Learning to Infer Graphics Programs from Hand-Drawn Images

Kevin Ellis\textsuperscript{1}, Daniel Ritchie\textsuperscript{2}, Armando Solar-Lezama\textsuperscript{1}, Joshua B. Tenenbaum\textsuperscript{1}

\textsuperscript{1}: MIT. \textsuperscript{2}: Brown University.
Learning to Infer Graphics Programs from Hand-Drawn Images

Kevin Ellis\textsuperscript{1}, Daniel Ritchie\textsuperscript{2}, Armando Solar-Lezama\textsuperscript{1}, Joshua B. Tenenbaum\textsuperscript{1}

\textsuperscript{1}: MIT. \textsuperscript{2}: Brown University.
Learning to Infer Graphics Programs from Hand-Drawn Images
Kevin Ellis¹, Daniel Ritchie², Armando Solar-Lezama¹, Joshua B. Tenenbaum¹

¹: MIT. ²: Brown University.
Learning to Infer Graphics Programs from Hand-Drawn Images

Kevin Ellis\textsuperscript{1}, Daniel Ritchie\textsuperscript{2}, Armando Solar-Lezama\textsuperscript{1}, Joshua B. Tenenbaum\textsuperscript{1}

\textsuperscript{1}: MIT. \textsuperscript{2}: Brown University.

\begin{verbatim}
for (i < 3)
  rectangle(3*i,-2*i+4,3*i+2,6)
for (j < i + 1)
  circle(3*i+1,-2*j+5)
reflect(y=8)
for(i<3)
  if(i>0)
    rectangle(3*i-1,2,3*i,3)
    circle(3*i+1,3*i+1)
\end{verbatim}
Learning to Infer Graphics Programs from Hand-Drawn Images

Kevin Ellis\textsuperscript{1}, Daniel Ritchie\textsuperscript{2}, Armando Solar-Lezama\textsuperscript{1}, Joshua B. Tenenbaum\textsuperscript{1}

$^1$: MIT. $^2$: Brown University.
Learning to Infer Graphics Programs from Hand-Drawn Images
Kevin Ellis¹, Daniel Ritchie², Armando Solar-Lezama¹, Joshua B. Tenenbaum¹

¹: MIT. ²: Brown University.

Hand Drawing

\begin{verbatim}
 for (i < 3)
 rectangle(3*i,-2*i+4, 3*i+2,6)
 for (j < i + 1)
 circle(3*i+1,-2*j+5)
 reflect(y=8)
 for(i<3)
 if(i>0)
 rectangle(3*i-1,2,3*i,3)
 circle(3*i+1,3*i+1)
\end{verbatim}

LaTeX

\begin{verbatim}
 for (i < 3)
 rectangle(3*i,-2*i+4, 3*i+2,6)
 for (j < i + 1)
 circle(3*i+1,-2*j+5)
 reflect(y=8)
 for(i<3)
 if(i>0)
 rectangle(3*i-1,2,3*i,3)
 circle(3*i+1,3*i+1)
\end{verbatim}
Image $\rightarrow$ Program Pipeline

### Image (Observed)
- Rendering
- Learning + Stochastic search

### Spec/Drawing Commands (Latent)
- `line`, `line`, `rectangle`, `line`, ...

### Program (Latent)
- `for (j < 3)`
- `for (i < 3)`
- `if (...)`
- `line(...)`
- `line(...)`
- `rectangle(...)`

### Applications
- Extrapolation
- Error correction

### Image $\rightarrow$ Spec
- Spec $\rightarrow$ Program
- Learning + Program synthesis
Parsing images into \LaTeX{} \texttt{TikZ} Commands


Target image: $I$

Canvas: $\text{render}(S)$

CNN $\odot$ Image features $\rightarrow$ MLP $\rightarrow$ circle$(\text{STN (attention)} \rightarrow \text{MLP}, X=7, \text{STN (attention)} \rightarrow \text{MLP}, Y=12)$

Next drawing command

Renderer $256 \times 256 \times 2$

$16 \times 16 \times 10$

\begin{itemize}
  \item (a): hand drawing
  \item (b): noisy render of (a)'s spec
\end{itemize}
Parsing images into \LaTeX\ TikZ Commands


Target image: \( I \)
Canvas: \text{render}(S)

\begin{align*}
\text{CNN} & \quad \text{Image features} \\
\text{MLP} & \quad \text{MLP} \\
\text{circle(} & \quad \text{STN (attention)} \\
& \quad \text{STN (attention)} \\
X=7, & \quad Y=12) \\
\end{align*}

Next drawing command
Renderer
\( 256 \times 256 \times 2 \)
\( 16 \times 16 \times 10 \)

(a): hand drawing
(b): noisy render of (a)’s spec

\begin{align*}
\text{Average number of errors} & \quad \# \text{ objects} \\
\text{in sample} & \quad \text{out of sample} \\
\end{align*}
Parsing images into \LaTeX{} TikZ Commands


Target image: \( I \)

Canvas: render(\( S \))

\[
\begin{array}{c}
\text{CNN} \\
\text{Image features} \\
\text{STN (attention)} \\
\text{MLP} \\
\text{MLP} \\
\text{MLP}
\end{array}
\]

\( \text{circle(} \ X=7, \ Y=12 \text{)} \)

Renderer

256 \( \times \) 256 \( \times \) 2

16 \( \times \) 16 \( \times \) 10

(a): hand drawing

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

Average number of errors

\begin{itemize}
    \item in sample
    \item out of sample
\end{itemize}
Synthesizing high-level programs from specs (spec=draw commands)

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

\[
\text{program}(S) = \arg \min_{p \in \text{DSL}} \text{cost}(p) \\
p \text{ consistent w/ } S \\
\text{min cost} \approx \text{simple+short}
\]

DSL: Domain Specific Language: variables, arithmetic, loops, conditionals

---

Program \(\rightarrow\) Statement; \(\cdots\); Statement
Statement \(\rightarrow\) circle(Expression,Expression)
Statement \(\rightarrow\) rectangle(Expression,Expression,Expression,Expression,Expression)
Statement \(\rightarrow\) line(Expression,Expression,Expression,Expression,Expression,Expression,Boolean,Boolean)
Statement \(\rightarrow\) for(0 \(\leq\) Var < Expression) \{ if (Var > 0) \{ Program \}; Program \}
Statement \(\rightarrow\) reflect(Axis) \{ Program \}
Expression \(\rightarrow\) \(\mathbb{Z}\times\text{Var}+\mathbb{Z}\)
Axis \(\rightarrow\) \(\mathbb{X} = \mathbb{Z} \mid \mathbb{Y} = \mathbb{Z}\)
\(\mathbb{Z}\) \(\rightarrow\) an integer
Learning to quickly synthesize programs

Learn search policy $\pi(\text{program subspace}|\text{spec})$

Think of the subspace as an “ansatz”

OBJECTIVE: Small subspace for tractability while also being likely to contain good programs

Entire program search space
Learning to quickly synthesize programs

Learn search policy $\pi($program subspace$|$spec$)$

Think of the subspace as an “ansatz”

OBJECTIVE: Small subspace for tractability while also being likely to contain good programs

Entire program search space

- long programs
- short programs
Learning to quickly synthesize programs

Learn search policy $\pi(\text{program subspace}|\text{spec})$

Think of the subspace as an “ansatz”

OBJECTIVE: Small subspace for tractability while also being likely to contain good programs
Learning to quickly synthesize programs

Learn search policy $\pi(\text{program subspace}|\text{spec})$

Think of the subspace as an “ansatz”

OBJECTIVE: Small subspace for tractability while also being likely to contain good programs

Entire program search space
Learning to quickly synthesize programs

Learn search policy $\pi(\text{program subspace}|\text{spec})$

Think of the subspace as an “ansatz”

OBJECTIVE: Small subspace for tractability while also being likely to contain good programs

Entire program search space

$\pi(\text{short, no loop/reflect}|S) = \Box$
$\pi(\text{long, loops}|S) = \Box$
$\pi(\text{long, no loop/reflect}|S) = \Box$
$\pi(\text{long, reflects}|S) = \Box$

etc.
Learning to quickly synthesize programs

Learn search policy \( \pi(\text{program subspace}|\text{spec}) \)

Think of the subspace as an “ansatz”

**OBJECTIVE:** Small subspace for tractability while also being likely to contain good programs

---

**Entire program search space**

\[
\begin{align*}
\pi(\text{short, no loop/reflect}|S) &= \square \\
\pi(\text{long, loops}|S) &= \square \\
\pi(\text{long, no loop/reflect}|S) &= \square \\
\pi(\text{long, reflects}|S) &= \square \\
&\text{etc.}
\end{align*}
\]
Example programs

<table>
<thead>
<tr>
<th>Drawing</th>
<th>Spec</th>
<th>Program</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Line (2,15,4,15) Line (4,9,4,13) Line (3,11,3,14) Line (2,13,2,15) Line (3,14,6,14) Line (4,13,8,13)" /></td>
<td>Line (2,15,4,15) Line (4,9,4,13) Line (3,11,3,14) Line (2,13,2,15) Line (3,14,6,14) Line (4,13,8,13)</td>
<td>for (i &lt; 3) line (i, -1<em>i + 6, 2</em>i + 2, -1<em>i + 6) line (i, -2</em>i + 4, i, -1*i + 6)</td>
</tr>
<tr>
<td><img src="image" alt="Circle (2,8) Rectangle (6,9,7,10) Circle (8,8) Rectangle (6,12,7,13) Rectangle (3,9,4,10)" /></td>
<td>Circle (2,8) Rectangle (6,9,7,10) Circle (8,8) Rectangle (6,12,7,13) Rectangle (3,9,4,10)</td>
<td>reflect (y = 8) for (i &lt; 3) if (i &gt; 0) rectangle (3<em>i - 1, 2, 3</em>i, 3) circle (3<em>i + 1, 3</em>i + 1)</td>
</tr>
<tr>
<td><img src="image" alt="Line (3,10,3,14, arrow) Rectangle (11,8,15,10) Rectangle (11,14,15,15) Line (13,10,13,14, arrow)" /></td>
<td>Line (3,10,3,14, arrow) Rectangle (11,8,15,10) Rectangle (11,14,15,15) Line (13,10,13,14, arrow)</td>
<td>for (i &lt; 3) line (7,1,5<em>i + 2,3, arrow) for (j &lt; i + 1) if (j &gt; 0) line (5</em>j - 1,9,5<em>i,5, arrow) line (5</em>j + 2,5<em>j + 2,9, arrow) rectangle (5</em>i, 3, 5<em>i + 4, 5) rectangle (5</em>i, 9, 5*i + 4, 10) rectangle (2, 0, 12, 1)</td>
</tr>
</tbody>
</table>
Application: Error correction

learn prior over programs (simple≈better), jointly infer likely parse+program

Top-down influence upon perception

<table>
<thead>
<tr>
<th>Drawing</th>
<th>Neural net output</th>
<th>Corrected output</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Drawing" /></td>
<td><img src="image2.png" alt="Neural net output" /></td>
<td><img src="image3.png" alt="Corrected output" /></td>
</tr>
<tr>
<td><img src="image4.png" alt="Drawing" /></td>
<td><img src="image5.png" alt="Neural net output" /></td>
<td><img src="image6.png" alt="Corrected output" /></td>
</tr>
</tbody>
</table>
Application: Extrapolating drawings
for (i < 3)
rectangle(3*i,-2*i+4, 3*i+2,6)
for (j < i + 1)
circle(3*i+1,-2*j+5)
reflect(y=8)
for(i<3)
if(i>0)
rectangle(3*i-1,2,3*i,3)
circle(3*i+1,3*i+1)