Library learning for structured object concepts

Haoliang Wang [1] **Judith Fan** [1]

Abstract

The ability to represent semantic structure in the environment — objects, parts, and relations — is a core aspect of human visual perception and cognition. As a testament to this ability, humans are capable of expressing rich conceptual knowledge in simple iconic images. Here we leverage recent advances in program synthesis to develop a self-supervised and data-efficient algorithm for learning the the part-based structure of objects, as represented by graphics programs. Our algorithm iteratively learns higher-order subroutines that can be used to more compactly represent these objects, exploiting commonalities between learned subroutines to infer a part-based grammar. Our experiments explore how the resulting library of learned subroutines is jointly determined by the data distribution and the cost of learning additional subroutines.

1. Introduction

The ability to represent semantic structure in the environment — objects, parts, and relations — is a core aspect of human visual perception and cognition. As a testament to this ability, humans effortlessly grasp the correspondence between a real human face and a line drawing of a face, even without auxiliary cues such as color and texture (Fig. 1). How are visual concepts organized such that they robustly encode such abstract correspondences? Here we explore the notion that this robustness reflects the inherently generative and compositional organization of human conceptual knowledge (Palmer, 1977; Tenenbaum et al., 2011). In this paper we present: (1) a proof-of-concept method for learning higher-order structural units within objects, represented as subroutines in a graphics library; and (2) experiments exploring how the resulting library of learned subroutines is jointly determined by the data distribution and the cost of learning.

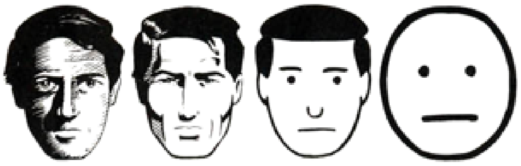


Figure 1. Human faces are configured in consistent ways across varying degrees of visual abstraction. (McCloud, 1994)

2. Related Work

2.1. Perceptual Organization

There is a long tradition within cognitive science of seeking to characterize the perceptual units by which humans parse the visual world (Palmer, 1977; Goldstone, 2003). Proposed solutions have ranged between manually specified volumetric primitives (Biederman & Ju, 1988) and features discovered via dimensionality reduction techniques (Lee & Seung, 1999), learned by neural networks (Yamins et al., 2014), or recovered by probabilistic inference (Austerweil & Griffiths, 2013). While many approaches have tended to focus on learning image-like internal representations, here we aim to learn a procedural representation that inherently captures the compositionality of visual concepts (Lake et al., 2017), inspired by "vision-as-inverse-graphics" (Kulkarni et al., 2015).

2.2. Program Synthesis

Specifically, we use techniques from program synthesis to develop a self-supervised and data-efficient algorithm that learns the part-based structure of objects, represented by graphics programs. Prior work using program synthesis to model graphical data has typically used a fixed Domain-Specific Language (DSL) to fit target objects, defined by a set of carefully designed functional primitives (Lake et al., 2015; Ellis et al., 2018b). Here we leverage a recently developed program learning method (Ellis et al., 2018a) to iteratively augment the DSL with higher-order subroutines that exploit statistical regularities within objects, as well as commonalities between these learned subroutines to infer a part-based grammar for objects.

*Equal contribution [1]Department of Psychology, University of California San Diego. Correspondence to: Haoliang Wang <haw027@ucsd.edu>, Judith Fan <jefan@ucsd.edu>.

Workshop on Object-Oriented Learning at ICML 2020. Copyright 2020 by the author(s).

3. Algorithm

Our algorithm proceeds through three main stages: program synthesis, subroutine learning, and grammar induction. In the first stage, our algorithm first searches for compact programs that hit the target specification guided by the base DSL (Table 1), similar to (Ellis et al., 2018b). In each iteration of the second stage, our algorithm proceeds through three steps: (1) proposing "*fragments*" from the induced program that may be good candidate subroutines to learn; (2) scoring these fragments according to how well they compress the input programs and their size; and (3) re-writing all programs using the highest scoring fragment (i.e., subroutine) and adding it to the library. In the third stage, the algorithm compresses this augmented DSL into a more compact part-based grammar (Figure 2).

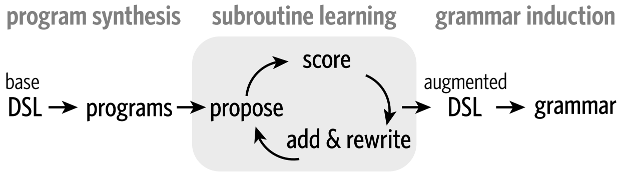


Figure 2. Schematic overview of algorithm.

3.1. Graphics Program Synthesis

As a proof-of-concept, we apply our algorithm to sketches of "smiley faces." Each smiley face is represented by a *specification* composed from a small set of graphical symbols, such that every stroke is depicted using either a `circle` or a `line`. Each smiley face consists of 9 possible features : head (`circle`), two types of mouth (`circle` or `line`), two types of left eye (`circle` or `line`), two types of right eye (`circle` or `line`), left glasses lens (`circle`) and right glasses lens (`circle`).

Given that each graphical element can be represented by a `circle` or `line`, we are able to begin with a very simple base DSL (Table 1). We use the Sketch program synthesizer (Solar Lezama, 2008) — no pun intended — to synthesize compact graphics programs that hit each input specification (Table 2). Given the base DSL and a specification S, we use Bayesian inference to obtain the most likely program p that generated each specification S, given a DSL of primitives $\mathscr{D}$. Our algorithm recovers p maximizing:

$$P(p|S,\mathscr{D}) \propto P(S|p)P(p|\mathscr{D})$$

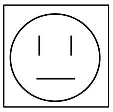
where $P(S|p)$ represents the likelihood of a specification given a program: here we adopt the all-or-nothing likelihood $\mathbb{1}[p \text{ consistent w/ } S]$. $P(p|\mathscr{D})$ represents the prior probability of the program: here we penalize long programs and set the prior of $p \propto \exp(-\text{length}(p))$.

Table 1. Base Domain-Specific Language (DSL).

`circle`	Unit circle at $(0,0)$ with radius 1
`line`	Unit line from $(0,-0.5)$ to $(0,0.5)$
`move`(x,y)	move to (x,y)
`scale`(r)	scale the size by ratio r
`rotate`(a)	rotate clockwise by angle a
`for`$(i,body)$	repeat *body* for i times

Table 2. Example smiley face (left), specification (middle), and program synthesized from that specification (right).

Drawing	**Specification**	**Induced Program**

	`Circle(4,3,9)`
	`Line(3,2, 3,3)`
	`Line(5,2, 5,3)`
	`Line(3,4, 5,4)`
		`((move 4 3 (scale 9`
		`(rotate 0 circle)))`
		`(move 4 4 (scale 2`
		`(rotate 2 line)))`
		`(move 3 2 (scale 1`
		`(rotate 0 (for 2`
		`line))))`

3.2. Library Learning

The programs induced above consist of sequences of drawing commands. However, because the order in which drawings commands are executed is unimportant for learning the part-based structure of smiley faces, we instead represent the programs as a `set` of λ-calculus expressions, where variables are prefixed with `$`, and we adopt De Bruijn indices to model bound variables. For example, this is the λ-calculus expression for the face in Table 2:

```
set((λ(move 4 3 (scale 9 (rotate 0 circle)) $0)),
    (λ(move 4 4 (scale 2 (rotate 2 line)) $0)),
    (λ(move 3 2 (scale 1 (rotate 0 (for 2 line))) $0)))
```

We propose fragments from the set of λ-calculus expressions to model the reuse of structure. We enumerate subsets of the set to get all possible combinations of features, yielding a very large number of fragment candidates.

To determine which ones to add to our base DSL, and the order in which they should be added, we developed a scoring function to rank these fragment candidates. This scoring function takes two pieces of information into account: minimum description length (MDL) and the size of the DSL ($\mathscr{L}$). The MDL term reflects the minimum number of lines of code we need to use to represent the original input programs using the candidate fragment and subroutines in the current DSL, which naturally favors learning large fragments. The DSL size term reflects the total number of lines of code that would be added to the base DSL, and its contribution is

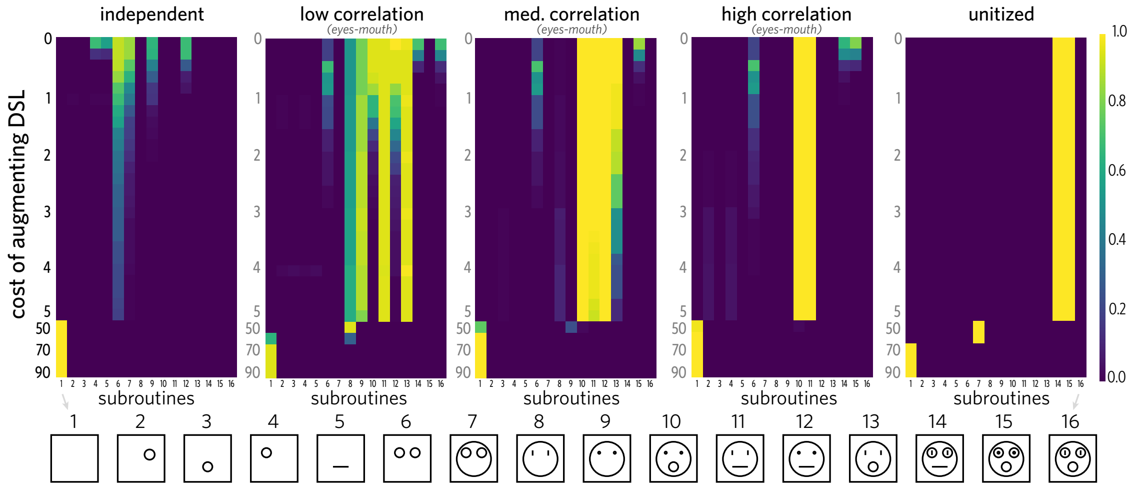


Figure 3. Each heat map corresponds to a different data distribution. Each row indicates a different value of w, the cost associated with learning a new subroutine. Each column corresponds to one of sixteen unique subroutines, shown at the bottom. Each cell represents the proportion of samples in which a given subroutine was learned.

controlled by a weight parameter, w, reflecting the cost of learning.

$$Loss(\mathscr{D}) = \sum_{p} MDL(p;\mathscr{D}) + w \times size(\mathscr{D})$$

The *MDL* of a program conditioned on the DSL is defined as:

$$MDL(p;\mathscr{D}) = \sum_{f\in\mathscr{D}} \mathbb{1}[\text{match}(p,f) \neq \perp]$$

where $\mathbb{1}[\text{match}(p,f) \neq \perp]$ indicates whether fragment f is a subset of program p.

On each iteration, the highest-scoring fragment is thus the one providing maximal compression of input programs while minimizing the expansion of the DSL. After the fragment with the highest score is selected and added into the base DSL, we rewrite the original programs in terms of this fragment. For example, if `Circle(4,3,9)` and `Line(3,2,3,3)` are combined into one subroutine, then the program shown above is re-written as below, where the red lines below represent this new subroutine:

```
set(set((λ(move 4 3 (scale 9 (rotate 0 circle)) $0)),
        (λ(move 4 4 (scale 2 (rotate 2 line)) $0))),
    (λ(move 3 2 (scale 1 (rotate 0 (for 2 line))) $0)))
```

Because we do not allow fragments contained within already learned subroutines to be proposed, this enforces the agent to learn higher-order abstractions over previously learned subroutines. The agent iterates through these three steps, and stops learning new subroutines when the *Loss* over the DSL does not decrease any further.

3.3. Grammar Induction by Library Compression

Once all new subroutines are added to the library, we conduct a depth-first tree search on each of the learned λ-calculus expressions and match their prefix names and parameters (`move(x,y)`, `scale(r)`, `rotate(a)`) to distill out the part-based *And-Or* grammar for smiley faces. Subroutines with the same prefix are merged into one *Or* node, subroutines with unique prefix are left as *And* nodes. Fuzzy matching of the prefix parameters is allowed.

4. Experiments

4.1. Dataset

The goal of our experiments was to explore how different constraints on learning, supplied either by external variables (i.e., the data distribution) or internal components of the model (i.e., the cost of increasing the library size), jointly influenced the subroutines our algorithm learned. Building on classic work investigating perceptual learning in cognitive science (Goldstone, 2003; Austerweil & Griffiths, 2013), we considered five data distributions, which varied in how strongly particular facial features co-occurred. For our experiments, we generated 100 samples containing 100 smiley faces from each of the following data distributions:

- **Independent**: All features are sampled independently with p=0.5, yielding faces that are often incoherent (e.g., only possessing a single eye or no mouth).
- **Low Correlation**: All faces are coherent (i.e., containing at least a head, two eyes, and a mouth), and the probability that the type of eyes and mouth will be

congruent (i.e., line eyes with line mouth) is equal to 0.4. Glasses appear with p=0.03.

- **Med Correlation**: Same as above, but the probability of congruent eyes & mouth is equal to 0.8.
- **High Correlation**: Same as above, but the probability of congruent eyes & mouth is equal to 1.0.
- **Unitized**: All faces are of two types, both appearing with $p = 0.5$: one with eyes and mouth consisting of lines and one with eyes and mouth consisting of circles, both always wearing glasses.

4.2. Manipulating the cost of learning

In our experiments, we explored the consequences of varying w, the *cost of learning*, over a wide range , $0 \leq w \leq 100$. We found that when w is small, complex configurations of facial features tend to be learned in their entirety (Fig. 3). As a consequence, the learned subroutines tend to be idiosyncratic to a small number of faces, and thus not reusable across contexts. As w increases, learning complex subroutines is disfavored, forcing the agent to discover regularities in the data distribution and learn higher-order abstractions that are reusable across a larger proportion of the dataset. However, when w grows too large, learning anything new becomes infeasible (see bottom left of all heatmaps in Fig. 3).

4.3. Manipulating the data distribution

Insofar as our algorithm can exploit statistical regularities in the co-occurrence of parts across different smiley faces, we hypothesized that it would be better able to learn part-like perceptual units (e.g., chunking both lenses comprising glasses into a single part) and meaningful correlations between different parts (i.e., that line-eyes and line-mouths tend to appear within the same face) when the data distribution supported these inferences. Over a wide range of w, we found that in the fully unitized distribution, our algorithm is more likely to learn complete faces (subroutines 14 & 15 in Fig. 3). When there is high correlation between types of eyes and mouth, two types of "base" faces are learned (subroutines 10 and 11). As the correlation decreases, more subroutines are learned (subroutine 8, 9, 12 and 13). Finally, in the fully independent distribution, the algorithm fails to consistently learn any particular subroutine, as expected (more columns omitted from Fig. 3 for clarity).

4.4. Sequence of subroutine learning

In the learning process, subroutines are learned across successive iterations Fig 4A). As new subroutines are added to the base library, the size of the library continues to increase. Since input programs can be written in increasingly reusable subroutines, the description length continues to decrease.

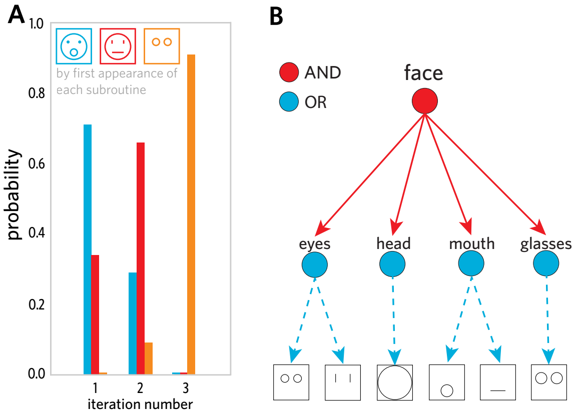


Figure 4. (A) Probability of first appearance of each subroutine across iterations, for the high-correlation data distribution and where w=0.9. (B) Grammar induced by comparing and getting unique primitive prefixes.

Across the first three iterations, the decrease of MDL dominates the overall loss term; subsequently, the change in MDL is negligible and cannot offset the cost of increasing the size of the DSL, thus learning stops. We observed that larger subroutines tended to be learned early on, and smaller ones learned later (Fig 4A), reflecting the greater ability of larger subroutines to compress the input programs, even after accounting for their size.

4.5. Learning a grammar from a library of learned subroutines

Using the high-correlation data distribution with w=0.9 as a case study, we also explored the induction of a part-based *AND-OR* grammar. Our approach was to exploit recurring patterns across the 'prefixes' to our learned subroutines. For example, subroutine `(λ(move 3 2 (scale 1 (rotate 0 (for 2 line))) $0))` and `(λ(move 3 2 (scale 1 (rotate 0 (for 2 circle))) $0))` share the same prefix and are merged into an *OR* node representing the *eyes*; likewise, `(λ(move 4 4 (scale 2 (rotate 2 line)) $0))` and `(λ(move 4 4 (scale 2 (rotate 2 circle)) $0))` are merged into the *OR* node representing the *mouth*, yielding the grammar shown in Fig 4B.

5. Discussion

We presented a proof-of-concept method for learning higher-order structural units within objects, represented as subroutines in a graphics library, and investigated how the resulting library of learned subroutines is jointly determined by the data distribution and the cost of learning. Future work will extend our proof-of-concept framework to model human perceptual learning and semantic structure in human sketches.

6. Acknowledgments

We thank Nadia Polikarpova, Shraddha Barke and Rose Kunkel for helpful discussion and feedback.

References

Austerweil, J. L. and Griffiths, T. L. A nonparametric bayesian framework for constructing flexible feature representations. *Psychological review*, 120(4):817, 2013.

Biederman, I. and Ju, G. Surface versus edge-based determinants of visual recognition. *Cognitive Psychology*, 20 (1):38–64, 1988.

Ellis, K., Morales, L., Sablé-Meyer, M., Solar-Lezama, A., and Tenenbaum, J. Library learning for neurally-guided bayesian program induction. In *NeurIPS*, 2018a.

Ellis, K., Ritchie, D., Solar-Lezama, A., and Tenenbaum, J. B. Learning to infer graphics programs from hand-drawn images. *NIPS*, 2018b.

Goldstone, R. L. Learning to perceive while perceiving to learn. In *Perceptual organization in vision*, pp. 245–290. Psychology Press, 2003.

Kulkarni, T. D., Kohli, P., Tenenbaum, J. B., and Mansinghka, V. Picture: A probabilistic programming language for scene perception. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4390–4399, 2015.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40, 2017.

Lee, D. D. and Seung, H. S. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755): 788–791, 1999.

McCloud, S. *Understanding Comics*. HarperPerennial, 1994. ISBN 9780613027823. URL https://books.google.com/books?id=oJ1vPwAACAAJ.

Palmer, S. E. Hierarchical structure in perceptual representation. *Cognitive psychology*, 9(4):441–474, 1977.

Solar Lezama, A. *Program Synthesis By Sketching*. PhD thesis, EECS Department, University of California, Berkeley, Dec 2008. URL http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-177.html.

Tenenbaum, J. B., Kemp, C., Griffiths, T. L., and Goodman, N. D. How to grow a mind: Statistics, structure, and abstraction. *science*, 331(6022):1279–1285, 2011.

Yamins, D. L., Hong, H., Cadieu, C. F., Solomon, E. A., Seibert, D., and DiCarlo, J. J. Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the National Academy of Sciences*, 111(23):8619–8624, 2014.