NExRI: Neural Explicit Rule Induction for Abstract Visual Reasoning

Anonymous authors
Paper under double-blind review

Abstract

Abstract visual reasoning requires inducing a relation from a few observed instances and re-applying it to novel ones. Existing models encode this relation implicitly in network weights, leaving it unobservable and unverifiable. We introduce **NExRI** (**N**eural **Ex**plicit **R**ule **I**nduction), a framework for end-to-end explicit rule induction on neural features, realized by two coupled modules. *IVaR* constructs contextual states that expose the latent relation by predicting the third panel from the first two and iteratively refining the prediction-observation discrepancy through verify-and-refine updates. *RaXO* turns these states into explicit rule objects that are validated on unseen evidence and executed on candidate completions. Instantiated as polynomial equations, the induced rules keep the fit tractable and the whole framework differentiable end-to-end. Experiments demonstrate state-of-the-art performance across multiple abstract visual reasoning benchmarks.

1 Introduction

Abstract visual reasoning (AVR) requires inducing a relation from a few observed instances and re-applying it to novel ones, offering a stringent test of generalization. Its canonical benchmark, Raven's Progressive Matrices (RPM), uses two rows to demonstrate a rule applied to a third (Zhang et al., 2019a). The rule must be inferred from sparse, visually diverse examples, where each instance is a composition of multiple attributes and even minor perturbations can break the relation. Large vision-language models fall short here, failing to extract a transferable rule from such sparse evidence (Zhang et al., 2026a). This capacity underpins visual analogy (Sadeghi et al., 2015), few-shot concept learning (Nie et al., 2020), and rule induction from grounded observation (Zhang et al., 2021), making AVR central to general-purpose visual intelligence (Małkiński & Mańdziuk, 2023).

Existing solvers fall into three families. Comparison-based methods encode it implicitly and rank candidates by compatibility over panel representations or completed matrices (Zhang et al., 2019b; Mondal et al., 2023). Predictive reasoning models take prediction discrepancy as the central signal and score candidates via learned computations over predictive evidence (Yang et al., 2025; Li et al., 2026). Explicit rule-induction models instead construct a rule from the observed context, typically in discrete symbolic or language-mediated form (Zhang et al., 2021; He et al., 2024). These families offer complementary implicit and explicit treatments of the latent relation.

Despite this progress, AVR still faces great challenges in generalizing to novel scenarios. The central remaining challenge is that explicit relational modeling and neural feature learning have not been reconciled in a single end-to-end framework (Garnelo & Shanahan, 2019). Neural solvers keep the relation implicit in network parameters, so the relation is never available as an object that can be checked against evidence it did not see, and selection rests on a training prior that compositional generalization is designed to defeat (Yang et al., 2025). Symbolic solvers expose a rule, but only in discrete symbolic or language-mediated forms that forgo end-to-end feature learning, require attribute parsers or annotations, and cannot be verified on learned latent states (Zhang et al., 2021).

We introduce **NExRI**, a framework for **N**eural **Ex**plicit **R**ule **I**nduction, where rules are explicitly induced from a few observed instances, represented as inspectable and re-executable objects, and learned end-to-end on neural features. NExRI decomposes this into two coupled tasks, each realized by a dedicated module, as shown in Figure 1. **IVaR** first predicts the third panel and treats the residual as evidence, then iteratively refines this evidence through a verify-and-refine loop, yielding neural features that make the underlying rule observable. **RaXO** (Rule-as-an-eXecutable-Object)

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

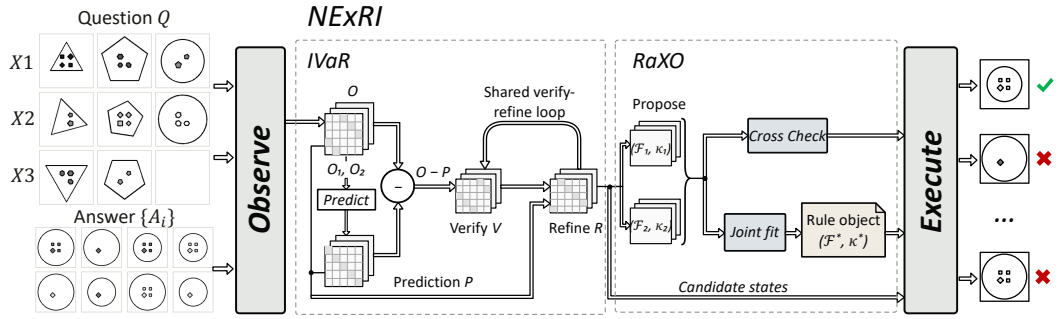


Figure 1: Overview of **NExRI**. **IVaR** first constructs neural features via a shared predict-and-verify and a verify-and-refine loop, making the underlying rule observable. Then, **RaXO** explicitly induces the rule from these features, validates it via cross-check, and jointly fits common constraints. The induced rule is instantiated as an inspectable and re-executable object, trained end-to-end.

turns these features into an explicit rule object, validates it against evidence not observed during induction, and fits a common constraint executed on candidate completions. The two modules instantiate NExRI by construction: the induced rule is differentiable and trainable end-to-end as a neural object, yet explicit, inspectable, and re-executable.

IVaR produces contextual states that expose the latent relation through two coupled loops. A predict-and-verify (PaV) loop (Yang et al., 2025) predicts the third panel from the first two and treats the prediction-observation discrepancy as residual evidence. However, this signal is static and cannot accumulate across iterations. IVaR thus adds a verify-and-refine (VaR) loop, where gated residual updates accumulate residuals into a verification state, and attention-weighted reads merge them with the original observations and predictions into a refinement state. Iterating this process enables later steps to revisit evidence under an evolving interpretation. Crucially, the discrepancy is a learned reasoning signal, not a calibrated violation measure, keeping feature construction end-to-end trainable.

Instead of implicit modeling, RaXO represents each rule as an explicit object: a function paired with a reference value that jointly defines an invariant for complete rows. A compiler derives this object from the contextual states of one complete row. *Cross Check* transfers each rule unchanged to the other complete row and assigns a soft support weight from its agreement there. All hypotheses proceed to *Joint Fit*, which reconciles each pair of proposals into a common rule. These rules remain fixed during candidate scoring, with evidence combined using the original support weights. Rules thus serve not as post-hoc explanations, but as the actual mechanism for evaluating candidates. We instantiate rules as polynomial equations, following prior work (He et al., 2024). Polynomials are one of many possible realizations, but their linear coefficients keep fitting tractable and the whole framework differentiable end-to-end.

Our contributions are threefold. 1) We introduce NExRI, the first framework to unify explicit rule modeling with neural feature learning in a single end-to-end network: relations are verifiable polynomial invariants on learned states, trained jointly by differentiating through the fitting step. 2) IVaR, a recurrent reasoning block, produces contextual states that expose the latent relation through coupled predict-and-verify and verify-and-refine loops. 3) RaXO turns these states into explicit rule objects: each complete row proposes a function and a reference, validated on the other row before a common constraint is fitted and executed on candidates. Experiments show state-of-the-art performance across abstract visual reasoning benchmarks, with ablations examining each component.

2 Related Work

AVR Tasks. AVR tasks assess whether a model can identify a relation persisting despite changes in its expressing objects (Małkiński & Mańdziuk, 2023). PGM and RAVEN establish the RPM protocol: two complete rows constrain a relation that a candidate must preserve in completing a third row: PGM varies attribute values and relational compositions, while RAVEN provides structured scene configurations (Barrett et al., 2018; Zhang et al., 2019a). I-RAVEN and RAVEN-FAIR refine

candidate generation to remove shortcuts, targeting latent relations rather than surface statistics (Hu et al., 2021; Benny et al., 2021). A-I-RAVEN omits attribute-rule pairings, probing generalization to novel scenarios (Małkiński & Mańdziuk, 2025). I-RAVEN-Mesh adds visual components and associated rules, testing composition from newly added elements (Małkiński & Mańdziuk, 2025). Bongard-LOGO and Bongard-OpenWorld extend the task to concept discrimination from positive and negative examples, inferring relations from few instances (Nie et al., 2020; Wu et al., 2024). Across these benchmarks, a relation must be induced from minimal complete evidence, such as two rows in RPM, a handful of instances in Bongard, and applied to instances with recombined attribute values and relations. Since each instance is under-determined on its own while training offers many, the question is how cross-instance regularities compensate.

AVR Solution Frameworks. Existing AVR solutions fall into three families. Early AVR models encode relations implicitly in the network. WReN aggregates pairwise panel relations and scores candidates (Barrett et al., 2018). CoPINet contrasts candidates against context panels (Zhang et al., 2019b). MRNet processes panels at multiple resolutions (Benny et al., 2021). STSN and Slot Abstractors extract object-centric representations (Mondal et al., 2023; 2024). Cross-instance regularities are captured by training across many instances, but at inference the relation is not derived from the current instance's rows. The network relies on a prior learned over the training distribution, leaving each instance's under-determination unresolved.

Predictive reasoning models make prediction and its discrepancy the central signal. PredRNet builds hierarchical prediction-error features that capture relational consistency across abstractions (Yang et al., 2025). Later work extends this line. HP^{2}AI stacks predict-and-verify blocks to iteratively refine predictions (He et al., 2024). CPCN uses predictive coding to disentangle relational rules from visual appearance (Zhang et al., 2025). DSRF augments this with multi-view features and gated attention to suppress irrelevant surface details (Li et al., 2026). DIRCR combines row-wise prediction with global inference under rule-contrastive supervision to resolve local ambiguities (Zhang et al., 2026b). The rule, however, remains implicit: encoded in the prediction network's parameters rather than constructed as an explicit object, and cannot be inspected.

Explicit rule-induction models build a rule from the observed context. TRIVR derives row-wise rules under a "2+1" formulation (He et al., 2025). PrAE abduces and executes probabilistic symbolic operators over scene representations (Zhang et al., 2021). ECo-NSR separates perception from symbolic reasoning via segmentation, representation, and analogical rule learning (Langer & Schmid, 2025). These methods improve interpretability and cross-configuration generalization, but still face the underdetermination problem: multiple rules can fit the observed context equally well, yet no mechanism disambiguates which one determines the correct answer.

3 Proposed NExRI

3.1 Overview of NExRI

An RPM presents two complete context rows $\boldsymbol{X}_1, \boldsymbol{X}_2$, the two observed panels of a query row $\boldsymbol{X}_3$, and K candidate panels $\{\boldsymbol{A}_i\}_{i=1}^{K}$. The task is to infer the relation shared by the complete rows and select the candidate that preserves it. A central difficulty is that existing models encode this relation implicitly in network weights: they can score candidates but cannot state what relation they rely on, and the latent relation is unobservable and unverifiable. NExRI instead treats the relation as an explicit rule object, a function $\mathcal{F}$ paired with a reference $\boldsymbol{\kappa}$, such that the complete rows satisfy

$$\mathcal{F}(\boldsymbol{X}_1) \approx \mathcal{F}(\boldsymbol{X}_2) \approx \boldsymbol{\kappa}. \tag{1}$$

A candidate is scored by how closely its completed row preserves this value, with agreement defined on function outputs rather than states, and the invariant remains puzzle-specific.

As shown in Figure 2, NExRI realizes this as an end-to-end trainable pipeline centered on making the rule an explicit, checkable object. A shared encoder maps panels to feature representations, and observation projection organizes them into attribute-specific views. IVaR converts these views into contextual row states that expose the latent relation. RaXO induces rule objects from the complete rows in this space, assigns soft support through cross-row verification, and executes the fitted rules on each candidate completion, with a classification head aggregating their weighted evidence into the answer score. Since every stage is differentiable and the rule is fixed by the complete rows alone, the framework trains end-to-end while each candidate is scored under a predetermined criterion.

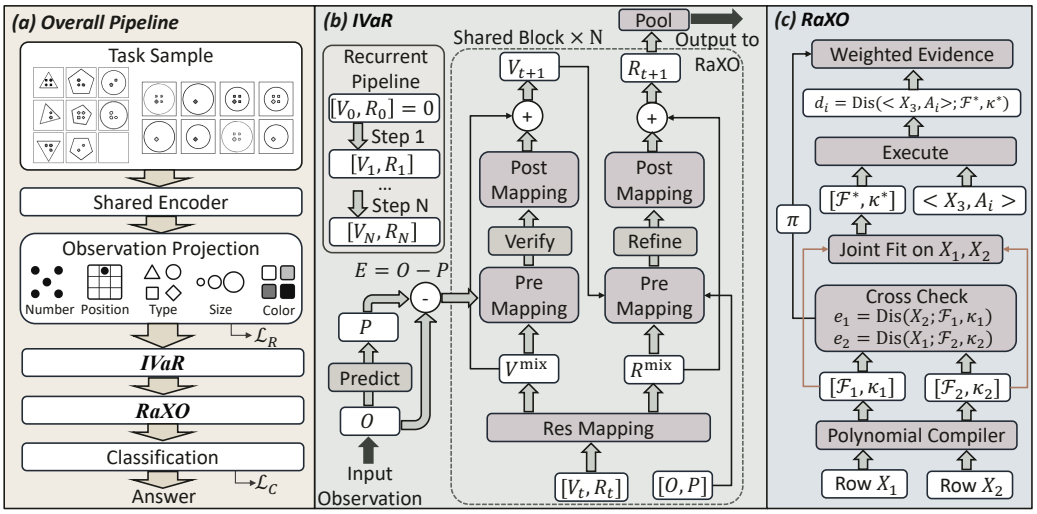


Figure 2: (a) Overall pipeline of **NExRI**. (b) **IVaR**: a recurrent block predicts the third panel, treats the residual as evidence, and iteratively refines it through verify-and-refine updates, exposing the latent relation in contextual states. (c) **RaXO**: rule objects are explicitly induced from these states, validated against the other complete row, and executed on candidates.

3.2 Iterative Verify-and-Refine (IVaR) Module

A relation between panels is not directly observable from raw features. It must be inferred from how panels differ and what remains consistent across them. IVaR constructs contextual row states that expose the latent relation through two coupled loops. The *predict-and-verify loop* predicts the third panel from the first two and treats the prediction-observation discrepancy as residual evidence. The *verify-and-refine loop* then iteratively accumulates this evidence into verification and refinement states, allowing the model to revisit the same evidence under an evolving interpretation.

3.2.1 Predict-and-Verify Loop

The input is a tensor $\boldsymbol{O} = (\boldsymbol{O}_1, \boldsymbol{O}_2, \boldsymbol{O}_3)$, with one feature map per panel position and $\boldsymbol{O}_j \in \mathbb{R}^{C \times L}$. Inspired by Yang et al. (2025), a predictor estimates the third panel from the first two:

$$\widehat{\boldsymbol{O}}_3 = f_{\text{pred}}(\boldsymbol{O}_1, \boldsymbol{O}_2), \quad \boldsymbol{P} = (\boldsymbol{O}_1, \boldsymbol{O}_2, \widehat{\boldsymbol{O}}_3), \quad \boldsymbol{E} = \boldsymbol{O} - \boldsymbol{P}. \tag{2}$$

The first two discrepancy maps are zero, while the third captures the residual between observation and prediction. We initialize $\boldsymbol{V}_0 = \boldsymbol{R}_0 = \boldsymbol{0}$ and keep $\boldsymbol{O}, \boldsymbol{P}, \boldsymbol{E}$ fixed across all steps.

Unlike prior work where the prediction network is trained to fit the relation and its error serves as a violation signal (Yang et al., 2025), here the predictor is not the locus of relation induction. Trained only via downstream reasoning objectives, it produces an initial discrepancy for the *verify-and-refine loop* to interpret, rather than a calibrated violation measure. The relation itself is constructed through iterative refinement, not encoded in the predictor's weights, so the predictor need not anticipate the final rule, and the discrepancy remains explicit evidence throughout refinement.

3.2.2 Verify-and-Refine Loop

A discrepancy alone does not identify its source, so IVaR maintains and refines two recurrent states: a verification state $\boldsymbol{V}_t$ accumulating evidence from $\boldsymbol{E} = \boldsymbol{O} - \boldsymbol{P}$, and a refinement state $\boldsymbol{R}_t$ integrating the original observations and predictions. Starting from $[\boldsymbol{V}_0, \boldsymbol{R}_0] = 0$, N shared blocks iteratively update this pair. At each step, a *Res Mapping* exchanges information between the two states. The *Verify Branch* updates $\boldsymbol{V}_{t+1}$, and the *Refine Branch* updates $\boldsymbol{R}_{t+1}$ using the original inputs and the new verification state. All parameters are shared across steps, rows, and views.

Residual Mapping. A doubly stochastic mixing exchanges information between the two states:

$$\boldsymbol{V}_t^{\text{mix}} = (1 - \rho_t)\boldsymbol{V}_t + \rho_t \boldsymbol{R}_t, \qquad \boldsymbol{R}_t^{\text{mix}} = \rho_t \boldsymbol{V}_t + (1 - \rho_t)\boldsymbol{R}_t. \tag{3}$$

This linear mixing lets each state absorb information from the other, while its doubly stochastic form does not increase the combined state norm during mixing.

Verify Branch. A single pass over $\boldsymbol{E}$ produces diffuse, uninterpretable evidence, so this branch accumulates discrepancy information over iterations. Its *Pre Mapping* dynamically reads current sources into a common space via learned attention,

$$\mathrm{Read}_t(\{\boldsymbol{S}_j\}) = \sum_j \alpha_{t,j} \boldsymbol{W}_j \boldsymbol{S}_j, \quad \boldsymbol{\alpha}_t = \mathrm{softmax}(\boldsymbol{q}_t), \tag{4}$$

where the block supplies the logits $\boldsymbol{q}_t$. At each step, it reads $\boldsymbol{E}$ alongside $\boldsymbol{V}_t^{\mathrm{mix}}$, which is zero at $t = 0$:

$$\boldsymbol{U}_{V,t} = \mathrm{Read}_{V,t}(\boldsymbol{E}, \boldsymbol{V}_t^{\mathrm{mix}}), \quad \boldsymbol{V}_{t+1} = \boldsymbol{V}_t^{\mathrm{mix}} + g_{V,t}\mathcal{G}_V(\boldsymbol{U}_{V,t}). \tag{5}$$

This re-weights evidence at every step. $\mathcal{G}_V$ applies gated feature interactions across panel positions, and the *Post Mapping* gate $g_{V,t}$ controls how much new evidence enters the persistent state before the residual addition, regulating the contribution of each update.

Refine Branch. The Refine Branch converts the accumulated verification evidence into a corrected representation by integrating the original observations $\boldsymbol{O}$, predictions $\boldsymbol{P}$, the updated verification state $\boldsymbol{V}_{t+1}$, and $\boldsymbol{R}_t^{\mathrm{mix}}$ through a dynamic read:

$$\boldsymbol{U}_{R,t} = \mathrm{Read}_{R,t}(\boldsymbol{O}, \boldsymbol{P}, \boldsymbol{V}_{t+1}, \boldsymbol{R}_t^{\mathrm{mix}}), \quad \boldsymbol{R}_{t+1} = \boldsymbol{R}_t^{\mathrm{mix}} + g_{R,t}\mathcal{G}_R(\boldsymbol{U}_{R,t}). \tag{6}$$

This dynamic re-weighting keeps refinement conditioned on verification evidence at each step. $\mathcal{G}_R$ combines verified evidence with the original context, producing a progressively corrected representation, while the write gate $g_{R,t}$ regulates the update. Verification and refinement thus form two coupled loops of evidence accumulation and evidence-guided reinterpretation, sharing all parameters across steps, rows, and views to enable iterative correction without introducing new parameters.

Spatial Readout (Pool). Pool compresses $\boldsymbol{R}_N$ into compact row states. Let m, q, j, l index the component, view, panel position, and spatial location, respectively. For each view q, the pooled vector passes through a shared projection ψ:

$$\boldsymbol{r}_{m,q,j} = \sum_l a_{j,m,l} \boldsymbol{R}_N(q, j, l), \quad \sum_l a_{j,m,l} = 1, \quad \boldsymbol{x}_{m,q,j} = \tanh(\psi(\boldsymbol{r}_{m,q,j})) \in [-1, 1]^n. \tag{7}$$

Here, n is the state dimension, and the spatial weights are reused across observation views.

3.3 Rule-as-an-eXecutable-Object (RaXO) Module

RaXO instantiates each relation as a function $\mathcal{F}$ paired with a reference $\boldsymbol{\kappa}$, such that complete rows satisfy $\mathcal{F}(\boldsymbol{X}_i) \approx \boldsymbol{\kappa}$. As shown in Figure 2c, it consists of five stages per component-view pair (index v omitted). The *Polynomial Compiler* proposes candidate rules from each complete row. *Cross Check* validates each proposal on the other complete row, converting cross-row cost into support weights $\boldsymbol{\pi}$. *Joint Fit* reconciles the paired proposals into a common rule $(\mathcal{F}^\star, \boldsymbol{\kappa}^\star)$. *Execute* applies this rule to each candidate row. *Weighted Evidence* aggregates the resulting costs under $\boldsymbol{\pi}$ into the evidence score. We instantiate rules as polynomials, which are nonlinear in the states yet affine in their coefficients, keeping fitting tractable and end-to-end differentiable.

Polynomial Compiler. The compiler turns one complete row $\boldsymbol{X}_i$ into H alternative rule objects, since the context may admit more than one relation. Each hypothesis contains D polynomial functions indexed by degree d. For a row $\boldsymbol{X} = (\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ of three panel states, let $\boldsymbol{\Phi}_d(\boldsymbol{X})$ contain the nonconstant monomials of total degree at most d, excluding $\boldsymbol{c}^d$. The compiler emits coefficients $\boldsymbol{\beta}_{i,h,d}$, instantiating a shared polynomial template

$$\mathcal{F}_d(\boldsymbol{X}; \boldsymbol{\beta}) = \boldsymbol{c}^d + \boldsymbol{\Phi}_d(\boldsymbol{X})\boldsymbol{\beta}, \quad \mathcal{F}_{i,h} = \big(\mathcal{F}_d(\,\cdot\,; \boldsymbol{\beta}_{i,h,d})\big)_{d=1}^{D}, \quad \boldsymbol{\kappa}_{i,h} = \big(\mathcal{F}_d(\boldsymbol{X}_i; \boldsymbol{\beta}_{i,h,d})\big)_{d=1}^{D}. \tag{8}$$

Each rule is thus a re-executable computation rather than a vector bound to $\boldsymbol{X}_i$. Decoupling instantiation from execution lets the same object transfer unchanged to the other complete row and to each candidate. Coefficients are shared across the n coordinates, bounded by $\|\boldsymbol{\beta}_{i,h,d}\|_\infty \le b_d$ via a scaled tanh, with the fixed $\boldsymbol{c}^d$ coefficient anchoring the scale and excluding the zero function. Their affine dependence on $\boldsymbol{\beta}$ keeps the fit tractable despite nonlinearity in the states.

Cross Check. A proposal satisfies its own row by construction, so agreement there is vacuous. Its function and reference are therefore held fixed while it is re-run on the other complete row. This check separates the evidence used to generate a proposal from the evidence used to weight it during candidate evaluation. Agreement is measured by

$$\mathrm{Dis}\big(\boldsymbol{X}; \mathcal{F}_{i,h}, \boldsymbol{\kappa}_{i,h}\big) = \frac{1}{nD} \sum_{d=1}^{D} \frac{\|\mathcal{F}_d(\boldsymbol{X}; \boldsymbol{\beta}_{i,h,d}) - \boldsymbol{\kappa}_{i,h,d}\|_2^2}{1 + \|\boldsymbol{\beta}_{i,h,d}\|_2^2}, \tag{9}$$

whose denominator prevents large coefficients from amplifying perturbations of the bounded states. With $\bar{i}$ denoting the row other than i, the two directions are

$$e_h^{(i)} = \mathrm{Dis}\big(\boldsymbol{X}_{\bar{i}}; \mathcal{F}_{i,h}, \boldsymbol{\kappa}_{i,h}\big), \qquad i \in \{1, 2\}, \tag{10}$$

and a learned scale $\tau > 0$ converts $\boldsymbol{e}^{(i)} = (e_h^{(i)})_{h=1}^{H}$ into support over hypotheses,

$$\boldsymbol{\pi}^{(i)} = \mathrm{softmax}\big(-\tau \boldsymbol{e}^{(i)}\big), \qquad \boldsymbol{\pi} = \tfrac{1}{2}\big(\boldsymbol{\pi}^{(1)} + \boldsymbol{\pi}^{(2)}\big). \tag{11}$$

Support is soft, not a hard filter: all H hypotheses proceed to fitting and *Weighted Evidence*, preserving ambiguity. Weights stay fixed during fitting and execution.

Joint Fit. The two complete rows can propose different coefficients and references for the same hypothesis, so candidates must be scored by one shared criterion. *Joint Fit* reconciles the two proposals into a common rule. Per hypothesis and degree, it solves

$$\min_{\|\boldsymbol{\beta}\|_\infty \le b_d,\, \boldsymbol{\kappa}} \quad \frac{1}{2} \sum_{i=1}^{2} \big(\|\boldsymbol{\beta} - \boldsymbol{\beta}_i\|_2^2 + \eta \|\boldsymbol{\kappa} - \boldsymbol{\kappa}_i\|_2^2\big) + \frac{\lambda}{n} \sum_{i=1}^{2} \|\mathcal{F}_d(\boldsymbol{X}_i; \boldsymbol{\beta}) - \boldsymbol{\kappa}\|_2^2, \tag{12}$$

balancing fidelity to the two source proposals against agreement between the two complete rows. Only $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ enter the fit. The incomplete query row and all candidates are excluded, so the shared rule is fixed before any completion is scored.

Execute. *Execute* applies the same fitted functions and references to each candidate row $\boldsymbol{X}_3^{(i)} = [\boldsymbol{X}_3; \boldsymbol{A}_i]$. The contextual states vary with the candidate, while the criterion used to evaluate them remains fixed. For hypothesis h, its cost is $d_h(i) = \mathrm{Dis}(\boldsymbol{X}_3^{(i)}; \mathcal{F}_h^\star, \boldsymbol{\kappa}_h^\star)$.

Weighted Evidence. Hypotheses are aggregated under their support using the same scale τ, giving $\ell(j) = \log \sum_h \pi_h \exp[-\tau d_h(j)]$. Component-view pairs are combined by a context-derived weight $w_v = \sigma(a_v) / \sum_u \sigma(a_u)$, where a_v is a shared linear readout of pair v's pooled refinement features averaged over the two complete rows and their panel positions, yielding $s_j = \sum_v w_v \ell_v(j)$ and $p_j = \exp(s_j) / \sum_{j'} \exp(s_{j'})$, with $\hat{\jmath} = \arg\max_j p_j$. Both weight families depend only on the complete context and stay fixed across candidates, so the score is a fixed weighted combination of execution costs rather than a learned compatibility function.

3.4 Training and Inference

We train jointly by $\mathcal{L} = \mathcal{L}_C + \lambda_R \mathcal{L}_R$, differentiating through *Joint Fit*. $\mathcal{L}_C$ is label-smoothed answer cross-entropy, and $\mathcal{L}_R$ is VICReg-style variance and cross-view covariance regularization (Bardes et al., 2022) over candidate panels before IVaR, without label-based selection. At inference, with all parameters frozen, only the relation coefficients and references are fitted to the two complete context rows, inducing each rule per puzzle from images alone.

4 Experimental Results

4.1 Experimental Setup

Datasets. We evaluate RAVEN (Zhang et al., 2019a), RAVEN-FAIR (Benny et al., 2021), I-RAVEN (Hu et al., 2021), I-RAVEN-Mesh, the four A-I-RAVEN attribute splits (Małkiński & Mańdziuk, 2025), and eight PGM regimes (Barrett et al., 2018). Bongard-LOGO (Nie et al., 2020) and Bongard-OpenWorld (Wu et al., 2024) test concept discrimination, and CLEVR-Matrices (Mondal et al., 2023) rendered-object reasoning. Tables report accuracy (%) on the official data splits, with standard deviations where available. Dataset-specific aggregation is defined in Section D.2.

Table 1: Accuracy (%) on RAVEN-family benchmarks, with SDs where available. **Bold** and underline mark best and second-best results. NExRI leads all eight settings.

	RAVEN family				A-I-RAVEN (OOD)				
Method	RAVEN	RAVEN-FAIR	I-RAVEN	I-RAVEN-Mesh	Color	Position	Size	Type	Avg.
WReN (2018)	16.8	30.3	23.8	25.7±0.2	16.9±0.5	17.3±0.4	12.4±0.5	15.1±0.7	19.8
CoPINet (2019b)	91.4	50.6	46.1	41.1±0.3	32.5±0.2	41.3±1.6	21.8±0.2	19.8±0.9	43.1
SRAN (2021)	54.3	72.9	60.8	57.8±0.2	38.3±1.0	56.9±0.7	34.4±3.0	30.7±2.2	50.8
SCL (2020)	83.9	90.1	95.0	80.9±1.5	65.1±2.0	76.7±7.1	65.6±2.4	49.5±1.8	75.9
Rel-Base (2020)	96.6	93.1	91.1	84.9±4.4	67.4±2.7	76.6±0.3	51.1±2.4	44.1±1.0	75.6
MRNet (2021)	89.7	88.4	83.5	79.5±2.0	33.6±8.2	62.6±2.6	20.6±5.0	19.4±0.3	59.7
STSN (2023)	96.9	95.4	95.7	48.7±11.5	39.3±6.9	36.1±19.9	38.4±16.6	39.1±5.0	61.2
DRNet (2024)	96.9	97.6	97.1	83.9±2.7	70.0±1.6	77.5±0.9	54.3±3.0	44.3±0.8	77.7
PredRNet (2025)	96.7	97.9	97.2	59.2±6.4	59.4±1.0	73.7±0.7	47.5±1.3	40.2±1.3	71.5
DSRF-L (2026)	99.1	99.1	99.4	92.3±1.3	70.8±1.1	78.7±0.8	80.8±1.1	55.6±0.8	84.5
NExRI (Ours)	**99.6**	**99.7**	**99.7**	**95.1**±0.8	**73.1**±0.9	**82.1**±0.7	**84.5**±0.9	**62.1**±1.1	**87.0**

Compared Methods. We compare with 20 methods from three groups. Implicit reasoning methods include WReN (Barrett et al., 2018), CoPINet (Zhang et al., 2019b), SRAN (Hu et al., 2021), SCL (Wu et al., 2020), Rel-Base (Spratley et al., 2020), MRNet (Benny et al., 2021), STSN (Mondal et al., 2023), DRNet (Zhao et al., 2024), ARII (Zhang et al., 2022), MLRN (Jahrens & Martinetz, 2020), and Slot Abstractors (Mondal et al., 2024). Predict-and-verify methods include PredRNet (Yang et al., 2025), HP^{2}AI (He et al., 2024), and DSRF-L (Li et al., 2026). Concept learning methods include B-SC and B-MoCo (Nie et al., 2020), Meta-Baseline (Chen et al., 2021), MetaOptNet (Lee et al., 2019), ProtoNet (Snell et al., 2017), and SNAIL (Mishra et al., 2018).

Implementation Details. For RAVEN-family benchmarks we use 80×80 panels, five observation views, and three spatial reads. IVaR performs two recurrent updates, and each of the four RaXO hypotheses contains polynomials of degrees one to three. Inference uses images alone. Diagnostic analyses use a frozen A-I-RAVEN-Type checkpoint at epoch 100 and are reported separately from the main benchmarks. Muon uses momentum 0.95, while Adam uses $\beta_1 = 0.9$ and $\beta_2 = 0.95$. The global gradient norm is clipped at 1. Experiments run on NVIDIA GeForce RTX 4090 hardware.

4.2 Comparison with State-of-the-Art Methods

RAVEN-Family. NExRI reaches 99.6, 99.7, and 99.7 on RAVEN, RAVEN-FAIR, and I-RAVEN versus 99.1, 99.1, and 99.4 for DSRF-L (Li et al., 2026) (Table 1). These near-ceiling results establish strong performance on standard matrix completion. I-RAVEN-Mesh adds a rule-governed mesh component and yields a larger gap of 95.1 versus 92.3. This result extends the improvement to an additional source of visual and relational structure beyond the original RAVEN attributes.

A-I-RAVEN tests previously unseen attribute-rule pairings, probing generalization beyond memorized pairings. NExRI leads all four splits, averaging 75.5 vs. 71.5 for DSRF-L (Li et al., 2026) and 55.2 for PredRNet (Yang et al., 2025) (Table 1), with gains over DSRF-L of 2.3, 3.4, 3.7, and 6.5 on Color, Position, Size, and Type, respectively. The largest gain occurs on Type, where DSRF-L has its lowest accuracy. This result is consistent with using each puzzle's context to define a fixed criterion for comparing candidates under unfamiliar attribute-rule pairings.

PGM Regimes. In Table 2, NExRI leads six of eight PGM regimes, including all five held-out regimes and neutral, averaging 65.4 versus 60.4 for PredRNet (Yang et al., 2025) and 54.9 for DSRF-L (Li et al., 2026). Relative to PredRNet, the largest gains occur on value transfer, while the already high HAP and HTP scores leave smaller room for improvement. NExRI exceeds DSRF-L across all eight regimes. Slot Abstractors (2024) leads interpolation and extrapolation at 91.6 and 39.3, ahead of NExRI by 0.2 and 0.5 points, respectively. NExRI thus improves the overall average while remaining competitive on value transfer.

Bongard-LOGO and -OpenWorld. Bongard tasks require applying a concept induced from support images to a query containing unseen instances. Table 3 shows that NExRI outperforms the compared baselines in every category on both benchmarks, achieving 97.3 vs. 76.9 for PredRNet (Yang et al., 2025) on LOGO. The gains span basic, free-form, compositional, and novel

Table 2: Accuracy on PGM. NExRI ranks first on six of eight settings and second on the other two.

		Attribute values			Held-out combinations				
Method	Avg.	N	I	E	HAP	HTP	HT	HALT	HASC
CoPINet (2019b)	-	56.4	51.2	16.4	-	-	-	-	-
WReN (2018)	32.4	62.6	64.4	17.2	27.2	41.9	19.0	14.4	12.5
MRNet (2021)	43.4	93.4	68.1	19.2	38.4	55.3	25.9	30.1	16.9
ARII (2022)	45.5	88.0	72.0	29.0	50.0	64.1	32.1	16.0	12.7
STSN (2023)	-	98.2	78.5	20.4	-	-	-	-	-
HP^{2}AI (2024)	-	99.3	80.0	22.6	-	-	-	-	-
Slot Abs. (2024)	51.9	91.5	**91.6**	**39.3**	63.3	78.3	20.4	16.7	14.3
PredRNet (2025)	60.4	99.1	75.3	22.9	99.3	98.9	41.2	33.4	13.1
DSRF-L (2026)	54.9	99.8	87.3	31.5	67.7	82.3	34.2	18.9	17.8
NExRI (Ours)	**65.4**	**99.9**	91.4	38.8	**99.4**	**99.2**	**42.4**	**33.6**	**18.1**

Table 3: Accuracy (%) on Bongard-LOGO (left) and Bongard-OpenWorld (right). Avg. equally weights the four displayed categories. Category definitions appear in Section D.2.

Method	FF	BA	CM	NV	Avg.
B-SC (2020)	66.3±0.6	73.3±1.3	63.5±0.3	63.9±0.8	66.8
B-MoCo (2020)	65.9±1.4	72.2±0.8	63.9±0.8	64.7±0.3	66.7
ProtoNet (2017)	64.6±0.9	72.4±0.8	62.4±1.3	65.4±1.2	66.2
PredRNet (2025)	76.6±0.1	92.9±0.2	69.0±0.9	69.0±1.1	76.9
NExRI (Ours)	**97.5**±0.7	**97.8**±0.9	**97.1**±0.8	**96.9**±0.4	**97.3**

Method	Short	Long	CS	Non-CS	Avg.
Meta-Base. (2021)	59.6	52.7	55.5	56.9	56.2
MetaOptNet (2019)	63.3	51.6	50.9	60.7	56.6
ProtoNet (2017)	59.2	57.7	51.8	61.0	57.4
SNAIL (2018)	66.1	61.5	63.6	64.1	63.8
NExRI (Ours)	**97.2**	**96.6**	**96.1**	**97.4**	**96.8**

concepts, consistent with using support-derived constraints to evaluate new examples. On OpenWorld, NExRI surpasses SNAIL (Mishra et al., 2018) by more than 30 percentage points in each category, with the largest margin on Long, whose concepts have longer descriptions.

CLEVR-Matrices. On CLEVR-Matrices, NExRI achieves the highest average accuracy of 99.9% and ties for first in Logic and Count (Table 4). As leading compared models also approach the performance ceiling, these results mainly validate NExRI's effectiveness on rendered objects.

Computational Cost. Table 5 reports NExRI at 10.14 GFLOPs and 5.13M parameters, both below DSRF-L (Li et al., 2026) (22.10 GFLOPs, 42.29M), demonstrating a favorable accuracy-efficiency trade-off relative to this comparable baseline. PredRNet Yang et al. (2025), however, remains markedly lighter (2.05 GFLOPs, 1.28M). NExRI's accuracy gains thus entail a computational premium over that method.

Table 4: Accuracy on CLEVR-Matrices.

Method	Logic	Location	Count	Avg.
MLRN (2020)	47.4	21.4	23.6	30.8
SCL (2020)	80.9	65.8	64.9	70.5
STSN (2023)	99.2	**100.0**	99.6	99.6
PredRNet (2025)	**100.0**	99.5	**99.9**	99.8
NExRI (Ours)	**100.0**	99.8	**99.9**	**99.9**

Table 5: GFLOPs and model size.

Method	GFLOPs ↓	Params (M) ↓
SRAN (2021)	11.27	45.75
MRNet (2021)	3.58	4.96
PredRNet (2025)	**2.05**	1.28
STSN (2023)	109.85	**0.57**
HP^{2}AI (2024)	12.82	6.38
DSRF-L (2026)	22.10	42.29
NExRI (Ours)	10.14	5.13

4.3 Ablation Studies

Table 6 summarizes ablations on A-I-RAVEN.

Ablation of Major Modules. We ablate the two major modules by replacing *IVaR* with *PaV*, and *RaXO* with a simple *Neural Scorer*. The baseline achieves 57.2%. Restoring IVaR alone improves accuracy to 62.3%, restoring RaXO alone to 66.3%, and restoring both yields 75.5%. Since the combined gain over the baseline exceeds the sum of the individual gains, the two modules exhibit a positive interaction.

Frozen Representations. To isolate the answer criterion, we freeze a trained IVaR representation and retrain each readout on these fixed states. RaXO achieves 75.1% accuracy versus 69.6% for the neural scorer, a 5.5-point improvement under identical features. The full model (75.5%) remains 0.4 points above the frozen RaXO variant.

Table 6: Component ablations on A-I-RAVEN. Avg. equally weights the four attribute splits.

Variants	Color	Position	Size	Type	Avg
PaV + Neural Scorer	60.8±1.1	74.5±0.6	51.2±1.4	42.4±1.2	57.2
IVaR + Neural Scorer	64.2±0.9	76.6±0.8	60.5±1.3	47.9±1.1	62.3
PaV + RaXO	66.9±1.0	78.3±0.6	67.9±1.2	52.2±1.3	66.3
IVaR + RaXO	73.1±0.9	82.1±0.7	84.5±0.9	62.1±1.1	75.5
Frozen IVaR + Neural Scorer	68.2±1.1	77.6±0.6	78.4±0.8	54.2±1.2	69.6
Frozen IVaR + RaXO	72.7±0.7	81.7±0.8	84.1±1.0	61.8±0.9	75.1
Discrepancy-only IVaR	70.0±1.1	79.2±0.8	80.6±1.0	57.8±1.3	71.9
Identity state mixing	72.7±0.8	81.8±0.6	84.0±1.0	61.7±1.2	75.1
Uniform hypothesis weights	71.5±1.0	80.7±0.9	82.1±1.1	60.2±1.2	73.6
Post-fit support weights	72.4±0.8	81.6±0.9	83.7±1.0	61.4±1.1	74.8

Table 7: Frozen Type diagnostics: 448 puzzles for weighting, 64 Center puzzles for pairing and transfer.

Weights	Acc.↑	NLL↓
Pre-fit	62.1	1.3
Post-fit	60.9	1.4
Uniform	60.9	1.4
Pairing	**Mean JS (×100)↓**	
Matched		3.4
Shuffled		19.3
Within-rule		8.8
Source	**Acc.↑**	**Margin (×100)↑**
Native	71.9	14.9
Same rule	70.3	14.2
Different rule	62.5	7.6

Individual Component Changes. Discrepancy-only IVaR, which removes observations and predictions as separate inputs, drops to 71.9%, indicating that retaining these sources benefits relational state construction. Identity state mixing replaces Residual Mapping with identity and retains 75.1%, suggesting a smaller contribution from mixing in this comparison. For RaXO, both uniform hypothesis weights (73.6%) and post-fit support weights (74.8%) underperform the default pre-fit weighting (75.5%). This ordering favors retaining cross-row support computed before joint fitting, when the checked row has not yet been used to fit the proposal being assessed.

Table 7 analyzes RaXO from a frozen Type checkpoint on A-I-RAVEN.

Support for Candidate Selection. We fix all components except the hypothesis weights. Pre-fit evaluates each row's proposal on the other row, post-fit evaluates on the rows used for fitting, and uniform assigns equal weight to all hypotheses. On this subset, pre-fit achieves the highest observed accuracy (62.1%) and lowest NLL (1.3), versus 60.9% and 1.4 for both post-fit and uniform. Since the fitted relations and candidate costs remain fixed, this comparison isolates the effect of hypothesis weighting. Cross-row evaluation supplies support before joint fitting, keeping proposal assessment separate from the agreement obtained by optimizing on both context rows.

Cross-Row Hypothesis Agreement. We compare support distributions from row pairs: matched (same puzzle), within-rule (different puzzles sharing the same rule), and shuffled (no rule restriction). JS divergence ($\times 100$) is 3.4 for matched pairs, 8.8 for within-rule, and 19.3 for shuffled. This ordering is consistent with sensitivity to both the rule and the puzzle context.

Testing Relation Transfer. We replace Type coefficients with Size or Color coefficients from a donor with the same rule or a different rule. Native retains the target's own coefficients. Same-rule transfer stays closer to native accuracy and margin (70.3%, 14.2 vs. 71.9%, 14.9) than different-rule transfer (62.5%, 7.6). Rule-matched donors retain more of the target's performance under this intervention.

5 Conclusion

We introduced NExRI, a framework that unifies explicit rule induction with neural feature learning in a single end-to-end network. IVaR accumulates predictive evidence through recurrent reasoning over contextual row states, while RaXO instantiates each rule as an explicit polynomial with context-induced coefficients held fixed during candidate evaluation. This separation of state construction from rule execution yields strong performance across RAVEN, A-I-RAVEN, PGM, Bongard, and CLEVR-Matrices, with substantial improvements on compositional and novel concepts in the Bongard evaluations.

The hardest held-out combinations remain challenging. Future work will investigate richer relation families and study cross-task transfer of learned coefficients.

AI USE STATEMENT

Generative AI tools were mainly used to assist with language editing. They also supported selected visualization tasks. The authors reviewed and verified the resulting text and visualizations.

ETHICS STATEMENT

This study uses publicly available visual reasoning benchmarks and does not involve recruiting human participants or collecting new personal data. We have identified no specific ethical concerns.

REPRODUCIBILITY STATEMENT

Section 3 and Appendix B describe the model formulation and inference procedure. Dataset configurations, splits, and evaluation metrics are specified in Appendix D.2. Appendix D.1 reports architecture, training objectives, and optimization settings, while Appendix E details the additional experimental protocols.

REFERENCES

Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations (ICLR)*, 2022.

David Barrett, Felix Hill, Adam Santoro, Ari Morcos, and Timothy Lillicrap. Measuring abstract reasoning in neural networks. In *International conference on machine learning (ICML)*, pp. 511–520. PMLR, 2018.

Yaniv Benny, Niv Pekar, and Lior Wolf. Scale-localized abstract reasoning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 12557–12565, 2021.

Yinbo Chen, Zhuang Liu, Huijuan Xu, Trevor Darrell, and Xiaolong Wang. Meta-baseline: Exploring simple meta-learning for few-shot learning. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9042–9051. IEEE, 2021.

Marta Garnelo and Murray Shanahan. Reconciling deep learning with symbolic artificial intelligence: representing objects and relations. *Current Opinion in Behavioral Sciences*, 29:17–23, 2019.

Wentao He, Jianfeng Ren, Ruibin Bai, and Xudong Jiang. Hierarchical perceptual and predictive analogy-inference network for abstract visual reasoning. In *Proceedings of the 32nd ACM International Conference on Multimedia (ACM MM)*, pp. 4841–4850, 2024.

Wentao He, Jianfeng Ren, Ruibin Bai, and Xudong Jiang. Two-stage rule-induction visual reasoning on rpms with an application to video prediction. *Pattern Recognition (PR)*, 160:111151, 2025.

Sheng Hu, Yuqing Ma, Xianglong Liu, Yanlu Wei, and Shihao Bai. Stratified rule-aware network for abstract visual reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 35, pp. 1567–1574, 2021.

Marius Jahrens and Thomas Martinetz. Solving raven's progressive matrices with multi-layer relation networks. In *2020 International joint conference on neural networks (IJCNN)*, pp. 1–6. IEEE, 2020.

Johannes Langer and Ute Schmid. Flexible segmentation for rule learning over object representations in abstract visual reasoning. In *Twelfth Annual Conference on Advances in Cognitive Systems*, 2025.

Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 10649–10657. IEEE, 2019.

Chengtai Li, Yuting He, Jianfeng Ren, Ruibin Bai, Yitian Zhao, and Xudong Jiang. Dsrf: A dynamic and scalable reasoning framework for solving rpms. volume 38, pp. 66719–66753, 2026.

Mikołaj Małkiński and Jacek Mańdziuk. A review of emerging research directions in abstract visual reasoning. *Information Fusion*, 91:713–736, 2023.

Mikołaj Małkiński and Jacek Mańdziuk. A-i-raven and i-raven-mesh: two new benchmarks for abstract visual reasoning. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, 2025.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In *International Conference on Learning Representations (ICLR)*, 2018.

Shanka Subhra Mondal, Taylor Webb, and Jonathan D Cohen. Learning to reason over visual objects. 2023.

Shanka Subhra Mondal, Jonathan D. Cohen, and Taylor Whittington Webb. Slot abstractors: Toward scalable abstract visual reasoning. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 36088–36105. PMLR, 21–27 Jul 2024.

Weili Nie, Zhiding Yu, Lei Mao, Ankit B Patel, Yuke Zhu, and Anima Anandkumar. Bongard-logo: A new benchmark for human-level concept learning and reasoning. volume 33, pp. 16468–16480, 2020.

Fereshteh Sadeghi, C. Lawrence Zitnick, and Ali Farhadi. Visalogy: Answering visual analogy questions. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 28. Curran Associates, Inc., 2015.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. volume 30, 2017.

Steven Spratley, Krista Ehinger, and Tim Miller. A closer look at generalisation in raven. In *European Conference on Computer Vision (ECCV)*, pp. 601–616. Springer, 2020.

Rujie Wu, Xiaojian Ma, Zhenliang Zhang, Wei Wang, Qing Li, Song-Chun Zhu, and Yizhou Wang. Bongard-openworld: Few-shot reasoning for free-form visual concepts in the real world. In *International Conference on Learning Representations (ICLR)*, volume 2024, pp. 20688–20718, 2024.

Yuhuai Wu, Honghua Dong, Roger Grosse, and Jimmy Ba. The scattering compositional learner: Discovering objects, attributes, relationships in analogical reasoning. *arXiv preprint arXiv:2007.04212*, 2020.

Zhenda Xie, Yixuan Wei, Huanqi Cao, Chenggang Zhao, Chengqi Deng, Jiashi Li, Damai Dai, Huazuo Gao, Jiang Chang, Kuai Yu, et al. mhc: Manifold-constrained hyper-connections. *arXiv preprint arXiv:2512.24880*, 2025.

Lingxiao Yang, Xiaohua Xie, Wei-Shi Zheng, Fang Fang, and Ru-Yuan Zhang. Neural prediction errors as a unified cue for abstract visual reasoning. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2025.

Beichen Zhang, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Haodong Duan, Dahua Lin, and Jiaqi Wang. Think visually, reason textually: Vision-language synergy in abstract reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 41203–41212, 2026a.

Chi Zhang, Feng Gao, Baoxiong Jia, Yixin Zhu, and Song-Chun Zhu. Raven: A dataset for relational and analogical visual reasoning. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5312–5322. IEEE, 2019a.

Chi Zhang, Baoxiong Jia, Feng Gao, Yixin Zhu, Hongjing Lu, and Song-Chun Zhu. Learning perceptual inference by contrasting. volume 32, 2019b.

Chi Zhang, Baoxiong Jia, Song-Chun Zhu, and Yixin Zhu. Abstract spatial-temporal reasoning via probabilistic abduction and execution. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9731–9741. IEEE, 2021.

Jiachen Zhang, Chengtai Li, Jianfeng Ren, Linlin Shen, Zheng Lu, and Ruibin Bai. Dircr: Dual-inference rule-contrastive reasoning for solving ravens. In *ICASSP 2026-2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 9062–9066. IEEE, 2026b.

Wenbo Zhang, Site Mo, Xianggen Liu, Sen Song, et al. Learning robust rule representations for abstract reasoning via internal inferences. volume 35, pp. 33550–33562, 2022.

Xinyu Zhang, Lingling Zhang, Yanrui Wu, Muye Huang, and Jun Liu. Cognitive predictive coding network: Rethinking the generalization in raven's progressive matrices. In *Proceedings of the 33rd ACM International Conference on Multimedia (ACM MM)*, pp. 4097–4106. Association for Computing Machinery, 2025. ISBN 9798400720352.

Kai Zhao, Chang Xu, and Bailu Si. Learning visual abstract reasoning through dual-stream networks. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 38, pp. 16979–16988, 2024.

A SIGNIFICANCE

NExRI contributes an operational view of what it means for a neural model to learn a relation. A relation is represented by a computation whose output should remain invariant across its instances. This view makes the induced relation itself an object that can be proposed, tested, revised, and reused. It gives visual reasoning research a concrete way to test an induced computation on additional evidence and examine whether it remains valid when its inputs change.

This perspective connects explicit relational modeling with learned representation. Explicitness resides in the computation performed on learned latent states. IVaR and RaXO instantiate this connection: state construction preserves evidence for relation induction, while relation execution determines how those states contribute to an answer. Training through execution makes the ability to support a reusable computation a criterion for learning the representation.

NExRI provides a general framework for context-based reasoning tasks that can be expressed through relation induction and query execution. Its interface consists of a support context for relation induction and query instances for execution, with grouping adapted to the task. RPM realizes the interface through row completion, while Bongard uses it for concept discrimination from positive and negative support examples. Task-specific grouping and verification adapt to the available context structure, as illustrated by the Bongard adaptation in Section D.1.

An executable relation also provides an interface for evaluating the reasoning process beyond final accuracy. Holding learned states fixed allows coefficients, support weights, or fitting choices to be changed separately and their effects on candidate evidence to be measured. The same explicit computation admits mathematical analysis of execution and ranking stability under perturbations of the learned states. These capabilities support a more direct study of which inferred relations are supported by evidence, how they affect decisions, and where their reuse fails. This interface offers a basis for studying richer relation families and transferring executable reasoning across tasks.

B ADDITIONAL METHOD DETAILS

We give implementation details and the fitting solver omitted from Sections 3.2 and 3.3. For the RPM formulation below, $s \in \{1, 2\}$ indexes complete context rows, $i \in \{1, \dots, K\}$ indexes candidate answers, and $v = (m, q)$ identifies a spatial component and observation view.

B.1 ITERATIVE VERIFY-AND-REFINE (IVAR)

Panel encoding and spatial reads. Context and candidate panels pass through the shared encoder in separate calls. Batch normalization computes minibatch statistics separately in these calls during training and uses stored running statistics at inference. The encoder fuses its final semantic map with the preceding higher-resolution layout map through pixel unshuffling, channel projection, and a sigmoid gate. Separate observation projections with channel normalization produce the views. For each panel, M spatial reads normalize content-dependent logits and positional biases over locations. These reads define content-dependent pooling regions.

Prediction and dynamic control. The predictor uses the first two observation maps and a learned prompt to estimate the third map. At step t, a shared controller averages $\boldsymbol{E}, \boldsymbol{P}, \boldsymbol{V}_t, \boldsymbol{R}_t$ over spatial locations while preserving the three panel positions. Concatenating these summaries gives a vector of dimension $12C$. Layer normalization, a $12C$-to-$2C$ linear map, GELU, and a nine-output head produce two Verify read logits, four Refine read logits, two write logits, and one mixing logit. Softmax normalizes the read weights within each branch. Sigmoids give the write gates, while a sigmoid scaled by $1/2$ gives $\rho_t \in (0, 1/2)$. All controls are computed from the states at the start of the step, before either branch updates.

Branch inputs and sequential writes. The dynamic read in Section 3.2 uses branch-specific, bias-free channel projections at every panel position and spatial location. Verify reads $(\boldsymbol{E}, \boldsymbol{V}_t^{\mathrm{mix}})$ with two softmax weights, starting from a zero mixed state. Refine reads $(\boldsymbol{O}, \boldsymbol{P}, \boldsymbol{V}_{t+1}, \boldsymbol{R}_t^{\mathrm{mix}})$, so its input includes the newly updated verification state. Verify uses the smooth normalization $\boldsymbol{z}/\sqrt{1 + \|\boldsymbol{z}\|_2^2}$, followed by separate gate and value projections and gated feature interactions across

panels. Refine applies channel projection and convolution to its combined sources. When residual dropout is enabled during training, it acts on the branch output before the write gate and residual addition. Observation projections remain specific to each view, while the controller and branch projections follow the parameter sharing of the recurrent block.

B.2 RULE-AS-AN-EXECUTABLE-OBJECT (RAXO)

For a fixed component-view pair, write a contextual row as $\boldsymbol{X} = (\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$, where each panel state lies in $[-1, 1]^n$.

Polynomial parameterization. The compiler is an MLP with one 96-unit GELU hidden layer. Its output biases initialize the adjustable coefficients near the power sums $\boldsymbol{a}^d + \boldsymbol{b}^d + \boldsymbol{c}^d$. The final weights are sampled from a zero-mean Gaussian with standard deviation 0.002. Joint fitting pairs proposals with the same hypothesis index from the two complete context rows. The same compiler produces every source proposal, with a fixed output slot for each hypothesis. The parameterization in Equation (8) uses 2, 8, and 18 adjustable coefficients for degrees one, two, and three, respectively. The implementation pads them into a shared 19-monomial tensor, with inactive entries fixed to zero.

Joint fitting solver. Consider one component-view pair, hypothesis, and degree in Equation (12). Write $\boldsymbol{\Phi}_s = \boldsymbol{\Phi}_d(\boldsymbol{X}_s)$, $\boldsymbol{q}_s = \boldsymbol{c}_s^d$, $\gamma = \lambda/n$, and $\overline{\boldsymbol{\beta}} = (\boldsymbol{\beta}_1 + \boldsymbol{\beta}_2)/2$. Eliminating the reference gives

$$\boldsymbol{\kappa}(\boldsymbol{\beta}) = \frac{\eta\,\overline{\boldsymbol{\kappa}} + \gamma \sum_{s=1}^{2} (\boldsymbol{q}_s + \boldsymbol{\Phi}_s \boldsymbol{\beta})}{\eta + 2\gamma}, \qquad \overline{\boldsymbol{\kappa}} = \tfrac{1}{2}(\boldsymbol{\kappa}_1 + \boldsymbol{\kappa}_2). \tag{13}$$

Let $\boldsymbol{S} = \boldsymbol{\Phi}_1 + \boldsymbol{\Phi}_2$ and $\boldsymbol{z}_s = \boldsymbol{q}_s - \overline{\boldsymbol{\kappa}}$. The reduced problem has the same minimizer as

$$\min_{\|\boldsymbol{\beta}\|_\infty \le b_d} \tfrac{1}{2}\boldsymbol{\beta}^\mathsf{T} \boldsymbol{A} \boldsymbol{\beta} - \boldsymbol{r}^\mathsf{T} \boldsymbol{\beta}, \tag{14}$$

where

$$\boldsymbol{A} = \boldsymbol{I} + \gamma \sum_s \boldsymbol{\Phi}_s^\mathsf{T} \boldsymbol{\Phi}_s - \frac{\gamma^2}{\eta + 2\gamma} \boldsymbol{S}^\mathsf{T} \boldsymbol{S}, \tag{15}$$

$$\boldsymbol{r} = \overline{\boldsymbol{\beta}} - \gamma \sum_s \boldsymbol{\Phi}_s^\mathsf{T} \boldsymbol{z}_s + \frac{\gamma^2}{\eta + 2\gamma} \boldsymbol{S}^\mathsf{T} \sum_s \boldsymbol{z}_s. \tag{16}$$

For $\eta > 0$ and $\gamma \ge 0$, the eliminated residual term is positive semidefinite, so $\boldsymbol{A} \succeq \boldsymbol{I}$. Regularization therefore makes the bounded optimization problem strongly convex with a unique optimum, including for rank-deficient context design matrices.

For the current degree, write $b = b_d$ and let Π_b clip coordinates to $[-b, b]$. Initialize $\boldsymbol{\beta}^{(0)} = \boldsymbol{y}^{(0)} = \Pi_b(\boldsymbol{A}^{-1}\boldsymbol{r})$. With $L_Q = 1 + \operatorname{tr}(\boldsymbol{A} - \boldsymbol{I})$ and $\omega = (\sqrt{L_Q} - 1)/(\sqrt{L_Q} + 1)$, each solver step is

$$\boldsymbol{\beta}^{(k+1)} = \Pi_b\left(\boldsymbol{y}^{(k)} - \frac{\boldsymbol{A}\boldsymbol{y}^{(k)} - \boldsymbol{r}}{L_Q}\right), \tag{17}$$

$$\boldsymbol{y}^{(k+1)} = \boldsymbol{\beta}^{(k+1)} + \omega(\boldsymbol{\beta}^{(k+1)} - \boldsymbol{\beta}^{(k)}). \tag{18}$$

Since $L_Q \ge \lambda_{\max}(\boldsymbol{A})$, $1/L_Q$ is a valid fixed step size. The projected iterations refine the clipped initialization while accounting for coupling between coefficients. The implementation takes 64 steps in FP32 and recovers the reference using Equation (13). It records $\|\boldsymbol{\beta} - \Pi_b(\boldsymbol{\beta} - (\boldsymbol{A}\boldsymbol{\beta} - \boldsymbol{r}))\|_\infty$ as a stationarity diagnostic. All hypotheses for the same pair and degree share $\boldsymbol{A}$, with separate right-hand sides. Training differentiates through the finite solver.

Evidence scale. The implementation parameterizes the scale shared by Cross Check and Weighted Evidence as $\tau = 1 + \operatorname{softplus}(\xi)$, with learned scalar ξ.

C Mathematical Properties

C.1 Residual Mixing

For one row, view, and update, the mixing matrix is nonnegative and doubly stochastic (Xie et al., 2025). It preserves the sum of the states: $\boldsymbol{V}_t^{\mathrm{mix}} + \boldsymbol{R}_t^{\mathrm{mix}} = \boldsymbol{V}_t + \boldsymbol{R}_t$. Expanding their squared norms gives

$$\|\boldsymbol{V}_t^{\mathrm{mix}}\|_F^2 + \|\boldsymbol{R}_t^{\mathrm{mix}}\|_F^2 = \|\boldsymbol{V}_t\|_F^2 + \|\boldsymbol{R}_t\|_F^2 - 2\rho_t(1-\rho_t)\|\boldsymbol{V}_t - \boldsymbol{R}_t\|_F^2. \tag{19}$$

Here, $\|\cdot\|_F$ denotes the Frobenius norm. For $\rho_t \in [0,1]$, Residual Mapping is nonexpansive in the combined state norm. With zero initialization and $N = 2$, the second update exchanges the evidence accumulated by the two branches in the first update.

C.2 Normalized Execution and Candidate Rankings

The cost in Equation (9) vanishes on $\{\boldsymbol{Y} : \mathcal{F}_d(\boldsymbol{Y}) = \boldsymbol{\kappa}_d \text{ for all } d\}$. We hold coefficients, references, support, aggregation weights, and scale fixed. Original and perturbed row states lie in $[-1,1]^{3\times n}$, as ensured by the hyperbolic-tangent readout.

Proposition 1 (Normalized execution stability). Let $\mathcal{A}_d = \{\boldsymbol{\alpha} \in \mathbb{N}_0^3 : 1 \le \|\boldsymbol{\alpha}\|_1 \le d\}$ index all nonconstant monomials, including the fixed leading term. Define $L_d^2 = \sum_{\boldsymbol{\alpha}\in\mathcal{A}_d} \|\boldsymbol{\alpha}\|_2^2$ and $L^2 = (Dn)^{-1}\sum_{d=1}^{D} L_d^2$. For any fixed executable relation and bounded row states $\boldsymbol{Y}, \boldsymbol{Y}'$, writing $q(\boldsymbol{Y}) = \mathrm{Dis}(\boldsymbol{Y}; \mathcal{F}, \boldsymbol{\kappa})$,

$$\left|\sqrt{q(\boldsymbol{Y})} - \sqrt{q(\boldsymbol{Y}')}\right| \le L\|\boldsymbol{Y} - \boldsymbol{Y}'\|_F. \tag{20}$$

The bound is uniform over coefficient values and fixed references.

Proof. For one state coordinate, let $\boldsymbol{z} \in [-1,1]^3$ and let $\boldsymbol{m}_d(\boldsymbol{z})$ collect the monomials indexed by $\mathcal{A}_d$. Each monomial satisfies $\|\nabla \boldsymbol{z}^{\boldsymbol{\alpha}}\|_2 \le \|\boldsymbol{\alpha}\|_2$, so the Jacobian of $\boldsymbol{m}_d$ has Frobenius norm at most L_d. The line segment between two bounded states stays in the cube, giving $\|\boldsymbol{m}_d(\boldsymbol{z}) - \boldsymbol{m}_d(\boldsymbol{z}')\|_2 \le L_d\|\boldsymbol{z} - \boldsymbol{z}'\|_2$. Order the polynomial coefficient vector as $\boldsymbol{\theta}_d = (1, \boldsymbol{\beta}_d)$, with the leading monomial first. Cauchy–Schwarz and summation over coordinates give

$$\frac{\|\mathcal{F}_d(\boldsymbol{Y}) - \mathcal{F}_d(\boldsymbol{Y}')\|_2}{\sqrt{1 + \|\boldsymbol{\beta}_d\|_2^2}} \le L_d\|\boldsymbol{Y} - \boldsymbol{Y}'\|_F. \tag{21}$$

Let $\boldsymbol{r}(\boldsymbol{Y})$ concatenate the residual vectors $(\mathcal{F}_d(\boldsymbol{Y}) - \boldsymbol{\kappa}_d)/\sqrt{Dn(1 + \|\boldsymbol{\beta}_d\|_2^2)}$ across degrees. Then $q(\boldsymbol{Y}) = \|\boldsymbol{r}(\boldsymbol{Y})\|_2^2$, while the preceding bound yields $\|\boldsymbol{r}(\boldsymbol{Y}) - \boldsymbol{r}(\boldsymbol{Y}')\|_2 \le L\|\boldsymbol{Y} - \boldsymbol{Y}'\|_F$. The reverse triangle inequality proves Equation (20). Normalization removes the coefficient-norm factor, while the bound retains its dependence on degree.

Proposition 2 (Candidate-ranking stability). For component-view pair v, suppose every candidate row changes by at most ϵ_v in Frobenius norm. Let $R_v = \max_{h,i} \sqrt{d_{v,h}(i)}$ be the largest original execution residual over its hypotheses and candidates. Define

$$B = \tau \sum_v w_v \left(2R_v L\epsilon_v + L^2\epsilon_v^2\right). \tag{22}$$

Every aggregated candidate score changes by at most B. Consequently, if the original winning score exceeds every other score by more than $2B$, the selected candidate is unchanged.

Proof. Proposition 1 implies $|d'_{v,h}(i) - d_{v,h}(i)| \le 2R_v L\epsilon_v + L^2\epsilon_v^2$. For fixed positive support weights, the gradient of $\log\sum_h \pi_h \exp(-\tau d_h)$ with respect to the cost vector has ℓ_1 norm τ. The mean-value theorem therefore bounds the evidence change by τ times the largest cost change. Aggregation with $w_v \ge 0$ and $\sum_v w_v = 1$ gives Equation (22). The gap between winning and competing scores can decrease by at most $2B$, proving the margin condition, which the final softmax preserves. The result characterizes candidate-ranking stability under perturbations of the latent states with the context-derived relation and scoring weights held fixed.

D EXPERIMENTAL SETUP

D.1 TRAINING CONFIGURATION AND OBSERVATION REGULARIZATION

The architecture and settings for the frozen Type diagnostics are summarized in Table 8. Parameter count and computational cost are reported in Table 5.

Table 8: Reference configuration for the frozen Type diagnostics.

Setting	Value
Input panels	80×80, grayscale
Observation / reasoning channels	160 / 160
Observation views / spatial components	5 / 3
Shared IVaR updates / state dimension	$N = 2$ / $n = 8$
Hypotheses / polynomial degrees	$H = 4$ / $d = 1, 2, 3$
Coefficient bounds by degree	$4, 12, 32$
Joint fitting	$\eta = \lambda = 1$, 64 projected steps
Training batch / learning rate / weight decay	128 / 10^{-3} / 10^{-5}
Diagnostic checkpoint	Epoch 100
Diagnostic arithmetic	BF16 features, FP32 relation execution

Muon updates matrix and convolutional parameters, while Adam handles vectors, designated normalization parameters, and output parameters. The reference optimizer uses Muon momentum 0.95, Adam moments $(0.9, 0.95)$, and global gradient clipping at 1. The answer objective uses label smoothing of 0.05.

Observation regularization. For a minibatch of B puzzles with K candidates, each component-view pair supplies BK samples. Observation regularization combines a standard-deviation floor of $\nu = 1$, with numerical stabilizer $\epsilon = 10^{-4}$, and a cross-view covariance penalty. The covariance penalty acts on pairs of distinct views. Distributed training gathers candidate features with gradients before computing the population moments. This block variant of variance-covariance regularization penalizes low variance and linear redundancy.

Data augmentation. The reference training pipeline applies the same sampled sequence of geometric transformations to all 16 panels, preserving the order of matrix rows and answer choices. Panels are resized to 80×80 by nearest-neighbor interpolation before augmentation. Horizontal and vertical flips are sampled independently, each with probability 0.3. Circular translation has probability 0.2, with each integer displacement sampled from 0 to 80 pixels on the resized panel. Shrinking followed by white padding has probability 0.2, with scale sampled uniformly from 0.8 to 0.99. Flips exchange left/right or up/down component labels where required. Circular translation retains the original source-component proxy labels. Validation and test preprocessing consists of resizing and normalization.

Bongard adaptation. For Bongard tasks, we retain context-conditioned polynomial execution. A polynomial relation is induced from the support context and held fixed when evaluating individual examples. Positive examples, negative examples, and queries follow the same processing path as RPM candidates: IVaR constructs their relational states, and RaXO scores their agreement with the induced relation. Positive and negative labels supervise concept discrimination, while a query's execution score determines its predicted concept membership. This adaptation uses support-based induction followed by example-wise execution, omitting the RPM-specific Cross Check between complete matrix rows.

D.2 BENCHMARK PROTOCOLS AND EXAMPLES

Accuracy is reported in percent and differences in percentage points. The Avg. columns in Tables 1 and 2 average eight datasets and eight regimes, respectively. The A-I-RAVEN OOD mean used in the text, Table 6, and Table 9 covers Color, Position, Size, and Type, excluding Mesh. For

CLEVR-Matrices and Bongard-LOGO, Avg. averages the three configurations and four concept categories, respectively. Baseline entries follow their published protocols, with each method retaining its reported pretraining, supervision, and optimization budget.

The following figures show example inputs from each benchmark. For matrix examples, the context appears above the eight candidates. A blank fourth column in a context is display padding.

D.2.1 RAVEN-FAMILY BENCHMARKS

RAVEN. RAVEN contains 70,000 problems: 42,000 for training and 14,000 each for validation and testing (Zhang et al., 2019a) (Figure 3). Each has eight context panels and eight candidates. Its seven configurations are Center, 2×2 Grid, 3×3 Grid, Left-Right, Up-Down, Out-InCenter, and Out-InGrid, abbreviated as Center, 2×2G, 3×3G, L-R, U-D, O-IC, and O-IG. Constant, Progression, Arithmetic, and Distribute Three govern number, position, type, size, and color. The original candidate generator permits answer-set shortcuts, motivating I-RAVEN and RAVEN-FAIR (Hu et al., 2021; Benny et al., 2021).

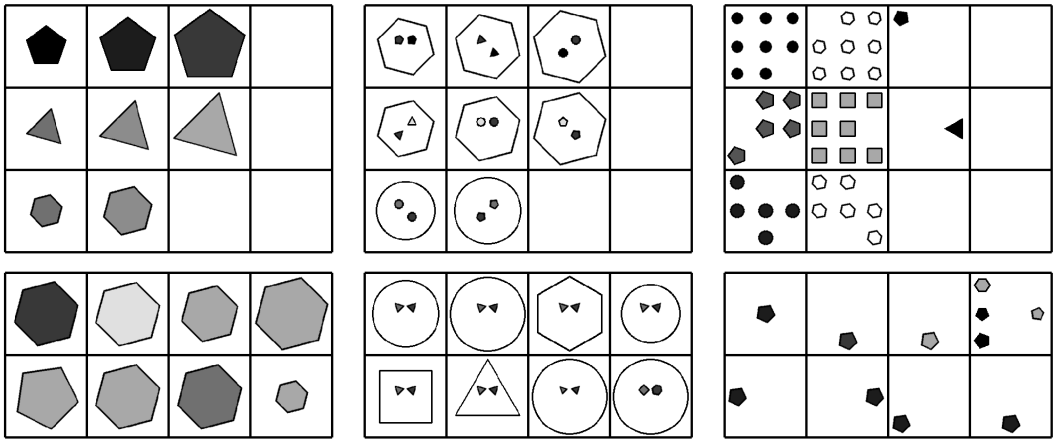


Figure 3: Examples from RAVEN (Zhang et al., 2019a).

I-RAVEN. I-RAVEN retains the RAVEN matrices and configurations but uses an Attribute Bisection Tree to balance distractor attributes, reducing answer-frequency shortcuts (Hu et al., 2021) (Figure 4).

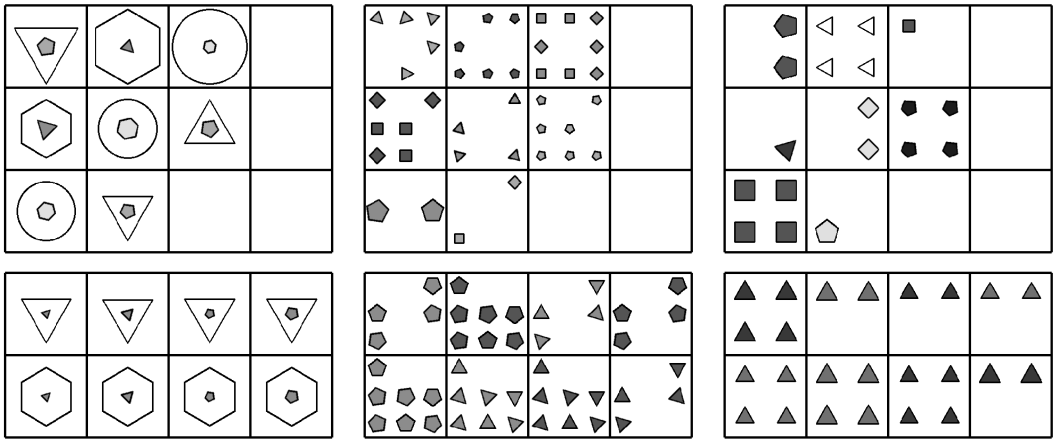


Figure 4: Examples from I-RAVEN (Hu et al., 2021).

RAVEN-FAIR. RAVEN-FAIR generates distractors by modifying an attribute of an existing option, including other distractors. This reduces the correct answer's privileged role in candidate construction (Benny et al., 2021) (Figure 5).

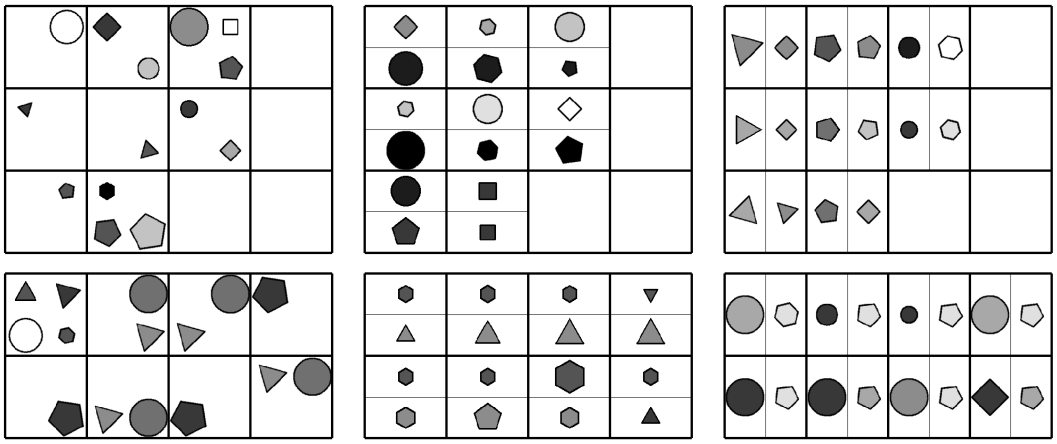


Figure 5: Examples from RAVEN-FAIR (Benny et al., 2021).

I-RAVEN-Mesh. I-RAVEN-Mesh adds rule-governed mesh lines and candidates that differ in this component (Małkiński & Mańdziuk, 2025) (Figure 6). The added mesh component extends the visual and relational content of the original matrices.

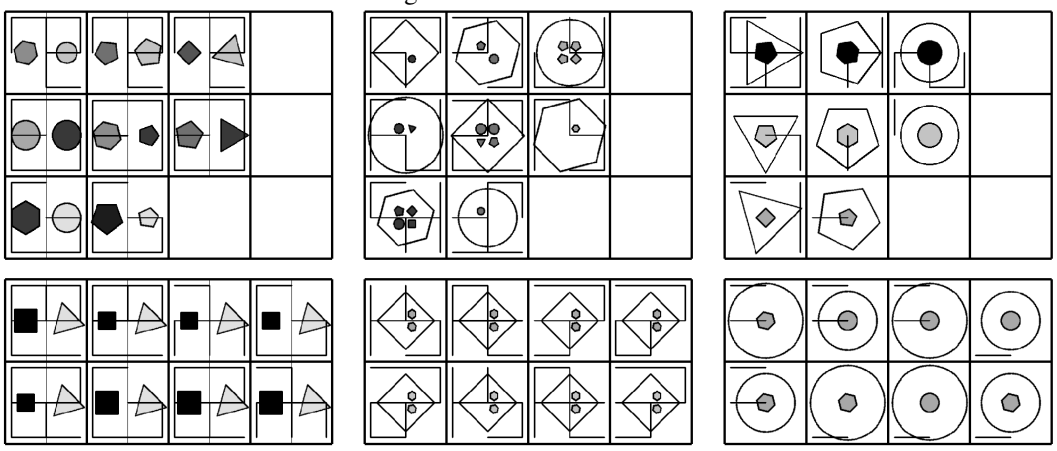


Figure 6: Examples from I-RAVEN-Mesh (Małkiński & Mańdziuk, 2025).

D.2.2 A-I-RAVEN Attribute Generalization

A-I-RAVEN withholds attribute-rule pairings (Małkiński & Mańdziuk, 2025). For each of its four splits, the designated attribute follows Constant during training and validation, then a nonconstant rule at test time. The other attributes can instantiate those rule families during training.

A-I-RAVEN-Color. The Color split applies this restriction to color (Figure 7).

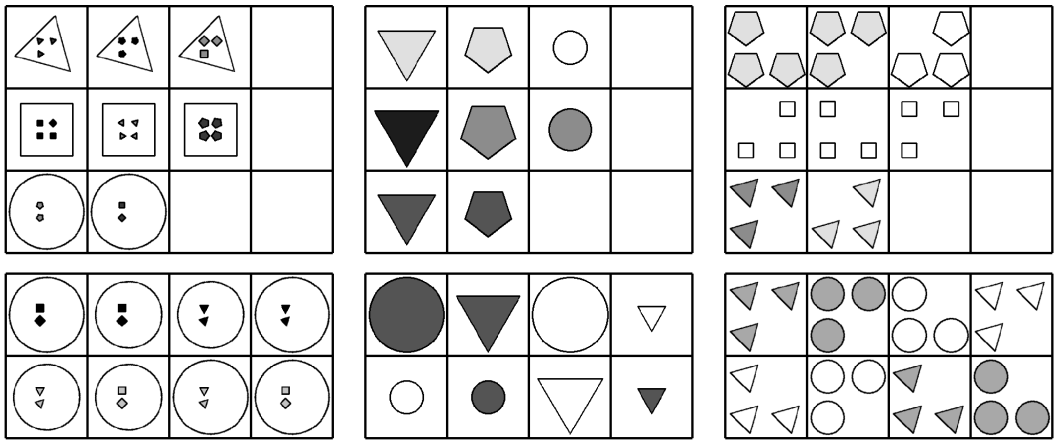


Figure 7: Examples from A-I-RAVEN-Color (Małkiński & Mańdziuk, 2025).

A-I-RAVEN-Position. The Position split concerns spatial arrangement, with number and position coupled in construction (Figure 8).

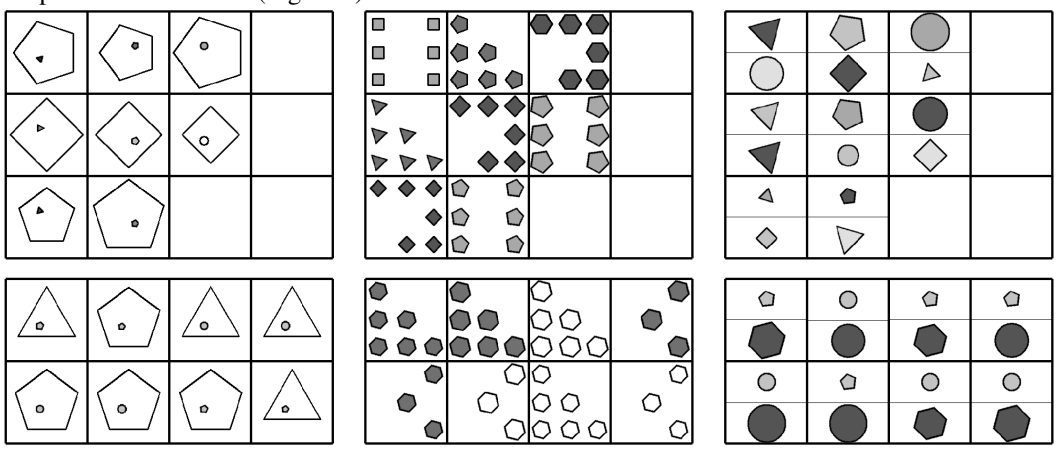


Figure 8: Examples from A-I-RAVEN-Position (Małkiński & Mańdziuk, 2025).

A-I-RAVEN-Size. The Size split concerns object size (Figure 9).

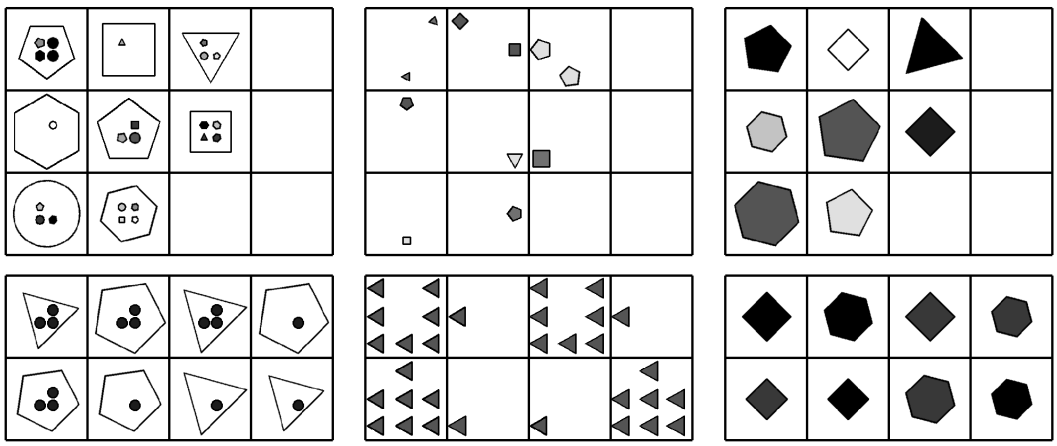


Figure 9: Examples from A-I-RAVEN-Size (Małkiński & Mańdziuk, 2025).

A-I-RAVEN-Type. The Type split concerns shape categories (Figure 10). It supplies the frozen-model diagnostics in Table 7, with the evaluation subset defined in Section E.1.

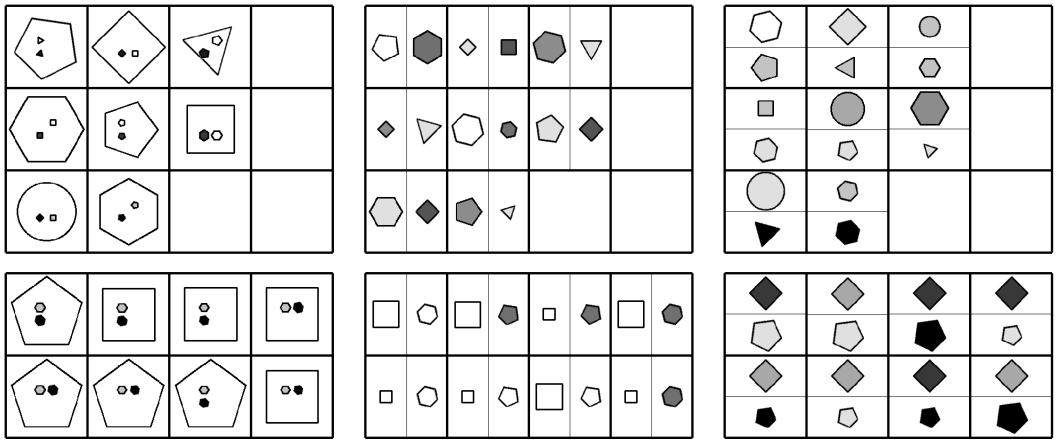


Figure 10: Examples from A-I-RAVEN-Type (Małkiński & Mańdziuk, 2025).

D.2.3 PGM AND CLEVR-MATRICES

PGM. PGM combines relation, object-class, and attribute triples into problems with eight context panels and eight candidates (Barrett et al., 2018) (Figure 11). The relation families are Progression, XOR, OR, AND, and Consistent Union. They apply to shapes or lines. The standard split has 1.2 million training, 20,000 validation, and 200,000 test problems, with eight regimes:

- **Neutral (N):** new instances from the same generating distribution.
- **Interpolation (I):** held-out values within the training attribute range.
- **Extrapolation (E):** attribute values outside the training range.
- **Held-out Attribute Pairs (HAP):** unseen pairs of familiar attributes.
- **Held-out Triple Pairs (HTP):** unseen co-occurrences of generating triples.
- **Held-out Triples (HT):** unseen triples of relations, object classes, and attributes.
- **Held-out Attribute Line Type (HALT):** held-out relations on line type.
- **Held-out Attribute Shape Color (HASC):** held-out relations on shape color.

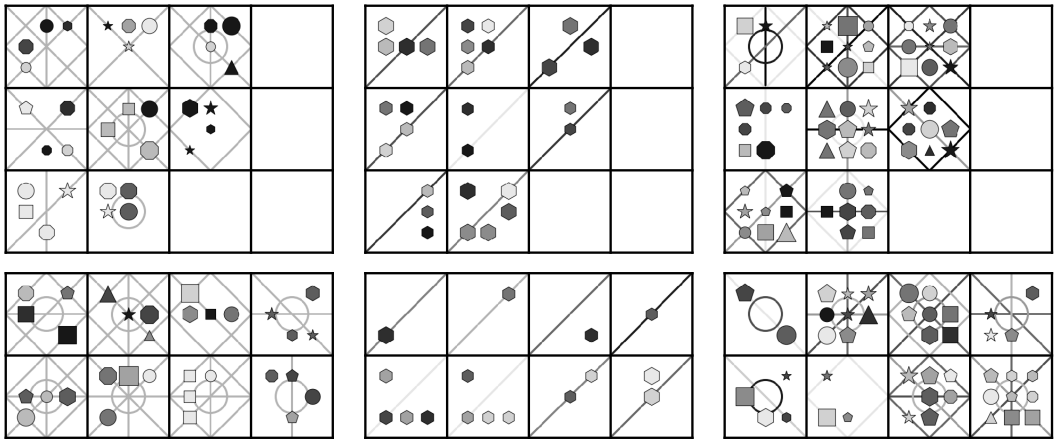


Figure 11: Examples from PGM (Barrett et al., 2018).

CLEVR-Matrices. CLEVR-Matrices uses three-dimensional objects and impartial candidate generation (Mondal et al., 2023) (Figure 12). Each of the Logic, Location, and Count configurations contains 16,000 training, 2,000 validation, and 2,000 test problems. Objects vary in shape, size, and color.

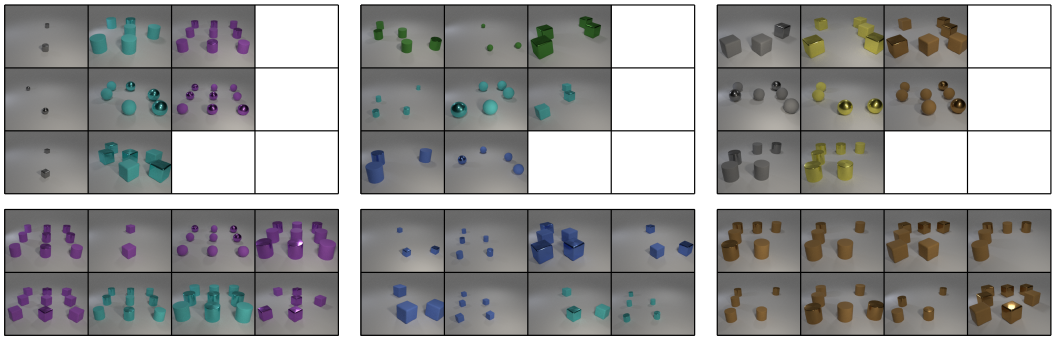


Figure 12: Examples from CLEVR-Matrices (Mondal et al., 2023).

D.2.4 Bongard Concept Discrimination

Bongard-LOGO. Bongard-LOGO contains 12,000 concept-discrimination problems rendered from LOGO programs (Nie et al., 2020) (Figure 13). Each support set has six positive and six negative examples. FF, BA, CM, and NV evaluate free-form problems, basic concepts, compositional generalization, and novel concepts, respectively.

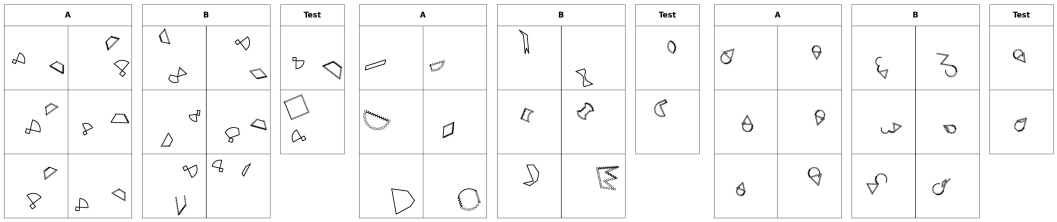


Figure 13: Bongard-LOGO examples (Nie et al., 2020). A and B show the two support classes, followed by test images.

Bongard-OpenWorld. Bongard-OpenWorld uses natural images with six positive and six negative support examples per problem (Wu et al., 2024) (Figure 14). Short/Long partition problems by concept length, while CS/non-CS partition them by the need for commonsense knowledge. Avg. in Table 3 is recomputed for every method as the unweighted mean of the four reported category accuracies. Since the two partitions overlap, this summary differs from overall test accuracy. CS and Non-CS in Table 3 denote the commonsense and non-commonsense categories, respectively.

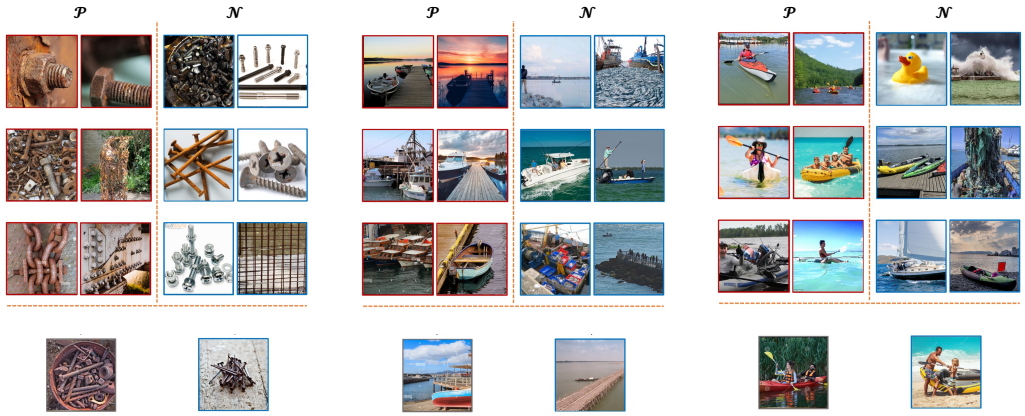


Figure 14: Bongard-OpenWorld examples (Wu et al., 2024). $\mathcal{P}$ and $\mathcal{N}$ show positive and negative support, with queries below the dashed dividers. Borders indicate dataset labels.

E Additional Analyses

E.1 RaXO Diagnostics

Evaluation subset. The frozen checkpoint in Section D.1 evaluates 448 of the 14,000 A-I-RAVEN-Type test puzzles: 64 per configuration, sampled uniformly without replacement with seed 20260920. Evaluation uses BF16 features, FP32 relation execution, and batch size eight. Context discrepancy averages rows, hypotheses, and component-view pairs within each puzzle, then puzzles. Bootstrap intervals estimate uncertainty across sampled puzzles at the fixed checkpoint.

Support weighting. The support-weighting comparison in Table 7 changes only hypothesis weights on all 448 puzzles. NLL is the mean negative natural-log probability assigned to the correct answer. Pre-fit support scores each row's proposals on the other complete row. Post-fit support instead scores the fitted relations on each context row. Both variants average two softmax distributions over hypotheses, whereas uniform support weights all four slots equally. Fitted relations, candidate costs, scale, and component-view aggregation weights remain fixed. The 95% interval for pre-fit minus uniform accuracy is $[-0.89, 2.90]$ percentage points, computed from 5,000 stratified whole-puzzle bootstrap resamples.

Row pairing. The row-pairing comparison in Table 7 compares saved directional support distributions on the 64 Center puzzles, using the first spatial component and all five views. Matched pairs use the same puzzle. Shuffled pairs use all other puzzles, while Within-rule pairs restrict these donors to the same view-specific rule. Jensen–Shannon (JS) divergence uses natural logarithms and is averaged over eligible donors, views, then target puzzles. Table 7 displays these values multiplied by 100. All comparisons use cached distributions, with donor pools restricted to other puzzles.

Coefficient transfer. On the same 64 Center puzzles, the coefficient-transfer comparison in Table 7 transfers Size or Color coefficients from donor puzzles into the first spatial component's Type view. Each donor replaces coefficients in all four hypothesis slots and all three degrees. Target Type rules are Progression or Distribute Three. Donors follow either the target rule or the other of these two rules, excluding Constant and Arithmetic. Each donor pool excludes the target puzzle and contains at least eight puzzles per source view and condition.

Native and transferred coefficients use references recomputed as mean polynomial outputs on the target's two context rows, with uniform hypothesis support. Target states, the other component-view pairs, scale, and aggregation weights remain fixed. Accuracy and the correct-answer margin over the highest-scoring distractor are averaged over donors, then the two source views, then the 64 targets, covering 3,528 donor executions. The margin is a difference between final candidate logits, displayed multiplied by 100 in Table 7. Rule labels define the offline interventions, while answer labels evaluate predictions.

Joint fitting versus averaging. On all 448 puzzles, this control changes only $\lambda = 0$ to $\lambda = 1$, retaining proposals, pre-fit support, and aggregation. Mean context discrepancy decreases by 23.0%, from 0.1717 to 0.1322. Correct answers change from 275 to 278, with 11 corrections and eight regressions. The paired accuracy difference has a 95% interval of $[-1.12, 2.68]$ percentage points, using 10,000 whole-puzzle bootstrap resamples within configurations with seed 20260924. The discrepancy reduction reflects closer agreement on the fitted context rows, accompanied by a net gain of three correct answers on this subset.

Replay precision and reproducibility. The cached controls reconstruct original and uniform log-probabilities with maximum absolute errors of 2.36×10^{-6} and 2.05×10^{-6}, respectively. Independent native polynomial execution differs from the saved costs by at most 3.98×10^{-7}. The source bundle records checkpoint and sample-manifest hashes, the intervention script, and outputs for each target-donor pairing.

E.2 HYPERPARAMETER SENSITIVITY

We vary the number of IVaR updates N, the observation-regularization weight λ_R, and the context-fit strength λ one at a time. All other settings remain at their defaults, with $n = 8$ and $D = 3$. Each configuration is trained from scratch with the same dataset-specific training protocol and evaluated on the four A-I-RAVEN OOD test splits. The λ_R sweep changes the training objective, whereas λ controls Joint Fit during both training and inference.

Table 9: Hyperparameter sensitivity on A-I-RAVEN. Accuracy (%) is reported for each OOD split. Avg. is the unweighted mean of the four split accuracies, rounded to one decimal. A dagger marks the default setting.

		A-I-RAVEN OOD				
Hyperparameter	Value	Color	Position	Size	Type	Avg.
IVaR updates N	1	71.2	80.8	82.4	60.4	73.7
	$2^\dagger$	73.1	82.1	84.5	62.1	75.5
	3	72.4	82.1	82.5	61.3	74.6
	4	72.2	80.1	84.4	61.3	74.5
	5	71.4	81.9	83.7	59.9	74.2
Regularization weight λ_R	0.025	70.8	81.4	83.6	60.9	74.2
	0.05	72.0	82.3	83.7	61.5	74.9
	$0.1^\dagger$	73.1	82.1	84.5	62.1	75.5
	0.2	71.7	81.9	82.5	60.4	74.1
	0.4	72.1	81.6	84.0	59.7	74.4
Context-fit strength λ	0.25	73.3	80.2	82.7	60.2	74.1
	0.5	71.1	80.9	83.1	60.3	73.9
	$1^\dagger$	73.1	82.1	84.5	62.1	75.5
	2	72.2	82.1	83.9	61.2	74.9
	4	71.7	81.3	83.8	61.5	74.6

IVaR updates. Two updates achieve the highest mean accuracy of 75.5%, compared with 73.7% for one update. The second update improves all four splits, while three, four, and five updates yield 74.6%, 74.5%, and 74.2%, respectively.

Regularization weight. The default $\lambda_R = 0.1$ gives the highest mean accuracy of 75.5%, while the other tested weights range from 74.1% to 74.9%. Although $\lambda_R = 0.05$ slightly improves Position accuracy, 0.1 performs best on the other three splits. The results favor an intermediate weight for encouraging feature variance and reducing cross-view redundancy alongside answer supervision.

Context-fit strength. The default $\lambda = 1$ also gives the highest mean accuracy. Weaker fitting at $\lambda = 0.25, 0.5$ yields 74.1% and 73.9%, while stronger fitting at $\lambda = 2, 4$ yields 74.9% and 74.6%. This pattern is consistent with balancing agreement on the two context rows against preservation of the source proposals.

E.3 Qualitative Comparisons and Failure Cases

We compare NExRI and DSRF-L on eight A-I-RAVEN OOD test puzzles (Małkiński & Mańdziuk, 2025), covering all four held-out attributes in each outcome group. Within the first 128 sorted test files per attribute/layout, eligible puzzles have a nonconstant held-out rule and a distractor changing only that attribute. We rank by nonconstant rules, rule families, then mean object count, retaining one example per stratum before evaluating the models. The evaluated examples are grouped into NExRI successes and shared failures.

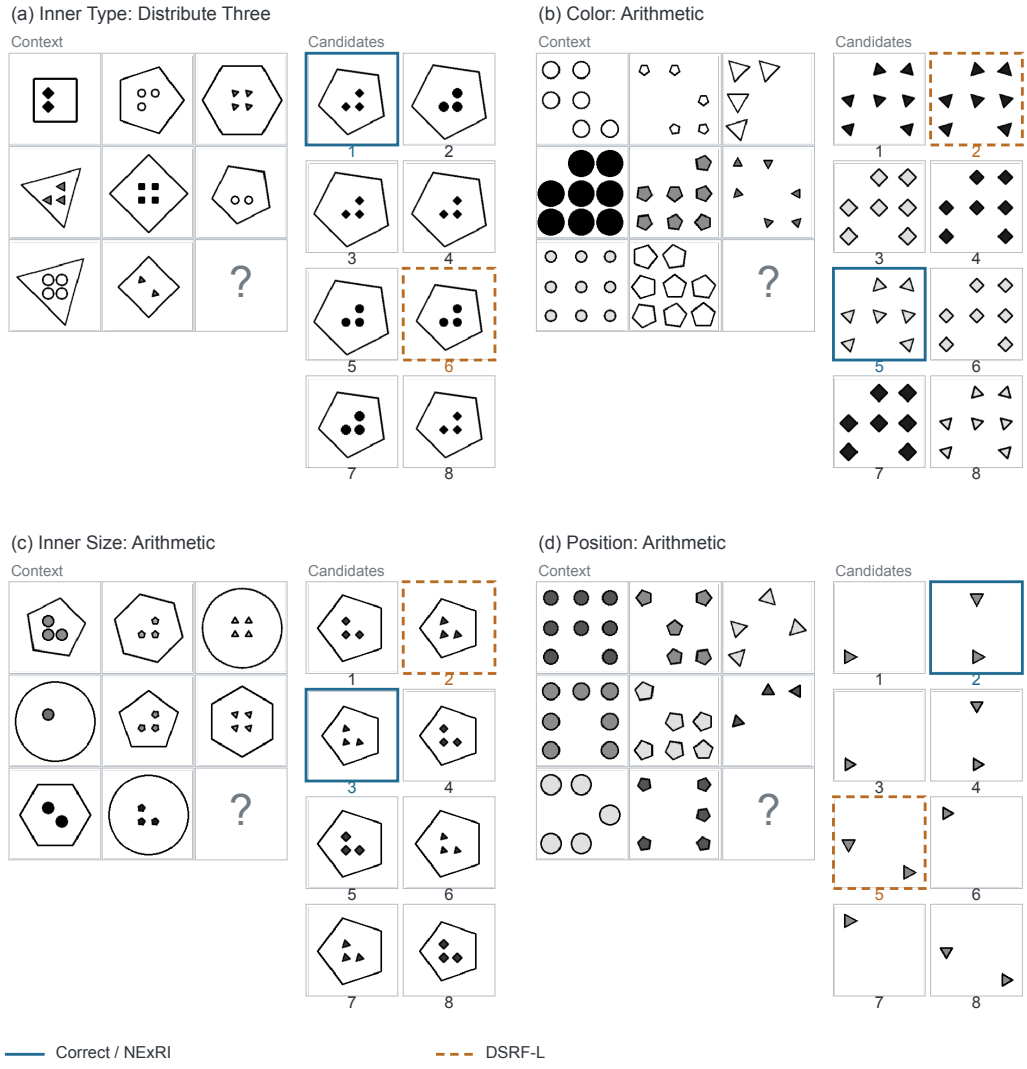


Figure 15: NExRI succeeds where DSRF-L fails. Blue outlines mark the correct answers selected by NExRI, and dashed orange outlines mark DSRF-L's predictions. Candidate indices are one-based.

Distinguishing partially correct completions. In Figure 15a,c, inner and outer objects obey different Type or Size relations. DSRF-L selects a distractor that changes only the inner attribute, while NExRI selects the completion satisfying both components. The same distinction appears for Color in panel b and Position in panel d, where the incorrect alternatives preserve the other annotated attributes. In panel d, the correct completion requires both the prescribed object count and spatial arrangement. These cases illustrate the importance of checking all attribute relations within each component.

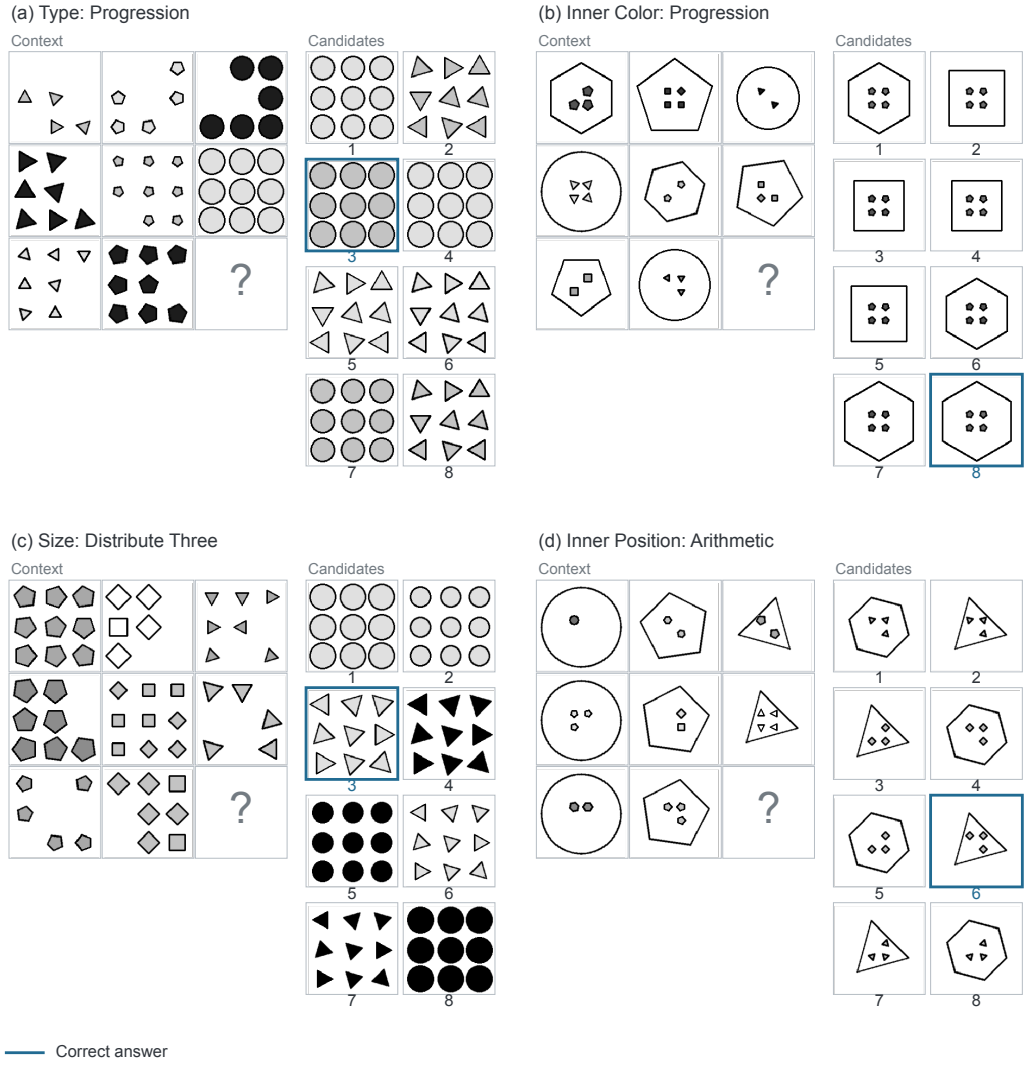


Figure 16: Failure cases for both NExRI and DSRF-L. Blue outlines mark the correct answers. Panels a,c use dense grids, while b,d combine inner and outer objects.

Concurrent attribute relations. The shared failures in Figure 16a,c require tracking several changing attributes at once. Panel a combines Type and Number progressions with separate Size and Color relations. Candidate 2 differs from the correct answer 3 only in Type. In panel c, candidates 3 and 6 agree in Number, Type, and Color, yet only candidate 3 satisfies the Size relation. The alternatives therefore contain substantial partial agreement, making a missed attribute sufficient to invalidate the completion.

Assigning rules to scene components. The nested scenes add a distinct requirement: inner and outer objects follow separate rules. In Figure 16b, candidates 7 and 8 differ only in inner Color, while the outer Type and Size must satisfy their own relations. In panel d, candidates 3 and 6 differ only in the inner spatial arrangement. The failures across these layouts expose the need to preserve attribute distinctions and apply each relation to the correct component. These cases identify a concrete challenge for relation induction: capturing concurrent attribute constraints and binding them to the corresponding scene components.

F LIMITATIONS

NExRI currently instantiates executable relations using a polynomial rule family and is evaluated on established visual reasoning benchmarks. Its extension to richer relation types and more open-ended reasoning settings remains to be explored. Future work will investigate these extensions while preserving the efficiency of context-based rule fitting and execution.