Interactive learning of cognitive programs

Sunayana Rane [1 2] **Miguel Lázaro-Gredilla** [1] **Dileep George** [1]

Abstract

Humans can identify and replicate high-level visual concepts easily from a few pairs of images. They can even learn from a single example, especially if they are able to ask a few clarifying questions. However, this ability has proven difficult to emulate in artificial systems. In this work, we design a system that infers a visual concept after being shown only one example of that concept. In order to achieve this, we equip the algorithm with an ability humans use to solve similar problems: interaction. After generating an initial guess, the algorithm also generates a small number of binary visual follow up questions. It then uses the answers to guide itself towards the correct concept. Concepts are represented as cognitive programs, and program induction generates candidate programs using tree search with probabilistic heuristics. Our Interactive Search Algorithm intelligently narrows the search space of candidate programs. The result is a system which, given only a single example of the concept and a few clarifying questions, successfully discovers 441 out of the 546 tested visual concepts.

1. Introduction

Humans, unlike artificial systems, can identify and replicate a high-level visual concept from a pair of images easily. Recent work (Lázaro-Gredilla et al., 2019) made promising progress towards learning simple visual concepts from a small number of input/output image pairs. For example, the concept "move bottom object left and change color" is learned from the images in Figure 1 below. However, the learning problem in (Lázaro-Gredilla et al., 2019) is posed in a way that does not resemble human learning from a teacher. Independent examples are given so that the concept that is common to all of them is discovered. A more human-like approach to convey a concept would be to present a single example, and then let the human ask follow-up questions. In this work, we design a system that infers a concept from only one example, as humans would. In order to achieve this, we equip the algorithm with the ability interact with the teacher.

Humans learn by interactively asking questions to clarify and solidify their internal representation of a concept. We present an interactive search algorithm that allows an artificial system to learn in the same way.

In our algorithm, the Simulated User thinks of a concept and produces a single pair of images to demonstrate it, as in Figure 1. The Learner then runs a preliminary search and produces a guess of the concept. It then produces another input/output pair demonstrating the concept it has guessed, and asks the Simulated User, "Is this what you meant?". Based on the Simulated User's response, it adjusts its search to most effectively narrow down and pinpoint the concept the Simulated User had in mind.

We demonstrate that this Interactive Search Algorithm works for a majority of the concepts covered in (Lázaro-Gredilla et al., 2019).

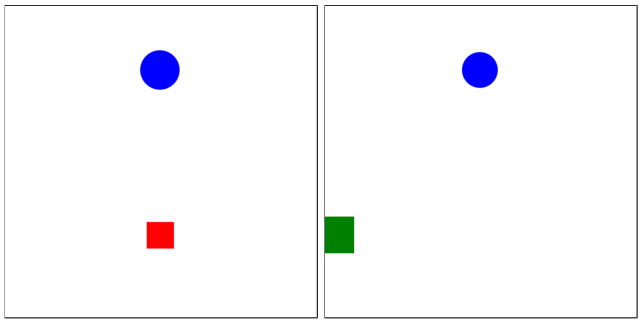

Figure 1. Input (left) and output (right) of the concept "move bottom object left and change color"

2. Background: Cognitive Programs

To ground this interactive learning and provide a basis for comparison with a non-interactive benchmark, we use an existing formulation for visual concept learning called the

[1]Vicarious, Inc [2]Work completed during a research internship at Vicarious. Correspondence to: Sunayana Rane <sunayana@alum.mit.edu>.

Presented at the Human-in-the-loop Learning Workshop at the 35^{th} International Conference on Machine Learning, Formerly Vienna, Austria.

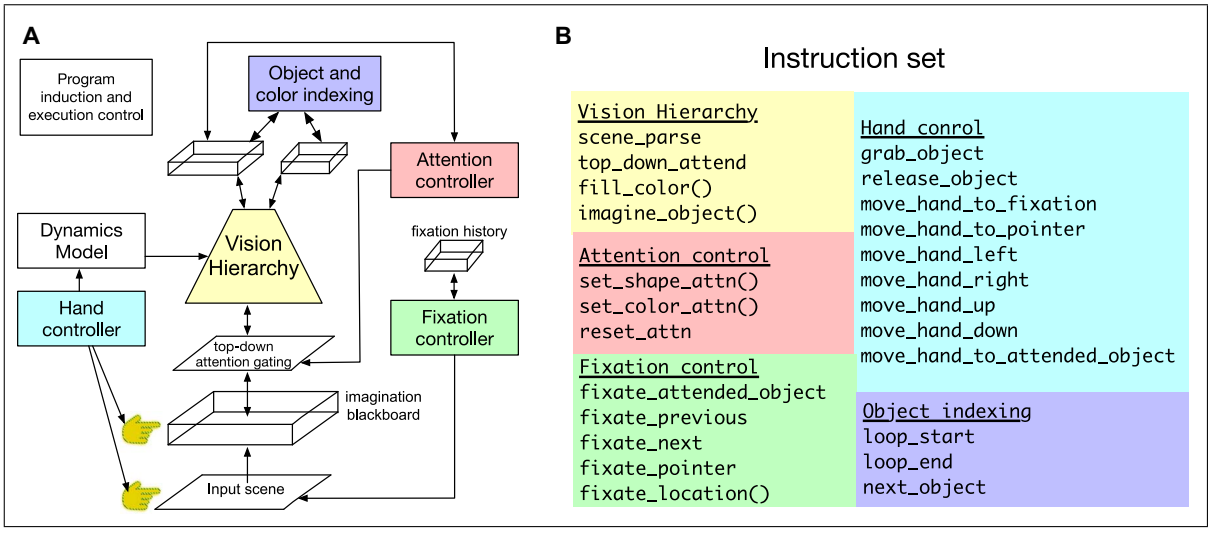

Figure 2. VCC architecture and program execution examples. Adapted with permission from (Lázaro-Gredilla et al., 2019).

Visual Cognitive Computer (VCC) (Lázaro-Gredilla et al., 2019). In the original work, this architecture was tested on robots to learn simple visual concepts from 10 input/output image pairs. The VCC consists of several important pieces that come together from cognitive science principles, including image schemas (Mandler & Cánovas, 2014), deictic mechanisms (Ballard et al., 1997), perceptual symbol systems (Barsalou, 1999), visual routines (Ullman), and mental imagery (Roelfsema & de Lange, 2016). These principles inspired the VCC's modules, including a visual hierarchy, attention mechanism, an imagination blackboard and a controller. These are designed to simplify the task of program induction (concept learning) from examples.

VCC contains an instruction set, as shown in Figure 2. In this framework a concept is represented as a series of instructions from that set, also known as a cognitive program. These instructions encode a concept that can be run on any input image, producing a resulting output image in which the concept has been applied. For instance, a very simple concept could be "Move the green object to the right until it touches another object".

When framed as such, concept learning is represented by program induction (Overlan et al., 2017; Gulwani et al., 2015). Program induction, in turn, becomes a search for a sequence of instructions that encapsulates and can reproduce the concept on a variety of inputs. Thus program induction becomes a tree-search problem.

The original VCC work applied a naive version of the explore-compress framework (Dechter et al., 2013), making it slow when trying to discover more sophisticated concepts (which translated into longer programs and therefore, deeper search). Later work (Sawyer et al., 2020) showed that a heuristic search that factorizes the different objects that appear in a scene provided speedups of several orders of magnitude. These heuristics make the search tractable for concepts that were not possible before. The explore-compress framework alternates between building a model based on the programs that it has discovered so far and searching for new programs following the learned model (in order of descending probablity according to the model). The VCC begins with a warm-up run to train the model on a bootstrap set. These heuristics are also incorporated into our own Interactive Search Algorithm. They are explained in detail in (Lázaro-Gredilla et al., 2019), and include:

1. A Markov model that prioritizes search by learned instruction transition probabilities, so that likely candidates are explored first.

2. Argument prediction using a convolutional neural network for guidance.

3. Identification of subroutines.

4. Pruning of invalid programs (e.g. those that dictate a physically impossible or invalid transition).

In addition to these heuristics, we also use object factorization and interaction to make the problem tractable with only one example input/output pair. We run our interactive learning search algorithm on the VCC framework to determine whether the visual concepts of the VCC framework can be learned with only one input example.

3. Object Factorization

In order to eventually make the program induction process truly interactive with a human user, induction must run fast enough to allow interaction in real-time. We use a Simulated User in place of a human in our experiments, which incorporates this time constraint. Since the original VCC-style program induction process is very slow, we incorporate object factorization, as described in (Sawyer et al., 2020).

Object factorization works by splitting the search process into subgoals; instead of trying to induce the entire program from scratch, we instead try to transform one object from its input state to its desired output state. Usually this transformation involves either a change in spatial coordinates, color, or both.

Object-factorized search begins with Dijkstra's algorithm, where the "distance" to a given program is the description length of the program, as calculated using the learned Markov transition matrix (Lázaro-Gredilla et al., 2019). Therefore, the candidates with likely instruction transitions are explored first. Search is run until a program emerges which successfully solves the concept for one object. Then, a new search is begun, with this successful program as the root node for search. The fact that a program matches the transition experienced by one object in every example pair demonstrates that it represents a valid sub-concept. These sub-concepts combine into increasingly good solutions as search continues, and finally into one cohesive concept represented by one cognitive program.

Object factorization also introduces some constraints, and it is most successful when combined with program mutation – when taking a simple one-object-at-a-time approach, some additional injected variation proves to be helpful (Sawyer et al., 2020). Program mutation works by changing (transposing, inserting, or deleting) single instructions in the candidate program. Any valid mutants that find at least as many objects as the candidate program are also taken to be candidates. Program mutation is inspired by directed evolution processes in protein engineering (Arnold, 1998). The variant of program mutation we use is described in detail in (Sawyer et al., 2020). Our Interactive Search Algorithm is built on top of this strategic search process.

4. Interactive Search Algorithm

We have created an interactive search algorithm that allows our system to infer the concept that a user has demonstrated in a single pair of images, and replicate it successfully. At the highest level, we are performing interactive search by using the user's feedback to continually narrow down our search space.

As such, the Learner chooses to ask questions that most effectively eliminate large sections of the search space, so that it needs the fewest number of questions to correctly infer the concept. This interactive search algorithm contains the following steps:

1. Given a single input and output example that the user has provided, the Learner uses program induction with object factorization to find a cognitive program that exhibits the correct behavior to generate that output from that input. This program is probably representative of that pair of images, but may not be a general solution that captures the "concept".

2. In order to search for a general solution, the Learner generates mutant programs. Mutant programs are slightly mutated versions of the original program, that are likely candidates for solving the general concept. We generate mutant programs by the protocol described in (Sawyer et al., 2020).

3. The Learner generates mutant inputs: to test how well each candidate program captures the concept, the Learner first runs it on a variety of inputs to generate sample outputs. To provide a variety of inputs, The Learner generates "mutant inputs", by incorporating variants of the original input from the Simulated User's given input/output pair. The Learner mutates the input by changing a single feature in the input image (either the color or the spatial coordinates of one or more objects).

4. Run candidate programs on mutant inputs, and isolate the input that best splits the programs as shown in Figure 3. In the Learner's interaction with the Simulated User, it uses the mutant input and generated output to ask the Simulated User, "Is this the concept you intended?". The answer will significantly reduce our search space.

5. Using the mutant input that best splits the programs, the outputs of the previous step creates branches, each with multiple programs that create the same output as shown in Figure 3. Here we define the "best" or "most even" split as the mutant input which creates the lowest standard deviation between branches (in other words, has a fairly equal number of programs in each branch). The Learner chooses the best mutant input using this criteria, then chooses a branch of this tree, and a program in that branch. It uses the mutant input and generated output from that program to interactively test whether that program generates the desired output on that mutant input (by asking the user, or in this case, the Simulated User):

   (a) If the user responds "yes, the generated images show the correct output on the mutant input", then

redefine your search space as ALL programs in this branch. Re-run the search recursively, from step 4.

(b) If the user responds "no, the generated images show an incorrect output", eliminate this branch and continue by moving to the next branch, repeating step 5.

6. Terminate after a max depth of 30 interaction if a solution has not been found.

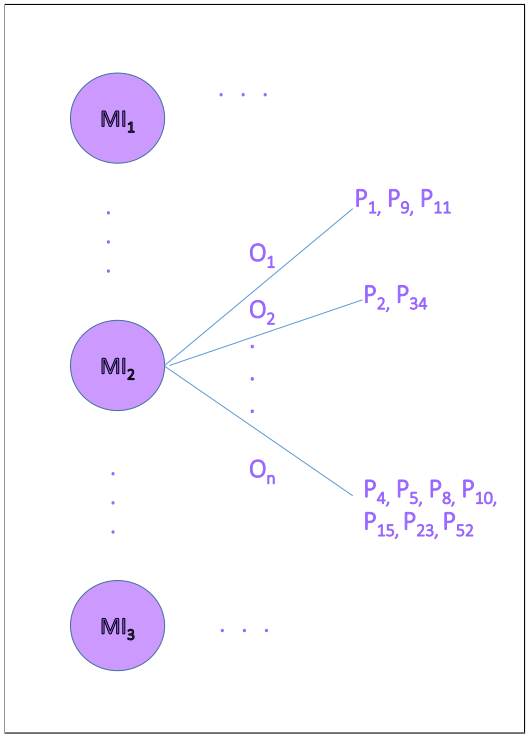

Figure 3. Data structure created by Interactive Search Algorithm to find the Mutant Input (MI) that most evenly splits the programs. Multiple programs often generate the same output for a given mutant input, and we can leverage this to intelligently devise a question which will eliminate an entire branch of candidate programs. MI is Mutant Input, O is Output, and P is Program.

4.1. Criteria for success

In interactive search, it is important to determine the criteria for what a "successful" search is. There are two potential ways to measure this:

1. End search when the user thinks that our generated input/output pairs capture the concept correctly
2. End search when there is only one possible branch left (a group of programs that produce the same outputs), and we have eliminated the remaining branches and programs as candidates. For this is happen, criteria 1 also has to be fulfilled.

Obviously criteria 1 is often much easier and faster to accomplish than criteria 2. However, we wanted to make sure we have generated a program that thoroughly captures the inherent human-generated concept, and does not simply fool the user into thinking it has done so. Therefore, in our evaluation, we have defined the second, more stringent criteria for success as a "sufficient solution", and present all results using that criteria. This is evident in step 5a of the protocol listed in the previous section – even if the user answers "yes" to one generated guess, we keep generating concepts and confirming with the user until we are sure. We are sure when there is only one branch left in the tree shown in Figure 3.

4.2. Sufficient and True Solutions

Once the Interactive Search Algorithm finds a successful program, we also test whether this generated program truly captures the concept originally chosen by the Simulated User.

A "sufficient" solution is one where the Simulated User affirms that the new pair of images represents the intended concept, *and* the solution belongs to the only branch logically remaining in the Interactive Search Tree. Thus even our lower bound for a solution, the "sufficient" solution, is very stringent. A "true" solution goes even further – a "true" solution is one that produces the same results as the original program, on all of several thousand diverse inputs.

We test whether true solutions are found by running both the program corresponding to the originally chosen concept and the newly generated program on a diverse set of several thousand mutant inputs and testing whether the resulting output images are identical.

If the original program and the generated program both represent the same concept, then both programs should produce identical output images.

However, getting absolutely identical results on thousands of mutant inputs is also a very tall order, as small differences do not necessarily mean the concept was not captured. Due to the nature of the instruction set which comprises each program, the different between sufficient and true solutions is often very slight. In fact, both sufficient and true solutions often satisfy the verbal description of the concept (see Qualitative Results for an example). Therefore, while we do measure the number of "true" solutions that are found, criteria 2 above is a more accurate and realistic metric for whether a solution is valid (especially in comparison to human concept learning).

Table 1. Interactive Search Algorithm solved 441 out of 546 original VCC concepts using only one input example

TOTAL CONCEPTS	546
SOLVED CONCEPTS	441
SUFFICIENT SOLUTIONS	369
TRUE SOLUTIONS	72

5. Results

Our Interactive Search Algorithm makes search tractable for a majority of the concepts covered in (Lázaro-Gredilla et al., 2019). In particular, it solved 441/546 (80.76 %) concepts successfully, with only one example given.

5.1. Quantitative Results

Of the 546 original concepts in VCC, Interactive Search using only a single demonstrated example of the concept yields 441 successfully solved concepts. Of these 441 successful solutions, 369 are sufficient solutions, and 72 are true solutions 1. Figure 4 and Figure 5 show that the Interactive Search Algorithm often yields at least a sufficient solution within a single interaction. True solutions, when found, are found within 10 interactions. All of these search processes use only a single example of the concept, as compared to 10 examples each in the original work (Lázaro-Gredilla et al., 2019). Interactive learning makes this previously intractable task possible.

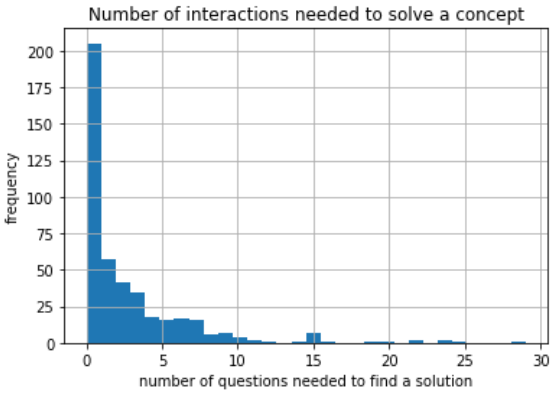

Figure 4. Most concepts are solved within a single interaction.

5.2. Qualitative Results

The Interactive Search Algorithm successfully solves the majority of concepts given only a single example. We also investigated the real difference between a "true" solution and a "sufficient" solution using qualitative observations. We notice two patterns:

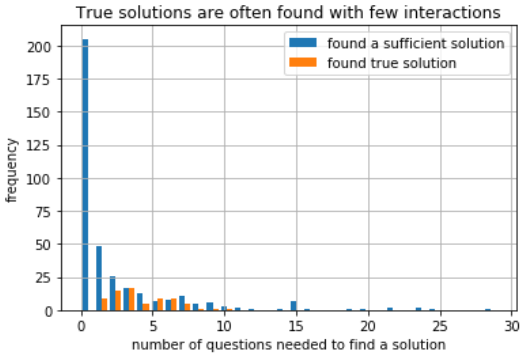

Figure 5. True solutions, when found, are found within 10 interactions.

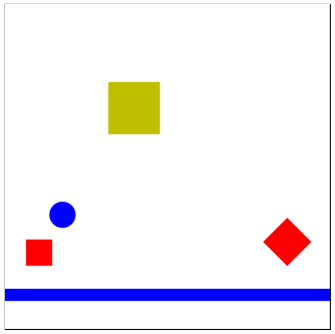

Figure 6. Confusing mutant input for concept "move red square object to left bottom corner" – there are two red square objects, and one is already in the left bottom corner

1. Some inputs were inevitably included that were confusing even for humans, due to our strategy of generating thousands of mutant inputs for thorough testing. If even one of these confusing inputs created different outputs, it would be labelled as a "sufficient" solution, which is a very stringent bar. For example, Figure 6 shows one mutant input for a concept called "move square red shape to left bottom corner". The "sufficient" solution program found for this concept was marked as such because it was not able to reproduce the concept on this mutant input. This is a reasonably confusing mutant input even for a human, because there are two red square objects, and one is already in the left bottom corner. It makes sense that this would cause slight differences between the outputs of the original and generated programs.

2. Some outputs, while not identical to the outputs of the original, are still technically valid representations of the given concept. For example, the Figure 7 shows

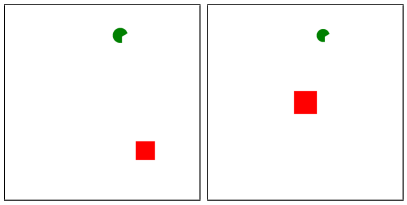

Figure 7. For the concept "move to middle", illustrated in the figure above, a generated solution may capture the concept without matching the output of the simulator exactly due to imperfections. In examples like these, the program is marked as unsolved even though the generated solution does solve the concept.

the concept "move to middle". The generated output, while not an exact match with the true output, does still solve the concept by moving the object to the middle area.

6. Conclusion

In this work, we develop an Interactive Search Algorithm for visual concept learning through program induction. The Interactive Search Algorithm is given only one input image and one output image representing the concept, in contrast with 10 image pairs given to the original, non-interactive algorithm in the VCC architecture (Lázaro-Gredilla et al., 2019). Our method successfully solves 441 out of 546 original VCC concepts using a single example, a previously intractable challenge in the original VCC experiments (Lázaro-Gredilla et al., 2019).

Our analysis of the difference between true and sufficient solutions also explores the ambiguity inherent in visual concept learning. There are often multiple programs that solve a given concept, even though their produced outputs might differ slightly. Furthermore, outputs that are very ambiguous or confusing even for humans are often confusing for the Interactive Search Algorithm too. Potential improvements include better heuristics for the search in program-induction, more diverse strategies for enhanced speed of search, and better criteria for "evenness of split" used to choose the best mutant input in each iteration. In the big picture, further comparisons between human concept learning and interactive program induction will be help us create better artificial systems for concept learning, and make progress towards the goal of human-level concept learning.

7. Acknowledgements

The authors would like to thank Danny Sawyer for his insightful comments and suggestions during this project.

References

Arnold, F. H. Design by directed evolution. *Accounts of Chemical Research*, 31(3):125–131, March 1998. doi: 10.1021/ar960017f. URL https://doi.org/10.1021/ar960017f.

Ballard, D. H., Hayhoe, M. M., Pook, P. K., and Rao, R. P. Deictic codes for the embodiment of cognition. *Behavioral and brain sciences*, 20(4):723–742, 1997.

Barsalou, L. W. Perceptual symbol systems. *Behavioral and brain sciences*, 22(4):577–660, 1999.

Dechter, E., Malmaud, J., Adams, R. P., and Tenenbaum, J. B. Bootstrap learning via modular concept discovery. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

Gulwani, S., Hernández-Orallo, J., Kitzelmann, E., Muggleton, S. H., Schmid, U., and Zorn, B. Inductive programming meets the real world. *Communications of the ACM*, 58(11):90–99, October 2015. doi: 10.1145/2736282. URL https://doi.org/10.1145/2736282.

Lázaro-Gredilla, M., Lin, D., Guntupalli, J. S., and George, D. Beyond imitation: Zero-shot task transfer on robots by learning concepts as cognitive programs. *Science Robotics*, 4(26):eaav3150, January 2019. doi: 10.1126/scirobotics.aav3150. URL https://doi.org/10.1126/scirobotics.aav3150.

Mandler, J. M. and Cánovas, C. P. On defining image schemas. *Language and cognition*, 6(4):510–532, 2014.

Overlan, M. C., Jacobs, R. A., and Piantadosi, S. T. Learning abstract visual concepts via probabilistic program induction in a language of thought. *Cognition*, 168:320–334, November 2017. doi: 10.1016/j.cognition.2017.07.005. URL https://doi.org/10.1016/j.cognition.2017.07.005.

Roelfsema, P. R. and de Lange, F. P. Early visual cortex as a multiscale cognitive blackboard. *Annual review of vision science*, 2:131–151, 2016.

Sawyer, D. P., Lázaro-Gredilla, M., and George, D. A model of fast concept inference with object-factorized cognitive programs, 2020.

Ullman, S. *Object Recognition and Visual Cognition*. MIT Press Cambridge MA.