Synthesizing Visual Concepts as Vision-Language Programs

Antonia Wüst[1] Wolfgang Stammer[2] Hikaru Shindo[1] Lukas Helff[1,3]

Devendra Singh Dhami[4] Kristian Kersting[1,3,5]

[1]AIML Lab, TU Darmstadt [2]Max Planck Institute for Informatics, SIC
[3]Hessian Center for AI (hessian.AI) [4]Uncertainty in AI Group, TU Eindhoven
[5]German Research Center for AI (DFKI)

Abstract

Vision-Language models (VLMs) achieve strong performance on multimodal tasks but often fail at systematic visual reasoning, especially in inductive reasoning problems. Neuro-symbolic methods promise to address this by inducing interpretable logical programs from images, though they usually rely on rigid, domain-specific perception modules for this. We propose Vision-Language Programs (VLP), which combine the perceptual flexibility of VLMs with the systematic reasoning of symbolic program synthesis. Rather than embedding reasoning inside the VLM, VLP leverages the model to produce structured visual descriptions that are compiled into neuro-symbolic programs. The resulting programs execute directly on images, remain consistent with task constraints, and provide human-interpretable explanations that enable easy shortcut mitigation. Our experiments across synthetic and real-world datasets demonstrate that VLPs outperform both direct and structured prompting of VLMs, particularly on tasks that require complex logical reasoning. [1]

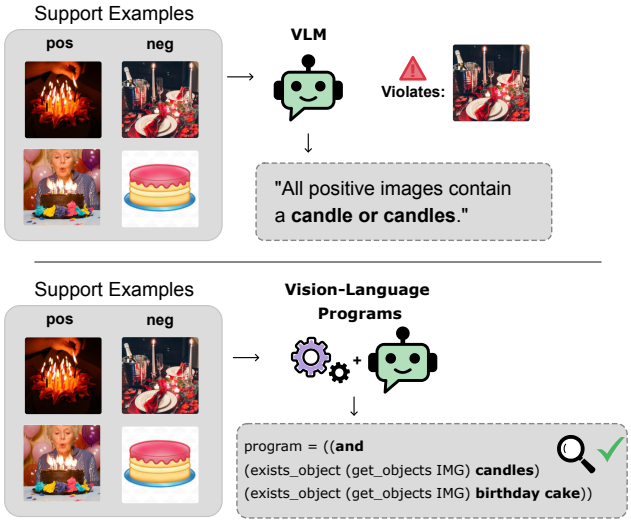

Figure 1. **VLMs cannot reliably perform inductive logic learning from images**, failing to capture visual compositions like *"candles **and** birthday cake"*. **Vision-Language Programs (VLP)** employ explicit symbolic reasoning to overcome such visual reasoning errors while maintaining perceptual flexibility.

1. Introduction

Vision–language models (VLMs) have achieved impressive results across multimodal tasks, yet they continue to struggle with visual reasoning. Studies reveal frequent failures in both perception and reasoning, even on relatively simple tasks [11, 16, 24, 25, 41, 46, 49, 50]. In inductive visual reasoning tasks where models must propose rules that distinguish between image sets (*cf*. Fig. 1), VLMs often fail by generating statements that violate the task constraints. In this example, the VLM proposes the rule "contains candle or candles", incorrectly satisfying one of the negative images. Such errors highlight the gap between pattern recognition and systematic reasoning in VLMs.

Recent work attempts to address this gap through test-time scaling, where models "think" longer via extended chain-of-thought generation [33, 42]. While effective in some cases, this approach is computationally expensive and prone to contradictions or repetitive loops [12, 30, 43, 48]. This raises the question of whether language-based inference alone is sufficient for robust reasoning.

Neuro-symbolic (NeSy) AI offers a promising alternative by integrating neural processing with structured symbolic inference [6, 17, 20, 26, 28, 29, 45]. This paradigm

[1]Project page: ml-research.github.io/vision-language-programs

has demonstrated improved robustness, compositional generalization, and interpretability, *e.g.*, over monolithic VLMs under domain shifts [15]. Hereby, program synthesis [9], which induces interpretable and logically consistent programs from examples, provides a particularly natural mechanism for implementing this integration in visual reasoning tasks. However, existing neuro-symbolic approaches for visual reasoning face critical limitations. Methods either require explicit queries to drive program generation [10, 15, 37, 39], limiting their applicability to inductive reasoning tasks, or depend on domain-specific image encoders, preventing generalization across diverse visual domains [28, 29, 45].

We therefore propose combining VLMs with program synthesis in form of VISION-LANGUAGE PROGRAMS (VLP) to overcome these shortcomings. Importantly, instead of embedding reasoning within the VLM, VLP leverages the model solely to produce structured visual descriptions that can be compiled into symbolic programs, decoupling perception from reasoning. VLP automatically induces rules from small sets of labeled image examples by first discovering candidate symbols through VLM-based analysis, then defining VLM functions that extract structured representations from images, which can be composed with symbolic reasoning functions in a domain-specific language. This allows the resulting programs to leverage neural perception at execution time while maintaining symbolic interpretability and logical consistency. This dualistic process allows VLP to *execute* directly on images.

Evaluations on both synthetic and real-world datasets demonstrate that even small VLMs, when embedded in our framework, surpass direct prompting, especially on tasks requiring complex logical reasoning. This hybrid approach thus leverages the perceptual priors of VLMs while enabling symbolic reasoning, marking a crucial step toward models that not only achieve strong performance but also provide transparent, structured decision processes.

To this end, we introduce: (i) VISION-LANGUAGE PROGRAMS, a framework combining VLMs with program synthesis to induce symbolic rules from labeled images without hand-crafted detectors; (ii) a domain-specific language integrating compositional VLM perception functions with symbolic reasoning operators; (iii) a probabilistic synthesis procedure that discovers and ranks programs by accuracy and likelihood; (iv) comprehensive empirical evidence showing our approach outperforms direct prompting, particularly on logically complex tasks; and (v) analysis demonstrating how programmatic structure enables shortcut detection and mitigation through transparent decision processes.

2. Related Work

Neuro-Symbolic Concept Induction. Neuro-symbolic AI [6, 17, 26] seeks to integrate the strengths of deep neural networks with the structure of symbolic reasoning. A common approach is to extract structured representations from raw perceptual inputs, then perform rule learning over the resulting symbols [34, 38, 45, 47], for instance via inductive logic programming (ILP) [28, 29] or probabilistic logic frameworks [20, 31, 32]. While effective for complex synthetic scenes, their applicability to open-world settings is often limited due to reliance on pre-defined predicates and domain-specific object detectors for constructing intermediate symbolic representations.

Program-based Visual Reasoning. A prominent recent approach to complex visual reasoning leverages programs for systematic image analysis. Frameworks such as VisProg [10], ViperGPT [39], CodeVQA [37], and NeP-Tune [15] employ foundation models to generate executable programs conditioned on natural language instructions or questions. Their approaches demonstrate the power of combining symbolic representations with foundation models for structured reasoning. While these approaches generate programs based on task-specific queries, our approach VLP focuses on inducing programs from labeled visual examples only, thereby uncovering programmatic representations that explain the conceptual differences between them.

Program synthesis [8] has long been successful in rule induction, with early efforts focusing on domains such as list processing and text editing [4, 9]. A benefit of explicit program synthesis over LLM-prompted approaches is the guarantee of syntactically valid and executable programs, eliminating formatting errors that can hinder downstream reasoning. More recent work extends these ideas to abstract reasoning tasks, such as the Abstract Reasoning Corpus [3], by leveraging foundation models to discover higher-level concepts [1, 40]. However, the intersection of program synthesis and foundation models for visual rule induction remains largely unexplored. An first step in this direction was taken by Wüst et al. [45], who proposed a program synthesis approach for visual concept induction, whereby programs are induced from positive and negative visual samples. However, their method relies on domain-specific object detectors to convert images into symbolic representations, limiting its applicability. In contrast, VLP extends this idea by directly synthesizing programs from natural images, eliminating the need for domain-specific pretraining and enabling broader applicability.

3. Vision Language Programs

We introduce VISION-LANGUAGE PROGRAMS (VLP), a framework designed to induce neuro-symbolic programs that explain the underlying visual rule from a set of labeled image examples. VLP combines the perceptual strengths of VLMs with structured reasoning in a Domain-Specific Language (DSL), allowing for explicit, interpretable, and systematic reasoning grounded in visual perception.

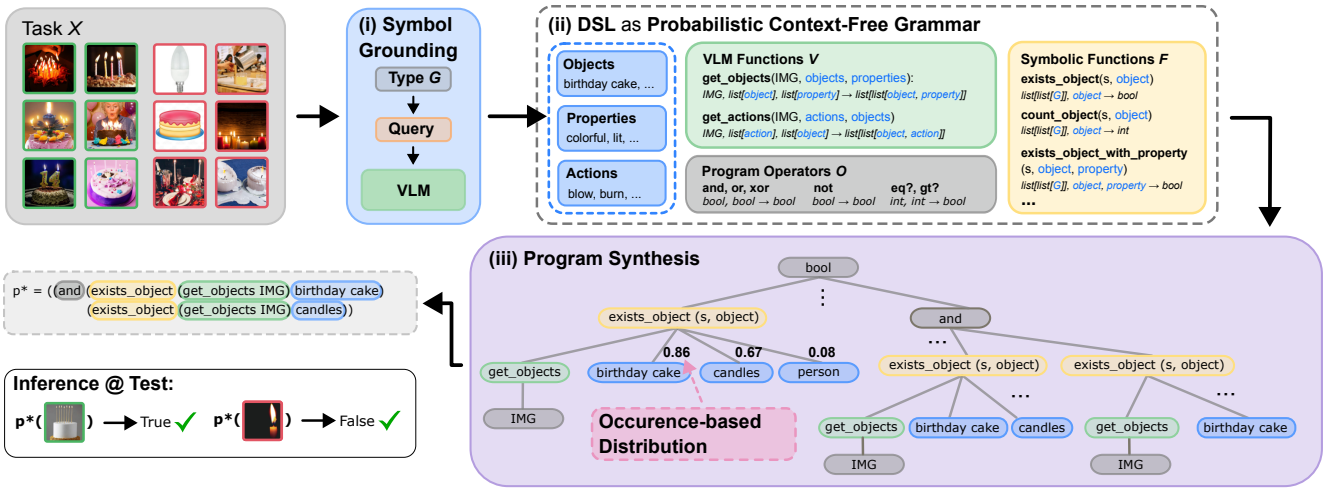

Figure 2. **Overview of VISION-LANGUAGE PROGRAMS synthesis**. Relevant variables are first discovered from the input examples (i) and used to construct a task-specific DSL, including VLM-based functions (ii). Program synthesis (iii) then searches this space to retrieve the most probable program that also achieves the highest accuracy on the input.

VLP operates in three consecutive stages. First, during *symbol grounding (i)*, relevant object, property, and action symbols are grounded for the task at hand. Second, a *Probabilistic Context-Free Grammar (ii)* is formed from the Domain-Specific Language (DSL) and the previously grounded symbols, including symbolic and neural, VLM-based functions. This forms a probabilistic interface for program search based on visual inputs. Finally, *program synthesis (iii)* searches over the solution space created by the grammar to synthesize a program that best distinguishes positive from negative examples. The resulting output program captures the semantic relations between the input images and can be executed on new test examples to infer new labels. We next detail each stage.

3.1. Problem Setup

We formulate inductive visual reasoning as the task of discovering a latent visual rule that explains a set of example images (denoted as *few-shot* examples in the remainder). Formally, let

$$\mathcal{X} = \{(I_1, y_1), (I_2, y_2), \ldots, (I_n, y_n)\}$$

denote a task, where each I_i is an input image and $y_i \in \{0, 1\}$ is the corresponding binary label. A label $y_i = 1$ indicates that I_i satisfies the latent visual rule (positive example), whereas $y_i = 0$ indicates that it does not (negative example). Each task is additionally associated with a set of held-out *query* samples for evaluation.

3.2. Symbol Grounding

The first stage of VLP establishes an interface between continuous visual inputs and discrete symbolic representations. This process, which we refer to as *symbol grounding*, maps perceptual information from images into structured, type-constrained symbols that form the atomic units for subsequent reasoning. We define three fundamental symbol types

$$\mathcal{G} = \{\texttt{object}, \texttt{property}, \texttt{action}\}.$$

These types constrain the semantic roles that individual symbols play in downstream program construction. Rather than relying on a fixed vocabulary, the vocabulary is dynamically adjusted based on the task at hand. This preserves generality and adaptation to novel domains and unseen visual compositions. In detail, given a reasoning task $\mathcal{X}$, VLP provides task-specific groundings of these abstract types by querying a pretrained VLM $\mathcal{M}$. For each symbol type $G_i \in \mathcal{G}$, the model proposes a set of groundings E_i:

$$\mathcal{M}(G_i, \mathcal{X}) = E_i, \quad E_i = \{e_{i,1}, \ldots, e_{i,m_i}\},$$

where each $e \in E_i$ denotes an individual grounding for symbol type G_i, and m_i is the number of symbols proposed for that type. For example, in Fig. 2 the symbols `birthday cake` and `candles` ground the type `object`, while `colorful` and `lit` ground the `property` type and `blow` and `burn` ground type `action`. The process is guided by type-specific queries to $\mathcal{M}$ (*cf*. Sec. H.2).

The result is a unified pool of symbols $\mathcal{E} := \bigcup_{i=1}^{|\mathcal{G}|} E_i$ that captures all objects, properties, and actions relevant to the current task. This pool acts as the semantic substrate for forming visual concepts. We note that although we consider three types in $\mathcal{G}$ in this work, the framework is not restricted to them. The types can be adapted or extended as needed.

3.3. Vision-Language DSL

The second element of VLP is a Domain-Specific Language (DSL) [5] that formalizes an interface between perception and reasoning. Unlike task-specific DSLs, ours is VLP-specific: it defines a general symbolic interface that remains invariant across domains and tasks. While the grounded symbols capture task-specific semantics, the DSL defines the syntax and functional structures shared across all tasks. It includes a small set of syntactic primitives such as `bool` and `int`, and a dedicated type `IMG`, representing the input domain. Beyond these primitives, the DSL defines three function types: *VLM functions*, *symbolic functions*, and *program operators*. VLM functions map perceptual features from images to symbolic representations, symbolic functions encode logical or arithmetic operations over these symbolic representations, and program operators compose these components into executable reasoning programs.

VLM functions $\mathcal{V}$ equip VLP with a perceptual interface to extract symbolic information from visual inputs using VLMs. Each function $v \in \mathcal{V}$ takes as input an image I together with a function-specific subset of the grounded symbol pool $\mathcal{E}_v \subseteq \mathcal{E}$, and outputs a structured symbolic representation $s \in \mathcal{S}$:

$$v(I, \mathcal{E}_v) = s.$$

In essence, VLM functions translate raw visual inputs into structured symbolic states. For example, the VLM function `get_objects` (see Fig. 2) relies only on the object and property groundings (*i.e.*, $\mathcal{E}_{\text{get\_objects}} = E_{\text{object}} \cup E_{\text{property}}$), extracting a structured object–property mapping for the given image. For the first input image (top left in Fig. 2), this mapping could be represented as:

```
[[birthday cake], [candles, colorful]].
```

The output provides a symbolic representation of the semantics of an image for downstream reasoning. Since $\mathcal{E}$ is constant across task images, we omit this input in the remainder of the paper and write functions like `get_objects` and `get_actions` as depending only on the image I.

Symbolic functions $\mathcal{F}$ form the core reasoning primitives of VLP. They operate directly on the symbolic representations s and capture the relationships, attributes, and interactions among them. Each function represents an interpretable reasoning step. For instance, the function `exists_object(s, e)` in Fig. 2 evaluates whether an object e appears in the representation s, returning a boolean. Other symbolic functions count objects or properties, verify the presence of specific actions, or symbol combinations. Collectively, these functions form the reasoning vocabulary of VLP, enabling structured queries over visual elements.

Program operators $\mathcal{O}$ specify how reasoning primitives in VLP can be composed. They encompass logical connectives (`AND`, `OR`, `NOT`) and comparison operators ($=, <, >$), enabling the construction of complex executable programs that represent abstract visual concepts.

3.4. DSL to Probabilistic Context-Free Grammar

To formalize program synthesis, VLP defines a Probabilistic Context-Free Grammar (PCFG) [5] derived directly from the type system of the DSL. This PCFG serves as a generative prior over the space of valid programs, enabling a systematic and efficient search for viable candidates.

Formally, for the grammar $\Gamma = (N, \Sigma, R, S, P)$, the nonterminals N correspond to the set of all valid return types in our DSL:

$$\mathcal{T} = \mathcal{G} \cup \{\texttt{IMG}, \texttt{bool}, \texttt{int}, \mathcal{S}\}.$$

The terminals Σ comprise the DSL functions ($\mathcal{V} \cup \mathcal{F} \cup \mathcal{O}$) and the grounded symbols $\mathcal{E}$. Since the synthesized program must classify images, the start symbol S is strictly fixed to the return type `bool`.

Grammar Construction. The production rules R are deterministically generated from the type signatures of the DSL elements. A function $f \in \Sigma$ that takes arguments of types $\tau_1, \ldots, \tau_k$ and returns τ_{out} yields the rule:

$$\tau_{out} \to f(\tau_1, \ldots, \tau_k).$$

For example, the boolean operations, VLM functions, and grounded symbols yield production rules such as:

$$\begin{aligned} \texttt{bool} &\to \textbf{exists\_object}(\mathcal{S}, \texttt{object}) \\ \mathcal{S} &\to \textbf{get\_objects}(\texttt{IMG}) \\ \texttt{object} &\to \textbf{birthday cake}. \end{aligned}$$

Recursively applying these rules expands the start symbol into a complete, type-consistent reasoning program.

Probabilistic Weighting. Because VLP is training-free, the production rules for structural DSL operations are assigned uniform weights. We introduce a data-driven inductive bias exclusively through the terminal rules for the grounded symbols $e \in \mathcal{E}$.

Rather than normalizing probabilities across symbol types, which would artificially penalize types with larger vocabularies by reducing their relative mass, we directly assign a score $w(e)$ to each symbol e. This weight is based on the product of the symbol's frequency and its precision in the positive support set:

$$w(e) = \frac{n_{\text{pos}}}{N_{\text{pos}}} \cdot \frac{n_{\text{pos}}}{n_{\text{pos}} + n_{\text{neg}}}$$

where n_{pos} and n_{neg} denote the number of occurrences of e in positive and negative examples respectively, and N_{pos} is the total count of positive examples. To prevent zero-weight assignments when $n_{\text{pos}} = 0$, we apply a small smoothing constant $w(e) = 0.01$.

The unnormalized weights are used directly during program search to calculate the generative prior of a program, ensuring that highly discriminative symbols are prioritized regardless of how many other symbols share their type.

3.5. Program Synthesis

Given the grammar Γ, VLP performs program synthesis by searching for an executable program p that best explains the task $\mathcal{X}$. Each program transforms an input image I_i into a boolean prediction $\hat{y}_i = p(I_i)$, whose correctness is evaluated against the ground-truth label y_i.

Program Search. The search systematically explores the space of candidate programs defined by the grammar. Specifically, we employ a top-down enumerative search starting from the start symbol *bool*. The search iteratively expands nonterminals, prioritizing derivations with higher cumulative rule weights to efficiently navigate the vast program space. For each fully expanded candidate program p, we compute its accuracy on $\mathcal{X}$:

$$\text{Acc}(p) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\big[p(I_i) = y_i\big]$$

as well as its generative prior $W(p)$, which is the product of the weights of all production rules used to construct p. To improve efficiency, the outputs of all VLM functions $\mathcal{V}$ are precomputed for every $I_i \in \mathcal{X}$ prior to search.

Candidate programs are then ranked by a two-level criterion: *Primary:* accuracy $\text{Acc}(p)$; *Secondary:* prior weight $W(p)$ to break ties. The top-ranked program p^* is selected as the final solution (*cf*. Fig. 2, bottom left).

4. Experiments

In the following, we evaluate VISION-LANGUAGE PROGRAMS on several datasets and tasks to assess neuro-symbolic VLP across various visual reasoning settings. We hereby address the following research questions:

- **(RQ1)** Can VLP improve concept learning performance over VLMs across varying visual domains?
- **(RQ2)** Can VLPs with non-reasoning models improve over dedicated reasoning models?
- **(RQ3)** Do VLP models exhibit advantages over VLMs with an increased number of training samples?
- **(RQ4)** Do VLP facilitate knowledge incorporation, *e.g.*, for shortcut mitigation?

Data. For the evaluation of our method, we use a set of different datasets across multiple domains that are based on synthetic and real-world images. Specifically, we evaluate on the datasets Bongard-HOI [13], Bongard-OpenWorld [44] and Bongard-RWR[19], which are based on real-world images and incorporate a diverse set of visual concepts. For a real-world dataset that provides more complex logical rules, we utilize COCOLogic [36] and create 10 tasks from it, one for each class. For the synthetic dataset, we use CLEVR-Hans3 [34], where we leverage the three classes to construct three tasks with complex logical rules from it. The Bongard datasets provide 12 samples from which the target rule should be induced, 6 positives and 6 negatives. For the other two datasets, we take 20 balanced support samples. Additional details are provided in Suppl. A.

Experimental Setup. We incorporate and compare to a selection of open source VLMs with varying sizes. Namely, we utilize InternVL3-8B and InternVL3-14B [2], Kimi-VL-A3B-Instruct [18], as well as Qwen2.5-VL-72B [22] and Qwen3-VL-30B-A3B-Instruct [23]. We prompt a model three times using sampling-based generation and perform program synthesis using the Heap Search Algorithm from [5] with a 10-second budget. The maximum program depth is 4 for Bongard-OpenWorld and Bongard HOI, and 6 for COCOLogic and CLEVR-Hans3. The DSL used for the experiments is provided in Suppl. G.

In the context of **RQ1** we compare the base models with and without VLP integration. In the context of **RQ2** we compare instruction-tuned models with VLP to dedicated reasoning models Kimi-VL-A3B-Thinking [18], Qwen3-VL-30B-A3B-Thinking [23] and gpt-5 [21]. For **RQ3**, we evaluate all models from RQ1 using an increased number of support samples and average their results.

For evaluations regarding knowledge incorporation (**RQ4**) we use CLEVR-Hans3 to investigate how DSL edits can be used to improve performance and mitigate shortcut learning. Across all evaluations, model performance is measured using balanced accuracy, reflecting each model's ability to correctly classify the query (test) images.

VLPs boost VLMs across domains (RQ1). In our first evaluation, we investigate the potential of our neuro-symbolic VLP framework to leverage the power of VLMs in diverse visual concept learning tasks. For this, we compare the performance of five different base VLMs with and without VLP processing on five datasets.

We observe in Tab. 1 that VLPs obtain substantially higher results across all models, with improvements of up to 13.5% on average across all datasets. Interestingly, model size appears to have minimal effect on VLP performance; however, smaller models (*e.g.*, InternVL3-8B and Qwen2.5-VL-7B) tend to benefit most from VLP-based processing. The strongest improvements across all mod-

Table 1. **Comparison of base VLMs and VLMs with VLP (averaged over three runs).** Balanced accuracy (%) on 6-shot Bongard tasks and 10-shot logical reasoning benchmarks. Improvements (green / gray) denote changes relative to the baseline. Best results per model are shown in **bold**; overall best results per dataset column are underlined.

Model	Avg.	Bongard Tasks (6-shot)			Logical Reasoning (10-shot)	
		Bongard-HOI	Bongard-OW	Bongard-RWR	COCOLogic	CLEVR-Hans3
InternVL3-8B	57.4	60.5	59.2	47.2	71.5	48.3
w/ **VLP**	**70.9** (+13.5)	**77.7** (+17.2)	**67.5** (+8.3)	**53.9** (+6.7)	**81.0** (+9.5)	**74.4** (+26.1)
InternVL3-14B	61.5	66.9	62.5	51.4	**78.4**	48.3
w/ **VLP**	**68.3** (+6.8)	**74.3** (+7.4)	**64.7** (+2.2)	**52.8** (+1.4)	76.7 (-1.7)	**73.3** (+25.0)
Kimi-VL-A3B-Instruct	58.5	59.8	58.6	46.4	**77.9**	50.0
w/ **VLP**	**65.5** (+7.0)	**69.4** (+9.6)	**59.4** (+0.8)	**52.5** (+6.1)	70.1 (-7.8)	**76.1** (+26.1)
Qwen2.5-VL-7B-Instruct	60.1	65.2	**66.2**	**49.7**	73.2	46.1
w/ **VLP**	**69.5** (+9.4)	**68.8** (+3.6)	62.9 (-3.3)	49.2 (-0.5)	**80.5** (+7.3)	**86.1** (+40.0)
Qwen3-VL-30B-A3B-Instruct	63.4	69.0	**68.5**	55.8	73.9	50.0
w/ **VLP**	**68.9** (+5.5)	**74.5** (+5.5)	66.3 (-2.2)	**58.3** (+2.5)	**79.1** (+5.2)	**66.1** (+16.1)

els occur on the compositionally complex CLEVR-Hans3 dataset, where the synthetic images are likely more out-of-distribution for pretrained VLMs, usually including 5 to 10 objects. This pattern suggests that structured reasoning offers greater advantages when perceptual uncertainty is higher. Notably, none of the model encoders were specifically finetuned on these datasets, demonstrating that VLP grants domain-independent flexibility.

Fig. 3 illustrates this effect of VLP's decoupling of perception from reasoning on a Bongard-RWR task. When prompted directly, the base Qwen3-VL model proposes a complex rule but fails to identify the simple underlying concept of "round vs. non-round objects," ultimately classifying both test images as positive. In contrast, the same base model with VLP successfully discovers the program

p^*= `(exists_property (get_objects IMG) round)`

by first identifying individual objects in each image, then inferring through program search that all positive images contain objects with the property `round`. The resulting Vision-Language Program p^* correctly classifies all test images.

The only dataset where VLP does not achieve the overall best performance is Bongard-OpenWorld. Upon examining the failure cases on this dataset, we identified numerous annotation errors (*cf*. Suppl. F). In such cases, standard VLM prompting tends to generate plausible but loosely defined rules that accommodate these inconsistencies. In contrast, VLP 's structured approach induces programs that remain logically consistent with the (incorrectly labeled) few-shot examples, but consequently fail the query samples.

Within the context of RQ1, we additionally investigate whether the observed improvements arise from the use of structured symbolic image representations (*e.g.*, `get_objects`, `get_actions`) by comparing LLM-based reasoning on symbolic inputs with VLP's reasoning in Suppl. Sec. C.1. The results indicate that, in fact, VLP's performance gains are primarily driven by the symbolic search process rather than the representation format. Conclusively, the induced rules by VLP are more reliable than rules obtained by prompting the model directly, even with structured symbolic representations at hand.

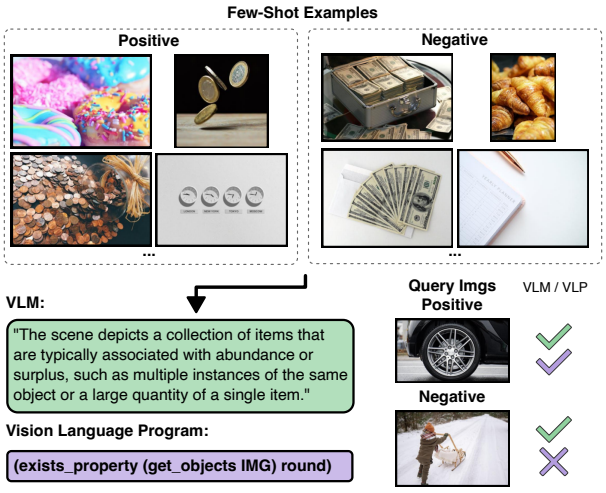

Figure 3. **Qualitative comparison on Bongard-RWR.** Direct VLM (Qwen3) prompting produces an incorrect rule about "abundance", misclassifying a query image. Qwen3 w/ VLP discovers a correct program that identifies round objects and achieves perfect query classification accuracy.

In summary, across these evaluations, we conclude that our neuro-symbolic VLP framework consistently enhances the reasoning capabilities of VLMs by effectively combining visual perception with symbolic search.

Table 2. **Comparison between *thinking* and *non-thinking* models with VLP**. VLP performs comparably or better than thinking models while using significantly fewer tokens, and can even build upon thinking-based approaches for further improvements.

	COCOLogic		CLEVR-Hans3	
	Acc	Tokens	Acc	Tokens
Kimi w/ Think	52.2	106,446	30.0	46,941
Kimi w/ **VLP**	**68.3**	**43,958**	**83.3**	**6,584**
Qwen3 w/ Think	**81.8**	108,346	46.7	106,922
Qwen3 w/ **VLP**	78.8	**22,341**	**68.3**	**5,052**
GPT-5 w/ Think	78.5	115,813	65.0	98,046
GPT-5 w/ **VLP**	81.8	**17,404**	68.3	**8,058**
GPT-5 w/ Think+**VLP**	**84.3**	183,387	**70.0**	84,642

Symbolic reasoning of VLPs outperforms VLM-based reasoning (RQ2). In our next evaluation, we compare the performance of VLP with instruction-tuned base VLMs to dedicated "reasoning" models. We focus on COCOLogic and CLEVR-Hans3 for this comparison, as these datasets exhibit the highest compositional and logical complexity, making them ideal testbeds for distinguishing structured symbolic reasoning from pure language-based inference. We compare Kimi-VL-A3B and Qwen3-VL-30B-Instruct with VLP to their "Thinking" counterparts. Additionally, we evaluate gpt-5 (high reasoning effort) against gpt-5-chat (no reasoning) and gpt-5 (low reasoning effort), both with VLP. We conduct the experiments with one run per model.

The results in Tab. 2 show that integrating VLP with non-reasoning models yields substantial improvements over dedicated reasoning models in all cases except Qwen3 on COCOLogic. Moreover, VLP-based reasoning requires substantially fewer tokens than dedicated reasoning models, despite executing more prompts, thereby demonstrating greater computational efficiency. Interestingly, integrating gpt-5 with low reasoning effort into VLP achieves improvements over both gpt-5 with Thinking and gpt-5-chat with VLP, while increasing token count only moderately on COCOLogic and even reducing it on CLEVR-Hans3. This suggests that VLP reasoning provides an orthogonal enhancement to the thinking mode of VLMs, yielding complementary performance benefits.

Overall, our findings suggest that VLP achieves strong reasoning performance while maintaining computational efficiency. We thus answer RQ2 affirmatively.

VLPs reliably integrate larger numbers of samples (RQ3). In many cases, having more samples for a specific task can provide valuable additional evidence for a concept and help refine it by excluding alternative explanations or shortcuts. In our third set of evaluations, we therefore examine how VLP handles larger numbers of input images,

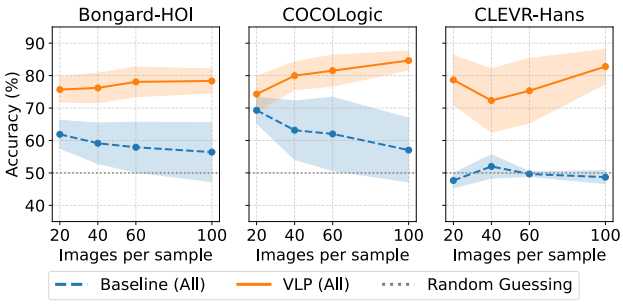

Figure 4. **VLP performance improves as more input images are provided**, in contrast to baselines, which stagnate or decline. Results are aggregated over models from Table 1.

particularly compared to the base VLM's image processing capacity. As shown in Fig. 4, we increase the number of input images per task from 20 to 100 across the Bongard-HOI, COCOLogic, and CLEVR-Hans3 datasets. For a fair comparison, the number of test images remains fixed in all runs, and we only consider concepts with sufficient support samples (*cf*. Suppl. A). For each dataset, we report the average performance across all VLM models from Tab. 1, comparing the base models with and without VLP.

We observe that performance improves for models using VLPs, as the additional evidence helps identify more accurate programs. For example, for the concept "*carrying surfboard*" in Bongard-HOI (*cf*. Suppl. Fig. 6), InternVL3-14B with 20 task examples discovers the program `((exists_action (get_actions IMG) holding))`, which does not specify the object being held but still covers 96% of the few-shot examples. With 100 examples, however, the model retrieves a more precise program, including the actions `holding` and `walking` in combination with the object `surfboard` (*cf*. Sec. B.1).

In contrast, the base VLMs do not benefit from an increased number of examples. Since they process all input images jointly, individual samples likely receive less attention, resulting in overly abstract learned representations. *E.g.*, for the previous *surfing* task with 20 images, InternVL3-14B produces the concept "*The activity involves walking with or actively surfing on a surfboard in a beach or ocean setting*," which is overly general and mistakenly includes negative examples (*surfing* instead of *carrying*).

Overall, these results highlight that VLP can effectively leverage larger sets of input images by synthesizing more specific and semantically grounded programs, while base VLMs struggle to capitalize on the additional data. We hence answer RQ3 affirmatively.

VLPs enable interaction with the Solution Space of the Program (RQ4). An important feature of VLPs is their inherent modularity, which allows users to inspect and in-

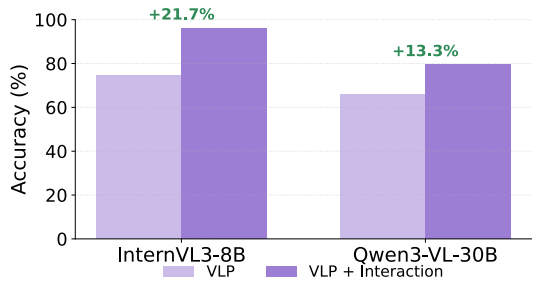

Figure 5. **VLP performance on CLEVR-Hans3 with DSL edits.** For InternVL3 `size`-related VLM functions were added, for Qwen3 shortcut-related colors (`red`, `gold`) removed.

teract with individual processing steps. This enables targeted debugging of failure cases and refinement of solutions through step-specific feedback. To investigate how this facilitates intuitive interactions with the task's solution space, valuable for model debugging and shortcut mitigation [7, 27, 34, 35], we investigate the solutions of the models InternVL3-8B and Qwen3-VL on the dataset CLEVR-Hans3 from Tab. 1.

Here, we observed that, *e.g.*, InternVL3-8B seldom considers the size of the objects, which is a relevant property for the rules. Looking at the grounded properties and the object-property representations, we observed that even though the size properties, *small* and *large*, are discovered, they are not always used in the object-property representation obtained by `get_objects`. An explanation for this could be that the concept *size* is very relative and depends on the context. Fortunately, thanks to the flexibility of VLP one can very simply add VLM functions to the DSL that directly ask for the size of a given object (*cf*. Sec. H.4) in *relation* to the other objects. As Fig. 5 (left) highlights, by incorporating these functions into the DSL the performance on the query samples improves substantially, *i.e.*, InternVL3-14B can now utilize this bias to achieve 96% accuracy.

In contrast, when inspecting the results of Qwen3-VL, we identified that the VLPs indeed make use of the VLM-proposed properties, however in some cases a VLP falsely considers the colors "red" and "gold" as relevant properties. Manual inspection of corresponding images (*e.g.* of task #2 in Suppl. Fig. 11), indeed identified these features as potential shortcuts for the task. Equipped with such discoveries a human user can easily interact with the DSL by removing "red" and "gold" from the available properties and by this targeted revision improve the model's performance by 13.3% (*cf*. Fig. 5 (right)).

These evaluations demonstrated that even with limited data, solution quality can be increased by incorporating additional task knowledge, *e.g.*, in a human-in-the-loop setting. We therefore conclude RQ4 positively.

5. Discussion

Across our evaluations, we have observed that VLP effectively improves visual inductive reasoning in terms of predictive performance improvements, but also model debugging. Below, we discuss additional considerations.

Our failure analysis indicates that models occasionally exhibit errors limiting overall effectiveness (*cf*. Suppl. E). These include formatting errors (*e.g.*, Kimi produced malformed representations in ∼13% of COCOLogic cases) and incomplete property descriptions, where VLMs omit relevant properties despite perceiving them (*cf*. Sec. E.2). Such perception failures constrain performance by limiting symbolic knowledge for reasoning. Notably, unlike opaque end-to-end VLMs, VLP enables tracing errors to individual images, providing actionable insights for debugging.

Looking forward to improving perceptual grounding quality, VLMs could be prompted for each symbol individually rather than all at once (*e.g.*, `get_objects`), though this would substantially increase prompt overhead. We view the current approach as an effective trade-off, with future work potentially pre-selecting promising symbols during search to reduce computational costs. Additionally, VLP currently lacks explicit object representations, limiting its use for spatial relations. Extending VLP with structured object representations would enable applications in expert domains like healthcare or mechanical engineering, where interactive human feedback is valuable.

Finally, a compelling frontier for future research lies in the automated expansion of the DSL, by enabling the framework to dynamically adapt its primitive set (including all functions, not only symbols) to the specificities of downstream tasks. We provide an initial feasibility analysis and proof-of-concept for this adaptive approach in Suppl. D.

6. Conclusion

In this work, we introduced VISION-LANGUAGE PROGRAMS (VLP), a framework that integrates the perceptual flexibility of VLMs with the systematic reasoning capabilities of program synthesis. By leveraging VLMs to generate structured symbolic descriptions that can be compiled into executable programs, VLP enable explicit composition, disambiguation, and, importantly, human-level interpretability beyond the natural language outputs of VLMs. Our empirical results across synthetic and real-world datasets show that VLP consistently improve generalization and predictive performance compared to direct prompting. Our work demonstrates that VLMs can be naturally integrated into a programmatic framework as callable functions rather than monolithic end-to-end predictors, bringing together the richness of neural vision-language representations with the reliability and controllabilty of symbolic program synthesis.

Acknowledgments

This work was supported by the Priority Program (SPP) 2422 in the subproject "Optimization of active surface design of high-speed progressive tools using machine and deep learning algorithms" funded by the German Research Foundation (DFG). The Eindhoven University of Technology authors received support from their Department of Mathematics and Computer Science and the Eindhoven Artificial Intelligence Systems Institute. Furthermore, this work has benefited from early stages of the Cluster of Excellence "Reasonable AI" funded by the German Research Foundation (DFG) under Germany's Excellence Strategy (EXC-3057), funding will begin in 2026.

We gratefully acknowledge support from the hessian.AI Service Center (funded by the Federal Ministry of Research, Technology and Space, BMFTR, grant no. 16IS22091) and the hessian.AI Innovation Lab (funded by the Hessian Ministry for Digital Strategy and Innovation, grant no. S-DIW04/0013/003).

References

[1] Shraddha Barke, Emmanuel Anaya Gonzalez, Saketh Ram Kasibatla, Taylor Berg-Kirkpatrick, and Nadia Polikarpova. Hysynth: Context-free LLM approximation for guiding program synthesis. *Advances in Neural Information Processing Systems*, 37, 2024. 2

[2] Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, and Lewei Lu. InternVL: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 5

[3] François Chollet. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019. 2

[4] Kevin Ellis, Lionel Wong, Maxwell Nye, Mathias Sable-Meyer, Luc Cary, Lore Anaya Pozo, Luke Hewitt, Armando Solar-Lezama, and Joshua B. Tenenbaum. Dreamcoder: growing generalizable, interpretable knowledge with wake-sleep Bayesian program learning. *Philosophical Transactions of the Royal Society A*, 381, 2023. 2

[5] Nathanaël Fijalkow, Guillaume Lagarde, Théo Matricon, Kevin Ellis, Pierre Ohlmann, and Akarsh Nayan Potta. Scaling neural program synthesis with distribution-based search. In *AAAI Conference on Artificial Intelligence*, 2022. 4, 5

[6] Artur d'Avila Garcez, Tarek R. Besold, Luc De Raedt, Peter Földiak, Pascal Hitzler, Thomas Icard, Kai-Uwe Kühnberger, Luis C. Lamb, Risto Miikkulainen, and Daniel L. Silver. Neural-symbolic learning and reasoning: contributions and challenges. In *AAAI Spring Symposium Series*, 2015. 1, 2

[7] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard S. Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2020. 8

[8] Sumit Gulwani. Automating string processing in spreadsheets using input-output examples. *ACM SIGPLAN Notices*, 46(1):317–330, 2011. 2

[9] Sumit Gulwani, Oleksandr Polozov, and Rishabh Singh. Program synthesis. *Foundations and Trends in Programming Languages*, 4, 2017. 2

[10] Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. 2

[11] Lukas Helff, Wolfgang Stammer, Hikaru Shindo, Devendra Singh Dhami, and Kristian Kersting. V-LoL: A diagnostic dataset for visual logical learning. *Journal of Data-centric Machine Learning Research*, 2024. 1

[12] Lukas Helff, Ahmad Omar, Felix Friedrich, Antonia Wüst, Hikaru Shindo, Rupert Mitchell, Tim Woydt, Patrick Schramowski, Wolfgang Stammer, and Kristian Kersting. SLR: Automated synthesis for scalable logical reasoning. *arXiv preprint arXiv:2506.15787*, 2025. 1

[13] Huaizu Jiang, Xiaojian Ma, Weili Nie, Zhiding Yu, Yuke Zhu, and Anima Anandkumar. Bongard-HOI: Benchmarking few-shot visual reasoning for human-object interactions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 5, 1

[14] 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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017. 1

[15] Danial Kamali and Parisa Kordjamshidi. NePTune: A neuro-pythonic framework for tunable compositional reasoning on vision-language. In *The Fourteenth International Conference on Learning Representations*, 2026. 2

[16] Amita Kamath, Jack Hessel, and Kai-Wei Chang. What's "up" with vision-language models? Investigating their struggle with spatial reasoning. In *Conference on Empirical Methods in Natural Language Processing*, 2023. 1

[17] Henry Kautz. The third AI summer: AAAI Robert S. Engelmore memorial lecture. *AI Magazine*, 2022. 1, 2

[18] Kimi Team. Kimi-VL technical report, 2025. 5

[19] Mikołaj Małkiński, Szymon Pawlonka, and Jacek Mańdziuk. Reasoning limitations of multimodal large language models. a case study of bongard problems. In *Forty-second International Conference on Machine Learning*, 2025. 5, 1, 13

[20] Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. DeepProbLog: Neural probabilistic logic programming. In *Advances in Neural Information Processing Systems*, 2018. 1, 2

[21] OpenAI. GPT-5. https://openai.com/index/introducing-gpt-5/, 2025. Accessed: 2025-11-04. 5

[22] Qwen Team. Qwen2.5-VL, 2025. 5

[23] Qwen Team. Qwen3 technical report, 2025. 5

[24] Pooyan Rahmanzadehgervi, Logan Bolton, Mohammad Reza Taesiri, and Anh Totti Nguyen. Vision language models are blind. In *Asian Conference on Computer Vision*, 2024. 1

[25] Gabriel Herbert Sarch, Snigdha Saha, Naitik Khandelwal, Ayush Jain, Michael J. Tarr, Aviral Kumar, and Katerina Fragkiadaki. Grounded reinforcement learning for visual reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. 1

[26] Md Kamruzzaman Sarker, Lu Zhou, Aaron Eberhart, and Pascal Hitzler. Neuro-symbolic artificial intelligence. *AI Communications*, 34, 2021. 1, 2

[27] Patrick Schramowski, Wolfgang Stammer, Stefano Teso, Anna Brugger, Franziska Herbert, Xiaoting Shao, Hans-Georg Luigs, Anne-Katrin Mahlein, and Kristian Kersting. Making deep neural networks right for the right scientific reasons by interacting with their explanations. *Nature Machine Intelligence*, 2, 2020. 8

[28] Hikaru Shindo, Viktor Pfanschilling, Devendra Singh Dhami, and Kristian Kersting. αILP: thinking visual scenes as differentiable logic programs. *Machine Learning*, 2023. 1, 2

[29] Hikaru Shindo, Viktor Pfanschilling, Devendra Singh Dhami, and Kristian Kersting. Learning differentiable logic programs for abstract visual reasoning. *Machine Learning*, 2024. 1, 2

[30] Parshin Shojaee, Seyed Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. 1

[31] Arseny Skryagin, Wolfgang Stammer, Daniel Ochs, Devendra Singh Dhami, and Kristian Kersting. Neural-probabilistic answer set programming. In *International Conference on Principles of Knowledge Representation and Reasoning*, 2022. 2

[32] Arseny Skryagin, Daniel Ochs, Devendra Singh Dhami, and Kristian Kersting. Scalable neural-probabilistic answer set programming. *Journal of Artificial Intelligence Research*, 2023. 2

[33] Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025. 1

[34] Wolfgang Stammer, Patrick Schramowski, and Kristian Kersting. Right for the right concept: Revising neuro-symbolic concepts by interacting with their explanations. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 2, 5, 8, 1

[35] David Steinmann, Felix Divo, Maurice Kraus, Antonia Wüst, Lukas Struppek, Felix Friedrich, and Kristian Kersting. Navigating shortcuts, spurious correlations, and confounders: From origins via detection to mitigation. *arXiv preprint arXiv:2412.05152*, 2024. 8

[36] David Steinmann, Wolfgang Stammer, Antonia Wüst, and Kristian Kersting. Object centric concept bottlenecks. *Advances in Neural Information Processing Systems*, 2025. 5, 1

[37] Sanjay Subramanian, Medhini Narasimhan, Kushal Khangaonkar, Kevin Yang, Arsha Nagrani, Cordelia Schmid, Andy Zeng, Trevor Darrell, and Dan Klein. Modular visual question answering via code generation. In *Annual Meeting of the Association for Computational Linguistics*, 2023. 2

[38] Vishal Sunder, Ashwin Srinivasan, Lovekesh Vig, Gautam Shroff, and Rohit Rahul. One-shot information extraction from document images using neuro-deductive program synthesis. In *International Workshop on Neural-Symbolic Learning and Reasoning*, 2019. 2

[39] Dídac Surís, Sachit Menon, and Carl Vondrick. ViperGPT: Visual inference via python execution for reasoning. In *IEEE/CVF International Conference on Computer Vision*, 2023. 2

[40] 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. 2

[41] Yiqi Wang, Wentao Chen, Xiaotian Han, Xudong Lin, Haiteng Zhao, Yongfei Liu, Bohan Zhai, Jianbo Yuan, Quanzeng You, and Hongxia Yang. Exploring the reasoning abilities of multimodal large language models (MLLMs): A comprehensive survey on emerging trends in multimodal reasoning. *arXiv preprint arXiv:2401.06805*, 2024. 1

[42] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 2022. 1

[43] Guojun Wu. It's not that simple. An analysis of simple test-time scaling. *arXiv preprint arXiv:2507.14419*, 2025. 1

[44] 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*, 2024. 5, 1

[45] Antonia Wüst, Wolfgang Stammer, Quentin Delfosse, Devendra Singh Dhami, and Kristian Kersting. Pix2Code: Learning to compose neural visual concepts as programs. In *Conference on Uncertainty in Artificial Intelligence*, 2024. 1, 2

[46] Antonia Wüst, Tim Tobiasch, Lukas Helff, Inga Ibs, Wolfgang Stammer, Devendra Singh Dhami, Constantin A. Rothkopf, and Kristian Kersting. Bongard in wonderland: Visual puzzles that still make AI go mad? In *International Conference on Machine Learning*, 2025. 1, 13

[47] Yuan Yang and Le Song. Learn to explain efficiently via neural logic inductive learning. In *International Conference on Learning Representations*, 2020. 2

[48] Zhiyuan Zeng, Qinyuan Cheng, Zhangyue Yin, Yunhua Zhou, and Xipeng Qiu. Revisiting the test-time scaling of o1-like models: Do they truly possess test-time scaling capabilities? In *Annual Meeting of the Association for Computational Linguistics*, 2025. 1

[49] Yizhe Zhang, He Bai, Ruixiang Zhang, Jiatao Gu, Shuangfei Zhai, Josh Susskind, and Navdeep Jaitly. How far are we from intelligent visual deductive reasoning? *ICLR Workshop: How Far Are We From AGI*, 2024. 1

[50] Kankan Zhou, Eason Lai, Wei Bin Au Yeong, Kyriakos Mouratidis, and Jing Jiang. Rome: Evaluating pre-trained vision-language models on reasoning beyond visual common sense. *Findings of the Association for Computational Linguistics (EMNLP)*, 2023. 1