Learning to Infer Graphics Programs from Hand-Drawn Images

Kevin Ellis[1], Daniel Ritchie[2], Armando Solar-Lezama[1], Joshua B. Tenenbaum[1]

[1]: MIT. [2]: Brown University.

Learning to Infer Graphics Programs from Hand-Drawn Images

Kevin Ellis[1], Daniel Ritchie[2], Armando Solar-Lezama[1], Joshua B. Tenenbaum[1]

[1]: MIT. [2]: Brown University.

Learning to Infer Graphics Programs from Hand-Drawn Images

Kevin Ellis[1], Daniel Ritchie[2], Armando Solar-Lezama[1], Joshua B. Tenenbaum[1]

[1]: MIT. [2]: Brown University.

Learning to Infer Graphics Programs from Hand-Drawn Images

Kevin Ellis[1], Daniel Ritchie[2], Armando Solar-Lezama[1], Joshua B. Tenenbaum[1]

[1]: MIT. [2]: Brown University.



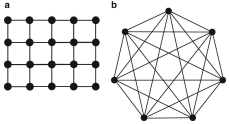
Learning to Infer Graphics Programs from Hand-Drawn Images

Kevin Ellis[1], Daniel Ritchie[2], Armando Solar-Lezama[1], Joshua B. Tenenbaum[1]

[1]: MIT. [2]: Brown University.







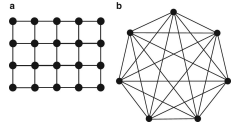
Learning to Infer Graphics Programs from Hand-Drawn Images

Kevin Ellis[1], Daniel Ritchie[2], Armando Solar-Lezama[1], Joshua B. Tenenbaum[1]

[1]: MIT. [2]: Brown University.



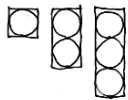




Hand Drawing **Program**

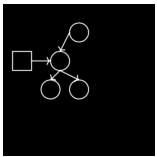
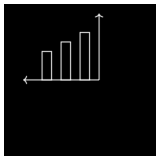
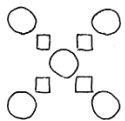
Drawing





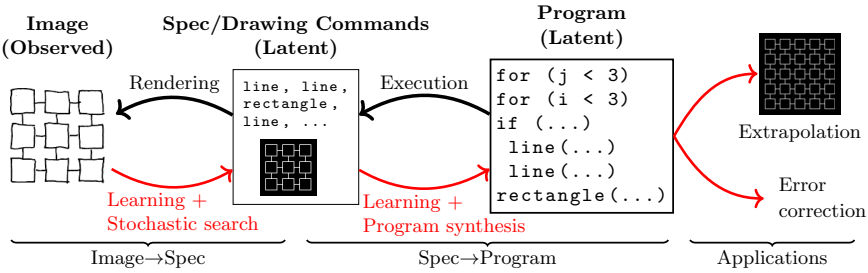
```
for (i < 3)
 rectangle(3*i,-2*i+4,
           3*i+2,6)
 for (j < i + 1)
  circle(3*i+1,-2*j+5)
```

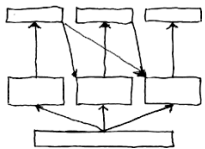
LaTeX

```
reflect(y=8)
 for(i<3)
  if(i>0)
   rectangle(3*i-1,2,3*i,3)
  circle(3*i+1,3*i+1)
```

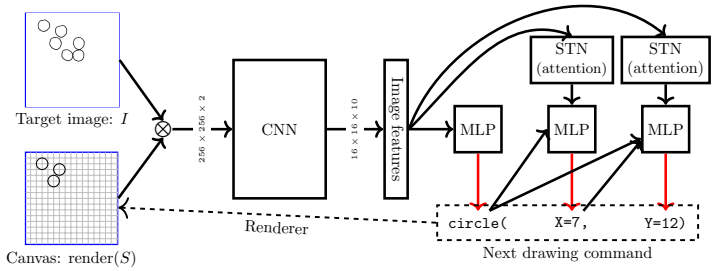
Image→Program Pipeline



Drawing	Spec	Program
	`Line(3,10,3,14,arrow)` `Rectangle(11,8,15,10)` `Rectangle(11,14,15,15)` `Line(13,10,13,14,arrow)` *... etc. ...; 16 lines*	`for(i<3)` `line(7,1,5*i+2,3,arrow)` `for(j<i+1)` `if(j>0)` `line(5*j-1,9,5*i,5,arrow)` `line(5*j+2,5,5*j+2,9,arrow)` `rectangle(5*i,3,5*i+4,5)` `rectangle(5*i,9,5*i+4,10)` `rectangle(2,0,12,1)`

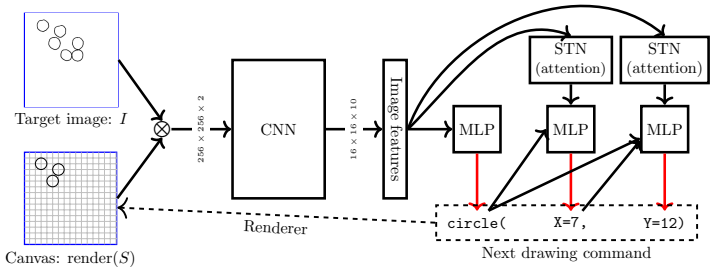
Parsing images into LaTeX TikZ Commands

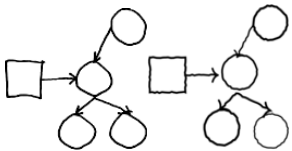
Neurally Guided Procedural Modeling (Ritchie et al 2016) + Attend, Infer, Repeat (Eslami et al 2016)



Parsing images into LaTeX TikZ Commands

Neurally Guided Procedural Modeling (Ritchie et al 2016) + Attend, Infer, Repeat (Eslami et al 2016)



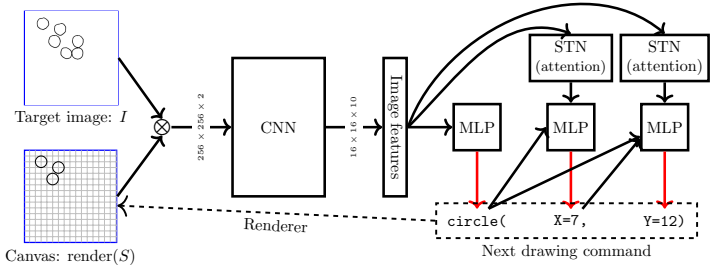


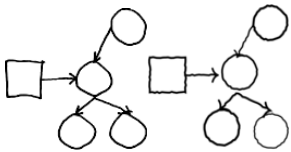
(a): hand drawing

(b): noisy render of (a)'s spec

Parsing images into LaTeX TikZ Commands

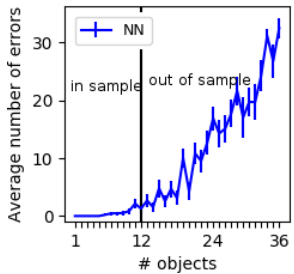
Neurally Guided Procedural Modeling (Ritchie et al 2016) + Attend, Infer, Repeat (Eslami et al 2016)





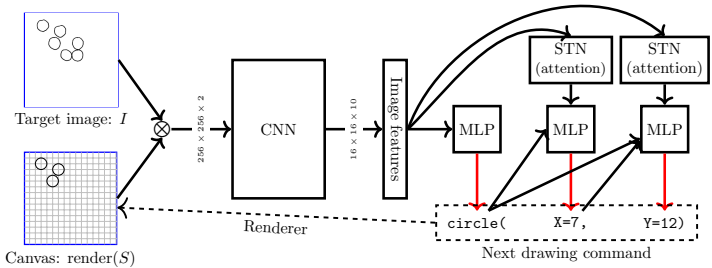
(a): hand drawing

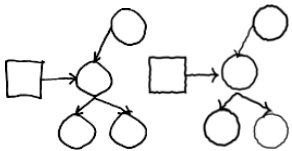
(b): noisy render of (a)'s spec



Parsing images into LaTeX TikZ Commands

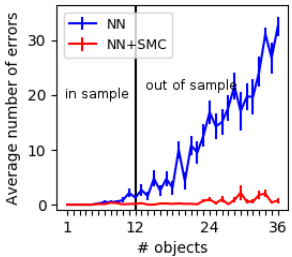
Neurally Guided Procedural Modeling (Ritchie et al 2016) + Attend, Infer, Repeat (Eslami et al 2016)





(a): hand drawing

(b): noisy render of (a)'s spec



Synthesizing high-level programs from specs (spec=drawing commands)

Constraint-based program synthesis; SAT solver (Solar-Lezama 2008)

$$\text{program}(S) = \underset{\substack{p \in \text{DSL} \\ p \text{ consistent w/ } S}}{\arg\min} \text{cost}(p)$$

min cost≈simple+short

DSL: **D**omain **S**pecific **L**anguage: variables, arithmetic, loops, conditionals

Program→	Statement; ⋯ ; Statement
Statement→	`circle`(Expression,Expression)
Statement→	`rectangle`(Expression,Expression,Expression,Expression)
Statement→	`line`(Expression,Expression,Expression,Expression,Boolean,Boolean)
Statement→	`for`(0 ≤ Var < Expression) { `if` (Var > 0) { Program }; Program }
Statement→	`reflect`(Axis) { Program }
Expression→	ℤ×Var+ℤ
Axis→	`X = ` ℤ |`Y = Z`
ℤ →	an integer

Learning to quickly synthesize programs

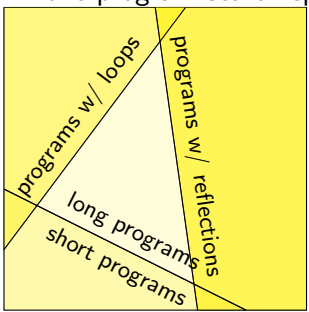
Learn search policy π(program subspace|spec)

Think of the subspace as an "ansatz"

OBJECTIVE (cf Bias-Optimal Search, Schmidhuber 2004):

$$\pi^* = \arg\min_{\pi} \sum_{\text{spec}} \min_{\substack{\text{all subspaces} \\ \text{subspace solves spec}}} \frac{\mathbb{E}[\text{time to exhaustively search the subspace}]}{\pi(\text{subspace}|\text{spec})}$$

Entire program search space



π(short, no loop/reflect|S) = ☐

π(long, loops|S) = ☐

π(long, no loop/reflect|S) = ☐

π(long, reflects|S) = ☐

etc.

Learning to quickly synthesize programs

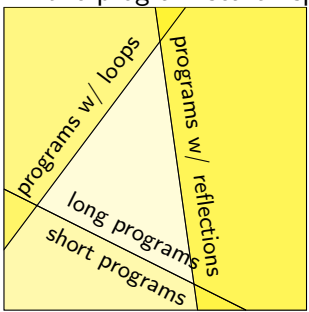
Learn search policy π(program subspace|spec)

Think of the subspace as an "ansatz"

OBJECTIVE (cf Bias-Optimal Search, Schmidhuber 2004):

$$\pi^* = \arg\min_{\pi} \sum_{\text{spec}} \min_{\substack{\text{all subspaces} \\ \text{subspace solves spec}}} \frac{\mathbb{E}[\text{time to exhaustively search the subspace}]}{\pi(\text{subspace}|\text{spec})}$$

Entire program search space



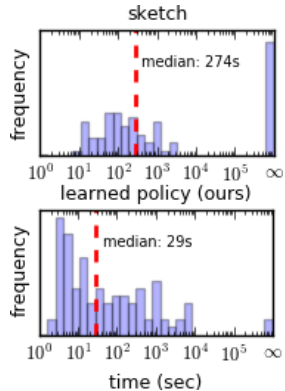
π(short, no loop/reflect|S) = ☐

π(long, loops|S) = ☐

π(long, no loop/reflect|S) = ☐

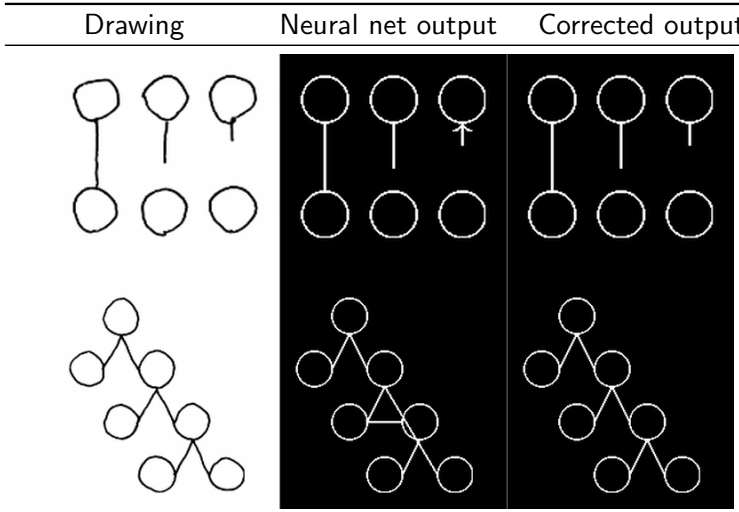
π(long, reflects|S) = ☐

etc.

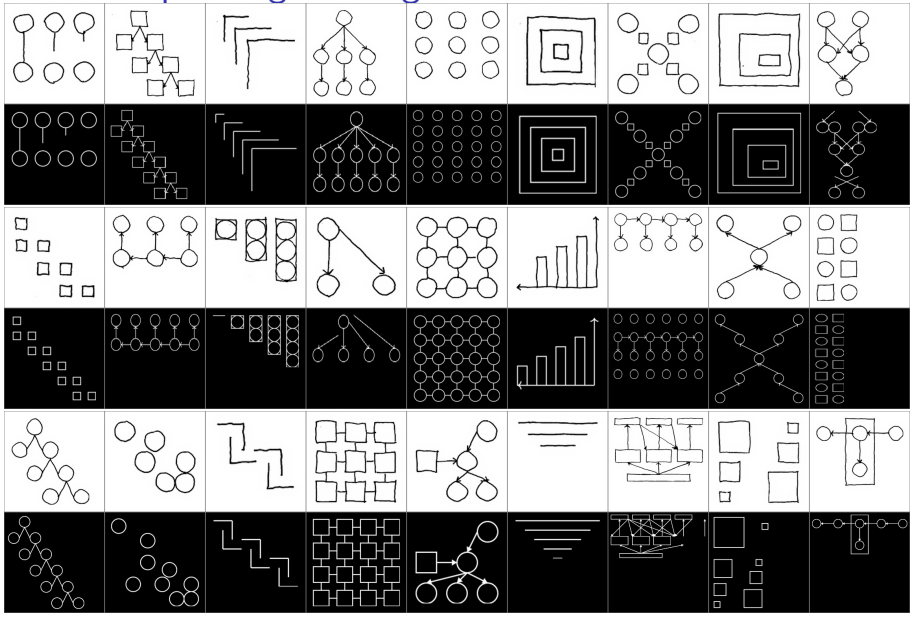


Application: Error correction

learn prior over programs (simple≈better), jointly infer likely parse+program
Top-down influence upon perception



Application: Extrapolating drawings



Visual input→Program: Poster AB #25