RIPA: RULE INDUCTION FROM PROBABILISTIC ATOMS FOR CLOSED-LOOP MULTIMODAL INDUCTIVE REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Multimodal Large Language Models (MLLMs) have made substantial progress in visual understanding, yet remain less reliable in visual inductive reasoning and provide limited rule-level interpretability. Neuro-symbolic methods, by contrast, support explicit rule induction and traceable inference, but their integration with MLLMs for rule-guided prediction and image-specific rule verification remains insufficiently explored. In this paper, we propose *RIPA*, a closed-loop framework integrating neuro-symbolic rule induction with MLLM reasoning. RIPA first uses an MLLM to extract probabilistic atoms describing object attributes, relations, and overall scene structure from images. A Template-Driven Differentiable Rule Learner (TDRL) then induces sparse class-specific rules from these atoms and stores selected rules in a rule bank. During inference, TDRL scores candidate classes and retrieves their associated rules. These scores and rules, together with the image and probabilistic atoms, are fed back to the MLLM for prediction and image-specific rule verification. Finally, a consistency-aware module fuses the TDRL scores and MLLM outputs using rule agreement and conflict signals to produce the final prediction, thereby closing the reasoning loop. Experiments on two established object-centric visual reasoning benchmarks and our RAVEN-Custom benchmark show that RIPA improves over the neuro-symbolic MLLM baseline ILP-CoT by 10% on average. Moreover, RIPA exports explicit symbolic rules and rule-level evidence, making its predictions interpretable and traceable.

1 INTRODUCTION

Visual inductive reasoning requires a model to identify regularities across observations, combine the conditions that define a concept, and apply the induced rule to unseen instances. Multimodal Large Language Models (MLLMs), such as Flamingo (Alayrac et al., 2022), BLIP-2 (Li et al., 2023), and LLaVA (Liu et al., 2023), have substantially advanced visual perception and vision–language understanding. Yet recognizing the content of individual images does not necessarily imply the ability to infer task-level rules from multiple examples. Recent evaluations reveal a persistent gap between image-level parsing and cross-example rule induction in MLLMs (Cao et al., 2025). As illustrated in Figure 1, a model can correctly identify objects, attributes, and relations in each image while still inducing an incomplete or inconsistent rule. This raises the central question of our work: how can explicit rules be induced from visual evidence and used to improve the inductive reasoning of MLLMs?

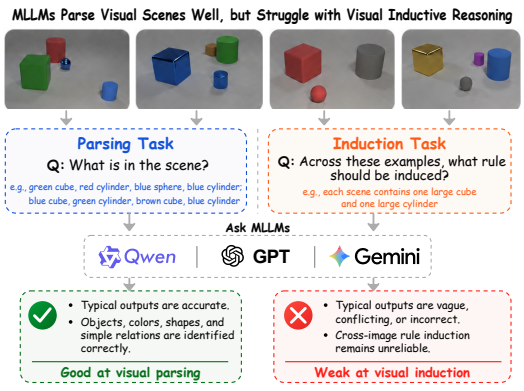


Figure 1: Image-level parsing does not guarantee cross-example rule induction. An MLLM can correctly identify visual elements in individual scenes while failing to infer a consistent task-level rule.

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.

One approach is to prompt MLLMs to articulate task hypotheses through intermediate reasoning. Chain-of-Thought (CoT) prompting can improve arithmetic, commonsense, and symbolic reasoning (Wei et al., 2022), but its rationales may not faithfully reflect model decisions (Turpin et al., 2023; Lanham et al., 2023) or ensure adherence to task-specific rules (Lyu et al., 2023). In visual induction, uncertain evidence must be aggregated across examples. An MLLM may therefore describe individual images plausibly while omitting necessary conditions, producing contradictory hypotheses, or applying rules inconsistently (Cao et al., 2025; Turpin et al., 2023; Lyu et al., 2023).

Neuro-symbolic learning addresses these limitations by separating perceptual grounding from structured reasoning and learning or executing explicit rules over object-centric representations (Yi et al., 2018; Mao et al., 2019; Manhaeve et al., 2018; Shindo et al., 2024). Recent work combines MLLMs with symbolic rule induction, but does not feed the induced rules back to the MLLM for image-specific visual verification (Peng et al., 2025). How these rules can guide prediction while being validated against each test image therefore remains open.

We propose *RIPA*, a closed-loop framework that uses induced rules to connect probabilistic visual evidence with MLLM prediction and verification. An MLLM extracts *probabilistic atoms* describing object attributes, relations, and scene structure. A *Template-Driven Differentiable Rule Learner (TDRL)* preserves this uncertainty when learning sparse, class-specific rules. It exports selected rules to a human-readable *rule bank*, whose structures and validation-derived quality scores remain fixed during inference.

During inference, TDRL selects candidate classes and retrieves their high-quality rules. Given the image, probabilistic atoms, candidates, and rules, the MLLM predicts the class and checks each rule against the visual evidence. The final prediction combines TDRL and MLLM scores with *rule-level agreement and conflict signals*. This *closed-loop reasoning* is decision-level feedback: induced rules guide the MLLM, whose verification informs the final decision without updating TDRL or the rule bank.

Our main contributions are summarized as follows:

- We propose RIPA, a closed-loop framework that externalizes rules induced from probabilistic visual evidence, uses them to guide MLLM inference, and incorporates image-specific rule verification into the final decision.
- We introduce TDRL, a template-driven differentiable rule learner that preserves perceptual uncertainty and learns sparse, class-specific rules that can be exported to a fixed, human-readable rule bank.
- We develop a rule-grounded MLLM reasoning and consistency-aware fusion mechanism that combines TDRL and MLLM predictions with rule agreement and conflict signals.
- We evaluate RIPA on three benchmarks. RIPA improves accuracy over prompting-based and neuro-symbolic MLLM baselines while providing explicit rules and traceable image-level evidence.

2 Related Work

Multimodal Models for Visual Reasoning. Existing MLLMs support multimodal in-context learning and visual instruction following through pretrained vision–language alignment (Alayrac et al., 2022; Li et al., 2023; Liu et al., 2023). To extend these models beyond direct perception, prior work commonly uses few-shot prompting and Chain-of-Thought reasoning to elicit task hypotheses and intermediate rationales (Wei et al., 2022). However, generated rationales may not faithfully reflect model decisions or remain consistent with task-specific rules (Turpin et al., 2023; Lanham et al., 2023; Lyu et al., 2023), particularly when reasoning requires abstract regularities to be inferred across multiple visual examples (Cao et al., 2025). RIPA differs from these prompting-based approaches by learning explicit symbolic rules from training examples and reusing them as structured guidance for MLLM prediction and visual-evidence verification.

Neuro-Symbolic Visual Reasoning and Differentiable Rule Learning. Neuro-symbolic visual reasoning combines neural perception with structured representations and symbolic inference, as exemplified by systems that execute learned programs over recovered scene structures (Yi et al.,

2018; Mao et al., 2019). Differentiable logic frameworks introduce continuous semantics for learning and inference under uncertain neural predictions (Badreddine et al., 2022; Manhaeve et al., 2018; 2021), while differentiable ILP methods relax discrete rule induction for gradient-based optimization (Evans & Grefenstette, 2018; Shindo et al., 2021b; Sen et al., 2022; Rocktäschel & Riedel, 2017). More recent approaches, including NSFR, αILP, and NEUMANN, learn or execute logical programs over probabilistic object-centric representations (Shindo et al., 2021a; 2023; 2024). RIPA builds on this line of work but externalizes the induced rules into a fixed rule bank and reuses them to guide MLLM prediction and image-specific rule verification.

External Rules and Formal Reasoning for Large Models. Prior work improves large-model reasoning through executable programs or formal solvers, constrained reasoning traces, and executable hypothesis search. PAL (Gao et al., 2023) and Logic-LM (Pan et al., 2023) delegate reasoning to programs or symbolic solvers, while Faithful CoT (Lyu et al., 2023), SatLM (Ye et al., 2023), SymbCoT (Xu et al., 2024), and LeanReasoner (Jiang et al., 2024) impose structured or formally verifiable reasoning. Hypothesis Search (Wang et al., 2024) searches over executable hypotheses. These methods mainly operate on textual or symbolic representations, whereas RIPA learns rules over probabilistic atoms and feeds them back to an MLLM for verification. ILP-CoT (Peng et al., 2025) is the closest prior work, but uses discrete symbolic descriptions without feeding induced rules back to the MLLM for visual verification.

3 PRELIMINARIES

3.1 LOGIC PROGRAMMING

Logic programming represents entities, attributes, and relations using facts and rules. We consider a function-free first-order language following the Datalog setting (Ceri et al., 1989). An atom has the form $p(t_1, \ldots, t_n)$, where p is an n-ary predicate and each t_i is a variable or constant. Function symbols and list-valued terms are excluded. An atom containing only constants is called a ground atom.

Given the predicates and constants allowed by a task-specific template $\mathcal{T}$, we denote the resulting finite Herbrand base by $\mathcal{B}_\mathcal{T}$. For an input image x, the perception module assigns each ground atom $a \in \mathcal{B}_\mathcal{T}$ a soft truth value $\nu_x(a) \in [0, 1]$. The probabilistic atom representation of x therefore defines the valuation, $\nu_x : \mathcal{B}_\mathcal{T} \to [0, 1]$.

A rule is written as: $h \leftarrow b_1, \ldots, b_K$, where the head atom h is derived when all body atoms $b_1, \ldots, b_K$ hold. A set of facts and rules forms a logic program P, and $P \models q$ indicates that query q is entailed by P. In inductive logic programming (ILP), rule learning aims to induce clauses that explain observed examples under a prescribed language bias (Muggleton, 1991; Muggleton & De Raedt, 1994).

3.2 NEURO-SYMBOLIC LEARNING AND DIFFERENTIABLE RULE INDUCTION

Neuro-symbolic learning combines neural perception with symbolic reasoning: neural modules map raw inputs to soft symbolic representations, while logical components perform structured inference over those representations (Yi et al., 2018; Mao et al., 2019; Tsamoura et al., 2021; Manhaeve et al., 2018). Differentiable rule induction replaces discrete rule selection and Boolean operators with continuous relaxations, allowing logical structures to be optimized by gradient-based methods under uncertain perceptual evidence (Rocktäschel & Riedel, 2017; Evans & Grefenstette, 2018; Shindo et al., 2021b; Sen et al., 2022).

We use the product T-norm for fuzzy conjunction and its dual probabilistic-sum T-conorm for fuzzy disjunction (Klement et al., 2000; Badreddine et al., 2022):

$$T_\Pi(z_1, \ldots, z_n) = \prod_{i=1}^{n} z_i, \qquad S_\Pi(z_1, \ldots, z_n) = 1 - \prod_{i=1}^{n} (1 - z_i). \tag{1}$$

For two truth values, these operators reduce to $T_\Pi(a, b) = ab$ and $S_\Pi(a, b) = a + b - ab$. Both operators are differentiable on $[0, 1]$ and recover Boolean conjunction and disjunction when all inputs are binary.

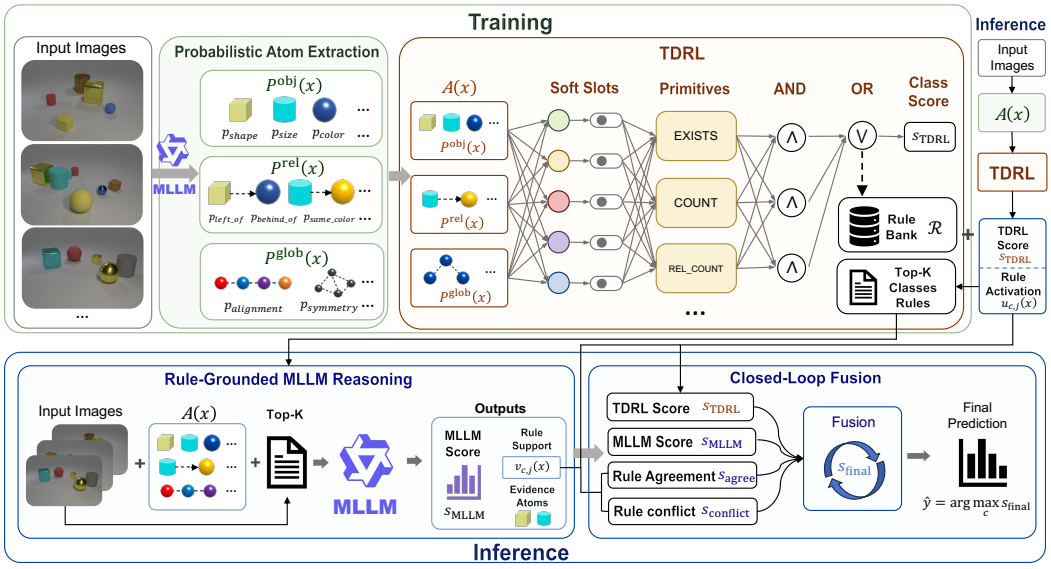


Figure 2: **Overview of the RIPA pipeline**. During training, probabilistic atoms are extracted from input images and used by TDRL to learn class-specific rules, which are exported to a fixed rule bank. During inference, the image, probabilistic atoms, and retrieved rules guide MLLM reasoning, and the final prediction is obtained by fusing TDRL scores, MLLM scores, rule agreement and rule conflict.

4 Methodology

This section presents the proposed RIPA framework. Figure 2 illustrates the overall pipeline of RIPA.

4.1 Probabilistic Atom Extraction

RIPA first uses an MLLM to extract a probabilistic atom representation from each input image. This allows the system to perform differentiable reasoning while preserving perceptual uncertainty. For an input image x, the probabilistic atom representation is written as: $A(x) = P^{\text{obj}}(x) \cup P^{\text{rel}}(x) \cup P^{\text{glob}}(x)$, where $P^{\text{obj}}(x)$, $P^{\text{rel}}(x)$, and $P^{\text{glob}}(x)$ are sets of probabilistic ground atoms describing object attributes, pairwise relations, and *global properties* characterizing the overall scene structure, respectively.

Object-level atoms describe attributes of individual objects. RIPA represents each attribute as a probability distribution over its value space. Relational atoms describe pairwise spatial and attribute relations between object pairs, such as left-of, close-to, or same-color. For tasks requiring holistic structural reasoning, RIPA allows global atoms to describe layout regularity, alignment, symmetry, or row-column patterns. Here, layout refers to the spatial arrangement of multiple objects within the image as a whole.

4.2 Template-Driven Differentiable Rule Learning

Given an input image x, the trained TDRL maps its probabilistic atoms $A(x)$ to class scores, $s_{\text{TDRL}}(x) = f_{\text{TDRL}}(A(x); \Theta_{\mathcal{T}})$, where $\Theta_{\mathcal{T}}$ denotes the learnable TDRL parameters. The *template space* $\mathcal{T}$ specifies the admissible attributes, relations and primitives.

An *object slot* denotes a learnable latent object role analogous to an object variable in a symbolic rule. Each slot is softly matched to all detected objects according to its learned attribute constraints and can therefore be grounded to different object instances across images. For each class c, TDRL maintains a candidate-rule set $\mathcal{R}_c^{\text{cand}}$ containing M rules. Each candidate rule contains S object slots together with a set of object-level and relation-level primitives.

For a candidate rule r, we omit the rule index from the local notation for clarity. For an attribute a with value space $\mathcal{V}_a$, object slot s of rule r is parameterized by an *attribute-value selector* and an *attribute gate*,

$$\pi_s^a = \mathrm{softmax}\left(\theta_s^a\right) \in \Delta^{|\mathcal{V}_a|-1}, \qquad \alpha_s^a = \mathrm{sigmoid}\left(g_s^a\right) \in [0,1],$$

where θ_s^a and g_s^a are learnable parameters optimized during training, π_s^a specifies the attribute-value distribution selected by the slot, and α_s^a controls whether attribute a is included in the slot.

Let $p_i^a \in \Delta^{|\mathcal{V}_a|-1}$ denote the probabilistic attribute distribution of object i in image x, and let $\mu_i \in \{0,1\}$ be its validity mask. The soft match between object i and slot s of rule r is then defined as:

$$M_{i,s}(x) = \mu_i \prod_{a \in \mathcal{A}_{\mathrm{obj}}} \left[\alpha_s^a \left\langle p_i^a, \pi_s^a \right\rangle + (1 - \alpha_s^a)\right].$$

When α_s^a approaches zero, attribute a contributes the neutral value 1 and is excluded from the slot concept.

TDRL evaluates each rule using *differentiable primitives*, which provide differentiable relaxations of symbolic conditions such as existence, counting, and pairwise relations. Let $h_t(x)$ denote the soft truth value of primitive t for image x. With the *primitive gate* $\lambda_t = \mathrm{sigmoid}(g_t)$, its gated truth value is: $\tilde{h}_t(x) = \lambda_t h_t(x) + (1 - \lambda_t)$, so a disabled primitive contributes the neutral truth value 1.

Following the fuzzy operators defined in Eq. (1), RIPA uses the product T-norm to combine primitives within each rule and the probabilistic-sum T-conorm to aggregate alternative candidate rules for the same class:

$$B_r(x) = T_\Pi\left(\{\tilde{h}_t(x)\}_{t \in \mathcal{P}_r}\right) = \prod_{t \in \mathcal{P}_r} \tilde{h}_t(x),$$

$$s_{\mathrm{TDRL}}(c \mid x) = S_\Pi\left(\{\rho_r B_r(x)\}_{r \in \mathcal{R}_c^{\mathrm{cand}}}\right) = 1 - \prod_{r \in \mathcal{R}_c^{\mathrm{cand}}} \left(1 - \rho_r B_r(x)\right),$$

where $\mathcal{P}_r$ is the set of primitives instantiated in rule r, $\mathcal{R}_c^{\mathrm{cand}}$ is a set of candidate rules, and ρ_r is a learnable *rule gate* controlling whether the entire candidate rule contributes to the class score. Thus, each candidate rule represents a conjunction of symbolic conditions, while multiple candidate rules capture alternative reasoning patterns for the same class.

TDRL is trained end-to-end using classification supervision and structural regularization,

$$\mathcal{L}_{\mathrm{TDRL}} = \mathcal{L}_{\mathrm{CE}} + \lambda_{\mathrm{sp}}\mathcal{L}_{\mathrm{sp}} + \lambda_{\mathrm{div}}\mathcal{L}_{\mathrm{div}} + \lambda_{\mathrm{int}}\mathcal{L}_{\mathrm{int}} + \lambda_{\mathrm{ne}}\mathcal{L}_{\mathrm{ne}}.$$

Here, $\mathcal{L}_{\mathrm{sp}}$ encourages sparse attribute, primitive, and rule gates, while $\mathcal{L}_{\mathrm{div}}$ reduces redundancy among slots and alternative candidate rules. $\mathcal{L}_{\mathrm{int}}$ moves learnable counting thresholds toward integers, and $\mathcal{L}_{\mathrm{ne}}$ prevents an active rule from disabling all of its primitives. Formal definitions of these structural regularizers are provided in Appendix B.3.

After training, TDRL removes inactive conditions and rules and discretizes each retained selector by selecting its highest-probability value. Each retained rule is then exported as an explicit conjunction of symbolic conditions, while multiple retained rules for the same class are combined disjunctively to form the final class rule.

4.3 Rule Bank Construction

The exported rules form a fixed *rule bank*: rule structures, predicate selections, thresholds, and validation-derived quality scores are not updated on test samples. Retained candidate rules are ranked by validation precision, coverage, average activation strength, and rule gates. For class c, the bank is $\mathcal{R}_c = \{(r_{c,1}, q_{c,1}), \ldots, (r_{c,K}, q_{c,K})\}$, $\mathcal{R} = \bigcup_c \mathcal{R}_c$, where $r_{c,j}$ is the j-th rule and $q_{c,j}$ its validation-estimated quality.

The rule bank provides an interpretable structure for TDRL, and also serves as external symbolic knowledge for MLLM reasoning.

4.4 RULE-GROUNDED MLLM REASONING

During inference, TDRL selects a small set of candidate classes and retrieves their high-quality rules from the fixed bank. A structured prompt provides the MLLM with the image, probabilistic atoms, candidates, and rules. The MLLM returns class prediction scores and a visual-support score for each retrieved rule, together with evidence atoms used for verification. It thus acts as both a multimodal classifier and an image-specific rule verifier, supplying rule-level evidence for closed-loop fusion.

4.5 CLOSED-LOOP FUSION

In the final stage, we introduce rule-level agreement and conflict signals for closed-loop fusion. Intuitively, if a high-quality rule is strongly activated by TDRL and receives a high visual-support score from the MLLM verifier, the class associated with this rule should receive higher confidence. Conversely, if TDRL activates a rule but the MLLM finds insufficient visual evidence for it, the corresponding class should be penalized.

Let $u_{c,j}(x)$ denote the input-dependent activation of rule $r_{c,j}$ computed by TDRL, and let $v_{c,j}(x)$ denote the support assigned to this rule by the MLLM verifier. Formally,

$$u_{c,j}(x) = f_{\text{act}}\left(A(x), r_{c,j}; \Theta_{\mathcal{T}}\right) \in [0,1], \qquad v_{c,j}(x) = f_{\text{verify}}\left(x, A(x), r_{c,j}\right) \in [0,1]$$

The validation-derived rule quality is written as $q_{c,j} = Q\left(r_{c,j}; \mathcal{D}_{\text{val}}\right) \in [0,1]$, where f_{act} is the TDRL rule-activation function, f_{verify} is the MLLM rule-verification function, and Q is the rule-quality evaluation function. RIPA then computes class-level agreement and conflict as:

$$s_{\text{agree}}(c \mid x) = \sum_j q_{c,j} u_{c,j}(x) v_{c,j}(x), \qquad s_{\text{conflict}}(c \mid x) = \sum_j q_{c,j} u_{c,j}(x)\left(1 - v_{c,j}(x)\right).$$

The final class score is defined as:

$$s_{\text{final}}(c \mid x) = \alpha s_{\text{TDRL}}(c \mid x) + \beta s_{\text{MLLM}}(c \mid x) + \gamma s_{\text{agree}}(c \mid x) - \delta s_{\text{conflict}}(c \mid x). \tag{2}$$

Then the final predicted class is: $\hat{y} = \arg\max_c s_{\text{final}}(c \mid x)$. Equation 2 combines class predictions with evidence about individual rules. Its first two terms combine the TDRL and MLLM class scores; the last two reward visually supported rule activations and penalize unsupported ones. The weights $\alpha, \beta, \gamma, \delta$ are selected on validation data and fixed at inference. Feedback therefore adjusts the final prediction while keeping TDRL and the rule bank fixed on test images.

5 EXPERIMENTS

We evaluate RIPA on three visual reasoning benchmarks, focusing on predictive accuracy, rule interpretability, and the contribution of adding rule-grounded MLLM reasoning and fusion to TDRL.

5.1 BASELINES AND EVALUATION SETTINGS

Baselines. We compare RIPA with MLLM Few-shot, which predicts labels from support examples; MLLM CoT (Wei et al., 2022), which first summarizes candidate visual rules before making a prediction; and ILP-CoT (Peng et al., 2025), which combines MLLM-based abstraction with inductive logic programming. We also compare with NEUMANN (Shindo et al., 2024), a neuro-symbolic method for differentiable logic-program induction. The MLLM-based experiments use Qwen3-VL (Bai et al., 2025), with additional evaluations using GPT-5.4 (OpenAI, 2026). Hyperparameters and fusion weights are selected only on validation data. Appendix B gives atom vocabularies, templates, and implementation settings; Appendix C gives baseline protocols and prompts; and Appendix D gives dataset splits.

Evaluation Metrics. We report classification accuracy as the primary quantitative metric. For CLEVR-Hans (Stammer et al., 2021; Peng et al., 2025), both validation and test accuracy are reported. For Kandinsky Patterns (Müller & Holzinger, 2021) and RAVEN-Custom, which we construct from the original RAVEN benchmark (Zhang et al., 2019), we report test accuracy.

Table 1: Experimental results on CLEVR-Hans and Kandinsky Patterns. Accuracy (%) is reported on validation and test splits for CLEVR-Hans, and on the test split for Kandinsky Patterns. Best results within each MLLM group are shown in bold.

MLLM	Method	Hans3		Hans7		Kandinsky Patterns				
		Val	Test	Val	Test	*twopairs*	*closeby*	*red-triangle*	*online-pair*	Avg.
Qwen3-VL-8B-Instruct	Few-shot	34.76	29.60	16.88	16.02	56.00	62.00	54.50	52.00	56.13
	CoT	58.89	53.33	43.54	34.62	53.50	54.50	50.75	53.00	52.94
	ILP-CoT	88.37	81.85	60.46	54.69	58.25	66.75	53.75	60.25	59.75
	RIPA	**88.67**	**89.69**	**96.80**	**84.53**	**98.75**	**94.25**	**78.25**	**79.00**	**87.56**
Qwen3-VL-235B-A22B-Instruct	Few-shot	42.49	41.36	14.81	14.71	53.25	52.50	57.75	49.75	53.31
	CoT	82.26	59.33	50.82	45.85	50.25	54.00	49.25	78.75	58.06
	ILP-CoT	97.38	97.47	69.49	70.97	74.25	93.25	89.00	68.50	81.25
	RIPA	**98.27**	**98.58**	**98.44**	**87.12**	**99.75**	**100.0**	**99.50**	**97.75**	**99.25**
GPT-5.4	Few-shot	43.47	38.93	52.08	51.05	96.00	83.75	86.25	66.50	83.13
	CoT	61.64	50.58	52.50	51.26	**100.0**	89.75	97.75	71.75	89.81
	ILP-CoT	83.51	83.07	63.89	60.17	**100.0**	86.50	**100.0**	91.75	94.56
	RIPA	**91.60**	**90.88**	**95.03**	**80.85**	99.75	**99.00**	96.75	**94.00**	**97.38**
NEUMANN (w/o pretrain)		66.82	66.63	85.72	85.71	48.25	50.00	51.00	49.50	49.69
NEUMANN		96.67	97.43	88.42	86.96	100.0	100.0	100.0	100.0	100.0

5.2 CLEVR-HANS

Dataset and Task. In our experiments, CLEVR-Hans (Stammer et al., 2021), derived from CLEVR (Johnson et al., 2017), serves as a confounded visual classification benchmark designed to evaluate whether a model learns class-defining compositional rules rather than relying on shortcut correlations. We use the standard CLEVR-Hans3 and CLEVR-Hans7 subsets. Hans3 contains three classes whose definitions primarily involve object existence and attribute conjunctions, whereas Hans7 extends the task to seven classes and introduces more complex counting and relational conditions.

Result Analysis. RIPA improves test accuracy over the prompting baselines for all three MLLMs (Table 1). Its gains over ILP-CoT are larger on Hans7 than on Hans3 for every backbone. Hans7 includes counting and relational conditions, whereas Hans3 mainly requires object existence and attribute conjunctions. RIPA represents these conditions as explicit rules and checks their visual support during inference.

RIPA with GPT-5.4 improves over the corresponding baselines, but trails Qwen3-VL-235B-A22B-Instruct on both Hans test sets. In the Hans3 atom-quality evaluation (Appendix E), Qwen3-VL-235B-A22B-Instruct produces more accurate attribute predictions and a lower Brier score than GPT-5.4.

RIPA with Qwen3-VL-235B-A22B-Instruct outperforms the no-pretraining variant of NEUMANN on both CLEVR-Hans test sets. The fully pretrained NEUMANN result provides additional context: its Slot Attention perception module uses different perceptual supervision and inductive biases from RIPA's MLLM-derived atoms. These comparisons evaluate complete visual reasoning pipelines and reflect differences in both perception and reasoning.

5.3 KANDINSKY PATTERNS

Dataset and Task. Kandinsky Patterns (Müller & Holzinger, 2021) is a rule-based visual concept benchmark composed of simple colored geometric objects. It has been widely adopted for evaluating neuro-symbolic visual reasoning and differentiable logic learning (Shindo et al., 2023; 2024;

Table 2: RAVEN-Custom test accuracy (%) with the 235B Qwen3-VL model.

Method	Center	2×2	3×3	Avg.
MLLM Few-shot	42.25	54.00	52.00	49.42
MLLM CoT	69.00	50.00	52.25	57.08
ILP-CoT	70.25	74.75	64.75	69.92
RIPA	**85.00**	**75.50**	**69.50**	**76.67**

Table 3: Component ablations: test accuracy (%). Kandinsky and RAVEN-Custom scores are averages across tasks and configurations, respectively.

Variant	Hans3	Hans7	Kandinsky	RAVEN-Custom
Hard atoms	70.43	79.76	75.06	72.06
TDRL only	91.42	80.06	94.25	65.96
RIPA	**98.58**	**87.12**	**99.25**	**76.67**

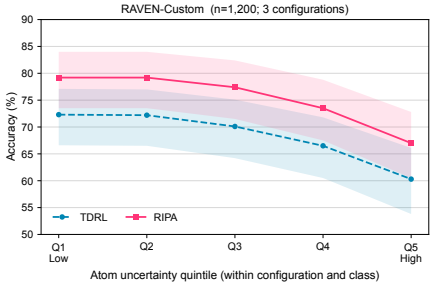


Figure 3: RAVEN-Custom test accuracy by atom-uncertainty quintile. Samples are stratified by configuration and class before pooling; shading shows 95% Wilson confidence intervals.

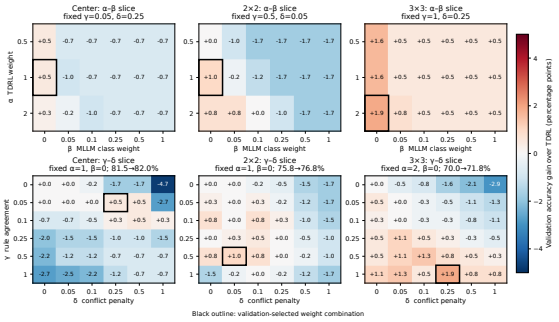


Figure 4: RAVEN-Custom validation sensitivity to fusion weights. Values show accuracy changes (percentage points) relative to TDRL only; black outlines mark validation-selected settings.

Lorello et al., 2025). We consider four binary classification tasks: *twopairs*, *closeby*, *red-triangle*, and *online-pair*. The *twopairs* task involves pair formation and counting, *closeby* evaluates spatial proximity, *red-triangle* requires color–shape conjunctions, and *online-pair* tests object alignment together with pairwise attribute constraints.

Result Analysis. RIPA achieves the highest average accuracy for all three MLLMs (Table 1). The largest average improvement over ILP-CoT is obtained with Qwen3-VL-8B-Instruct. Under GPT-5.4, CoT and ILP-CoT remain competitive on individual tasks, including *twopairs* and *red-triangle*. CoT also falls below few-shot prediction on the 8B backbone, so requesting a free-form rationale does not ensure higher accuracy.

5.4 RAVEN-CUSTOM

Dataset and Task. RAVEN (Zhang et al., 2019) is a visual abstract reasoning benchmark based on Raven's Progressive Matrices. Each problem contains eight context panels and eight candidate answers, and the original objective is to select the candidate that correctly completes the matrix. To adapt RAVEN to the rule-induction setting considered in this work, we construct RAVEN-Custom, a binary classification variant governed by a predefined set of transformation rules. A positive example satisfies the complete transformation conjunction, whereas a negative example pairs the context panels with an incorrect candidate. Details of the construction procedure, rule definitions, and data splits are provided in Appendix D.3.

Result Analysis. RIPA obtains the highest test accuracy in all three configurations, averaging 76.67% compared with 69.92% for ILP-CoT (Table 2). The gains over ILP-CoT are 14.75, 0.75, and 4.75 percentage points on Center, 2×2, and 3×3, respectively. Unlike the single-scene tasks, RAVEN-Custom requires transformations to be checked across three ordered panels. RIPA

represents these conditions as explicit transformation rules and verifies their visual support. Its accuracy nevertheless decreases from Center to the two grid configurations, as does that of ILP-CoT. The gain is modest on 2×2, and all methods remain below 70% on 3×3.

Uncertainty and Fusion Analyses. Figure 3 groups test samples into five atom-uncertainty quintiles within each configuration and class before pooling. RIPA has higher observed accuracy than TDRL in all five groups, including the highest-uncertainty group. Accuracy decreases for both methods as uncertainty increases: the full framework retains an advantage across the displayed range, but its predictions remain sensitive to less certain atom representations.

Figure 4 reports changes in validation accuracy over TDRL only while varying pairs of fusion weights. In the α–β slices, increasing the MLLM class-score weight does not consistently improve accuracy. The γ–δ slices show larger gains for 3×3, but increasing agreement or conflict weights is not uniformly beneficial. The best-performing regions differ across configurations, indicating that the balance between rule support and conflict matters when choosing fusion weights. These are validation-set sensitivity results; Table 3 reports the test-set comparison with TDRL only.

5.5 Interpretability Analysis

RIPA exports rules that specify conditions over attributes, relations, counts, global properties, and rule confidence. These rules approximately recover object-existence patterns on Hans3, alignment and attribute-agreement constraints on Kandinsky, and number, type, and size transformations on RAVEN-Custom. Predictions can therefore be traced to rule activations over probabilistic atoms rather than only to class labels or free-form explanations, while errors can be attributed to perception, template expressiveness, or unsupported activations. Appendix D provides exported rules and case-level visualizations for all three benchmarks.

5.6 Ablation Analysis

Table 3 compares the full framework with hard-atom inputs and TDRL-only prediction. The TDRL-only variant uses the same trained TDRL, probabilistic atoms, and rule bank as RIPA, with only subsequent MLLM reasoning and fusion disabled. Relative to this variant, RIPA improves accuracy by 7.16, 7.06, 5.00, and 10.71 percentage points on CLEVR-Hans3, CLEVR-Hans7, Kandinsky, and RAVEN-Custom, respectively. This comparison measures the joint contribution of rule-grounded MLLM reasoning and fusion with the symbolic component held fixed. Hard-atom inputs also reduce accuracy relative to RIPA on all four groups, supporting the use of probabilistic visual evidence.

6 Conclusion

RIPA connects probabilistic rule induction with rule-grounded MLLM prediction and verification through a fixed rule bank and decision-level feedback. Across the evaluated visual reasoning tasks, it improves accuracy over prompting-based and neuro-symbolic MLLM baselines while exposing rules and supporting evidence. The complete framework exceeds TDRL-only prediction by 5.00–10.71 percentage points, supporting the added value of the combined reasoning and fusion stages. Perceptual uncertainty and fixed template expressiveness remain limitations. Future work should improve visual grounding, uncertainty calibration, and the adaptability of the rule representation.

Reproducibility Statement

The training, rule-bank construction, and inference procedures are summarized in Algorithm 1. Appendix B specifies the task-specific template spaces, differentiable primitives, structural regularizers, and implementation settings; Appendix C describes the multimodal backbones and baseline protocols. Dataset construction, splits, and exported rules are provided in Appendix D, and Appendix E describes the atom-quality evaluation protocol and metrics. The evaluation settings and metrics are given in Section 5.1. Hyperparameters and fusion weights are selected on validation data and fixed for test evaluation.

AI Use Statement

Generative AI tools were used to assist with language polishing. The use of multimodal language models for probabilistic atom extraction and rule-grounded prediction and verification is part of the proposed method and is described in Sections 4.1 and 4.4; the evaluated models and protocols are detailed in Appendix C. We take responsibility for the final content of this work, including its text, claims, results, and references.

References

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: A visual language model for few-shot learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 23716–23736, 2022.

Samy Badreddine, Artur d'Avila Garcez, Luciano Serafini, and Michael Spranger. Logic tensor networks. *Artificial Intelligence*, 303:103649, 2022. doi: 10.1016/j.artint.2021.103649.

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, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025. doi: 10.48550/arXiv.2511.21631. URL https://arxiv.org/abs/2511.21631.

Xu Cao, Yifan Shen, Bolin Lai, Wenqian Ye, Yunsheng Ma, Joerg Heintz, Jintai Chen, Meihuan Huang, Jianguo Cao, Aidong Zhang, and James M. Rehg. What is the visual cognition gap between humans and multimodal LLMs? In *The Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=78lTuD6wiO.

Stefano Ceri, Georg Gottlob, and Letizia Tanca. What you always wanted to know about Datalog (and never dared to ask). *IEEE Transactions on Knowledge and Data Engineering*, 1(1):146–166, 1989. doi: 10.1109/69.43410.

Richard Evans and Edward Grefenstette. Learning explanatory rules from noisy data. *Journal of Artificial Intelligence Research*, 61:1–64, 2018. doi: 10.1613/jair.5714.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10764–10799. PMLR, 2023.

Dongwei Jiang, Marcio Fonseca, and Shay B. Cohen. LeanReasoner: Boosting complex logical reasoning with Lean. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 7497–7510, Mexico City, Mexico, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.416. URL https://aclanthology.org/2024.naacl-long.416/.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2901–2910, 2017.

Erich Peter Klement, Radko Mesiar, and Endre Pap. *Triangular Norms*, volume 8 of *Trends in Logic*. Kluwer Academic Publishers, Dordrecht, 2000. doi: 10.1007/978-94-015-9540-7.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, Saurav Kadavath, Shannon Yang, Thomas Henighan, Timothy Maxwell, Timothy Telleen-Lawton, Tristan Hume, Zac Hatfield-Dodds, Jared Kaplan, Jan Brauner, Samuel R. Bowman, and Ethan Perez. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*, 2023. doi: 10.48550/arXiv.2307.13702. URL https://arxiv.org/abs/2307.13702.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven C. H. Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19730–19742. PMLR, 2023.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 34892–34916, 2023.

Luca Salvatore Lorello, Marco Lippi, and Stefano Melacci. The KANDY benchmark: Incremental neuro-symbolic learning and reasoning with kandinsky patterns. *Machine Learning*, 114:161, 2025. doi: 10.1007/s10994-025-06798-x.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. Faithful chain-of-thought reasoning. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 305–329, Nusa Dua, Bali, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.ijcnlp-main.20. URL https://aclanthology.org/2023.ijcnlp-main.20/.

Robin Manhaeve, Sebastijan Dumančić, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. DeepProbLog: Neural probabilistic logic programming. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/dc5d637ed5e62c36ecb73b654b05ba2a-Abstract.html.

Robin Manhaeve, Sebastijan Dumančić, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Neural probabilistic logic programming in DeepProbLog. *Artificial Intelligence*, 298:103504, 2021. doi: 10.1016/j.artint.2021.103504.

Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B. Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *International Conference on Learning Representations*, 2019.

Stephen Muggleton. Inductive logic programming. *New Generation Computing*, 8(4):295–318, 1991. doi: 10.1007/BF03037089.

Stephen Muggleton and Luc De Raedt. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19–20:629–679, 1994. doi: 10.1016/0743-1066(94)90035-3.

Heimo Müller and Andreas Holzinger. Kandinsky patterns. *Artificial Intelligence*, 300:103546, 2021. doi: 10.1016/j.artint.2021.103546.

OpenAI. Introducing GPT-5.4. https://openai.com/index/introducing-gpt-5-4/, March 2026. Published March 5, 2026.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3806–3824, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.248. URL https://aclanthology.org/2023.findings-emnlp.248/.

Yifei Peng, Yaoli Liu, Enbo Xia, Yu Jin, Wang-Zhou Dai, Zhong Ren, Yao-Xiang Ding, and Kun Zhou. Abductive logical rule induction by bridging inductive logic programming and multimodal large language models. *arXiv preprint arXiv:2509.21874*, 2025. doi: 10.48550/arXiv.2509.21874. URL https://arxiv.org/abs/2509.21874.

Tim Rocktäschel and Sebastian Riedel. End-to-end differentiable proving. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/b2ab001909a8a6f04b51920306046ce5-Abstract.html.

Prithviraj Sen, Breno W. S. R. de Carvalho, Ryan Riegel, and Alexander Gray. Neuro-symbolic inductive logic programming with logical neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8212–8219, 2022. doi: 10.1609/aaai.v36i8.20795.

Hikaru Shindo, Devendra Singh Dhami, and Kristian Kersting. Neuro-symbolic forward reasoning. *arXiv preprint arXiv:2110.09383*, 2021a. doi: 10.48550/arXiv.2110.09383. URL https://arxiv.org/abs/2110.09383.

Hikaru Shindo, Masaaki Nishino, and Akihiro Yamamoto. Differentiable inductive logic programming for structured examples. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 5034–5041, 2021b. doi: 10.1609/aaai.v35i6.16637.

Hikaru Shindo, Viktor Pfanschilling, Devendra Singh Dhami, and Kristian Kersting. αILP: Thinking visual scenes as differentiable logic programs. *Machine Learning*, 112(5):1465–1497, 2023. doi: 10.1007/s10994-023-06320-1.

Hikaru Shindo, Viktor Pfanschilling, Devendra Singh Dhami, and Kristian Kersting. Learning differentiable logic programs for abstract visual reasoning. *Machine Learning*, 113(11–12):8533–8584, 2024. doi: 10.1007/s10994-024-06610-2.

Wolfgang Stammer, Patrick Schramowski, and Kristian Kersting. Right for the right concept: Revising neuro-symbolic concepts by interacting with their explanations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3619–3629, 2021.

Efthymia Tsamoura, Timothy Hospedales, and Loizos Michael. Neural-symbolic integration: A compositional perspective. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 5051–5060, 2021. doi: 10.1609/aaai.v35i6.16639.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Advances in Neural Information Processing Systems*, volume 36, pp. 74952–74965, 2023.

Ruocheng Wang, Eric Zelikman, Gabriel Poesia, Yewen Pu, Nick Haber, and Noah D. Goodman. Hypothesis search: Inductive reasoning with language models. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=G7UtIGQmjm.

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

Jundong Xu, Hao Fei, Liangming Pan, Qian Liu, Mong-Li Lee, and Wynne Hsu. Faithful logical reasoning via symbolic chain-of-thought. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13326–13365, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.720. URL https://aclanthology.org/2024.acl-long.720/.

Xi Ye, Qiaochu Chen, Isil Dillig, and Greg Durrett. SatLM: Satisfiability-aided language models using declarative prompting. In *Advances in Neural Information Processing Systems*, volume 36, pp. 45548–45580, 2023.

Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Joshua B. Tenenbaum. Neural-symbolic VQA: Disentangling reasoning from vision and language understanding. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/5e388103a391daabe3de1d76a6739ccd-Abstract.html.

Chi Zhang, Feng Gao, Baoxiong Jia, Yixin Zhu, and Song-Chun Zhu. RAVEN: A dataset for relational and analogical visual reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5317–5327, 2019.

A ALGORITHM OF RIPA

Algorithm 1 summarizes the training, rule-bank construction, and closed-loop inference procedures of RIPA.

Algorithm 1: Training and Inference of RIPA

Require: Training set $\mathcal{D}_{\mathrm{tr}}$, validation set $\mathcal{D}_{\mathrm{val}}$, test image x, and template space $\mathcal{T}$
Ensure: Predicted label $\hat{y}$

Extract and cache $A(x)$ for $x \in \mathcal{D}_{\mathrm{tr}} \cup \mathcal{D}_{\mathrm{val}}$
Initialize TDRL parameters $\Theta_{\mathcal{T}}$
for each training epoch **do**
    for $(x, y) \in \mathcal{D}_{\mathrm{tr}}$ **do**
        $s_{\mathrm{TDRL}}(x) \leftarrow f_{\mathrm{TDRL}}\left(A(x); \Theta_{\mathcal{T}}\right)$
        Update $\Theta_{\mathcal{T}}$ by minimizing $\mathcal{L}_{\mathrm{TDRL}}$
    end for
end for
$\mathcal{R} \leftarrow \mathrm{ExportAndRank}\left(\Theta_{\mathcal{T}}, \mathcal{D}_{\mathrm{val}}\right)$
Extract $A(x)$ and obtain $s_{\mathrm{TDRL}}(c \mid x)$ and $u_{c,j}(x)$
$\mathcal{C}^{\mathrm{cand}}(x) \leftarrow \mathrm{TopK}_c\ s_{\mathrm{TDRL}}(c \mid x)$
Retrieve $\mathcal{R}^{\mathrm{ret}}(x)$ from the fixed rule bank $\mathcal{R}$
Query the MLLM with x, $A(x)$, and $\mathcal{R}^{\mathrm{ret}}(x)$
Obtain $s_{\mathrm{MLLM}}(c \mid x)$ and $v_{c,j}(x)$
for $c \in \mathcal{C}^{\mathrm{cand}}(x)$ **do**
    Compute $s_{\mathrm{agree}}(c \mid x)$ and $s_{\mathrm{conflict}}(c \mid x)$
    Compute $s_{\mathrm{final}}(c \mid x)$
end for
return $\hat{y} = \arg\max_{c \in \mathcal{C}^{\mathrm{cand}}(x)} s_{\mathrm{final}}(c \mid x)$

B METHOD DETAILS

B.1 TASK-SPECIFIC TEMPLATE SPACE

TDRL does not search over unrestricted logic programs. Instead, each task defines a finite template space

$$\mathcal{T} = (\mathcal{S}, \mathcal{A}, \mathcal{V}_{\mathrm{rel}}, \mathcal{G}, \mathcal{P}, \Omega)$$

where $\mathcal{S}$ is the set of learnable object slots, $\mathcal{A}$ is the object-attribute vocabulary, $\mathcal{V}_{\mathrm{rel}}$ is the relation vocabulary, $\mathcal{G}$ contains optional global predicates characterizing overall scene structure, $\mathcal{P}$ is the admissible differentiable primitive library, and Ω specifies structural constraints such as the maximum number of candidate rules, distinct-object requirements, valid relation pairs, and allowed RAVEN transformations. Figure 5 provides a visual overview of the template spaces used in the experiments.

B.2 DIFFERENTIABLE PRIMITIVE LIBRARY

For consistency with the main text, let $\mathcal{R}_c^{\mathrm{cand}}$ denote the set of candidate rules maintained for class c. In the local definitions below, we fix a candidate rule $r \in \mathcal{R}_c^{\mathrm{cand}}$ and omit its index unless it is needed.

Soft object templates. For object slot s and attribute a, TDRL learns an attribute-value selector and an attribute gate,

$$\pi_s^a = \mathrm{softmax}(\theta_s^a), \qquad \alpha_s^a = \mathrm{sigmoid}(g_s^a),$$

where θ_s^a and g_s^a are rule-specific learnable parameters optimized during training. For an input image x, let p_i^a denote the probabilistic distribution of attribute a for object i, and let $\mu_i \in \{0, 1\}$ be its validity mask. The soft match between object i and slot s is

$$M_{i,s}(x) = \mu_i \prod_{a \in \mathcal{A}_{\mathrm{obj}}} \left[\alpha_s^a \left\langle p_i^a, \pi_s^a \right\rangle + 1 - \alpha_s^a\right].$$

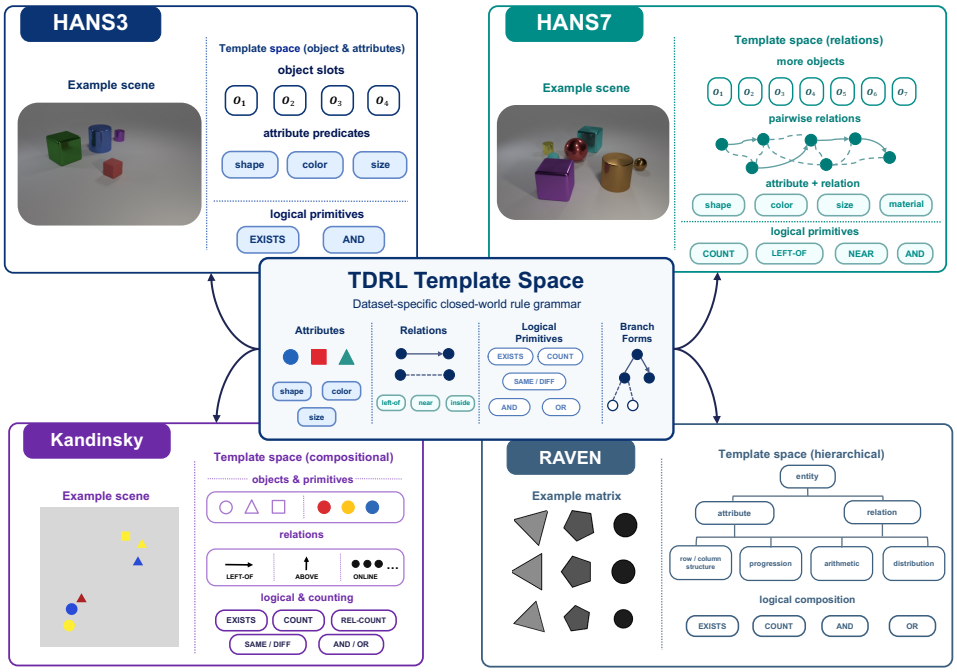


Figure 5: Task-specific template spaces used by TDRL. Hans3 uses existential object templates; Hans7 adds counting and directional-relation templates; Kandinsky adds pairwise similarity, difference, proximity, and global alignment predicates characterizing overall scene structure; RAVEN uses ordered three-panel transformation primitives over component attributes. The diagram shows admissible structures rather than learned rules.

The product ranges over the attributes of a single object. A disabled attribute contributes the neutral value 1, while μ_i ensures that padded or nonexistent objects have zero matching scores.

Existence and object counting. Let $O_{\max}$ denote the fixed maximum number of detected objects. The differentiable existential primitive is the probabilistic OR over object–slot matches,

$$h_s^{\text{exists}}(x) = 1 - \prod_{i=1}^{O_{\max}} \left(1 - M_{i,s}(x)\right).$$

The soft number of matching objects and the thresholded counting primitive are

$$N_s(x) = \sum_{i=1}^{O_{\max}} M_{i,s}(x),$$

$$h_s^{\text{count}}(x) = \text{sigmoid}\left(\frac{N_s(x) - \tau_s}{T_{\text{cnt}}}\right),$$

where τ_s is learnable and T_{cnt} controls the softness around the threshold.

Pairwise relation primitives. For source slot s and target slot s', TDRL learns a relation selector $\pi_{s,s'}^{\text{rel}}$ over the admissible relation types. Let $R_{i,j}$ denote the probabilistic relation distribution between objects i and j. Their pair score is

$$z_{i,j,s,s'}(x) = \mathbf{1}[i \neq j] M_{i,s}(x) M_{j,s'}(x) \times \left\langle R_{i,j}, \pi_{s,s'}^{\text{rel}} \right\rangle.$$

Relational existence is

$$h_{s,s'}^{\text{rel-exists}}(x) = 1 - \prod_{i \neq j} \left(1 - z_{i,j,s,s'}(x)\right).$$

Hans3 distinct-object matching. Hans3 contains two existential slots. To prevent both slots from grounding to the same object, its rule score uses distinct ordered object pairs,

$$h^{\text{distinct}}(x) = 1 - \prod_{i \neq j} \left(1 - M_{i,1}(x) M_{j,2}(x)\right).$$

Thus, the two slot conditions must be satisfied by two different visible objects.

Hans7 relational counting. For Hans7, relational counting has an "exists-a-source" interpretation. For each source object,

$$N^{\text{out}}_{i,s,s'}(x) = \sum_{j \neq i} z_{i,j,s,s'}(x),$$

and the primitive is

$$h^{\text{rel-count}}_{s,s'}(x) = 1 - \prod_{i} \left[1 - \text{sigmoid}\left(\frac{N^{\text{out}}_{i,s,s'}(x) - \tau^{\text{rel}}_{s,s'}}{T_{\text{cnt}}}\right)\right].$$

This primitive is high when at least one source object participates in sufficiently many selected relations.

Kandinsky pair relations. Kandinsky relations are derived from probabilistic shape and color atoms and the MLLM-predicted object centers. For objects i and j,

$$\begin{aligned} s^{\text{shape}}_{ij} &= \left\langle p^{\text{shape}}_i, p^{\text{shape}}_j \right\rangle, \\ s^{\text{color}}_{ij} &= \left\langle p^{\text{color}}_i, p^{\text{color}}_j \right\rangle, \\ s^{\text{same-both}}_{ij} &= s^{\text{shape}}_{ij} s^{\text{color}}_{ij}, \\ s^{\text{same-shape-diff-color}}_{ij} &= s^{\text{shape}}_{ij} \left(1 - s^{\text{color}}_{ij}\right). \end{aligned}$$

Let d_{ij} be the Euclidean distance between normalized centers and let κ be an adaptive distance threshold estimated from the median object span. The proximity score is

$$s^{\text{close}}_{ij} = \text{clip}\left(1 - \frac{d_{ij}}{\kappa}, 0, 1\right).$$

For Kandinsky relational counting, all valid ordered pairs are accumulated,

$$N^{\text{rel}}_{s,s'}(x) = \sum_{i \neq j} z_{i,j,s,s'}(x),$$

$$h^{\text{rel-count}}_{s,s'}(x) = \text{sigmoid}\left(\frac{N^{\text{rel}}_{s,s'}(x) - \tau^{\text{rel}}_{s,s'}}{T_{\text{cnt}}}\right).$$

Attribute-difference primitives use

$$d^{\text{shape}}_{ij} = 1 - s^{\text{shape}}_{ij}, \qquad d^{\text{color}}_{ij} = 1 - s^{\text{color}}_{ij},$$

and are aggregated with probabilistic OR over valid pairs. The REL-DIFF-BOTH primitive additionally multiplies a selected relation score by both difference terms. The global *all-aligned* atom characterizes overall scene structure and is computed from the residuals of object centers to their best-fitting line; short or only approximately collinear layouts receive a low value.

RAVEN constant primitive. For attribute a, component d, and three panel distributions $p^a_{d,1}$, $p^a_{d,2}$, and $p^a_{d,3}$, the constant score is

$$h^{a,\text{const}}_d = \left\langle p^a_{d,1}, p^a_{d,2} \right\rangle \left\langle p^a_{d,2}, p^a_{d,3} \right\rangle.$$

For position, dot products are replaced by fuzzy Jaccard similarities because a panel may occupy multiple spatial positions.

RAVEN progression primitive. For $\delta \in \{-2, -1, +1, +2\}$, let T_δ be the discrete transition matrix that maps value v to $v + \delta$ when the result remains in the attribute vocabulary. The progression score is

$$h^{a,\mathrm{prog}}_{d,\delta} = \left(p^a_{d,1}\right)^\top T_\delta p^a_{d,2} \cdot \left(p^a_{d,2}\right)^\top T_\delta p^a_{d,3}.$$

A softmax selector learns the progression step used by each candidate rule and attribute.

RAVEN arithmetic primitive. Let $\star_+$ and $\star_-$ denote probabilistic addition and subtraction over discrete values. The arithmetic scores are

$$h^{a,\mathrm{add}}_d = \left\langle p^a_{d,1} \star_+ p^a_{d,2}, p^a_{d,3} \right\rangle,$$
$$h^{a,\mathrm{sub}}_d = \left\langle p^a_{d,1} \star_- p^a_{d,2}, p^a_{d,3} \right\rangle.$$

Arithmetic is disabled for the type attribute because type IDs are categorical rather than numeric quantities.

RAVEN distribute-three primitive. The distribute-three score is high when the three panel values are pairwise different,

$$h^{a,\mathrm{dist3}}_d = \mathrm{clip}\Bigg[1 - \frac{1}{3}\Big(\left\langle p^a_{d,1}, p^a_{d,2} \right\rangle + \left\langle p^a_{d,1}, p^a_{d,3} \right\rangle + \left\langle p^a_{d,2}, p^a_{d,3} \right\rangle\Big), 0, 1\Bigg].$$

For each attribute, a softmax selector chooses among constant, progression, arithmetic, and distribute-three. When explicit relation-family or progression-step atoms are available, they are used as calibrated auxiliary evidence; the panel-level primitives remain sufficient for the current perception-only cache.

B.3 Structural Regularization

This section provides the explicit definitions of the structural regularizers in $\mathcal{L}_{\mathrm{TDRL}}$. The regularizers are instantiated according to the primitives available in each task-specific template. Terms that are not applicable to a template are set to zero. Let $\mathcal{R}^{\mathrm{cand}} = \bigcup_c \mathcal{R}^{\mathrm{cand}}_c$ denote the complete set of candidate rules. When summing across rules, we restore the rule index r to the local parameters introduced above.

Classification loss. For a mini-batch $\{(x_b, y_b)\}_{b=1}^B$, TDRL converts its class scores into logits before applying cross-entropy. For Hans3, Hans7, and Kandinsky,

$$\ell_c(x) = \log(s_{\mathrm{TDRL}}(c \mid x) + \epsilon_{\log}) + b_c,$$

where $\epsilon_{\log} > 0$ prevents taking the logarithm of zero, b_c is a learned class bias for Hans7 and Kandinsky, and $b_c = 0$ for Hans3. RAVEN-Custom directly produces a two-class logit vector from its binary rule score. The classification term is

$$\mathcal{L}_{\mathrm{CE}} = -\frac{1}{B}\sum_{b=1}^{B} \log \frac{\exp(\ell_{y_b}(x_b))}{\sum_c \exp(\ell_c(x_b))}.$$

Gate-sparsity regularization. Let $\mathfrak{G}_\mathcal{T}$ denote the collection of gate tensors available in template $\mathcal{T}$. It contains the attribute gates $\alpha^a_{r,s}$, primitive gates $\lambda_{r,t}$, and rule gates ρ_r, together with any task-specific primitive gates. We define

$$\mathcal{L}_{\mathrm{sp}} = \sum_{G \in \mathfrak{G}_\mathcal{T}} \frac{1}{|G|} \sum_{g \in G} g^\kappa,$$

where $\kappa = 1$ for Hans7 and Kandinsky and $\kappa = 2$ for Hans3. Normalization constants can be absorbed into λ_{sp}. Minimizing $\mathcal{L}_{\mathrm{sp}}$ suppresses unnecessary attributes, primitives, and candidate rules, thereby producing shorter exported rules.

Slot and rule diversity. To represent the concept learned by object slot s of candidate rule r, we concatenate its gated attribute selectors:

$$\mathbf{z}_{r,s} = \mathrm{concat}_{a \in \mathcal{A}_{\mathrm{obj}}} \left(\alpha^a_{r,s} \pi^a_{r,s} \right).$$

The slot-diversity penalty measures the squared cosine similarity between different slots in the same candidate rule:

$$\mathcal{L}_{\mathrm{slot}} = \frac{1}{Z_{\mathrm{slot}}} \sum_c \sum_{r \in \mathcal{R}^{\mathrm{cand}}_c} \sum_{s \neq s'} \cos^2 \left(\mathbf{z}_{r,s}, \mathbf{z}_{r,s'} \right),$$

where Z_{slot} is the number of valid ordered slot pairs in the sum.

A candidate-rule representation is obtained by concatenating all slot representations in that rule:

$$\mathbf{z}^{\mathrm{rule}}_r = \mathrm{concat}_s \, \mathbf{z}_{r,s}.$$

The rule-diversity penalty is

$$\mathcal{L}_{\mathrm{rule}} = \frac{1}{Z_{\mathrm{rule}}} \sum_c \sum_{\substack{r, r' \in \mathcal{R}^{\mathrm{cand}}_c \\ r \neq r'}} \cos^2 \left(\mathbf{z}^{\mathrm{rule}}_r, \mathbf{z}^{\mathrm{rule}}_{r'} \right),$$

where Z_{rule} is the number of valid ordered rule pairs in the sum. The diversity term groups these two components:

$$\mathcal{L}_{\mathrm{div}} = \omega_{\mathrm{slot}} \mathcal{L}_{\mathrm{slot}} + \omega_{\mathrm{rule}} \mathcal{L}_{\mathrm{rule}},$$

where ω_{slot} and ω_{rule} are separate regularization coefficients. When only one slot or one candidate rule is available, the corresponding term is set to zero. Minimizing $\mathcal{L}_{\mathrm{div}}$ discourages different slots or candidate rules from encoding redundant rule conditions.

Integer-threshold regularization. Let $\mathcal{H}_{\mathrm{cnt}}$ contain all learnable thresholds used by object-counting and relation-counting primitives, including $\tau_{r,s}$ and $\tau^{\mathrm{rel}}_{r,s,s'}$. We define

$$\mathcal{L}_{\mathrm{int}} = \frac{1}{|\mathcal{H}_{\mathrm{cnt}}|} \sum_{\tau \in \mathcal{H}_{\mathrm{cnt}}} \left(\tau - \mathrm{round}(\tau) \right)^2.$$

This term encourages the learned soft thresholds to approach integers, making the exported counting conditions easier to interpret. For templates without learnable counting thresholds, $\mathcal{L}_{\mathrm{int}} = 0$.

Non-empty-rule regularization. For candidate rule r, let

$$\bar{\lambda}_r = \frac{1}{|\mathcal{P}_r|} \sum_{t \in \mathcal{P}_r} \lambda_{r,t}$$

denote its average primitive-gate activation. The non-empty-rule penalty is

$$\mathcal{L}_{\mathrm{ne}} = \frac{1}{|\mathcal{R}^{\mathrm{cand}}|} \sum_{r \in \mathcal{R}^{\mathrm{cand}}} \rho_r \, \mathrm{ReLU} \left(\epsilon_{\mathrm{ne}} - \bar{\lambda}_r \right),$$

where ϵ_{ne} is a small minimum-activation margin, set to 0.12 in the Hans7 and Kandinsky implementations. The rule gate ρ_r ensures that the penalty is applied only when the candidate rule itself is selected. Thus, an active rule is discouraged from disabling all of its primitives.

Together, $\mathcal{L}_{\mathrm{sp}}$ favors concise rules, $\mathcal{L}_{\mathrm{div}}$ reduces redundant slots and candidate rules, $\mathcal{L}_{\mathrm{int}}$ improves the readability of counting thresholds, and $\mathcal{L}_{\mathrm{ne}}$ prevents sparsity regularization from producing active but vacuous rules.

B.4 Implementation Settings

Table 4 lists the main TDRL and closed-loop settings used by the current implementation. The four Kandinsky tasks share the same optimization and fusion settings, while RAVEN-Custom is trained separately for the Center, 2×2, and 3×3 configurations.

Table 4: Key implementation settings. Candidate Rules denotes the number of candidate rules maintained per class during TDRL training, whereas Top-J Rules denotes the number of retained rule-bank entries retrieved for inference.

Dataset / Task	Epochs	Learning Rate	Batch	Candidate Rules	Slots	Top-K Classes	Top-J Rules	$(\alpha, \beta, \gamma, \delta)$
Hans3	150	2×10^{-3}	256	1	2	2	1	$(1.5, 0.1, 0.1, 1.0)$
Hans7	150	8×10^{-4}	128	2	4	4	2	$(1.0, 1.0, 0.5, 0.5)$
Kandinsky	150	1×10^{-3}	128	2	2	2	2	$(1.5, 0.3, 0.2, 0.8)$
RAVEN Center	150	5×10^{-3}	512	1	1	2	1	$(1.0, 0.0, 0.05, 0.25)$
RAVEN 2×2	150	5×10^{-3}	512	1	1	2	1	$(1.0, 0.0, 0.5, 0.05)$
RAVEN 3×3	150	5×10^{-3}	512	1	1	2	1	$(2.0, 0.0, 1.0, 0.25)$

C Baselines and MLLM Details

C.1 Multimodal Backbones

Qwen3-VL-8B-Instruct. Qwen3-VL-8B-Instruct (Bai et al., 2025) is the smaller open-weight backbone used for probabilistic atom extraction and closed-loop verification.

Qwen3-VL-235B-A22B-Instruct. Qwen3-VL-235B-A22B-Instruct (Bai et al., 2025) is the higher-capacity Qwen3-VL backbone used in the main large-model experiments and in RAVEN-Custom. In our implementation, it is accessed through a local OpenAI-compatible vLLM endpoint. The same output schemas and task instructions are used as for the 8B model.

GPT-5.4. GPT-5.4 (OpenAI, 2026) is evaluated as a proprietary high-capacity MLLM accessed through an API. We treat it as a black-box multimodal model and do not rely on undisclosed architectural assumptions. The prompt content, candidate labels, and output constraints are kept aligned with the Qwen experiments.

C.2 MLLM Few-Shot

MLLM Few-shot directly predicts the query label from labeled support images and does not construct an explicit rule bank. We use 20 support images per class. This support size was selected in preliminary validation because it provided the best empirical trade-off between coverage of within-class variation and prompt length. Smaller support sets produced unstable hypotheses and more hallucinated class rules, whereas larger sets did not yield a consistent validation improvement.

For each task, the prompt is organized as follows:

1. a short task definition and the complete candidate-label set;
2. 20 labeled support images for each class, grouped by class;
3. an instruction to compare recurring visual evidence across support examples;
4. the unlabeled query image;
5. a strict output constraint requiring only one label from the candidate set.

Listing 1: Schematic organization of the MLLM Few-shot prompt.

```
System: You are given a visual classification task.
Use only the labeled support images to infer the class pattern
    .
Do not invent labels or return an explanation.

Candidate labels: <label_1, ..., label_C>

```

```
Support examples grouped by class:
Class <label_1>: <20 labeled images>
...
Class <label_C>: <20 labeled images>

Query image: <image>
Return exactly one label from the candidate set.
```

C.3 MLLM CHAIN-OF-THOUGHT

MLLM CoT (Wei et al., 2022) uses the same 20 support images per class but separates rule abstraction from query classification. In the first stage, the model summarizes the recurring conditions associated with each class, including object attributes, counts, relations, and overall scene structure. In the second stage, it checks the query image against these summaries and predicts a label. The rule summary is generated anew for each prompt and is not validated by an external symbolic executor.

Listing 2: Schematic organization of the MLLM CoT prompt.

```
Stage 1: From the labeled support images, summarize one
candidate visual rule for every class. Include all necessary
attribute, count, relation, and scene-level conditions.

Stage 2: Inspect the query image and test it against every
candidate rule. Select the best-supported class.

Output:
Class rules: <concise class-wise summaries>
Prediction: <one candidate label>
```

Compared with direct few-shot prediction, this protocol makes the intermediate hypothesis visible. However, because the hypothesis is free-form and generated within the same forward pass, it may omit a required conjunct, merge multiple alternatives, or introduce an unsupported relation.

C.4 ILP-COT

ILP-CoT (Peng et al., 2025) combines MLLM-based abstraction with inductive logic programming. The MLLM first converts the support images into symbolic descriptions and proposes candidate literals in a task-specific vocabulary. An ILP stage then searches for clauses that separate positive and negative examples. The resulting clauses are used to classify the query. In our comparison, the candidate label set and support budget are aligned with the prompt-based baselines. ILP-CoT differs from RIPA in three important respects: its symbolic descriptions are treated as discrete inputs, its rule search is not optimized jointly over probabilistic atoms, and the learned clauses are not fed back to an MLLM for image-specific test-time verification.

D DATASETS AND EXPORTED RULES

D.1 CLEVR-HANS

CLEVR-Hans (Stammer et al., 2021) is derived from CLEVR (Johnson et al., 2017) and introduces class-correlated nuisance attributes in the training distribution. The benchmark evaluates whether a model learns the underlying compositional rule rather than relying on a shortcut that fails under distribution shift. We use CLEVR-Hans3 and CLEVR-Hans7.

Hans3 contains three classes and primarily tests conjunctions of object-existence conditions. Each object is represented by shape, size, material, and color atoms. TDRL uses two learnable slots and distinct-object matching. Hans7 contains seven classes and adds general-position atoms, ordered directional relations, object counts, relational counts, and alternative rules. At most ten objects are retained in both variants.

Figure 6 compares the manually defined rules with TDRL exports. On Hans3, the exported rules recover the central object combinations, although some retain additional color or material literals. On Hans7, TDRL recovers the intended existence, counting, and relational rule families. The learned count thresholds remain soft, and some exported relation labels differ from the manually specified relations. The comparison distinguishes recovery of the rule family from recovery of every literal.

D.2 KANDINSKY PATTERNS

Kandinsky Patterns (Müller & Holzinger, 2021) contains synthetic images composed of colored geometric objects. We use four balanced binary tasks: *twopairs*, *closeby*, *red-triangle*, and *online-pair*. Every task contains $2{,}000$ training, 400 validation, and 400 test images, with equal numbers of true and false examples in each split.

The *twopairs* task requires two shape-based pair structures, including one same-shape/same-color pair and one same-shape/different-color pair. The *closeby* task requires at least one close object pair. The *red-triangle* task requires a red triangle and a nearby object with different shape and color. The *online-pair* task combines scene-level collinearity with a repeated shape-color pair. Figure 7 shows representative positive and negative images together with the ground-truth and exported rules. Across all four tasks, the exported rules preserve the main relational or scene-level structure; differences mainly appear as small shifts in learned count thresholds or source-target slot assignments.

D.3 RAVEN-CUSTOM

The original RAVEN benchmark (Zhang et al., 2019) is an eight-choice matrix-completion task. RAVEN-Custom converts it into fixed-rule, row-level binary classification. We use three configurations: Center, 2×2 (Distribute-Four), and 3×3 (Distribute-Nine). For each original problem, the first two entries of the third row are paired once with the correct candidate to form a true row and once with one sampled incorrect candidate to form a false row. The negative candidate is sampled with a fixed seed, producing a balanced $1{:}1$ binary task.

For each configuration, $1{,}400$ original problems are divided into $1{,}000$ training, 200 validation, and 200 test problems. After creating one true and one false row from every problem, each configuration contains $2{,}000$ training, 400 validation, and 400 test rows. Across all three configurations, the dataset therefore contains $6{,}000$ training, $1{,}200$ validation, and $1{,}200$ test rows.

Each row contains three ordered panels. RIPA extracts component-level distributions over number, occupied position, type, size, and color. TDRL evaluates constant, progression, arithmetic, and distribute-three transformations. Configuration identity is observed metadata used to route the relevant candidate rule; it is not itself sufficient to predict true or false. Figure 8 compares the predefined transformation conjunctions with the exported rules for the three configurations. The figure should be read literal by literal: a positive rule is supported only when every enabled attribute transformation holds.

E IMPACT OF MLLM CHOICE ON PROBABILISTIC ATOM QUALITY

To examine whether the choice of MLLM affects probabilistic atom extraction, we conduct an intrinsic evaluation on the CLEVR-Hans3 test set. Hans3 is used because its oracle scene descriptions provide exact object identities and attribute annotations, allowing the extracted probabilistic atoms to be directly compared with oracle atoms. We additionally evaluate the downstream TDRL performance obtained from each atom source while keeping the rule learner, training procedure, and optimization settings unchanged.

E.1 ORACLE ALIGNMENT PROTOCOL

Oracle atoms are constructed from the CLEVR scene descriptions associated with Hans3. For each test image n, the predicted and oracle objects are aligned by a one-to-one matching $\mathcal{M}_n$ that maximizes average attribute agreement over $\min(\widehat{O}_n, O_n)$ pairs. The matching is used to align attribute-level evaluation, while unmatched predicted and oracle objects contribute to the entity-count preci-

sion and recall defined below. The intrinsic atom-quality metrics are computed on the Hans3 test split.

E.2 Atom-Quality Metrics

Structured-output validity. The valid-output rate measures whether an MLLM response can be parsed and contains all required fields:

$$\text{ValidRate} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{1}\left[\text{response } n \text{ is valid}\right].$$

Entity-count F1 and count error. Because each $\mathcal{M}_n$ contains $\min(\widehat{O}_n, O_n)$ aligned pairs, entity-count precision, recall, and F1 measure consistency in the number of recovered objects:

$$P_{\text{ent}} = \frac{\sum_{n=1}^{N}|\mathcal{M}_n|}{\sum_{n=1}^{N}\widehat{O}_n}, \quad R_{\text{ent}} = \frac{\sum_{n=1}^{N}|\mathcal{M}_n|}{\sum_{n=1}^{N}O_n},$$
$$F1_{\text{ent}} = \frac{2P_{\text{ent}}R_{\text{ent}}}{P_{\text{ent}}+R_{\text{ent}}}.$$

The object-count mean absolute error is

$$\text{CountMAE} = \frac{1}{N}\sum_{n=1}^{N}\left|\widehat{O}_n - O_n\right|,$$

where $\widehat{O}_n$ and O_n denote the predicted and oracle object counts, respectively.

Attribute accuracy. For all matched object–attribute instances, attribute accuracy is

$$\text{AttrAcc} = \frac{1}{|\mathcal{I}|}\sum_{(n,i,a)\in\mathcal{I}} \mathbf{1}\left[\arg\max_{v} p^{a}_{n,i,v} = y^{a}_{n,i}\right]$$

where $\mathcal{I}$ contains the object–attribute instances induced by the matched pairs in $\{\mathcal{M}_n\}_{n=1}^{N}$, $p^{a}_{n,i,v}$ is the predicted probability of attribute value v, and $y^{a}_{n,i}$ is the corresponding oracle value.

Brier score. The quality of the complete attribute probability distribution is evaluated using the multiclass Brier score:

$$\text{BS} = \frac{1}{|\mathcal{I}|}\sum_{(n,i,a)\in\mathcal{I}}\frac{1}{|\mathcal{V}_a|}\sum_{v\in\mathcal{V}_a}\left(p^{a}_{n,i,v} - \mathbf{1}\left[y^{a}_{n,i}=v\right]\right)^2.$$

A lower Brier score indicates that the predicted probability vectors are closer to the oracle atoms.

Expected calibration error. Let $\widehat{p}_k$ denote the maximum predicted probability of attribute instance k, and let B_m denote the m-th confidence bin. The expected calibration error is

$$\text{ECE} = \sum_{m=1}^{M}\frac{|B_m|}{|\mathcal{I}|}\left|\text{acc}(B_m) - \text{conf}(B_m)\right|.$$

We use equal-width confidence bins and compute ECE over all matched attribute predictions.

Oracle downstream gap. To determine whether atom-extraction errors affect rule induction, a separate TDRL model is trained on the training-split cache for each atom source, selected using the corresponding validation-split cache, and evaluated on the corresponding test-split cache under identical settings. Test-split oracle atoms are used only for evaluation and never for training or model selection. The downstream oracle gap is

$$\Delta_{\text{oracle}} = \text{Acc}^{\text{oracle}}_{\text{TDRL}} - \text{Acc}^{\text{MLLM}}_{\text{TDRL}}.$$

A smaller gap indicates that the extracted atoms preserve more of the task-relevant symbolic information contained in the oracle representation.

Table 5: Probabilistic atom quality on the CLEVR-Hans3 test set relative to oracle atoms extracted from CLEVR scene descriptions.

Atom Source	Valid Rate ↑	Entity-count F1 ↑	Count MAE ↓	Attr. Acc. ↑	Brier ↓	ECE ↓	Δ_{oracle} ↓
Oracle atoms	1.000	1.000	0.000	1.000	0.000	0.000	0.000
Qwen3-VL-8B-Instruct	0.994	0.975	0.313	0.967	0.028	0.013	0.102
Qwen3-VL-235B-A22B-Instruct	1.000	0.999	0.013	0.997	0.003	0.025	0.013
GPT-5.4	1.000	0.998	0.026	0.963	0.029	0.049	0.095

Result analysis. Among the three MLLMs in Table 5, Qwen3-VL-235B-A22B-Instruct has the lowest object-count error, the highest attribute accuracy, and the lowest Brier score. Its downstream TDRL accuracy is 1.3 percentage points below the oracle-atom setting.

Qwen3-VL-8B-Instruct has a larger object-count error and a downstream oracle gap of 10.2 percentage points. Its ECE is the lowest among the MLLMs, but its attribute accuracy is below that of the 235B model. GPT-5.4 has near-perfect entity-count F1, but lower attribute accuracy and larger Brier and calibration errors than the 235B model; its downstream gap is 9.5 percentage points.

Valid-output rates are at least 99.4%, yet downstream oracle gaps range from 1.3 to 10.2 percentage points. A correctly structured response therefore does not ensure accurate object counts or attributes. TDRL multiplies atom truth values within each rule, so errors in these values can change rule activations. The 235B model has both the smallest oracle gap and the highest RIPA accuracy on Hans3 (Table 1).

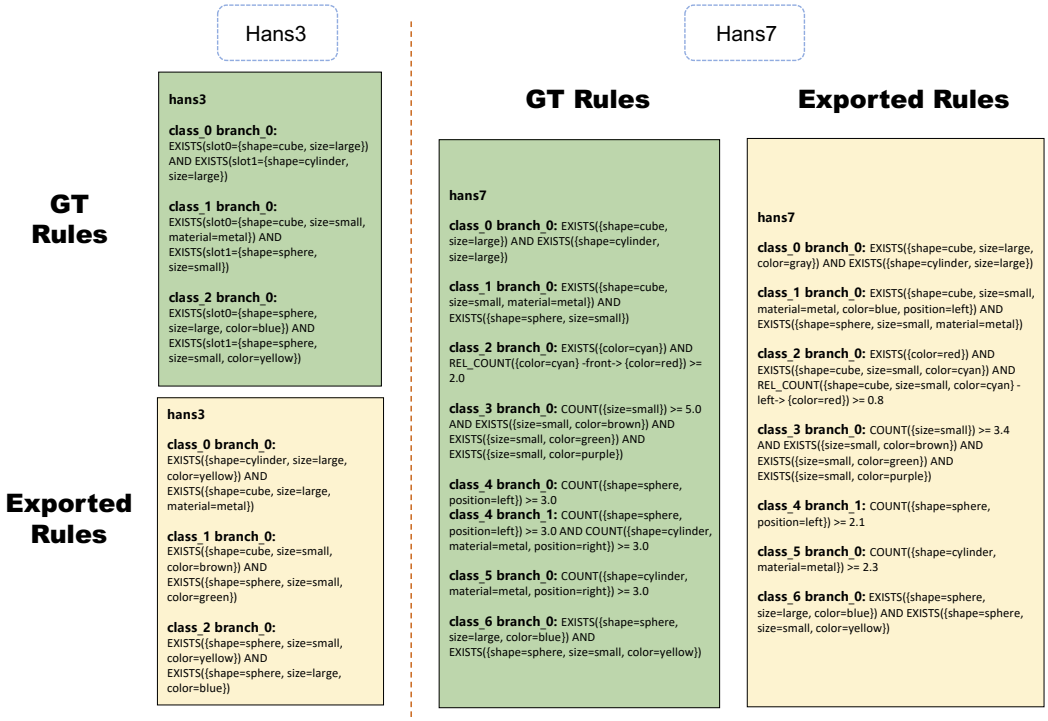


Figure 6: Ground-truth and exported rules on CLEVR-Hans. Hans3 primarily contains two-object existential conjunctions, whereas Hans7 additionally contains counting and relational conditions. The exported rules recover the main rule families and class-specific object anchors. Differences include extra attributes, soft count thresholds, and changed relation labels.

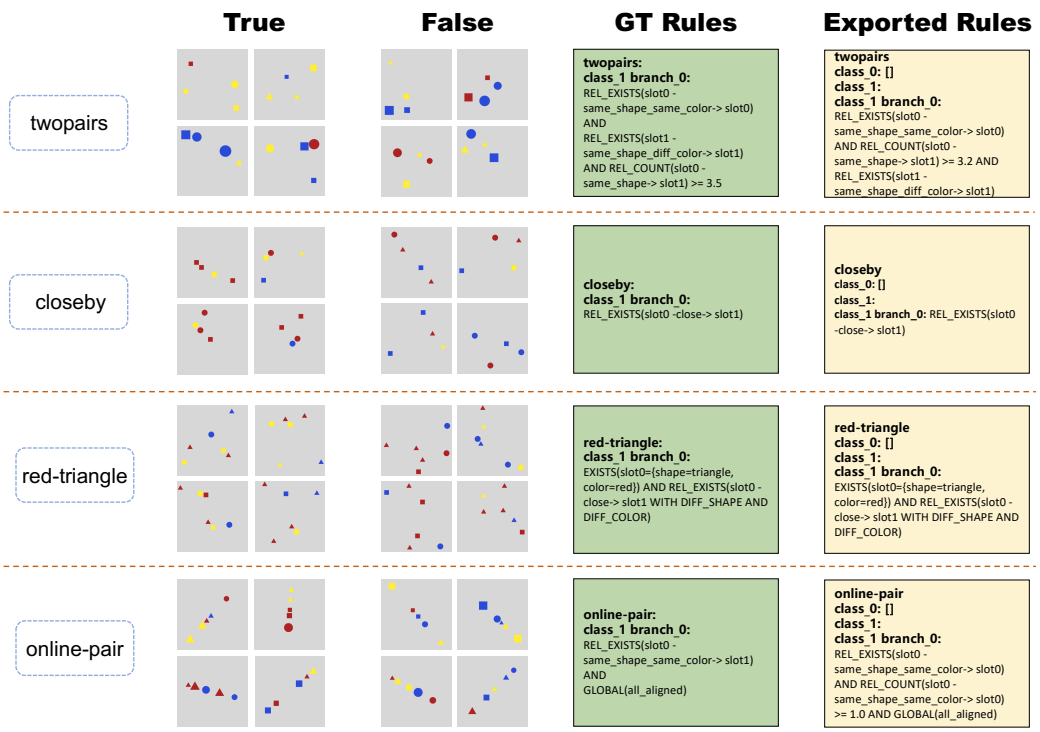


Figure 7: Representative positive and negative images, ground-truth rules, and TDRL exports for the four Kandinsky tasks. The learned rules recover the defining proximity, repeated-pair, attribute-difference, and scene-level alignment structures. Small deviations appear mainly in learned relation-count thresholds and equivalent slot assignments.

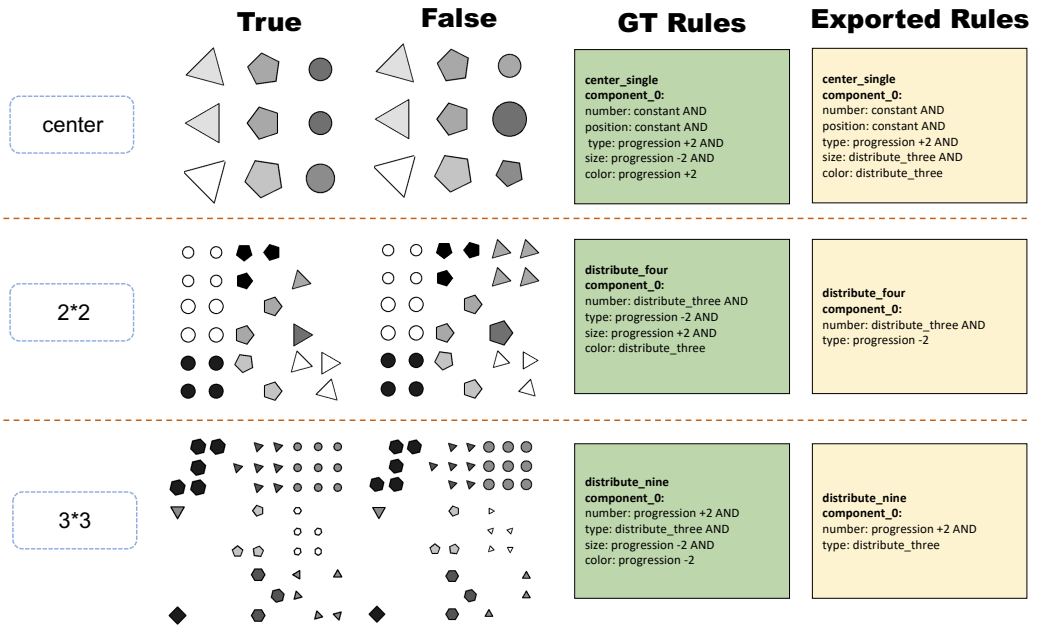


Figure 8: Ground-truth and exported rules for the Center, 2×2, and 3×3 RAVEN-Custom configurations. Each positive rule is a conjunction of component-level transformations over number, position, type, size, and color. The comparison shows which transformation families are recovered and where exported attribute selectors, progression steps, or enabled literals differ from the ground truth.