Composing Neural Learning and Symbolic Reasoning with an Application to Visual Discrimination

Adithya Murali[1*], **Atharva Sehgal**[2*], **Paul Krogmeier**[1] and **P. Madhusudan**[1]
[1]Department of Computer Science, University of Illinois at Urbana-Champaign
[2]Department of Computer Science, University of Texas at Austin
adithya5@illinois.edu, atharvas@utexas.edu, paulmk2@illinois.edu, madhu@illinois.edu

Abstract

We consider the problem of combining machine learning models to perform higher-level cognitive tasks with clear specifications. We propose the novel problem of Visual Discrimination Puzzles (VDP) that requires finding interpretable discriminators that classify images according to a logical specification. Humans can solve these puzzles with ease and they give robust, verifiable, and interpretable discriminators as answers. We propose a compositional neurosymbolic framework that combines a neural network to detect objects and relationships with a symbolic learner that finds interpretable discriminators. We create large classes of VDP datasets involving natural and artificial images and show that our neurosymbolic framework performs favorably compared to several purely neural approaches.

1 Introduction

Deep learning has made significant strides in solving specialized tasks, especially in areas such as vision and NLP [Goodfellow *et al.*, 2016]. In this paper we study how these specialized models can be *composed and integrated* into solutions for high-level tasks with clear specifications. We are especially interested in solutions for settings without a lot of data.

We believe the problem of integrating learned models is important and can have many applications. For example, a robot may need to formulate a complex plan that utilizes pretrained vision components to detect objects and prediction of human behavior. It is challenging to utilize pretrained components to achieve a high-level task. In this paper, we investigate neurosymbolic techniques for higher level symbolic reasoning using neural components for such problems.

1.1 Visual Discrimination Puzzles

In this paper, we propose a new high-level task called a *Visual Discrimination Puzzle* (VDP). Figure 1 shows an example of a VDP[1]. The first row contains some *example images* E (a, b, and c), and the second row consists of *candidate images* C

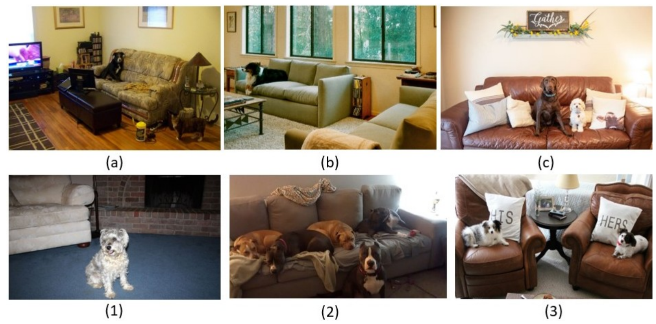

Figure 1: *A Visual Discrimination Puzzle*

(1, 2, and 3). To solve the puzzle we are asked to answer the following question:

Which candidate image is most similar to all of the example images? Explain why.

Humans have no prior experience with VDPs, but seem to be able to solve them with ease. We invite the reader to try solving the puzzle in Figure 1 before reading further.

When solving a VDP, different people can (and do, in our limited experience) come up with different answers. Many people select #3 as the answer to the puzzle in Figure 1 and explain that, in each example image as well as candidate image #3, *all dogs are sitting on sofas* while this is not true in the other candidate images.

A natural formalization of the specification for VDP puzzles is the following:

> *Is there a property P that holds in all example images and exactly one candidate image, but does not hold in the other candidate images?*

We call such a property P a *discriminator*. Solving a VDP reduces to finding a discriminator.

Observe the following salient aspects of the problem. First, the set E of example images is small. A VDP solver must learn the common concept P using only 3-5 images. Second, unlike visual question answering (VQA), where one *computes* a query over a scene, solving VDPs requires *searching* in a large space of discriminators for one that satisfies the puzzle specification. Third, the logical specification of the problem makes the solution sensitive to every image. A solver cannot merely identify a candidate image that is similar to all example

[*]Contact Authors

[1]See Appendix in the full version of the paper for image attributions: https://arxiv.org/abs/1907.05878

images— it must also *exclude* the other candidates. Consequently, a candidate c may be a solution for one puzzle but not in another where different candidate images are present. For example, if we modified the *non-answer* image #1 in Figure 1 so that the dog *is* on the sofa, the solution might be different from #3 as the concept 'all dogs are on sofas' no longer corresponds to a unique candidate. Even swapping the solution candidate image with an example image could potentially change the solution (as there could be a *simpler* discriminator for this new puzzle that chooses a different solution). E.g., P can choose #1 in the first puzzle and Q can be a simpler discriminator that chooses #2 in the second puzzle, but is not a solution for the first puzzle. Finally, we would accept an answer from a person or machine only if it is justified— we want a precise, interpretable concept P that can be evaluated on images to see that it indeed satisfies the puzzle specification.

1.2 A Neurosymbolic Framework

We propose that effective integrative frameworks can be obtained by composing learned neural models for vision and symbolic reasoning, where the latter caters to the higher-level task specification. Designing such a framework poses several challenges, including **(1) Interface:** determining the exact communication between the neural and symbolic components; **(2) Interpretability:** explaining decisions of symbolic components, potentially in terms of inputs received from neural components; and **(3) Robustness:** symbolic components should ideally be robust to vagaries of individual neural components and to different implementations.

In this paper, we instantiate a neurosymbolic architecture for solving VDPs that addresses these three challenges in novel ways. In this work, the neural components are realized with state-of-the-art vision models that are trained offline on thousands of images. Given an image, the vision model detects a *scene graph* consisting of objects, their labels, bounding boxes or relative positions, and their relationships. Given a puzzle, we extract one scene graph for each image in the puzzle.

We propose an interface to the symbolic component using *first-order logic scene models*, which can be automatically computed from scene graphs. The symbolic component uses discrete search (realized efficiently using a SAT solver) to synthesize a discriminator P expressible in *first-order logic* over scene models, which identifies a candidate c as a solution to a given puzzle. The symbolic synthesis not only solves the puzzle by finding an appropriate candidate, but also justifies the choice using a first-order formula that is eminently interpretable. For the puzzle in Figure 1, our system would potentially find the discriminator:

$$\forall x.\, (\mathsf{dog}(x) \implies \exists y.\, \mathsf{sofa}(y) \land \mathsf{sitting\_on}(x, y))$$

The robustness of such a system certainly depends on the robustness of the vision model (for example, it is hard to solve the puzzle in Figure 1 if a dog is not detected). However, there are other robustness properties of interest. In particular, if the vision component improves and detects new objects or relationships, we would like any discriminator found before the improvement to *remain* a discriminator with respect to richer scene models that contain these new objects and relationships. We call this the *extension property*. We build a domain-specific logic called *First-Order Scene Logic* (FO-SL), prove that it has the extension property, and use it to express discriminators.

The problem of synthesizing quantified discriminators in FO, and in particular FO-SL, is a relatively new problem (as opposed to *program* synthesis). Reducing the problem to off-the-shelf synthesizers does not scale, and we build our own SAT-based symbolic solvers for synthesizing quantified discriminators.

Evaluation. We create three VDP datasets. Two of these are based on real-world scenes and contain ~ 9000 puzzles, and the other (synthetic) dataset is based on the CLEVR domain and consists of 825 puzzles. We implement and evaluate our framework on the real-world datasets and show that it is effective and robust. It solves 68% and 80% of the puzzles in the two datasets and gives sensible discriminators. We perform ablation studies that examine the effectiveness of the domain-specific logic FO-SL as well as the synthesis algorithm. The ablation for FO-SL involves implementing a second synthesis solver based on a different technique. We also compare our framework with purely neural baselines based on image similarity models [Wang *et al.*, 2014] and prototypical networks [Snell *et al.*, 2017], and we show that they perform poorly ($\sim 40\%$). This comparison is made using the CLEVR VDP dataset, which is designed to have unique minimal discriminators in FO-SL. Finally, we also create a dataset called ODDONE consisting of 1872 puzzles, which involves a different specification (picking the "odd one out" from a set of images) adapted from the CLEVR VDP dataset. We use this dataset to evaluate how well various approaches adapt to new high-level specifications without retraining.

Contributions. The primary contributions of this paper are: (1) the VDP problem, which (cognitively) involves visual perception as well as a search for interpretable concepts, (2) sets of ~ 11600 VDP and ODDONE puzzles that span both natural scenes and synthetic scenes, (3) an instantiation of the neurosymbolic framework with a novel interface between neural and symbolic components, the FO-SL logic for robust visual discriminators, and an efficient discriminator synthesis algorithm based on SAT solving, (4) an evaluation including ablation studies and comparisons to purely neural baselines.

2 Related Work

The idea of learning in two phases, with a first phase involving specific concepts learned from a large dataset and a second phase involving few-shot learning with concepts from the first phase, is not new. The work on recognizing handwritten characters [Lake *et al.*, 2015] explores a similar idea. In that work, the first phase learns a generative model of handwritten characters using strokes and the second phase uses Bayesian learning. In our work, we use neural models for object detection and then SAT-based synthesis of first-order formulas. VDP puzzles, however, are different because they require the chosen candidate to be discriminated from other candidates, and this suggests the use of logic. Synthesizing programs to explain behavior and generalize has been explored in various other work recently [Ellis *et al.*, 2018; Liu *et al.*, 2019].

Synthesizing programs from discrete data has been studied by both the AI and programming languages communities; the former in inductive logic programming (ILP) [Nédellec, 1998] and the latter in program synthesis [Alur *et al.*, 2018; Gulwani *et al.*, 2015] (including the use of SAT/SMT solvers [Solar-Lezama *et al.*, 2006; Alur *et al.*, 2015]).

There has been a flurry of recent work in combining neural and symbolic learning techniques [Amizadeh *et al.*, 2020; Yi *et al.*, 2018; Mao *et al.*, 2019] for problems where large datasets are available. In some cases the goal is to learn a program (e.g., learning programs as models of natural scenes [Liu *et al.*, 2019], assisting programmers by learning from large code repositories [Raychev *et al.*, 2019]) , and several new techniques have emerged [Balog *et al.*, 2017; Parisotto *et al.*, 2017; Murali *et al.*, 2018; Devlin *et al.*, 2017; Bunel *et al.*, 2018; Sun *et al.*, 2018; Evans and Grefenstette, 2018] . In this context, our work is novel in that it combines the neural and symbolic components in two different layers, where the symbolic layer is used to synthesize interpretable logical discriminators and handle few-shot learning effectively.

A closely related problem is that of Visual Question Answering (VQA) [Antol *et al.*, 2015]. Neurosymbolic approaches [Yi *et al.*, 2018; Amizadeh *et al.*, 2020] have been used to disentangle visual and NLP capabilities from reasoning in order to solve VQA on artificially rendered images. VQA involves *queries* about a single image or scene and end-to-end learning algorithms are commonly used. However, VDP puzzles require *searching* through a large class of potential discriminators, which is inherently a higher-level task. In a sense, we are asking whether there is *some question* for which a VQA engine would say 'yes' on some images and 'no' on others. discriminator. The work in [Andreas *et al.*, 2018] solves few-shot classification problems in this manner, but does not handle specifications like that of VDPs, which involve negation.

The works on Neural Module Networks [Andreas *et al.*, 2016] and Concept Bottleneck Models [Koh *et al.*, 2020] explore methods of embedding symbolic representations or concepts into neural network architectures. However, it is unclear whether there are methods to *search* for concepts (say, satisfying specifications) using these architectures.

Another related problem is solving puzzles from Raven's Progressive Matrices [Santoro *et al.*, 2018; Zhang *et al.*, 2019]. While the puzzles are similar to ours in specification, the space of concepts is minuscule and does not require any synthesis. The work in [Zhang *et al.*, 2019] shows that simply enumerating the concepts achieves 100% accuracy on these datasets.

3 Composing Neural Learning and Symbolic Reasoning for Discriminating Scenes

Prior to solving VDPs, humans have first learned to distinguish objects (*dogs, sofas*, detect relationships (*dog sitting on sofa*), poses (*woman standing*), and other attributes (*cat has closed eyes*) by drawing from a rich visual experience accumulated from childhood. Upon first encountering a VDP puzzle, humans do not have the same rich experience to go by (they likely haven't solved any VDPs previously). Despite this lack of prior experience, they quickly formulate a mechanism to identify a solution by *composing high-level concepts that use learned lower-level concepts*.

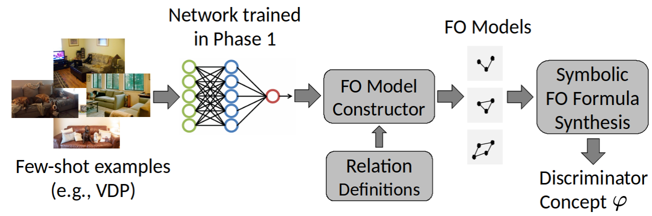

Figure 2: *Compositional Framework Combining Neural Learning and Symbolic Reasoning*

Our framework is built on this intuition and involves two similar phases. We propose a neurosymbolic approach to solve VDPs, as illustrated in Figure 2 (see Appendix for image attributions). Phase 1 involves *long-term learning* using large training sets that is independent of any high-level task. Phase 2 depends on the specification of a particular task (solving a VDP), and it utilizes the concepts learned in Phase 1 to meet the higher-level specification (few-shot discrimination of scenes). More precisely, we propose:

Neural Learning Algorithms for Phase 1. Learning a scene representation in terms of objects ("dog"), attributes ("dog is black"), and relationships ("dog is sitting on sofa") — called a scene graph — is a well-studied problem in literature, and convolutional neural networks (CNNs) are effective models. In this paper we use YOLOV4 [Wang *et al.*, 2021], a CNN-based object detector trained on the IMAGENET and COCO datasets to predict multiple objects with *bounding boxes* and *class labels*.

Interfacing Phase 1 and Phase 2 using Scene Models. The interface between the output of the neural component and the input of the symbolic synthesis component is an important challenge. We propose a novel interface, namely *First-Order Scene Models*, that capture object classes, attributes, and relationships.

Symbolic Synthesis Algorithms for Phase 2. In this phase we build novel algorithms to synthesize *quantified first-order logic* formulas that satisfy the VDP puzzle specification and which discriminate between the *finite* first-order scene models for each image.

If solving VDPs were the only goal, we could train models on a large class of puzzles. Our goal instead is to focus on a different kind of solution; we want to study how to solve puzzles and tasks afresh, i.e., without access to a rich experience in solving them.

4 Synthesis of First-Order Logic Discriminators

4.1 First-Order Logic and Scene Models

We work with first-order *relational* logic over a signature $\Sigma = (\mathcal{L}, \mathcal{R})$, where $\mathcal{R}$ is a finite set of relation symbols and $\mathcal{L} \subseteq \mathcal{R}$ is a set of unary symbols we call *labels* (which we use to model categories of objects). Each relation symbol is associated with an arity $n \in \mathbb{N}, n > 0$.

The syntax of first-order logic formulas is given by:

$$\textit{Formulas} \ \ \varphi ::= R(x_1, \ldots, x_k) \mid x_i = x_j \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \neg\varphi \mid \varphi \Rightarrow \varphi \mid \exists x.\, \varphi \mid \forall x.\, \varphi$$

We use the standard notions of models and semantics for first-order logic (see a standard logic textbook [Enderton, 2011]).

Given a set of images X for a VDP puzzle, we build a first-order model for each image $I \in X$ by feeding I to the pretrained neural network. The model's universe corresponds to the set of objects detected by the network. We model the class labels identified by the network as unary predicates $\mathcal{L}$ (e.g. $cat(x)$ or $person(x)$), and the identified relationships between objects as relations $\mathcal{R}$. The resulting *First-Order Scene Models* are used by the symbolic synthesizer.

4.2 First-Order Scene Logic and the Extension Property

We argue that a first-order logic over scene models (that are obtained from neural networks) should satisfy a particular *robustness property*. In a suitable logic, we would like any discriminator to remain a discriminator whenever *additional* objects or relationships are discovered by a vision model.

Suppose that we have found a discriminator for a puzzle, say, *all dogs are on sofas*. We would like it to remain a discriminator if new *irrelevant* objects and relationships are detected and added to the scene model. For example, if we add to a scene model a previously unrecognized pen then our discriminator should still work as a solution.

Standard first-order logic does not have this property. For example, in the puzzle in Figure 1, if only dogs and sofas were recognized then we could express "all dogs are on sofas" using the formula $\forall x.\exists y.(\textit{sofa}(y) \wedge (x \neq y \Rightarrow \textit{on}(x, y)))$, which says that all objects other than a sofa are on a sofa. If the vision model begins to recognizes tables in the images, then the formula fails to be a discriminator (it would be false on the example images).

We define *model extensions* as follows:

Definition 1 (Model Extension)**.** *A model M' over $\mathcal{R}'$* extends *a model M over $\mathcal{R}$ (where $\mathcal{R} \subseteq \mathcal{R}'$) if M' agrees with M on the interpretation of all relations in $\mathcal{R}$.*

Thus, given (1) the imprecision of vision models, (2) that different systems will likely have different detection rates for various object classes and relationships, and (3) that the choice of visual system crucially affects the scene model and consequently the discriminators, we propose the following property for discriminators:

Definition 2 (Extension Property)**.** *A logic* has the extension property *if any discriminator φ for a set of models $\{M_i\}$ remains a discriminator for any extended models $\{M'_i\}$.*

We formulate *First-order Scene Logic* (FO-SL) based on guarded logics, wherein any quantified object is always *guarded* by an assertion that it has a specific object label:

$$\text{FO-SL} ::= \forall x.L(x) \Rightarrow \varphi \ \mid\ \exists x.L(x) \wedge \varphi \ \mid\ \psi$$
$$\psi ::= R(\overline{x}) \ \mid\ \psi \vee \psi \ \mid\ \psi \wedge \psi \ \mid\ \neg\psi \ \mid\ \psi \Rightarrow \psi$$

In the grammar above, L ranges over label relations, e.g. $cat(x)$, and R ranges over relation symbols (attributes and object relationships).

FO-SL can express properties of all cats in a scene, but not of all objects of *any* kind (a variable cannot range over cats, pens, paintings, and specks of dust). With this we can show (see Appendix for proof):

Theorem 1. *The guarded fragment has the extension property.*

We thus propose the use of FO-SL as our space of possible discriminators.

4.3 Solving VDP by Synthesizing Formulas

We solve VDPs by synthesizing formulas that serve as discriminators for scene models. Let us fix a puzzle with example images E and candidate images C, as well as a signature Σ that is determined by a given vision model. Let the corresponding scene models be $E_M = \{e_1^M, \ldots, e_u^M\}$ and $C_M = \{c_1^M, \ldots, c_v^M\}$.

Definition 3 (Discriminator)**.** *An first-order sentence φ is a* discriminator *for (E_M, C_M) if there is a model $\widehat{c}^M \in C_M$ such that:*
(D1) For every model $e^M \in E_M$, $e^M \models \varphi$
(D2) $\widehat{c}^M \models \varphi$
(D3) For every $c^M \in C_M$ such that $c^M \neq \widehat{c}^M$, $c^M \not\models \varphi$

This definition formally captures the puzzle specification: a discriminator is a formula that is true in all example images (D1) and true in *exactly* one candidate image (D2 and D3).

Learning FO Discriminators

We describe an algorithm for synthesizing FO-SL discriminators to solve a given VDP puzzle. In particular, we build an algorithm that finds *conjunctive* discriminators.

We tried using state-of-the-art *program synthesis* engines that handle the SyGuS format (Syntax-Guided Synthesis) [Alur *et al.*, 2015] . These did not scale well (see Section 5.3), so we implemented our own solution using SAT solvers.

If k is the number of quantifiers in the discriminator, then we initially set k to 1 and iteratively increment it whenever we cannot find a discriminator with k quantifiers. We introduce a Boolean variable b_a for every atomic formula $a(\overline{x})$ that can occur in the matrix, i.e., the quantifier-free part of the formula. The intention is that b_a is true if and only if the atomic formula a occurs as a conjunct in the matrix of the discriminator. We also introduce k Boolean variables that determine whether the k quantifiers are existential or universal, as well as variables that determine the guards (labels in FO-SL) for each quantified variable. Given a valuation $\overline{b}$ for these variables, we can write a formula $\phi(\overline{b})$ that evaluates the discriminator encoded by $\overline{b}$ on all scene models. With extra Boolean variables that encode the choice of candidate scene model, we can formulate a constraint that reflects the specification of the puzzle and the definition of discriminator above.

We then use a SAT solver (such as Z3 [de Moura and Bjørner, 2008]) to determine whether the constraint is satisfiable. If the SAT solver finds a satisfying valuation, we can construct the FO-SL discriminator and the chosen candidate

ID	Class Description	ID	Class Description
1	All teddy bears on a sofa	2	There is an SUV
3	Onward lane (person on left sidewalk)	4	Fruit in separate piles (apples & oranges)
5	Kickoff position (ball b/w two people)	6	Laid out place setting (v/s dirty dishes)
7	Person kicking ball	8	Dog herding sheep
9	Parking spot	10	People carrying umbrellas
11	Bus filled with people	12	All dogs on sofas
13	Desktop PC	14	People wearing ties
15	Person sleeping on bench	16	All cats on sofas
17	Kitchen	18	TV is switched on
19	Two cats on same sofa	20	Cat displayed on TV

Table 1: Concept Class Descriptions for Natural Scenes Dataset

from the valuation. If the constraint is unsatisfiable, then we know there is no conjunctive discriminator with k quantifiers.

5 Evaluation

5.1 Datasets

We create 11,600 puzzles across four datasets. We describe these briefly; see the Appendix for details. We also invite the reader to browse the static website of VDPs[2] for a sample of puzzles across the datasets.

Natural Scenes. The Natural Scenes VDP dataset is created from 20 *base* real-world concept classes such as "all dogs are on sofas". For each class, we collect positive images that satisfy the concept and negative images that do not. We create puzzles by choosing all examples from the positive set and all candidates from the negative set except for one positive image (the *intended* candidate). We sample 3864 puzzles randomly. We provide a description of these concepts in Table 1.

GQA VDP dataset. The GQA VDP dataset is created automatically using the GQA dataset [Hudson and Manning, 2019], which is a VQA dataset. It consists of real-world scenes along with scene graphs, as well as questions and answers about the images. We use questions with yes/no answers such as: *Is there a fence made of wood?* We create puzzles as described above (with "yes" images being the positive category) and sample 5000 random VDPs. Note that the proposition corresponding to the question, i.e. *there is a fence made of wood*, is hence a discriminator.

CLEVR VDP dataset. The CLEVR domain [Johnson *et al.*, 2017] is an artificial VQA domain consisting of images with 3D shapes such as spheres, cubes, etc. The objects possess a rich combination of attributes such as shape, colour, size, and material. The CLEVR VDP dataset consists of 15 base concept classes as described in Table 2. Each concept class is a *schema* such as 'ShapeX and ShapeY have the same color', where the variables ShapeX and ShapeY denote distinct shapes. We create abstract VDPs whose unique minimal discriminator is contained in the FO-SL schema (e.g., $\forall x.ShapeX \Rightarrow \forall y.ShapeY \Rightarrow samecolor(x, y)$). We instantiate these variables with various combinations of shapes, colours, etc., and we sample 825 VDPs with unique solutions in the FO-SL conjunctive fragment.

ID	Concept Class Schema
1	Every `shapeX` has a `shapeY` to its left and right.
2	There is a `shapeX` of color `colorA` to the left of a `shapeY` of color `colorB`.
3	There is a `shapeX` to the right of every `shapeY`.
4	There is a `shapeX` and there is a `shapeY` to the left of all `shapeX`.
5	Every `shapeX` has a `shapeY` to its right.
6	All `shapeX`s and `shapeY`s have the same color.
7	There is no color such that there is only one `shapeX` of that color.
8	There is a leftmost `shapeX` and a rightmost `shapeX`.
9	All `shapeX` are to the left of all `shapeY`s and the rightmost `shapeY` is made of `materialQ`.
10	All `shapeX`s are to the left of a `shapeY` of color `colorB`.
11	All `shapeX`s are to the left of all `shapeY`s.
12	Every `shapeX` has a `shapeY` to its right.
13	Every `shapeX` has a `shapeY` behind it.
14	There are three `shapeX`s of the same color.
15	There is a `materialQ` `shapeX` to the left of all `shapeY`s.

Table 2: Concept Class Schema for CLEVR VDP Dataset. `shapeX` and `shapeY` (disequal) range over sphere, cylinder, and cube. `colorA` and `colorB` (disequal) range over 8 possible colors. `materialQ` can either be rubber or metal.

ODDONE Puzzles dataset. We create a dataset of discrimination puzzles that are different from VDPs. An ODDONE puzzle consists of 4 images with the objective of identifying the image that is the *odd one out*. We formalize this task similar to VDPs by demanding a concept φ that satisfies *exactly* three images. We create these from the CLEVR VDP dataset by choosing three example images and one non-answer candidate image from each puzzle. Finally, we only include the 1872 puzzles that have a unique minimal discriminator in FO-SL.

Note that only the CLEVR VDP and ODDONE datasets have unique discriminators in conjunctive FO-SL.

5.2 Implementation

We use a pretrained model of YOLOv4 [Wang *et al.*, 2021] for the Natural Scenes dataset, which outputs object labels and bounding boxes. For the CLEVR domain, we use a pretrained model from the work in [Yi *et al.*, 2018] . The model produces a set of objects with their coordinates in 3D space and other attributes like shape, color, or material. We compute first-order scene models based on the outputs automatically.

The symbolic synthesizer implements an algorithm that searches for conjunctive discriminators in FO-SL. The algo-

[2]The datasets, code, and the website of VDPs can be found at: https://github.com/muraliadithya/vdp

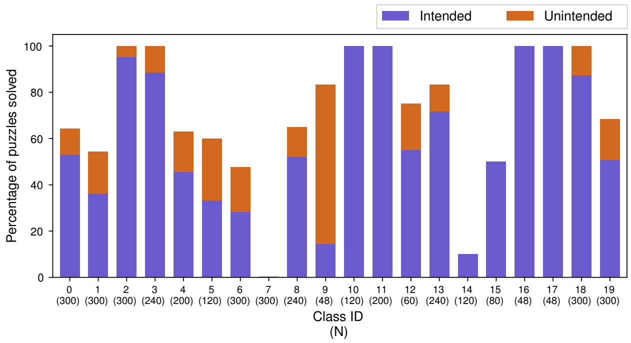

Figure 3: Evaluation on 3864 Natural Scenes Dataset puzzles: Class ID refers to the ID in Table 1, and (N) is the number of puzzles in each class. Intended: solver chose the intended discriminator. Unintended: solver did not choose the intended discriminator.

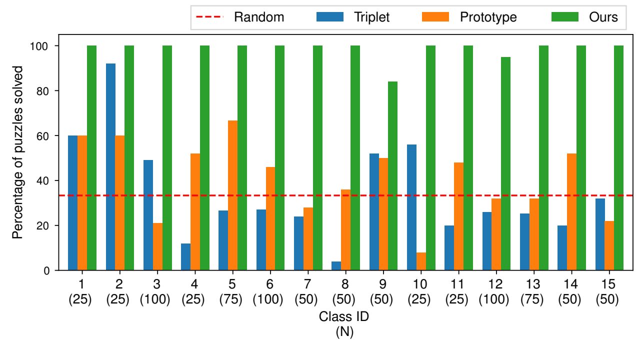

Figure 4: Comparison with neural baselines on 825 CLEVR VDP Dataset puzzles: Class ID refers to the ID in Table 2, and (N) is the number of puzzles in each class. **Triplet**: image similarity baseline. **Prototype**: prototypical network baseline. **Ours**: our solver. Dashed line represents accuracy of a random predictor.

rithm creates the constraints described in Section 4.3 and uses the SAT-solver Z3 [de Moura and Bjørner, 2008]. We also implement another synthesizer that searches for discriminators in full first-order logic to perform ablation studies. See the Appendix for more details about implementation.

5.3 Experiments

We report on the evaluation of our tool on the datasets and various ablation studies in terms of the research questions (RQs) below.

Experiments on Natural Scenes and GQA VDP Datasets
RQ1: Effectiveness

We evaluate our tool on the Natural Scenes dataset, applying the SAT-based symbolic synthesizer with a timeout. Since it is always possible to discriminate a finite set of (distinct) models with a complex (but perhaps unnatural) formula, we restrict our search to discriminators with complexity smaller than or equal to the target discriminator. We present the results in Figure 3. Our tool is effective and solves 68% of the 3864 puzzles, with an average accuracy of 71% per concept class.

Our tool finds varied discriminators with multiple quantifiers and complex nesting. For example, “Football in between two people” (kickoff position) is expressed by $\forall x. (\mathsf{person}(x) \Rightarrow \forall y. (\mathsf{sports\ ball}(y) \Rightarrow \exists z. (\mathsf{person}(z) \wedge \mathsf{left}(y, x) \wedge \mathsf{right}(y, z))))$, which has three quantifiers with alternation. The tool also generalizes intended discriminators.

Note that the Natural Scenes VDPs do not have unique solutions since smaller discriminators may exist. For example, in the concept class of “Laid out place settings”, a particular puzzle always displayed cups in example images. Our tool picked an unintended candidate satisfying “There is a cup” rather than the intended candidate satisfying a more complex discriminator (utensil arrangements). Our tool picks an unintended candidate in 17% of solutions. We provide more examples of discriminators found in the Appendix.

RQ2: Generality

How general is our framework? The GQA VDP dataset consists of automatically generated puzzles where neither the images nor the discriminators are curated. However, the pipeline fails if the vision model fails. In fact, most failures on the Natural Scenes dataset are of this kind. In our experience, although vision models are good at detecting objects, many report several bounding boxes for the same object and are not accurate. We experimented with SOTA scene graph generators (similar to the work in [Zellers *et al.*, 2018]) and found many issues.

To better study the generality of our formulation, we ablate the vision model errors and use the ground-truth scene graphs of images given in the GQA dataset to extract scene models. Our tool performs very well: it solves 81% of the puzzles (4% unintended). See the Appendix for examples of (un)solved puzzles.

RQ3: Failure Modes

The primary failure mode of the tool is failure of the vision model, as discussed above. This failure manifests as non-sensical discriminators. We leave the problem of designing frameworks that are robust despite vision model failures to future work.

We identify two other failure modes where the solver was not able to find any discriminator, namely (1) expressive power of the interface: we do not solve puzzles with discriminators like *There is a bag in the bottom portion of the image*, since information about the region of the image in which an object was present is not typically expressed in scene graphs. Many of the unsolved puzzles in the GQA VDP dataset belong to this failure mode; and (2) expressive power of the FO-SL fragment: consider the concept class *TV is switched on* in the Natural Scenes dataset, expressed as $\exists x. \mathsf{tv}(x) \wedge \exists y. \mathsf{displayed\ on}(y, x)$. This requires unguarded quantification and cannot be expressed in FO-SL. Another example is *There is a dog that is not white*, which requires negation and is not expressible in the conjunctive fragment.

RQ4: Ablation of Synthesis Algorithm

Our SAT-based synthesis algorithm is quite effective and finds discriminators in a few seconds. However, there are other possible synthesis engines, in particular those developed in the program synthesis literature for SyGuS problems [Alur *et al.*, 2015] . The specification of a discriminator can be encoded as a SyGuS problem, and so we perform an ablation study using CVC4SY [Reynolds *et al.*, 2019] (winner of SyGuS competitions) to find discriminators. CVC4SY did not scale and took at least 10 minutes for even moderately difficult puzzles, often not terminating after 30 minutes.

RQ5: Ablation of FO-SL

Evaluation of RQ1 shows that FO-SL is a rich logic that expresses many interesting discriminators. However, would a more expressive logic find more natural or better discriminators? We perform an ablation study on Natural Scenes VDPs by implementing a second solver for symbolic synthesis. This solver searches for discriminators in full first-order logic rather than FO-SL: the quantifiers are not guarded and the matrix need not be conjunctive. Note that FOL does not satisfy the extension property.

This solver is slower and times out for 57% of the puzzles. Among solved puzzles, solutions are sometimes more general, e.g. *There is something that is within all sofas*, instead of *All dogs on sofas*. However, we almost always obtain unnatural discriminators, e.g. *There is no chair and there is some non-sofa*. We therefore conclude that the guarded quantification in FO-SL is important, and full first-order logic is not suitable for finding natural discriminators. See the Appendix for more examples of unnatural discriminators as well as a detailed description of the solver.

Experiments on CLEVR VDP Dataset

RQ4: Neural Baselines

The CLEVR VDP dataset consists of puzzles with unique discriminators by construction, and our tool performs (unsurprisingly) very well (99%). Since solutions are unique, we can ask if solving VDPs is learnable using neural models.

We first construct a baseline from an image similarity model trained using triplet loss [Wang *et al.*, 2014]. To solve a VDP, we choose the candidate that maximizes the product of similarity scores with all the example images. We present the class-wise performance of this model in Figure 4. It does not perform very well and is rarely better than chance (33%). This shows that VDP solutions are not aligned well with commonly learned image features. Therefore, we evaluate another baseline based on prototypical networks [Snell *et al.*, 2017] by training on VDPs. Such networks aim to learn embeddings such that the average embedding of a class of images acts as a prototype, which can then be used to compute similarity with respect to the class. We fine-tune a ResNet18 [He *et al.*, 2016] + MLP architecture pretrained on CIFAR10 using 6 concept classes, validated against a held-out set of classes. We then evaluate it on the unseen classes and other unseen puzzles. This model achieves a slightly better overall accuracy of 40% as shown in Figure 4, but is still not performant.

Experiments on ODDONE Puzzles Dataset

RQ7: Adaptive Mechanisms

We evaluate the adaptability of mechanisms (without retraining) when the higher-level puzzle description is changed. Our framework is evidently highly adaptable and we can solve ODDONE puzzles without retraining by simply changing the synthesis objective (the constraint).

We evaluate the adaptability of the baseline models learned for VDP on the new task by adapting the scoring function (see Appendix) and find that they perform very poorly. The similarity and prototypical network baselines perform at 13% and 23% respectively, compared to a random predictor at 25%. Therefore, we conclude that the neural representations learned using the two baselines for one task do not lend themselves well to newer high-level task specifications without retraining.

6 Conclusions

The results of this paper argue that building symbolic synthesis and reasoning over outputs of a neural network lead to robust interpretable reasoning for solving puzzles such as VDP. For a future direction, the following problem formulation for VDPs seems interesting: given a VQA engine, find a property expressed in natural language that acts as a discriminator, as interpreted by the VQA engine. Solving the above would result in natural language discriminators (which would be more general than FO-SL), but the problem of searching over such discriminators seems nontrivial. We would also like to adapt the neurosymbolic approach in this paper to real-world applications that require learning interpretable logical concepts from little data, including obtaining differential diagnoses from medical images and health records, and learning behavioral rules for robots from interaction with humans and the environment.

Acknowledgments

This work is supported in part by a research grant from Amazon and a Discovery Partner's Institute (DPI) science team seed grant.

References

[Alur *et al.*, 2015] Rajeev Alur, Rastislav Bodík, Eric Dallal, Dana Fisman, Pranav Garg, Garvit Juniwal, Hadas Kress-Gazit, P. Madhusudan, Milo M. K. Martin, Mukund Raghothaman, Shambwaditya Saha, Sanjit A. Seshia, Rishabh Singh, Armando Solar-Lezama, Emina Torlak, and Abhishek Udupa. Syntax-guided synthesis. *Dependable Software Systems Engineering*, 2015.

[Alur *et al.*, 2018] Rajeev Alur, Rishabh Singh, Dana Fisman, and Armando Solar-Lezama. Search-based program synthesis. *Commun. ACM*, 61(12):84–93, November 2018.

[Amizadeh *et al.*, 2020] Saeed Amizadeh, Hamid Palangi, Oleksandr Polozov, Yichen Huang, and Kazuhito Koishida. Neuro-symbolic visual reasoning: Disentangling ”visual” from ”reasoning”. *ICML*, 2020.

[Andreas *et al.*, 2016] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. *CVPR*, 2016.

[Andreas *et al.*, 2018] Jacob Andreas, Dan Klein, and Sergey Levine. Learning with latent language. *ACL*, 2018.

[Antol *et al.*, 2015] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence

Zitnick, and Devi Parikh. Vqa: Visual question answering. *ICCV*, 2015.

[Balog *et al.*, 2017] Matej Balog, Alexander L. Gaunt, Marc Brockschmidt, Sebastian Nowozin, and Daniel Tarlow. Deepcoder: Learning to write programs. *ICLR*, 2017.

[Bunel *et al.*, 2018] Rudy Bunel, Matthew J. Hausknecht, Jacob Devlin, Rishabh Singh, and Pushmeet Kohli. Leveraging grammar and reinforcement learning for neural program synthesis. *ICLR*, 2018.

[de Moura and Bjørner, 2008] Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient smt solver. *TACAS*, 2008.

[Devlin *et al.*, 2017] Jacob Devlin, Jonathan Uesato, Surya Bhupatiraju, Rishabh Singh, Abdel-rahman Mohamed, and Pushmeet Kohli. Robustfill: Neural program learning under noisy i/o. *ICML*, 2017.

[Ellis *et al.*, 2018] Kevin Ellis, Daniel Ritchie, Armando Solar-Lezama, and Joshua B. Tenenbaum. Learning to infer graphics programs from hand-drawn images. *NIPS*, 2018.

[Enderton, 2011] Herbert B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, 2011.

[Evans and Grefenstette, 2018] Richard Evans and Edward Grefenstette. Learning explanatory rules from noisy data. *J. Artif. Int. Res.*, 61(1):1–64, Jan 2018.

[Goodfellow *et al.*, 2016] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[Gulwani *et al.*, 2015] Sumit Gulwani, José Hernández-Orallo, Emanuel Kitzelmann, Stephen H. Muggleton, Ute Schmid, and Benjamin Zorn. Inductive programming meets the real world. *Commun. ACM*, 58(11):90–99, October 2015.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CVPR*, 2016.

[Hudson and Manning, 2019] Drew A. Hudson and Christopher D. Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. *CVPR*, 2019.

[Johnson *et al.*, 2017] J. Johnson, B. Hariharan, Laurens van der Maaten, Li Fei-Fei, C. L. Zitnick, and Ross B. Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. *CVPR*, 2017.

[Koh *et al.*, 2020] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *ICML*, 2020.

[Lake *et al.*, 2015] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[Liu *et al.*, 2019] Yunchao Liu, Jiajun Wu, Zheng Wu, Daniel Ritchie, William T. Freeman, and Joshua B. Tenenbaum. Learning to describe scenes with programs. *ICLR*, 2019.

[Mao *et al.*, 2019] Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B. Tenenbaum, and Jiajun Wu. The Neuro-Symbolic Concept Learner: Interpreting Scenes, Words, and Sentences From Natural Supervision. *ICLR*, 2019.

[Murali *et al.*, 2018] Vijayaraghavan Murali, Letao Qi, Swarat Chaudhuri, and Chris Jermaine. Neural sketch learning for conditional program generation. *ICLR*, 2018.

[Nédellec, 1998] Claire Nédellec. Inductive logic programming, from machine learning to software engineering by f. bergadano and d. gunetti, the mit press, usa, 1996. *Knowl. Eng. Rev.*, 13(2):201–208, jul 1998.

[Parisotto *et al.*, 2017] Emilio Parisotto, Abdel-rahman Mohamed, Rishabh Singh, Lihong Li, Dengyong Zhou, and Pushmeet Kohli. Neuro-symbolic program synthesis. *ICLR*, 2017.

[Raychev *et al.*, 2019] Veselin Raychev, Martin T. Vechev, and Andreas Krause. Predicting program properties from 'big code'. *Commun. ACM*, 62(3):99–107, 2019.

[Reynolds *et al.*, 2019] Andrew Reynolds, Haniel Barbosa, Andres Nötzli, Clark Barrett, and Cesare Tinelli. cvc4sy: Smart and fast term enumeration for syntax-guided synthesis. *CAV*, 2019.

[Santoro *et al.*, 2018] Adam Santoro, Felix Hill, David G. T. Barrett, Ari S. Morcos, and Timothy P. Lillicrap. Measuring abstract reasoning in neural networks. *ICML*, 2018.

[Snell *et al.*, 2017] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *NIPS*, 2017.

[Solar-Lezama *et al.*, 2006] Armando Solar-Lezama, Liviu Tancau, Rastislav Bodik, Sanjit Seshia, and Vijay Saraswat. Combinatorial sketching for finite programs. *SIGOPS Oper. Syst. Rev.*, 2006.

[Sun *et al.*, 2018] Shao-Hua Sun, Hyeonwoo Noh, Sriram Somasundaram, and Joseph Lim. Neural program synthesis from diverse demonstration videos. *ICML*, 2018.

[Wang *et al.*, 2014] Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. Learning fine-grained image similarity with deep ranking. *CVPR*, 2014.

[Wang *et al.*, 2021] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Scaled-yolov4: Scaling cross stage partial network. *CVPR*, 2021.

[Yi *et al.*, 2018] Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Joshua B. Tenenbaum. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. *NeurIPS*, 2018.

[Zellers *et al.*, 2018] Rowan Zellers, Mark Yatskar, Sam Thomson, and Yejin Choi. Neural motifs: Scene graph parsing with global context. *CVPR*, 2018.

[Zhang *et al.*, 2019] Chi Zhang, Feng Gao, Baoxiong Jia, Yixin Zhu, and Song-Chun Zhu. Raven: A dataset for relational and analogical visual reasoning. *CVPR*, 2019.