and Reasoning Corpus (ARC) was proposed as a benchmark for this setting: given only a few input-output grid examples, a solver must infer the latent transformation and apply it to a new test input (Chollet, 2019). ARC therefore serves as a natural testbed for fluid intelligence in AI, emphasizing abstract reasoning, object-based reasoning, geometric reasoning, and counting while minimizing task-specific prior knowledge (Chollet, 2019; Wu et al., 2025).

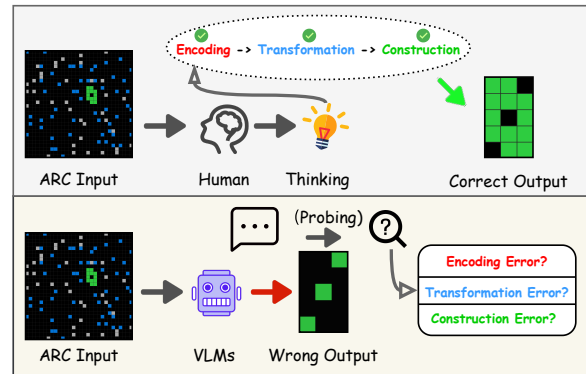


Figure 1: Case of ARC failure diagnosis. When a VLM produces a wrong output, the source of failure may lie in encoding, transformation, or construction, but end-to-end evaluation alone cannot distinguish among them.

Prior work has shown that VLMs and LLMs still struggle on ARC-style tasks, and that representation and prompting choices can materially affect performance (Xu et al., 2024; Acquaviva et al., 2022; Camposampiero et al., 2023; Min Tan and Motani, 2024). However, aggregate exact-match scores tell us only that models fail, not where they fail. Errors may arise from imperfect visual encoding of the input grid, failure to infer the underlying transformation, or failure to construct a globally consistent response even after partial reasoning success. As shown in Figure 1, a wrong ARC output may therefore reflect different underlying bottlenecks, while end-to-end evaluation cannot distinguish among them.

In this paper, we address this question from an information-processing perspective. Motivated by classic working-memory accounts of information processing (Baddeley and Hitch, 1974), we introduce an operational decomposition of inference into three stages: *encoding*, *transformation*, and *construction*. We instantiate this view with three controlled inference conditions: **Direct**, where a VLM predicts the answer directly from the task

input; **Transcribed-IR**, where the model first produces a text-grid intermediate representation (IR) and then reasons over it; and **Gold-IR**, where the corresponding gold text-grid representation is provided to remove transcription noise. These conditions provide interface-controlled diagnostic evidence over a coupled solving pipeline, allowing us to estimate which bottleneck is most consistent with the observed intervention pattern rather than attributing all failures to a single reasoning deficit.

Our findings show that VLM failures on ARC are stage-sensitive and benchmark-dependent. On **ARC-1**, the original ARC benchmark, Gold-IR improves GPT-5’s exact match from 22.9% to 68.6%. Because this comparison also changes the input representation, we additionally perform a matched image-conditioned control: replacing model-transcribed auxiliary text with gold text improves exact match from 8.6% to 20.0% (+11.4 points). Together, these results provide diagnostic evidence that encoding quality contributes substantially to ARC-1 failures under our evaluated interfaces. On **ARC-2**, a newer and harder successor benchmark, even the best Gold-IR result reaches only 4.2% exact match, which is consistent with a harder downstream transformation/composition bottleneck. To probe residual failures under more controlled conditions, we introduce **V-ARAOC**, a new diagnostic benchmark with atomic operation families. On V-ARAOC, even Gold-IR remains insufficient: for GPT-5, a lightweight shape-first constrained construction method raises Gold-IR EM from 6.8% to 86.3%. This strong intervention effect provides diagnostic evidence that output construction under shape and spatial constraints is a major source of residual error in this controlled benchmark. A $\text{Verify}@K$ analysis further shows that larger search improves candidate generation but leaves a selection gap. Together, these results suggest that ARC-style failures should be analyzed as benchmark-dependent information-processing bottlenecks rather than as a single reasoning deficit.

To summarize, our contributions are as follows:

Information-processing diagnosis. We decompose VLM-based ARC solving into *encoding*, *transformation*, and *construction*, and operationalize this view with three inference conditions: *Direct*, *Transcribed-IR*, and *Gold-IR*.

Benchmark-dependent diagnostic patterns.

We show that ARC failures do not follow a single diagnostic pattern: ARC-1 is highly

sensitive to encoding quality, whereas ARC-2 remains weak even after transcription noise is removed; expanded $\text{Verify}@K$ analysis further separates gains in candidate generation from persistent limits in selection.

V-ARAOC, intervention, and release. We introduce **V-ARAOC**¹, identify shape-sensitive construction as a residual failure mode, and show that a lightweight shape-first constrained construction method selectively repairs it.

2 Related Work

ARC as a benchmark for abstract reasoning.

The Abstraction and Reasoning Corpus (ARC) was introduced to evaluate fluid intelligence through few-shot grid transformations with minimal task-specific prior knowledge (Chollet, 2019). Early progress largely relied on DSL- and program-synthesis-based methods, including search-heavy ARC Kaggle systems and later approaches with richer abstractions and search control (Chollet et al., 2020; Xu et al., 2023; Lei et al., 2024). More recent work has explored learning-guided search and adaptation, while ARC-2 was introduced to reduce susceptibility to brute-force strategies and emphasize abstraction and compositional generalization (Chollet et al., 2025, 2026).

LLM/ VLM-based ARC solving. Recent studies apply LLMs and VLMs to ARC-style tasks, often by serializing 2D grids into text or related intermediate formats (Xu et al., 2024). These works show that performance is highly sensitive to representation, prompting, and communication format (Acquaviva et al., 2022; Camposampiero et al., 2023; Min Tan and Motani, 2024). Wu et al. (2025) further analyze ARC from the perspective of fluid intelligence and identify limitations such as skill composition, abstract input handling, and left-to-right decoding. However, the ARC studies reviewed above primarily evaluate end-to-end performance without explicitly isolating the stage responsible for failure.

Perception bottlenecks, multimodal reasoning, and diagnostic evaluation.

Recent work has begun to disentangle perception from reasoning in abstract visual benchmarks. Wang et al. (2026) argue that apparent reasoning failures can be substantially driven by perception bottlenecks; related

¹Project repository: <https://github.com/xuanZhou111/VARAOC>

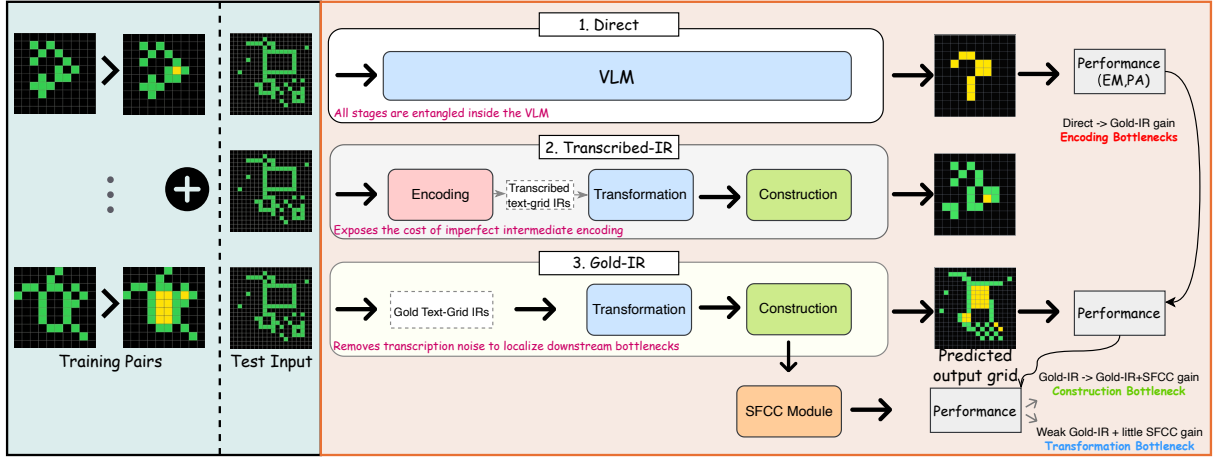


Figure 2: Overview of the information-processing decomposition and the three inference conditions. Direct entangles encoding, transformation, and construction inside an end-to-end VLM. Transcribed-IR exposes the cost of imperfect intermediate encoding, while Gold-IR removes transcription noise to localize downstream bottlenecks. Comparing Direct, Transcribed-IR, Gold-IR, and Gold-IR+SFCC yields intervention-based diagnostic signatures that help localize likely sources of failure across the coupled solving pipeline.

evidence comes from studies of nonverbal abstract reasoning (Ahrabian et al., 2024), visual deductive reasoning (Zhang et al., 2024), and reasoning-path fidelity (Bi et al., 2025). Our diagnostic perspective also follows broader work on behavioral testing and targeted evaluation in NLP (McCoy et al., 2019; Ribeiro et al., 2020; Kiela et al., 2021).

3 Information-Processing Decomposition

End-to-end ARC evaluation compounds multiple possible sources of failure. A low exact-match score may reflect inaccurate perception of the input grid, failure to infer the underlying rule, or failure to construct a valid output even after partial reasoning success. To separate these possibilities, we study ARC solving from an information-processing perspective and decompose inference into three stages: *encoding*, *transformation*, and *construction*. Figure 2 summarizes this decomposition and the three controlled inference conditions used throughout the paper.

3.1 Task Setting and Metrics

An ARC task consists of training input-output pairs $\{(x_i, y_i)\}_{i=1}^n$ and one or more test inputs. For evaluation, we treat each test input as a separate task-test case with the same training pairs. Let x^* denote the test input for a given evaluation case. Each grid is a 2D array over the color alphabet $\{0, \dots, 9\}$. The solver must infer the latent transformation from the training pairs and predict the output for x^* . While ARC is formally defined over digit grids,

standard evaluation renders them as colored images. Prior approaches have represented ARC grids using text or language-based intermediate representations (Xu et al., 2024; Camposampiero et al., 2023). We instead study VLMs that receive rendered images as input, using digit-grid representations only as intermediate or oracle signals.

Let the ground-truth test output be $y^* \in \{0, \dots, 9\}^{H \times W}$, where H and W denote the output height and width. A model prediction is parsed into either a valid rectangular grid $\hat{y}^* \in \{0, \dots, 9\}^{\hat{H} \times \hat{W}}$, where $\hat{H}$ and $\hat{W}$ denote the predicted height and width, or an invalid output.

We report **exact match (EM)** and **pixel accuracy (PA)**. EM is 1 iff the prediction is valid and matches y^* exactly in both shape and all cell values; otherwise EM is 0.

If the output is invalid, we define $PA = 0$. Otherwise, let $h = \min(H, \hat{H})$ and $w = \min(W, \hat{W})$. We compute PA on the ground-truth coordinate system with top-left anchoring and no translation search:

$$PA(y^*, \hat{y}^*) = \frac{1}{HW} \sum_{i=1}^h \sum_{j=1}^w \mathbf{1}[\hat{y}_{ij}^* = y_{ij}^*]. \quad (1)$$

Thus, PA is normalized by the ground-truth area, and positions outside the predicted grid are treated as incorrect.

We also report shape match, i.e., the fraction of evaluation cases for which the predicted grid has the correct height and width. In interface-sensitive

analyses, we additionally report parse rate to distinguish invalid outputs from cell-level prediction errors.

3.2 Three Inference Conditions

We instantiate the decomposition with three inference conditions that differ in how intermediate representations (IRs) are obtained. Our IR is a canonical text-grid format, i.e., a rectangular list-of-lists of integers in $\{0, \dots, 9\}$.

Direct. A VLM receives the ARC task as rendered grid images and predicts the final output directly. Unlike prior LLM-based approaches that serialize grids into digit-matrix text, the Direct condition preserves the visual input modality, so that failures may arise from any stage including visual encoding. This is the standard end-to-end setting for VLM-based ARC solving.

Transcribed-IR. The model first converts visual grids into text-grid IRs and then predicts the final output from those representations. This condition exposes the cost of imperfect intermediate encoding while keeping the same downstream transformation and construction interfaces.

Gold-IR. Gold IRs are provided in place of model-transcribed ones. This removes transcription noise and allows us to test whether performance remains limited by downstream transformation or construction. We use Gold-IR as a diagnostic isolation condition rather than as a strict upper bound for image-based solving.

Comparing these conditions provides interface-controlled diagnostic evidence over a coupled solving pipeline. Large Direct-to-Gold-IR gains indicate strong sensitivity to the quality and form of the encoded representation; weak Gold-IR performance indicates substantial residual difficulty after transcription noise is removed; and large gains from SFCC indicate strong sensitivity to the output-construction protocol. We interpret these patterns jointly with matched-modality, representation-sensitivity, and output-protocol analyses to identify the failure explanation most consistent with each benchmark. Together, these complementary interventions provide the basis for stage-sensitive failure attribution across benchmarks.

3.3 Diagnostic Probes

Beyond the three inference conditions, we use four targeted probes to test which stage is most likely to

limit performance.

Rendering and modality controls. We vary how grids are presented to the model, including whether the model receives image inputs, model-transcribed IR, self-ensemble transcriptions obtained by cell-wise majority vote over multiple independent reads, or gold IR. We also test sensitivity to basic rendering choices in a smaller-scale ablation. These controls probe the encoding stage and test whether better input representations improve downstream success.

Output protocol ablations. Under Gold-IR, we vary the output protocol while keeping the reasoning input fixed. Concretely, we compare alternative output protocols such as dense-grid and object-centric formats, and measure their effects on parse rate, exact match, and pixel accuracy. This probe tests whether downstream performance is sensitive to how the predicted solution must be externalized during construction.

Verify@K. We allow the model to generate K candidate solutions and select among them with an automatic verifier based on training-example consistency. Each candidate k contains predictions for all training inputs and one prediction for the test input. For each candidate k , we compute a training-consistency score $s_k = \sum_{i=1}^n \mathbf{1}[\hat{y}_i^{(k)} = y_i]$, where $\hat{y}_i^{(k)}$ is the k -th candidate’s predicted output for training input x_i . We then select $k^* = \arg \max_k s_k$, with ties broken by first occurrence, and take the final test prediction to be the test output associated with candidate k^* . This probe separates limitations in candidate generation from limitations in candidate selection: if larger K improves pass rates while verifier-selected accuracy improves much less, then correct candidates are being generated but not reliably chosen.

Shape-First Constrained Construction (SFCC).

SFCC explicitly constrains output construction by predicting the target shape before generating cell values. In all main SFCC results, we use **Predicted-SFCC**: the model first predicts the output height and width, and then generates cell values constrained to that predicted shape. We reserve **Oracle-SFCC** for a diagnostic upper-bound variant in which the gold output shape is provided. Further details on Oracle-SFCC are provided in Appendix A.5. This probe tests whether residual failures under Gold-IR can be reduced through

stronger construction control. On tasks where Gold-IR still leaves substantial error, large gains from Predicted-SFCC suggest a construction bottleneck rather than encoding noise. Further details on the supporting diagnostics are provided in Appendix A.4.

4 Benchmarks and Experimental Setup

4.1 ARC-1 and ARC-2

We evaluate on two ARC benchmarks that differ in difficulty. **ARC-1** is the original ARC benchmark (Chollet, 2019); We follow the fixed 100-task ARC training subset used in prior work (Wu et al., 2025); because several tasks contain multiple test instances, this yields 105 task-test cases, all of which are evaluated under every model and inference condition. We do not additionally balance the subset by operation family or difficulty. **ARC-2** is a newer and harder successor benchmark designed to reduce susceptibility to brute-force tactics and place greater emphasis on abstraction and compositional generalization (Chollet et al., 2026). We evaluate on all 120 tasks in the public ARC-AGI-2 evaluation split; because some tasks contain multiple test instances, this yields 167 task-test cases, all of which are evaluated under every model and inference condition. Using both benchmarks allows us to test whether the main bottleneck remains stable across benchmarks or shifts from encoding to downstream transformation and construction.

4.2 V-ARAO

To probe residual failures under more controlled conditions, we introduce **V-ARAO**, a diagnostic benchmark organized into atomic operation families. V-ARAO consists of 190 tasks across 19 families (10 tasks per family), covering both spatial primitives and higher-level atomic abstractions such as movement, copying, mirroring, scaling, cropping, counting, symmetry completion, and marker-guided translation. Further details on representative task examples and family-level analyses are provided in Appendix A.1, and full family definitions are given in Appendix A.2. Unlike ARC-1 and ARC-2, V-ARAO is intended for controlled diagnosis rather than broad benchmark comparison. More specifically, V-ARAO is designed to reduce broad task heterogeneity and isolate whether models can faithfully externalize the required output once rule complexity is controlled at the level of atomic operations. Its purpose is therefore not to

replace ARC as a general abstract-reasoning benchmark, but to provide a more targeted diagnostic setting in which residual failures can be interpreted more cleanly, especially when they arise from spatial construction rather than from upstream encoding noise or large-scale compositional ambiguity. V-ARAO is not derived from existing ARC instances and is not LLM-generated. Its operation families are manually specified, while individual instances and gold outputs are generated programmatically using deterministic family rules. We automatically verified that the evaluated set contains 190 matched tasks with no duplicate IDs, missing test inputs, or invalid target grids, and manually checked all 190 tasks for rule consistency, output correctness, ambiguity, and degeneracy. Full construction details are provided in Appendix A.3. We also conducted a human study with 10 participants (2 PhD, 4 Master’s, and 4 Bachelor’s students; none involved in the project), all of whom solved all 190 tasks correctly, with a median time of approximately 6 seconds per task. The instruction given to participants was: “For each ARC task, infer the transformation from the provided input–output examples and produce the output grid for the test input.” The human study serves as a validation of task clarity and human solvability. Human ease does not by itself imply that the benchmark lacks diagnostic value: prior studies likewise report high human solvability on ARC-style tasks while machine performance remains substantially lower (Johnson et al., 2021; LeGris et al., 2025). V-ARAO is deliberately designed around regular, atomic transformations, enabling controlled analysis of model failure signatures under human-unambiguous conditions. Its scope therefore centers on diagnostic analysis of specific reasoning and construction behaviors within ARC-style grid tasks.

4.3 Models and Implementation

We evaluate 11 proprietary and open-source VLMs: GPT-5, GPT-5-mini, and GPT-5-nano²; Gemini 2.5 (Comanici et al., 2025); GLM-4.5V (Team et al., 2026) and GLM-4.6V³; Qwen2.5-VL-72B, Qwen2.5-VL-32B, and Qwen2.5-VL-7B (Bai et al., 2025b); and Qwen3-VL-30B in both Instruct and Thinking modes (Bai et al., 2025a). Across all conditions, we use a canonical text-grid interme-

²Official GPT-5 system card: <https://openai.com/index/gpt-5-system-card/>. Accessed: 2026-05-25.

³Official Z.AI documentation for GLM-4.6V: <https://docs.z.ai/guides/vlm/glm-4.6v>. Accessed: 2026-05-25.

Model	ARC-1 (%)					ARC-2 (%)				
	Direct	Trans.-IR	Gold-IR	Δ	Shape	Direct	Trans.-IR	Gold-IR	Δ	Shape
Gemini 2.5	12.4	9.5	44.8	+32.4	93.3	0.0	0.0	0.6	+0.6	76.7
GLM-4.5V	2.9	1.9	11.4	+8.6	89.5	0.0	0.0	0.0	+0.0	67.1
GLM-4.6V	3.8	3.8	13.3	+9.5	85.7	0.0	0.0	0.0	+0.0	73.7
GPT-5	22.9	9.5	68.6	+45.7	96.2	1.2	0.0	4.2	+3.0	72.5
GPT-5-mini	3.8	1.9	66.7	+62.9	95.2	0.0	0.0	1.2	+1.2	52.1
GPT-5-nano	2.9	1.9	43.8	+40.9	89.5	0.0	0.0	0.0	+0.0	53.3
Qwen2.5-VL-72B	3.8	1.9	9.5	+5.7	87.6	0.0	0.0	0.0	+0.0	67.7
Qwen2.5-VL-32B	2.9	1.0	15.2	+12.4	86.7	0.0	0.0	0.0	+0.0	73.1
Qwen2.5-VL-7B	0.9	1.0	1.0	+0.0	92.4	0.0	0.0	0.0	+0.0	96.4
Qwen3-VL-30B (Instr.)	2.9	1.9	18.1	+15.2	88.6	0.0	0.0	0.0	+0.0	70.7
Qwen3-VL-30B (Think.)	4.8	3.8	42.9	+38.1	88.6	0.0	0.0	1.2	+1.2	66.5

Table 1: Main exact-match results across benchmarks. Δ = Gold-IR – Direct (percentage points), summarizing the gain from replacing the direct input interface with Gold-IR. Shape denotes Gold-IR shape match.

Model	G-		+SFCC		Δ EM
	EM	Shape	EM	Shape	
GPT-5	6.84	16.32	86.32	97.89	+79.48
GPT-5-mini	6.32	15.26	86.32	95.79	+80.00
Gemini 2.5	6.32	16.84	64.74	82.63	+58.42

Table 2: V-ARAOC results under SFCC in the Gold-IR setting. G-EM and G-Shape denote Gold-IR exact match and shape match, respectively; +SFCC EM and +SFCC Shape report the corresponding metrics after applying SFCC. All values are percentages. Δ EM denotes the exact-match gain from Gold-IR to Gold-IR+SFCC.

diate representation, a deterministic parser, and a unified evaluation pipeline. Unless otherwise stated, main results use a shared prompting protocol and deterministic decoding; Verify@ K uses stochastic decoding, and SFCC adds explicit shape-constrained construction. Additional prompt templates and model-specific configuration details are provided in Appendix A.6 and Appendix A.8.

5 Results and Analysis

We begin with a cross-benchmark comparison across inference conditions and then examine each benchmark in more detail. The central question is not only whether performance improves under stronger interfaces, but also which interventions matter on which benchmarks. This comparison provides the basis for the stage-sensitive interpretations developed in the following subsections.

5.1 Main Results Across Benchmarks

Overview. Table 1 summarizes exact-match results on ARC-1 and ARC-2 across the three inference conditions, and directly reports Δ (Gold-IR – Direct), which measures how performance changes

under a gold text-grid interface relative to direct image-based solving. Because this comparison changes both representation quality and input format, we complement it with matched-modality controls below. A consistent pattern emerges: the observed failure patterns are stage-sensitive and benchmark-dependent, with clear variation across benchmarks.

ARC-1: strong sensitivity to encoding quality.

On **ARC-1**, moving from *Direct* to *Gold-IR* yields large gains across strong models. For example, GPT-5 improves from 22.9% to 68.6% exact match (Δ = +45.7), GPT-5-mini from 3.8% to 66.7% (Δ = +62.9), and Gemini 2.5 from 12.4% to 44.8% (Δ = +32.4). These gains, together with high Gold-IR shape match, show that ARC-1 performance is highly sensitive to the quality and form of the encoded representation. The matched image-conditioned control in Section 5.2 provides more tightly controlled evidence for the contribution of transcription quality.

ARC-2: limited benefit from encoding control.

On **ARC-2**, by contrast, the same intervention is far less effective. Even under *Gold-IR*, exact match remains near zero for most models, with the best result reaching only 4.2% for GPT-5 (Δ = +3.0). For many models, Δ is effectively zero. This contrast suggests that ARC-2 is not explained primarily by noisy encoding; instead, the residual failures are more consistent with a substantially harder downstream transformation/composition bottleneck.

V-ARAOC: strong construction sensitivity.

The picture is different again on **V-ARAOC**. There, *Gold-IR* remains weak even though the benchmark is organized around atomic operation families

Data	Setting	EM	Shape	PA	Content@Shape
ARC-1	Direct	22.86	65.71	72.49	81.73
ARC-1	Trans.-IR	9.52	43.81	58.77	65.72
ARC-1	Gold-IR	68.57	96.19	91.04	93.10
ARC-2	Direct	1.20	44.31	37.02	35.96
ARC-2	Trans.-IR	0.00	45.64	32.15	22.97
ARC-2	Gold-IR	4.19	72.46	67.62	77.87

Table 3: Representative decomposition summary for GPT-5. Content@Shape denotes content accuracy conditioned on correct shape.

rather than full ARC-style compositional tasks. At the same time, explicit construction control can markedly improve performance (Table 2), showing that residual V-ARAOC errors are sensitive to the construction protocol once clean input representations are provided. Further details on model scale and reasoning modes are provided in Appendix A.7.

Summary. Taken together, the benchmarks exhibit distinct intervention signatures: ARC-1 is strongly affected by encoding quality, ARC-2 remains difficult after encoding noise is reduced, and V-ARAOC responds strongly to constrained construction. We interpret these patterns jointly across complementary interventions to support stage-sensitive diagnosis of the dominant failure sources associated with each benchmark. Further details on pixel accuracy and bootstrap confidence intervals are provided in Appendix A.10.

5.2 ARC-1: Diagnostic Evidence for Encoding Sensitivity

Gold-IR gains. ARC-1 provides the clearest evidence that performance is sensitive to encoding quality. Across strong models, replacing model-read intermediate representations with Gold-IR yields large exact-match gains (Table 1), showing that a substantial fraction of the observed performance gap can be recovered by supplying a cleaner intermediate representation.

Matched modality control. The headline Direct-to-Gold-IR comparison changes both representation quality and input modality. We therefore perform a more tightly matched image-conditioned control in which both conditions retain the image input and differ only in whether the auxiliary text-grid representation is model-transcribed or gold. On the matched 70-case overlap with parseable GPT-5 transcriptions, replacing transcribed text

Setting	EM	Shape	PA
Image + transcribed text	8.57	42.86	57.70
Image + gold text	20.00	90.00	77.82

Table 4: ARC-1 modality control on GPT-5. Replacing model-transcribed text with gold text substantially improves exact match, shape match, and pixel accuracy, providing matched-interface evidence that transcription quality contributes substantially to ARC-1 performance. All values are percentages.

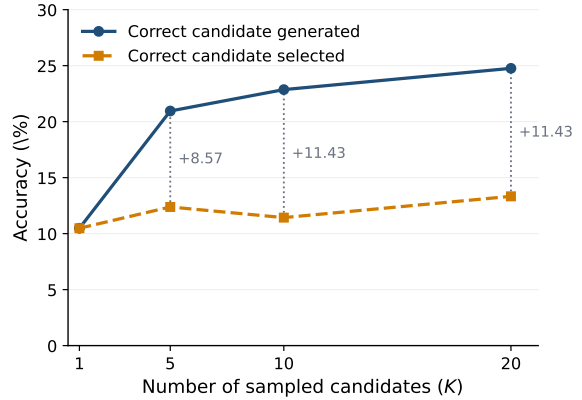


Figure 3: Verify@ K on ARC-1 for GPT-5-mini. Larger K substantially improves the chance that a correct candidate is generated (Pass@ K / OracleSelect@ K), but verifier-selected accuracy improves much less, leaving a persistent selection gap.

with gold text improves EM from 8.6% to 20.0% (+11.4 points), shape match from 42.9% to 90.0%, and PA from 57.7% to 77.8%. This controlled contrast reduces the modality confound and provides stronger evidence that transcription quality materially contributes to ARC-1 failures.

Verify@ K and selection gap. At the same time, larger search does not fully close the remaining gap. Figure 3 shows that increasing K substantially raises the probability that a correct candidate is generated, but verifier-selected accuracy improves much less. For GPT-5-mini, Pass@ K rises from 10.48% at $K = 1$ to 24.76% at $K = 20$, whereas verifier-selected accuracy increases only from 10.48% to 13.33%, leaving an 11.43-point selection gap at $K = 20$. Collectively, these results suggest that ARC-1 shows strong evidence of noisy encoding, with additional downstream losses from imperfect candidate selection once multiple candidates are available. Further details on the output-majority clustering sanity check are provided in Appendix A.4.

Rendering sensitivity. Rendering ablations further confirm encoding sensitivity: changing the rendering from 16-pixel cells without gridlines to 32-pixel cells with gridlines raises stage-1 pixel accuracy from 58.5% to 87.5% in a smaller-scale ARC-1 transcription probe. Further details are provided in Appendix A.4.

5.3 ARC-2: Residual Difficulty after Encoding Control

Marginal Gold-IR gains. ARC-2 exhibits a different failure profile from ARC-1. Replacing model-read representations with Gold-IR yields only marginal exact-match gains (Table 1): even the best Gold-IR result reaches only 4.2%, and for many models $\Delta(\text{Gold-IR} - \text{Direct})$ is effectively zero. This suggests that noisy encoding alone does not explain ARC-2 failures.

Partial structure without exact execution. A complementary auxiliary-text control points in the same direction. On the same ARC-1 subset, replacing raw transcribed text with majority-voted transcriptions raises shape match from 40.00% to 87.14% and EM from 7.14% to 8.57%, although it still remains below gold text. Appendix A.4 provides more details. This suggests that self-ensemble transcription can partially repair structural encoding errors, but does not eliminate downstream content errors. At the same time, Gold-IR performance is not uniformly poor in all respects. Several models achieve non-trivial Gold-IR shape match and moderate pixel accuracy despite near-zero exact match. For example, GPT-5 reaches 67.6% Gold-IR PA and 72.5% Gold-IR shape match while remaining at 4.2% exact match, and Gemini 2.5 reaches 58.9% PA and 76.7% shape match while remaining at 0.6% exact match. This dissociation suggests that models often capture partial structure or local content, but fail to execute the full transformation precisely enough to solve the task exactly. GPT-5 Gold-IR EM remains 4.2% for both shape-changing and same-shape ARC-2 tasks, suggesting that ARC-2 errors are not explained solely by output-canvas changes. Further details on this complementary shape-change analysis are provided in Appendix A.5. In this sense, ARC-2 does not reflect a complete failure to represent the task at all, but rather a failure to carry partial structural understanding through the full transformation process with sufficient precision and consistency. The near-zero exact-match results therefore mask a

more nuanced error profile: models often preserve fragments of the intended structure, yet still fall short of the exact compositional execution required for success.

Limited effect of SFCC on ARC-2. Construction-oriented repair is also ineffective on ARC-2. SFCC yields only mixed or negligible gains: GPT-5 improves from 0.0% to 1.8% under Transcribed-IR, but drops from 4.19% to 3.59% under Gold-IR, while GPT-5-mini and Gemini 2.5 show no exact-match gains. Taken together, these results are more consistent with a harder transformation/composition bottleneck than with an encoding-only or construction-only explanation. Further details are provided in Appendix A.9.

Output-protocol sensitivity. Residual ARC-2 performance also depends on how the solution is externalized. Under fixed Gold-IR inputs, changing only the output protocol from a dense grid to an object-style representation changes Gemini 2.5 parse rate from 89.2% to 31.1% and PA from 57.6% to 37.1%; for Qwen2.5-VL-72B, PA changes from 53.1% to 49.6%. These effects indicate that the output interface contributes to the residual downstream difficulty on ARC-2. Across both protocols, exact-match performance remains near zero, consistent with substantial remaining transformation and composition difficulty. Further details are provided in Appendix A.4.

5.4 V-ARAOC: Construction Sensitivity under Spatial Constraints

Gold-IR remains insufficient. V-ARAOC isolates a different failure regime from ARC-1 and ARC-2. Although the benchmark is organized around atomic operation families rather than full ARC-style compositional tasks, *Gold-IR* remains insufficient: all three evaluated models stay below 7% exact match under Gold-IR, and Gold-IR shape match also remains low (Table 2). This pattern suggests removing encoding noise alone does not resolve the dominant residual failures on V-ARAOC.

Family-level heterogeneity. Family-level analysis further indicates that these failures are highly uneven rather than uniformly distributed: some counting-style families are relatively easy, whereas several geometry- and shape-sensitive families remain difficult even under Gold-IR. Table 5 reports representative results across five V-

Family	GPT-5-mini		Gemini 2.5		GPT-5	
	EM	Shape	EM	Shape	EM	Shape
count_holes	1.0	1.0	1.0	1.0	1.0	1.0
count_objects	0.2	1.0	0.2	1.0	0.2	1.0
min_obj_crop	0.0	0.1	0.0	0.0	0.1	0.1
change_color	0.0	0.0	0.0	0.0	0.0	0.0
mirror_left	0.0	0.5	0.0	0.6	0.0	0.3

Table 5: Representative V-ARAOC Gold-IR family-level results across models. The pattern is highly non-uniform: `count_holes` is essentially solved, `count_objects` is often shape-correct but content-wrong, and geometry- or construction-sensitive families remain difficult even with clean input representations. `min_obj_crop` denotes `select_min_object_crop`.

ARAOC families. Notably, `count_holes` is essentially solved, `count_objects` is often shape-correct but content-wrong, and families such as `select_min_object_crop`, `change_color`, and `mirror_left` remain difficult despite clean input representations. This pattern motivates testing whether stronger control over structured output construction can recover these errors, particularly for families requiring globally consistent shape and layout.

SFCC reduces construction-sensitive errors. This interpretation is strongly supported by Shape-First Constrained Construction (SFCC). As shown in Table 2, explicitly predicting the target shape before generating cell values yields dramatic improvements under Gold-IR. For GPT-5, exact match rises from 6.8% to 86.3%, and shape match rises from 16.3% to 97.9%; similarly large gains appear for GPT-5-mini and Gemini 2.5. These results provide strong diagnostic evidence that the residual bottleneck on V-ARAOC is more consistent with construction difficulty than with encoding noise.

Predicted versus oracle shape. The main SFCC results use predicted rather than gold output dimensions: Predicted-SFCC first predicts the target height and width and then constrains generation to that predicted shape. Under Gold-IR, GPT-5 reaches 86.3% EM with Predicted-SFCC and 88.4% with Oracle-SFCC, while GPT-5-mini reaches 86.3% and 89.5%, respectively. The small gap between predicted and oracle shape conditions indicates that most of the SFCC gain is retained when the target dimensions are predicted by the model. This pattern further supports the sensitivity of V-ARAOC performance to structured output con-

struction. Further details on the Predicted-SFCC and Oracle-SFCC comparison are provided in Appendix A.5.

ARC-1 construction sanity check. To test whether the same intervention can repair genuine ARC construction errors, we additionally examine all five GPT-5-mini Gold-IR cases among the 105 ARC-1 task-test cases where the baseline output was parseable but had the wrong shape. Predicted-SFCC recovers the correct shape in 2/5 cases (0% $\rightarrow$ 40% shape match) and raises mean PA from 42.2% to 46.0%, while EM remains 0%. Oracle-SFCC reaches 100% shape match but only 20% EM. Because these five cases exhaust the observed baseline shape errors rather than forming a random sample, this is a targeted sanity check rather than an estimate of overall ARC-1 improvement. The result illustrates that shape control can repair some construction errors without eliminating the need for correct transformation and composition.

Transcribed-IR vs. Gold-IR contrast. SFCC also improves Transcribed-IR performance, but the gains are much smaller than under Gold-IR (e.g., GPT-5: 5.79% $\rightarrow$ 24.74% vs. 6.84% $\rightarrow$ 86.32%). This contrast is consistent with our stage-based interpretation: construction repair is partly masked by upstream transcription errors under noisy inputs, but becomes much more effective once encoding noise is removed. Family-level heatmaps further show that SFCC gains concentrate in shape-sensitive families rather than being uniformly distributed. Appendix A.1 provides further details.

6 Conclusion

We analyzed VLM failures on ARC-style abstract reasoning through an encoding–transformation–construction decomposition using Direct, Transcribed-IR, Gold-IR, and targeted interventions. The results reveal benchmark-dependent failure signatures: ARC-1 is strongly sensitive to encoding quality, ARC-2 remains difficult after transcription noise is removed, and V-ARAOC is highly sensitive to output-construction constraints. $\text{Verify}@K$ further exposes a gap between generating correct candidates and selecting them reliably. Overall, similar end-to-end failures can arise from different bottlenecks, motivating stage-sensitive diagnosis beyond aggregate accuracy.

Limitations

Gold-IR changes the input modality. The Gold-IR intervention removes transcription noise by providing gold intermediate representations as text, but this also changes the conditioning signal compared to image-based solving. We therefore treat Gold-IR as a diagnostic isolation condition rather than a strict upper bound for image-based VLM solving.

Dependence on the chosen intermediate representation. Our analysis is built around a canonical text-grid intermediate representation and a standardized writer to maximize comparability across models and settings. However, different intermediate representations or output protocols can shift both validity and accuracy, so some observed bottlenecks may depend in part on this design choice.

Hosted-model reproducibility. Because several evaluated models are accessed through hosted APIs whose implementations may evolve over time, exact model-specific numerical results may vary across future reruns. We provide model identifiers, decoding settings, prompts, and evaluation code to support auditability.

Limited verifier strength and search budget. Our multi-candidate generate-and-rerank diagnostic verifies candidates only through training-example consistency, which is an imperfect proxy for true rule correctness. Larger search budgets or stronger verifiers could change the balance between generation and selection errors.

Benchmark and coverage limits. ARC-1 results use 105 task-test cases derived from the fixed 100-task subset described in Section 4.1, while ARC-2 uses all 120 tasks in the public evaluation split, expanded into 167 task-test cases. Metrics and bootstrap confidence intervals are computed at the case level, so tasks with multiple test instances contribute multiple evaluation cases. Not every diagnostic is run on every model due to cost and throughput constraints. In addition, while V-ARAOC provides controlled family-level diagnosis, broader model coverage and more extensive human evaluation would strengthen the generality of the findings. Our claims are limited to ARC-style grid-based visual-symbolic reasoning; extending the diagnostic framework to natural images, diagrams, or broader multimodal benchmarks would require

domain-specific intermediate representations and construction constraints.

Pixel accuracy under shape mismatch. PA compares predictions against the ground truth using fixed top-left anchoring without offset search. When the predicted grid differs in size from the ground truth, only the top-left overlap region is evaluated. This is a reasonable default for ARC, where outputs have a fixed coordinate system, but it means that a prediction with correct content at a shifted position would receive reduced credit. We mitigate this by reporting shape match separately to distinguish canvas-size errors from content errors.

Ethical Considerations

This paper presents a diagnostic study of vision-language ARC solvers and introduces V-ARAOC as a controlled benchmark of atomic grid transformations. The work is methodological: we analyze failure modes under controlled interface substitutions and perturbations rather than building a system for real-world deployment. Because all tasks are synthetic grid puzzles, the direct societal risks are limited, but several considerations remain relevant.

Data, licensing, and release. We evaluate on ARC-AGI-1/2 and V-ARAOC. All inputs are abstract grid puzzles and do not contain personal, biometric, or user-generated content. We follow the terms of use of each benchmark and do not redistribute restricted task instances.

Privacy and harmful content. The study processes only synthetic integer-valued grids. No personal data is collected, inferred, or stored, and the tasks do not elicit natural-language content about identifiable individuals or real-world events. As a result, privacy and harmful-content risks are minimal.

Benchmark gaming and overfitting. Although our goal is failure attribution, diagnostic findings such as interface choices or construction constraints could be used to optimize narrowly for a benchmark without improving general reasoning. This risk is especially relevant for V-ARAOC: if atomic-family tasks are used directly for training, the benchmark may lose part of its diagnostic value. We therefore position V-ARAOC as an evaluation resource rather than a training target, and recommend maintaining clear separation between any

released training material and held-out diagnostic splits.

Use of AI assistants. AI assistants were used primarily for language editing and manuscript proof-reading during manuscript preparation, e.g., the camera-ready phase. All scientific content and final revisions were reviewed and approved by the authors.

Participant consent. Participants were informed that their responses would be used for research purposes and provided informed consent before participation.

Human study compensation. The 10 participants in the human evaluation (Section 4.2) were compensated at a rate that meets or exceeds the local standard for research participation. The study involved only abstract grid puzzles with no personal or sensitive content, and no personally identifiable information was collected.

Acknowledgments

This research was supported by the China Scholarship Council (CSC) under grant numbers 202308070070 (Xuan Zhou) and 202308070072 (Xuefeng Wei).

References

- Samuel Acquaviva, Yewen Pu, Marta Kryven, Theodoros Sechopoulos, Catherine Wong, Gabrielle E Ecanow, Maxwell Nye, Michael Henry Tessler, and Joshua B. Tenenbaum. 2022. Communicating natural programs to humans and machines. In *Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS '22*, Red Hook, NY, USA. Curran Associates Inc.
- Kian Ahrabian, Zhivar Sourati, Kexuan Sun, Jiarui Zhang, Yifan Jiang, Fred Morstatter, and Jay Pujara. 2024. *The curious case of nonverbal abstract reasoning with multi-modal large language models*. *Preprint*, arXiv:2401.12117.
- Alan Baddeley and Graham James Hitch. 1974. *Working memory*, volume 8, pages 47–90. Academic Press, United States.
- Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, and 45 others. 2025a. *Qwen3-vl technical report*. *Preprint*, arXiv:2511.21631.
- Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibao 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, and 8 others. 2025b. *Qwen2.5-vl technical report*. *Preprint*, arXiv:2502.13923.
- Jing Bi, Junjia Guo, Susan Liang, Guangyu Sun, Luchuan Song, Yunlong Tang, Jinxi He, Jiarui Wu, Ali Vosoughi, Chen Chen, and Chenliang Xu. 2025. *Verify: A benchmark of visual explanation and reasoning for investigating multimodal reasoning fidelity*. *Preprint*, arXiv:2503.11557.
- Giacomo Camposampiero, Loïc Houmard, Benjamin Estermann, Joël Mathys, and Roger Wattenhofer. 2023. *Abstract visual reasoning enabled by language*. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2643–2647.
- Raymond Bernard Cattell. 1963. *Theory of fluid and crystallized intelligence: A critical experiment*. *Journal of Educational Psychology*, 54:1–22.
- Francois Chollet, Mike Knoop, Gregory Kamradt, and Bryan Landers. 2025. *Arc prize 2024: Technical report*. *Preprint*, arXiv:2412.04604.
- Francois Chollet, Mike Knoop, Gregory Kamradt, Bryan Landers, and Henry Pinkard. 2026. *Arc-agi-2: A new challenge for frontier ai reasoning systems*. *Preprint*, arXiv:2505.11831.
- François Chollet. 2019. *On the measure of intelligence*. *Preprint*, arXiv:1911.01547.
- François Chollet, Katherine Tong, Walter Reade, and Julia Elliott. 2020. Abstraction and reasoning challenge. <https://kaggle.com/competitions/abstraction-and-reasoning-challenge>. Kaggle.
- Gheorghe Comanici, Eric Bieber, Mike Schaeckermann, Ice Pasupat, Naveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, Luke Marris, Sam Petulla, Colin Gaffney, Asaf Aharoni, Nathan Lintz, Tiago Cardal Pais, Henrik Jacobsson, Idan Szpektor, Nan-Jiang Jiang, and 3416 others. 2025. *Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities*. *Preprint*, arXiv:2507.06261.
- Aysja Johnson, Wai Keen Vong, Brenden M. Lake, and Todd M. Gureckis. 2021. *Fast and flexible: Human program induction in abstract reasoning tasks*. *Preprint*, arXiv:2103.05823.
- Douwe Kiela, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, Zhiyi Ma, Tristan Thrush, Sebastian Riedel, Zeerak Waseem, Pontus Stenetorp, Robin Jia, Mohit Bansal, Christopher Potts, and Adina Williams. 2021.

- Dynabench: Rethinking benchmarking in NLP.** In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4110–4124, Online. Association for Computational Linguistics.
- Solim LeGris, Wai Keen Vong, Brenden M. Lake, and Todd M. Gureckis. 2025. **A comprehensive behavioral dataset for the abstraction and reasoning corpus.** *Scientific Data*, 12(1):1380.
- Chao Lei, Nir Lipovetzky, and Krista A. Ehinger. 2024. **Generalized planning for the abstraction and reasoning corpus.** In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI’24/IAAI’24/EAAI’24. AAAI Press.
- R. Thomas McCoy, Ellie Pavlick, and Tal Linzen. 2019. **Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference.** In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.
- John Chong Min Tan and Mehul Motani. 2024. **Llms as a system of multiple expert agents: An approach to solve the abstraction and reasoning corpus (arc) challenge.** In *2024 IEEE Conference on Artificial Intelligence (CAI)*, pages 782–787.
- Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. **Beyond accuracy: Behavioral testing of NLP models with CheckList.** In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.
- V Team, Wenyi Hong, Wenmeng Yu, Xiaotao Gu, Guo Wang, Guobing Gan, Haomiao Tang, Jiale Cheng, Ji Qi, Junhui Ji, Lihang Pan, Shuaiqi Duan, Weihang Wang, Yan Wang, Yean Cheng, Zehai He, Zhe Su, Zhen Yang, Ziyang Pan, and 74 others. 2026. **Glm-4.5v and glm-4.1v-thinking: Towards versatile multimodal reasoning with scalable reinforcement learning.** *Preprint*, arXiv:2507.01006.
- Xinhe Wang, Jin Huang, Xingjian Zhang, Tianhao Wang, and Jiaqi W. Ma. 2026. **Your reasoning benchmark may not test reasoning: Revealing perception bottleneck in abstract reasoning benchmarks.** In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 18113–18124, San Diego, California, United States. Association for Computational Linguistics.
- Junjie Wu, Mo Yu, Lemao Liu, Dit-Yan Yeung, and Jie Zhou. 2025. **Understanding LLMs’ fluid intelligence deficiency: An analysis of the ARC task.** In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8339–8360, Albuquerque, New Mexico. Association for Computational Linguistics.
- Yudong Xu, Elias B. Khalil, and Scott Sanner. 2023. **Graphs, constraints, and search for the abstraction and reasoning corpus.** In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI’23/IAAI’23/EAAI’23. AAAI Press.
- Yudong Xu, Wenhao Li, Pashootan Vaezipoor, Scott Sanner, and Elias B. Khalil. 2024. **Llms and the abstraction and reasoning corpus: Successes, failures, and the importance of object-based representations.** *Preprint*, arXiv:2305.18354.
- Yizhe Zhang, He Bai, Ruixiang Zhang, Jiatao Gu, Shuangfei Zhai, Josh Susskind, and Navdeep Jaitly. 2024. **How far are we from intelligent visual deductive reasoning?** *Preprint*, arXiv:2403.04732.

A Appendix

A.1 Representative V-ARAO Task Examples

Figure 4 shows representative input-output examples from V-ARAO. Each family is designed to emphasize a single atomic transformation underlying ARC-style reasoning.

A.2 V-ARAO Family Definitions

V-ARAO is a visual diagnostic dataset organized into operation families, where each instance is constructed to emphasize a single atomic transformation underlying ARC-style reasoning. Table 6 summarizes the families with concise input/output templates and the intended skill probed.

Figures 5 and 6 summarize the family-level Gold-IR baseline patterns and the corresponding gains from SFCC.

A.3 V-ARAO Construction and Quality Control

V-ARAO is not derived from existing ARC task instances and is not LLM-generated. The operation taxonomy and family rules are manually designed, while concrete task instances are programmatically instantiated under explicit family specifications. Each family specification defines the input schema, output rule, valid grid-size range, color constraints, object-placement constraints, target-object selection rule, gold-output computation rule,

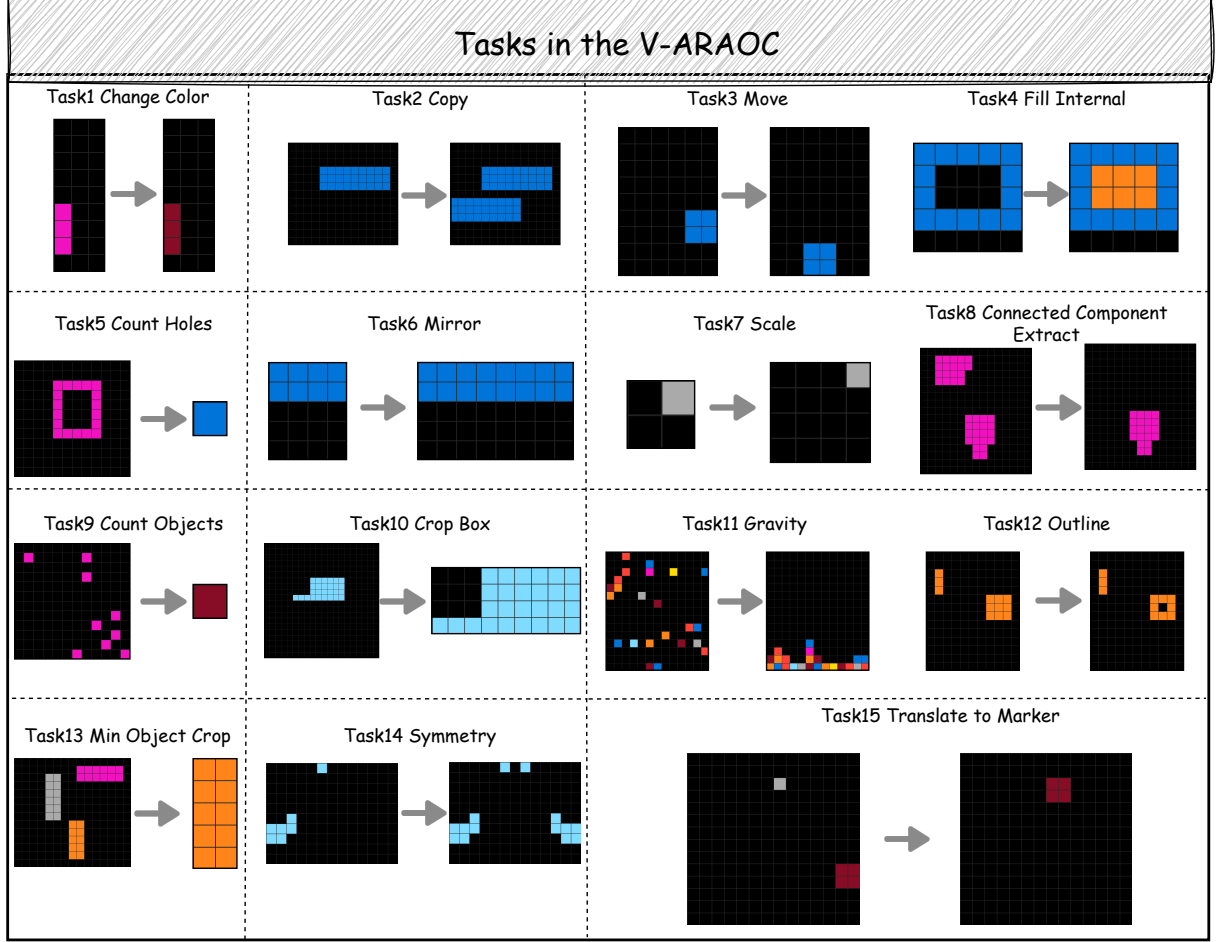


Figure 4: **V-ARAOC task families.** Each cell shows a representative input-output example for one atomic operation family, where the arrow denotes the intended transformation. Families cover spatial primitives (e.g., *move*, *copy*, *mirror*, *scale*), counting/selection (*count holes*, *count objects*, *component extract*, *crop*), dynamics (*gravity*), and higher-level abstractions (*symmetry*, *translate to marker*). Full family definitions and additional variants are provided in Appendix Table 6.

and rejection criteria for ambiguous or degenerate cases.

Each task contains three training demonstrations and one held-out test instance. Gold outputs are produced by deterministic family transformation rules rather than by model annotation. We applied automatic quality-control checks to the evaluated set: the metadata contains 190 rows, evaluation files contain 190 matched rows, and there are 0 duplicate task IDs, 0 missing evaluation rows, 0 missing test-input images, and 0 invalid target grids. We also manually verified all 190 final tasks for family-rule consistency, gold-output correctness, ambiguity, degeneracy, and whether the demonstrations support the intended latent rule.

A.4 Supporting Interface Diagnostics

This section reports additional interface-sensitive analyses that support the main stage-based conclusions but are not central enough to include in the main text.

Rendering ablations. Table 7 summarizes the rendering ablation for stage-1 transcription. We vary basic rendering choices, including cell size and the presence of gridlines, and measure their effects on parse rate and stage-1 pixel accuracy.

Output-protocol ablations. Table 8 summarizes the output-protocol comparison under fixed Gold-IR inputs. The two output interfaces differ substantially in parse rate and downstream accuracy for Gemini 2.5, while the corresponding differences are smaller for Qwen2.5-72B. Exact-match performance remains near zero across both protocols.

Family	Input	Output	Skill probed
cc_extract	Multiple same-color connected components	Keep a target component; clear the rest	Connected-component segmentation and selection
change_color	One or more objects/regions	Recolor a specified target; geometry unchanged	Target localization and color mapping
copy	A source object is present	Duplicate the object at a rule-specified location (optionally keep original)	Object extraction and duplication
copy_small	Smaller/sparser objects (often noise-sensitive)	Copy as in copy, but under finer granularity	Fine-grained recognition and precise copying
count_holes	Closed shapes with holes	Encode the number of holes (e.g., bars/blocks)	Topological “hole” counting
count_objects	Multiple discrete objects	Encode the object count with a fixed visual code	Instance segmentation and counting
crop_bbox	A target object	Crop the tight bounding-box subgrid	Localization and cropping-as-output
fill_internal	A closed boundary/outline	Fill the interior region with a specified (or rule-based) color	Interior-region identification and filling
gravity	Objects/pixels on a canvas	Let objects “fall” and stack until obstacles/bottom	Rule-driven spatial dynamics
mirror	A shape (and an implied axis)	Mirror across a horizontal/vertical axis	Symmetry mapping
mirror_left	Left half provides reference	Complete/mirror left-to-right	Oriented symmetry completion
mirror_right	Right half provides reference	Complete/mirror right-to-left	Oriented symmetry completion
move	A target object (often with cues/markers)	Translate the object to a rule-specified location	Localization and conditional translation
move_small	Small objects and/or subtle shifts	Move as in move with tighter precision	Precise read and precise write
outline	Solid object(s)	Keep only boundary pixels	Contour extraction / basic morphology
scale	An object with a scale rule	Resize (often integer factor) and place by rule	Discrete resampling and scaling
select_min_object_crop	Multiple objects of different sizes	Select the smallest object and output its crop	Segmentation → comparison → cropping
symmetry_complete	Partially symmetric structure	Fill missing parts to satisfy symmetry	Global-constraint completion
translate_to_marker	A source block and a marker point	Translate the block to align with the marker (marker removed)	Relational reasoning and exact translation

Table 6: V-ARAOC operation families and their intended skills.

Cell	Gridline	Parse	Stage1 PA
16	0	0.090	0.585
24	0	0.140	0.753
32	1	0.180	0.875

Table 7: Rendering ablation for stage-1 transcription (GPT-5-mini, ARC-1 subset). Parse denotes the fraction of valid transcriptions, and Stage1 PA denotes transcription pixel accuracy.

Model	Protocol	Parse	EM	PA
Gemini 2.5	grid	0.892	0.000	0.576
Gemini 2.5	object	0.311	0.006	0.371
Qwen2.5-72B	grid	0.970	0.000	0.531
Qwen2.5-72B	object	0.994	0.000	0.496

Table 8: Output-protocol ablation under Gold-IR inputs on ARC-2. The protocol alone can materially change parse rate and downstream accuracy, especially for Gemini 2.5.

Verify@ K numeric summary. Table 9 gives the numeric counterpart of the main-text Verify@ K figure after the 32k-token rerun and reparse pass.

Selector sanity check for Verify@ K . Table 10 summarizes the selector sanity check. Following the reviewer suggestion, we additionally evalu-

ated an output-majority clustering selector under the same GPT-5-mini ARC-1 Verify@ K protocol. The clustering selector improves over training-consistency selection at intermediate K , but it still does not close the gap to OracleSelect@ K .

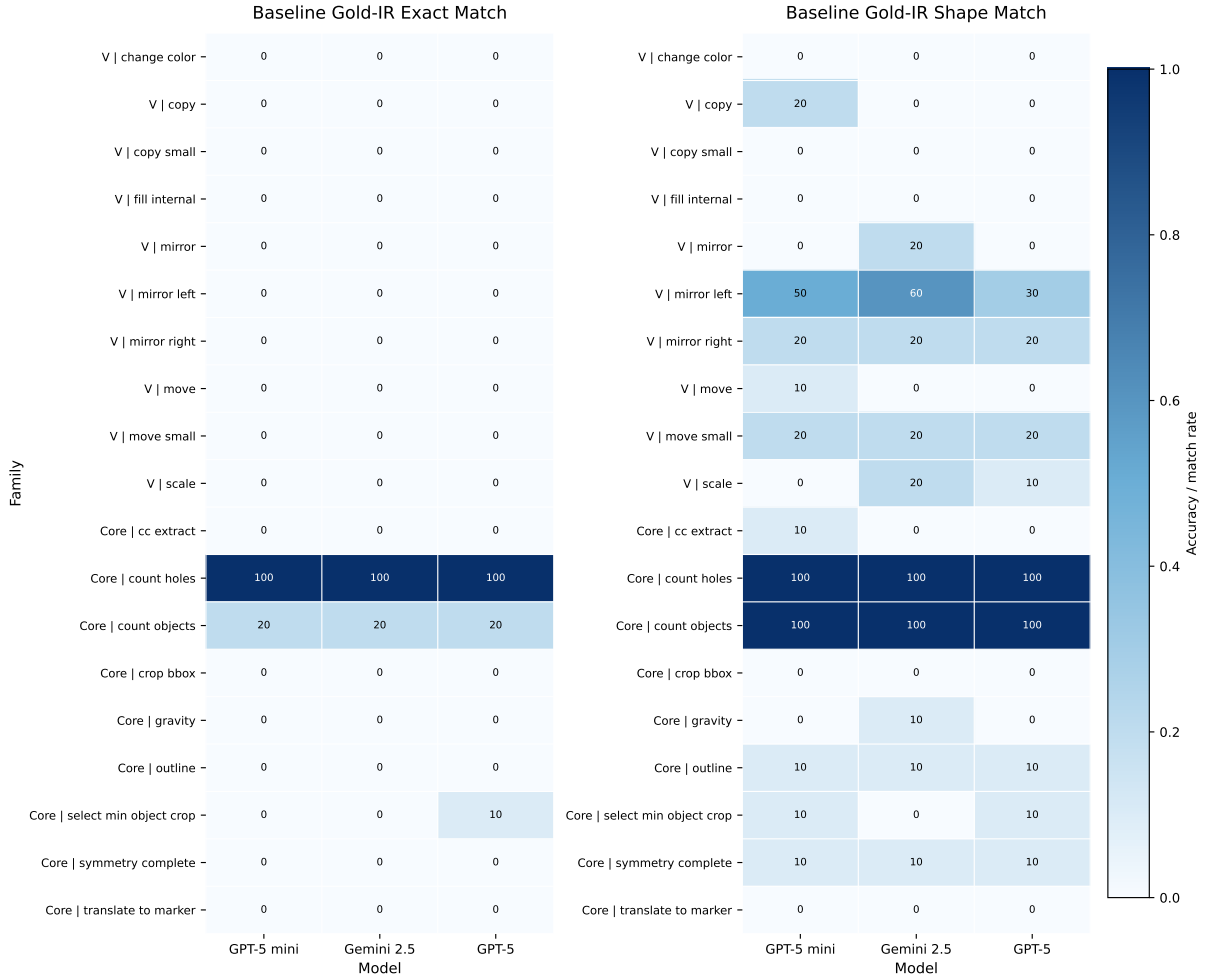


Figure 5: **Baseline Gold-IR family-level performance on V-ARAOC.** Left: exact match (EM). Right: shape match. Performance is highly uneven across families. count_holes is solved reliably by all three models, whereas many geometry- and construction-sensitive families remain near zero EM even under Gold-IR. The contrast between EM and shape match further shows that some families, such as count_objects and mirror_left, often achieve partially correct output extent without solving the full task exactly.

K	Pass@ K	OracleSel@ K	VerifierSel@ K
1	10.48	10.48	10.48
5	20.95	20.95	12.38
10	22.86	22.86	11.43
20	24.76	24.76	13.33

Table 9: Verify@ K on ARC-1 for GPT-5-mini. Larger K clearly improves candidate generation, but verifier-based selection remains much weaker than oracle selection.

Majority-vote transcription as an auxiliary hint.

Table 11 summarizes this auxiliary transcription analysis on the same 70 ARC-1 cases using GPT-5-mini as the solver. The single-transcription condition uses GPT-5 transcriptions, whereas the voted condition uses cell-wise majority voting over five GPT-5-mini reads.

K	OracleSel@ K	TrainConsist.	OutputMajorityCluster
5	20.0	11.4	14.3
10	23.8	10.5	16.2
20	25.7	13.3	13.3

Table 10: Selector sanity check for Verify@ K on ARC-1 with GPT-5-mini. Output-majority clustering improves over training-consistency selection at intermediate K , but does not close the gap to OracleSel@ K .

A.5 Full SFCC Results

Predicted-SFCC versus Oracle-SFCC. Table 12 compares Predicted-SFCC and Oracle-SFCC on V-ARAOC under Gold-IR. All main SFCC results use Predicted-SFCC, where the model predicts the output height and width before generating cell values. Oracle-SFCC provides the

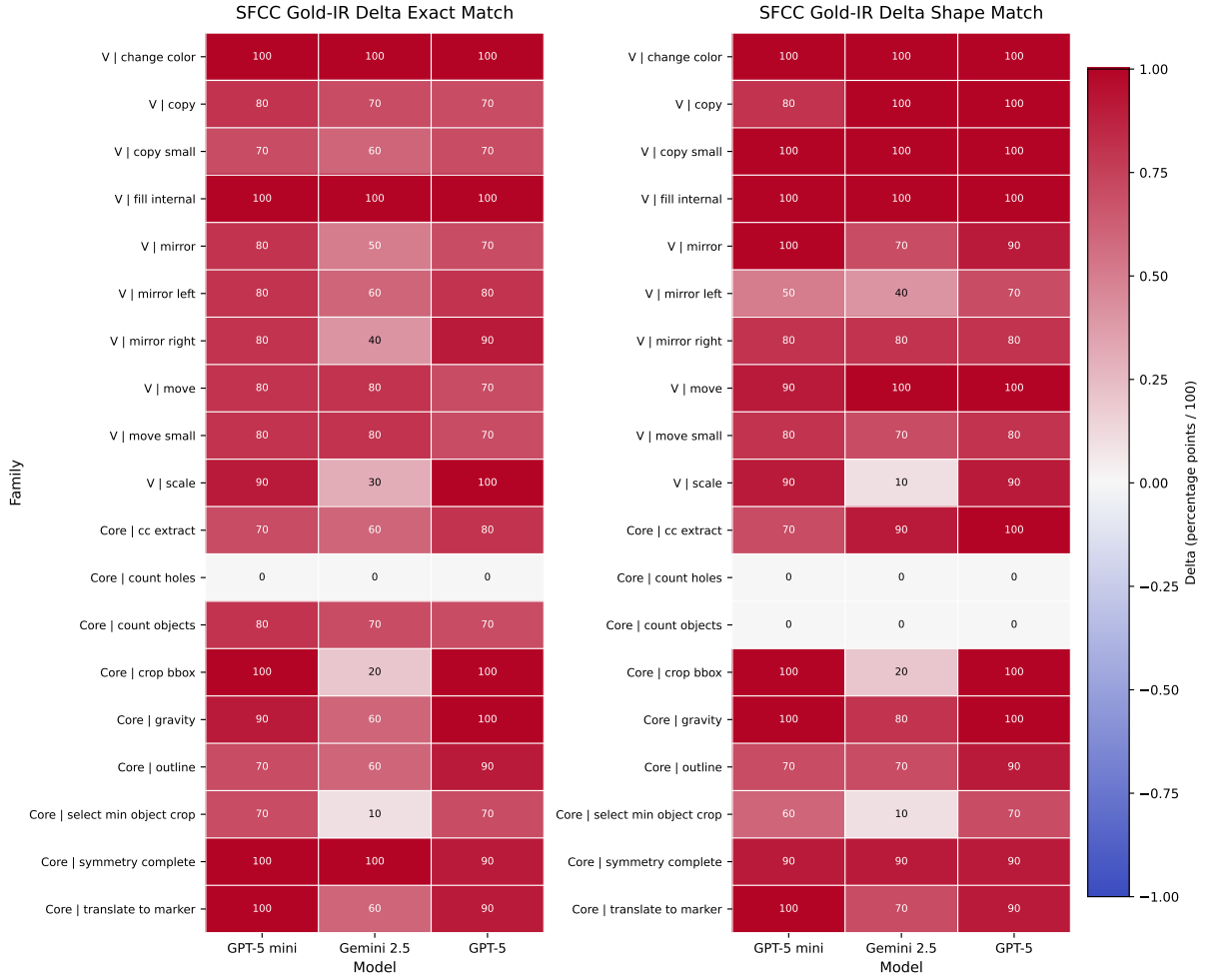


Figure 6: **Family-level gains from SFCC under Gold-IR on V-ARAOC.** Left: gain in exact match. Right: gain in shape match. Gains are computed relative to the Gold-IR baseline. SFCC improvements are concentrated in shape-sensitive families rather than being uniformly distributed across all operations. Families such as change_color, fill_internal, crop_bbox, gravity, and symmetry_complete show especially large gains, whereas count_holes shows essentially no improvement because it is already solved at baseline.

Setting	EM	Parse	Shape	PA
Gold	12.86	98.57	85.71	70.73
Transcribed	7.14	97.14	40.00	53.78
Voted	8.57	100.00	87.14	60.53

Table 11: Auxiliary transcription analysis on the same 70 ARC-1 image-plus-text cases with GPT-5-mini as the solver. The Transcribed condition uses a single GPT-5 transcription, whereas Voted uses cell-wise majority voting over five GPT-5-mini reads; the comparison therefore does not isolate the effect of voting alone.

gold output shape only as a diagnostic upper bound.

ARC-2 shape-change proxy. Table 13 reports the ARC-2 shape-change proxy. As a coarse structural analysis, we bucket ARC-2 task-test cases by whether the gold output shape differs from the

Model	Pred.-EM	Pred.-Shape	Oracle-EM	Oracle-Shape
GPT-5	86.3	97.9	88.4	100.0
GPT-5-mini	86.3	95.8	89.5	100.0

Table 12: Predicted-SFCC versus Oracle-SFCC on V-ARAOC under Gold-IR. Main SFCC results use Predicted-SFCC; Oracle-SFCC provides only a diagnostic upper bound with gold output shape.

test-input shape. EM remains low in both buckets, suggesting that ARC-2 failures are not explained solely by output-canvas changes.

Table 14 reports the full V-ARAOC SFCC comparison under both Transcribed-IR and Gold-IR for all three models.

ARC-2 bucket	N	EM	PA
Shape-changing outputs	48	4.2	36.4
Same-shape outputs	119	4.2	80.2

Table 13: GPT-5 Gold-IR performance on ARC-2 by whether the gold output shape differs from the test-input shape. EM remains low in both buckets, suggesting that failures are not explained solely by output-canvas changes.

A.6 Prompt Templates for Inference Conditions

This appendix provides the exact prompt templates used in our ARC-VLM and V-ARAOC experiments. Table 15 summarizes the prompt inventory across inference conditions. The prompts are organized by inference condition (*Direct*, *Transcribed-IR*, and *Gold-IR*) and differ primarily in input modality (image vs. text grids) and, in some settings, output protocol.

A.6.1 Shared ARC color legend

The numbers represent different colors:

- 0 = black
- 1 = blue
- 2 = red
- 3 = green
- 4 = yellow
- 5 = gray
- 6 = magenta
- 7 = orange
- 8 = cyan
- 9 = brown

A.6.2 Direct prompt (composite-image variant)

Use case. Direct condition with a single composite image containing the training examples and test input. The model predicts the final output grid directly.

Lets play a game where you are transforming an input grid of numbers into an output grid of numbers.

The numbers represent different colors:
(see Shared ARC color legend)

You will see a composite image of an ARC task: the top part contains several training examples (input → output), and the bottom part contains the test input.

Find the common pattern from the training examples and apply it to the test input.

What is the output grid? Please only output your answer without analysis in the following format:
Output grid:

A.6.3 Direct prompt (hybrid few-shot variant)

Use case. Hybrid few-shot variant used as the base prompt for Verify@ K (Appendix A.6.8): training IO pairs are provided as text grids, while the test input is provided as an image. This variant is distinct from the main Direct condition, which uses rendered grid images.

You will be playing a game that need to find common patterns from input examples and apply the pattern for prediction on new examples.

Lets play a game where you are transforming an input grid of numbers into an output grid of numbers.

The numbers represent different colors:
(see Shared ARC color legend)

Here are examples of input grids and its corresponding output grids:

Example input grid:
<TRAIN_INPUT_GRID_0>
Example output grid:
<TRAIN_OUTPUT_GRID_0>

Example input grid:
<TRAIN_INPUT_GRID_1>
Example output grid:
<TRAIN_OUTPUT_GRID_1>

... (repeat for all train examples) ...

The test input grid is provided as an image.
What is the output grid? Please only output your answer without analysis in the following format:
Output grid:

A.6.4 Gold-IR prompt (text-only few-shot)

Use case. Gold-IR condition: both training IO and the test input are provided as text grids, removing transcription noise from the encoding stage.

You will be playing a game that need to find common patterns from input examples and apply the pattern for prediction on new examples.
Lets play a game where you are transforming an input grid of numbers into an output grid of numbers.

The numbers represent different colors:
(see Shared ARC color legend)

Here are examples of input grids and its corresponding output grids:

Example input grid:
<TRAIN_INPUT_GRID_0>
Example output grid:
<TRAIN_OUTPUT_GRID_0>

... (repeat for all train examples) ...

The input grid is:
<TEST_INPUT_GRID>

Model	Setting	Base EM	SFCC EM	Base Shape	SFCC Shape	Base PA	SFCC PA
GPT-5	Transcribed-IR	5.79	24.74	11.58	40.53	59.41	83.53
GPT-5	Gold-IR	6.84	86.32	16.32	97.89	47.71	96.32
GPT-5-mini	Transcribed-IR	3.16	13.16	8.42	16.32	46.90	76.35
GPT-5-mini	Gold-IR	6.32	86.32	15.26	95.79	44.54	96.83
Gemini 2.5	Transcribed-IR	7.37	14.74	15.26	22.11	44.63	66.43
Gemini 2.5	Gold-IR	6.32	64.74	16.84	82.63	46.37	91.20

Table 14: Full V-ARAOC SFCC comparison. SFCC yields strong gains in both EM and shape under Gold-IR, and smaller but still positive gains under Transcribed-IR.

Condition / prompt	Stage	Input	Output format
Direct (composite-image)	–	composite image (train IO + test input)	text grid after “Output grid:”
Hybrid few-shot (Verify@ K base)	–	train text grids + test image	text grid after “Output grid:”
Transcribed-IR	Stage-1	test input image	text grid (strict; no extra text)
Transcribed-IR	Stage-2 (grid)	text grids (train + test)	text grid after “Output grid:”
Gold-IR (object ablation)	–	text grids (train + test)	strict JSON {height,width,grid}
Gold-IR	–	text grids (train + test)	text grid after “Output grid:”
Verify@ K	–	base prompt + constraint	strict JSON {train_preds,test_pred}

Table 15: Prompt inventory across inference conditions. All prompts include the same ARC color legend (Appendix A.6.1).

What is the output grid? Please only output your answer without analysis in the following format:
Output grid:

A.6.5 Transcribed-IR Stage-1 prompt (image transcription)

Use case. Transcribed-IR Stage-1: the model transcribes the grid exactly from an image. This prompt is also used in the smaller-scale rendering ablations.

You will be playing a game that need to find common patterns from input examples and apply the pattern for prediction on new examples.

Lets play a game where you are transforming an input grid of numbers into an output grid of numbers.

The numbers represent different colors:
(see Shared ARC color legend)

The test input grid is provided as an image.
Task: Transcribe the grid EXACTLY. Do NOT solve or explain.
Please only output your answer without analysis in the following format:
Output grid:
[[...],[...]]

Constraints:
– grid values must be integers in $[0,9]$
– the grid must be rectangular
– each row must have the same length
– no code fences, no extra text
– do NOT output any other keys (no height/width/grid fields)

A.6.6 Transcribed-IR Stage-2 prompt (grid protocol)

Use case. Transcribed-IR Stage-2: given text-grid representations for the training pairs and the test input, the model predicts the final output grid using the standard “Output grid:” marker.

You will be playing a game that need to find common patterns from input examples and apply the pattern for prediction on new examples.

Lets play a game where you are transforming an input grid of numbers into an output grid of numbers.

The numbers represent different colors:
(see Shared ARC color legend)

Here are examples of input grids and its corresponding output grids:

Example input grid:
<TRAIN_INPUT_GRID_0>
Example output grid:
<TRAIN_OUTPUT_GRID_0>

... (repeat for all train pairs) ...

The test input grid is provided below.
Test input grid:
<TEST_INPUT_GRID>

What is the output grid? Please only output your answer without analysis in the following format:
Output grid:

A.6.7 Gold-IR prompt (object-protocol ablation)

Use case. Output-protocol ablation under fixed Gold-IR inputs: the model must output a strict JSON object with explicit height, width, and grid fields.

What is the output grid?
Please output STRICT JSON ONLY (no markdown, no extra text) in the following format:
{`"height":H,"width":W,"grid":[[...],[...]]`}
Constraints:
- keys must be exactly: height, width, grid
- height/width must match the grid shape
- grid values must be integers in $[0,9]$

A.6.8 Verify@ K prompt (pass@ K with training consistency outputs)

Use case. Verify@ K : we append a strict JSON requirement to the base prompt (typically the Direct hybrid few-shot prompt in Appendix A.6.3). The model outputs predictions for each training input (for automatic consistency checking) and one prediction for the test input.

<BASE_PROMPT>

Now output STRICT JSON (no markdown, no extra text) with keys:
- `"train_preds"`: a list of N grids (your predicted output for each Example input grid, in order)
- `"test_pred"`: one grid for the test input
Format example:
{`"train_preds":[[[0]],[[1]],[[2]]],"test_pred":` }
`[[0]]`}

Here, N is the number of training examples in the task (typically $N = 3$).

A.7 Model Scale and Reasoning Mode Analysis

We provide a compact within-family comparison rather than a general scaling law, because the evaluated models differ in architecture, training data, and serving setup. Table 16 summarizes the main trends: stronger GPT-5-tier models and Qwen3-VL Thinking improve ARC-1 Gold-IR performance, while ARC-2 remains near zero; Qwen2.5-VL does not show monotonic exact-match gains with parameter scale.

A.8 Model Configuration and Decoding

Model identifiers used in our experiments. OpenAI models are called via the OpenAI API: `gpt-5`, `gpt-5-mini`, and `gpt-5-nano`. Google models are called via the Gemini API: `gemini-2.5-flash`. In addition, we evaluate

Comparison	ARC-1	ARC-2
GPT-5-nano / GPT-5-mini / GPT-5	43.8 / 66.7 / 68.6	0.0 / 1.2 / 4.2
Qwen2.5-VL 7B / 32B / 72B	1.0 / 15.2 / 9.5	0.0 / 0.0 / 0.0
Qwen3-VL Instruct / Thinking	18.1 / 42.9	0.0 / 1.2

Table 16: Within-family comparison of scale and reasoning modes. Values are Gold-IR exact-match percentages.

multiple open-source VLMs served through AI Ping’s OpenAI-compatible endpoint, including `qwen2.5-vl-72b`, `qwen2.5-vl-32b`, `qwen2.5-vl-7b`, `qwen3-vl-30b` (Instruct/Thinking), `glm-4.5v`, and `glm-4.6v`.⁴

API endpoints and cost reporting. For OpenAI, we follow the official API pricing page when reporting token costs.⁵ For Gemini, we use the official Gemini API pricing page.⁶ For AI Ping-hosted models, we report costs using service-provided price metadata exposed by the platform.⁷

Decoding and inference hyperparameters. Unless otherwise noted, we use deterministic decoding with temperature = 0 and top_p = 1 for the main inference-condition results, generating a single response per prompt (num_samples=1). For Verify@ K , we draw K independent samples using stochastic decoding (e.g., temperature = 1, top_p = 1) under the same prompt format, and select the candidate that maximizes training-set consistency under an automatic verifier. We cap the maximum output length at 32k tokens when supported by the API.

A.9 Additional ARC-2 SFCC Analysis

Table 17 reports the additional ARC-2 SFCC analysis. Unlike V-ARAOC, SFCC does not yield consistent exact-match improvements on ARC-2: the only increase is the small GPT-5 Transcribed-IR gain, while the remaining settings are unchanged or slightly worse. This pattern is consistent with substantial residual difficulty beyond output-shape construction.

⁴<https://aiping.cn/docs/UseCases/coding-assistant>

⁵<https://openai.com/api/pricing>

⁶<https://ai.google.dev/pricing>

⁷<https://aiping.cn/modelList>

Model	Setting	Baseline EM	SFCC EM
GPT-5	Transcribed-IR	0.00	1.80
GPT-5	Gold-IR	4.19	3.59
GPT-5-mini	Transcribed-IR	0.00	0.00
GPT-5-mini	Gold-IR	1.20	1.20
Gemini 2.5	Transcribed-IR	0.00	0.00
Gemini 2.5	Gold-IR	0.60	0.60

Table 17: ARC-2 SFCC results. Unlike V-ARAO, explicit shape-constrained construction does not yield consistent exact-match gains on ARC-2. All values are percentages.

A.10 Bootstrap Confidence Intervals for Main Inference-Condition Results

We report case-level bootstrap 95% confidence intervals (CIs) for EM and PA, treating each task-test pair as one evaluation case. Tables 18, 19, and 20 report the corresponding results for ARC-1, ARC-2, and V-ARAO, respectively.

Model	Direct EM [CI]	Direct PA [CI]	Transcribed-IR EM [CI]	Transcribed-IR PA [CI]	Gold-IR EM [CI]	Gold-IR PA [CI]	Shape
Gemini 2.5	0.1238 [0.0667, 0.1905]	0.4433 [0.3686, 0.5203]	0.0952 [0.0476, 0.1524]	0.2578 [0.1887, 0.3329]	0.4476 [0.3524, 0.5524]	0.8185 [0.7618, 0.8747]	0.9333
GLM-4.5V	0.0286 [0.0000, 0.0667]	0.3946 [0.3274, 0.4660]	0.0190 [0.0000, 0.0476]	0.2090 [0.1492, 0.2701]	0.1143 [0.0571, 0.1810]	0.7016 [0.6363, 0.7583]	0.8952
GLM-4.6V	0.0381 [0.0095, 0.0762]	0.3802 [0.3112, 0.4515]	0.0381 [0.0095, 0.0762]	0.3228 [0.2485, 0.3861]	0.1333 [0.0762, 0.1905]	0.6816 [0.6104, 0.7469]	0.8571
GPT-5	0.2286 [0.1524, 0.3048]	0.7249 [0.6753, 0.7724]	0.0952 [0.0476, 0.1524]	0.5877 [0.5295, 0.6451]	0.6857 [0.5905, 0.7714]	0.9104 [0.8725, 0.9457]	0.9619
GPT-5-mini	0.0381 [0.0095, 0.0762]	0.3585 [0.2945, 0.4272]	0.0190 [0.0000, 0.0476]	0.1483 [0.0948, 0.2068]	0.6667 [0.5714, 0.7619]	0.8729 [0.8188, 0.9209]	0.9524
GPT-5-nano	0.0286 [0.0000, 0.0667]	0.2712 [0.2084, 0.3392]	0.0190 [0.0000, 0.0476]	0.2840 [0.2202, 0.3455]	0.4381 [0.3524, 0.5240]	0.7702 [0.7040, 0.8299]	0.8952
Qwen2.5-VL-72B	0.0381 [0.0095, 0.0762]	0.3499 [0.2833, 0.4219]	0.0190 [0.0000, 0.0476]	0.2225 [0.1630, 0.2877]	0.0952 [0.0476, 0.1524]	0.6774 [0.6131, 0.7380]	0.8762
Qwen2.5-VL-32B	0.0286 [0.0000, 0.0667]	0.3178 [0.2505, 0.3880]	0.0095 [0.0000, 0.0288]	0.3914 [0.3308, 0.4546]	0.1524 [0.0857, 0.2286]	0.6742 [0.6012, 0.7375]	0.8667
Qwen2.5-VL-7B	0.0094 [0.0000, 0.0283]	0.1708 [0.1211, 0.2319]	0.0095 [0.0000, 0.0286]	0.4894 [0.4240, 0.5490]	0.0095 [0.0000, 0.0286]	0.4476 [0.3841, 0.5084]	0.9238
Qwen3-VL-30B (Instruct)	0.0286 [0.0000, 0.0667]	0.3617 [0.2895, 0.4320]	0.0190 [0.0000, 0.0476]	0.4054 [0.3421, 0.4739]	0.1810 [0.1143, 0.2667]	0.7249 [0.6596, 0.7830]	0.8857
Qwen3-VL-30B (Thinking)	0.0476 [0.0095, 0.0952]	0.4087 [0.3431, 0.4766]	0.0381 [0.0095, 0.0762]	0.2426 [0.1780, 0.3050]	0.4286 [0.3333, 0.5333]	0.7346 [0.6639, 0.8072]	0.8857

Table 18: ARC-1 results across inference conditions with bootstrap 95% confidence intervals.

Model	Direct EM [CI]	Direct PA [CI]	Transcribed-IR EM [CI]	Transcribed-IR PA [CI]	Gold-IR EM [CI]	Gold-IR PA [CI]	Shape
Gemini 2.5	0.0000 [0.0000, 0.0000]	0.1176 [0.0825, 0.1508]	0.0000 [0.0000, 0.0000]	0.0507 [0.0266, 0.0779]	0.0060 [0.0000, 0.0180]	0.5887 [0.5247, 0.6446]	0.7665
GLM-4.5V	0.0000 [0.0000, 0.0000]	0.1816 [0.1377, 0.2266]	0.0000 [0.0000, 0.0000]	0.1033 [0.0697, 0.1346]	0.0000 [0.0000, 0.0000]	0.5243 [0.4649, 0.5821]	0.6707
GLM-4.6V	0.0000 [0.0000, 0.0000]	0.2245 [0.1792, 0.2719]	0.0000 [0.0000, 0.0000]	0.1680 [0.1164, 0.2161]	0.0000 [0.0000, 0.0000]	0.5105 [0.4461, 0.5680]	0.7365
GPT-5	0.0120 [0.0000, 0.0299]	0.3702 [0.3230, 0.4198]	0.0000 [0.0000, 0.0000]	0.3215 [0.2707, 0.3738]	0.0419 [0.0178, 0.0719]	0.6762 [0.6240, 0.7232]	0.7246
GPT-5-mini	0.0000 [0.0000, 0.0000]	0.2166 [0.1687, 0.2596]	0.0000 [0.0000, 0.0000]	0.1138 [0.0823, 0.1519]	0.0120 [0.0000, 0.0299]	0.3809 [0.3229, 0.4449]	0.5210
GPT-5-nano	0.0000 [0.0000, 0.0000]	0.1293 [0.0949, 0.1676]	0.0000 [0.0000, 0.0000]	0.1555 [0.1166, 0.1999]	0.0000 [0.0000, 0.0000]	0.3401 [0.2790, 0.3982]	0.5329
Qwen2.5-VL-72B	0.0000 [0.0000, 0.0000]	0.2139 [0.1649, 0.2607]	0.0000 [0.0000, 0.0000]	0.1363 [0.1001, 0.1759]	0.0000 [0.0000, 0.0000]	0.5416 [0.4814, 0.6050]	0.6766
Qwen2.5-VL-32B	0.0000 [0.0000, 0.0000]	0.1793 [0.1344, 0.2249]	0.0000 [0.0000, 0.0000]	0.1993 [0.1368, 0.2609]	0.0000 [0.0000, 0.0000]	0.5469 [0.4849, 0.6087]	0.7305
Qwen2.5-VL-7B	0.0000 [0.0000, 0.0000]	0.0807 [0.0527, 0.1113]	0.0000 [0.0000, 0.0000]	0.2000 [0.1551, 0.2450]	0.0000 [0.0000, 0.0000]	0.2191 [0.1766, 0.2682]	0.9641

Table 19: ARC-2 results across inference conditions with bootstrap 95% confidence intervals.

Model	Direct EM [CI]	Direct PA [CI]	Transcribed-IR EM [CI]	Transcribed-IR PA [CI]	Gold-IR EM [CI]	Gold-IR PA [CI]	Shape
Gemini 2.5	0.1842 [0.1316, 0.2368]	0.2670 [0.2074, 0.3248]	0.0737 [0.0421, 0.1107]	0.2445 [0.1939, 0.2973]	0.0632 [0.0316, 0.1000]	0.1111 [0.0710, 0.1504]	0.1947
GPT-5-mini	0.0684 [0.0368, 0.1105]	0.0964 [0.0577, 0.1404]	0.0316 [0.0105, 0.0579]	0.1226 [0.0824, 0.1647]	0.0632 [0.0316, 0.1000]	0.1159 [0.0763, 0.1610]	0.1895

Table 20: V-ARAOC results across inference conditions with bootstrap 95% confidence intervals. Shape reports Gold-IR shape match.