Library Learning for Neurally-Guided Bayesian Program Induction
Kevin Ellis\textsuperscript{1}, Lucas Morales\textsuperscript{1}, Mathias Sablé-Meyer\textsuperscript{2},
Armando Solar-Lezama\textsuperscript{1}, Joshua B. Tenenbaum\textsuperscript{1}
\textsuperscript{1}: MIT. \textsuperscript{2}: ENS Paris-Saclay.

List processing:
\[ [5, 2, 9] \rightarrow [9, 2, 5] \]
\[ [1, 1, 2, 2] \rightarrow [2, 2, 1, 1] \]
\[ [1, 2, 3, 2] \rightarrow [2, 3, 2, 1] \]

Text editing:
P Kohli \rightarrow Dr. Kohli
Sumit Gulwani \rightarrow Dr. Gulwani
Danny Tarlow \rightarrow Dr. Tarlow

Symbolic regression:
\[ ax^3 + bx^2 + cx + d \]
\[ a/(x - b) \]

Explore/Compress/Compile (EC\textsuperscript{2}) learns to solve programming tasks like these by growing a library of code and training a neural net to search for programs written using the library.
Library learning

\[ [7 \ 2 \ 3] \rightarrow [7 \ 3] \]
\[ [1 \ 2 \ 3 \ 4] \rightarrow [3 \ 4] \]
\[ [4 \ 3 \ 2 \ 1] \rightarrow [4 \ 3] \]
Library learning

\[
\begin{align*}
[7 & 2 & 3] & \rightarrow [7 & 3] \\
[4 & 3 & 2 & 1] & \rightarrow [4 & 3]
\end{align*}
\]

**Library:**

\[
f_1(\ell, p) = \text{foldr } \ell \text{ nil } (\lambda (x \ a) \ (\text{if} \ (p \ x) \ (\text{cons} \ x \ a) \ a)))
\]

\(f_1: \text{Higher-order filter function}\)

\((\text{Get elements } x \text{ from } \ell \text{ where } (p \ x) \text{ returns true})\)
Library learning

\[
\begin{align*}
[7 \ 2 \ 3] & \rightarrow [7 \ 3] \\
[1 \ 2 \ 3 \ 4] & \rightarrow [3 \ 4] \\
[4 \ 3 \ 2 \ 1] & \rightarrow [4 \ 3]
\end{align*}
\]

\[f(\ell) = (f_1 \ \ell \ (\lambda (x) \ (> x \ 2)))\]

Library:

\[f_1(\ell,p) = (\text{foldr } \ell \ \text{nil} \ (\lambda (x \ a) \ (\text{if } (p \ x) \ (\text{cons } x \ a) \ a)))\]

\[(f_1: \text{Higher-order filter function})\]

\[(Get \ elements \ x \ from \ \ell \ where \ (p \ x) \ returns \ true)\]
Subset of 38 learned library routines for list processing

\[ f_0(\ell, r) = (\text{foldr } r \ \ell \ \text{cons}) \]

\[ f_0: \text{Append lists } r \text{ and } \ell \]

\[ f_1(\ell, p) = (\text{foldr } \ell \ \text{nil} \ (\lambda (x \ a) \ (\text{if } (p \ x) \ (\text{cons } x \ a) \ a))) \]

\[ f_1: \text{Higher-order filter function} \]

\[ f_2(\ell) = (\text{foldr } \ell \ 0 \ (\lambda (x \ a) \ (\text{if } (> a x) \ a \ x))) \]

\[ f_2: \text{Maximum element in list } \ell \]

\[ f_3(\ell, k) = (\text{foldr } \ell \ (\text{is-nil } \ell) \ (\lambda (x \ a) \ (\text{if } a \ a \ (= k x)))) \]

\[ f_2: \text{Whether } \ell \text{ contains } k \]
Explore/Compress/Compile as Bayesian Inference
Explore/Compress/Compile as Amortized Bayesian Inference
Explore/Compress/Compile as Amortized Bayesian Inference

**Explore**: Infer programs, fixing DSL and neural recognition model
Explore/Compress/Compile as Amortized Bayesian Inference

Explore: Infer programs, fixing DSL and neural recognition model

DSL

\(f_1, f_2, \ldots\)

Task

\[
\begin{align*}
[7 & 2 & 3] & \rightarrow [7 & 3] \\
[4 & 3 & 2 & 1] & \rightarrow [4 & 3]
\end{align*}
\]

Recog. model

Neurally-Guided Enumerative Search

Programs:

\((f_1 \ell (\lambda (x) (> x 2)))\)
Explore/Compress/Compile as Amortized Bayesian Inference

**Compress**: Update DSL, fixing programs
Explore/Compress/Compile as Amortized Bayesian Inference

**Compress**: Update DSL, fixing programs

progs. for task$_1$ → cons → + 1 1 → DSL+

progs. for task$_2$ → + car z → 1

DSL → prog task → prog task → prog task → is

DSL+ → prog task → prog task → prog task → is
Explore/Compress/Compile as Amortized Bayesian Inference

**Compile**: Train recognition model
Explore/Compress/Compile as Amortized Bayesian Inference

**Compile:** Train recognition model

```
DSL → sample program → run task
```

```
DSL
→ prog
↓
→ task
→ prog
↓
→ task
→ prog
↓
→ task
```
Explore/Compress/Compile as Amortized Bayesian Inference
In the style of FlashFill (Gulwani 2012). Starts with map, fold, etc.

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>+106 769-438</td>
<td>106.769.438</td>
</tr>
<tr>
<td>+83 973-831</td>
<td>83.973.831</td>
</tr>
<tr>
<td>Temple Anna H</td>
<td>TAH</td>
</tr>
<tr>
<td>Lara Gregori</td>
<td>LG</td>
</tr>
</tbody>
</table>
Text editing: Library learning builds on itself

Learned DSL primitives over 3 iterations (3 columns). Learned primitives call each other (arrows).
Programs with numerical parameters: Symbolic regression from visual input

Fits parameters by autograd-ing thru program

Recognition model looks at picture of function’s graph

<table>
<thead>
<tr>
<th>Programs &amp; Tasks</th>
<th>DSL</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Function 1" /></td>
<td>$f_0(x) = (+ x \text{ real})$</td>
</tr>
<tr>
<td>$f(x) = (f_1 \ x)$</td>
<td>$f_1(x) = (f_0 (* \text{ real } x))$</td>
</tr>
<tr>
<td><img src="image" alt="Function 2" /></td>
<td>$f_2(x) = (f_1 (* x (f_0 x)))$</td>
</tr>
<tr>
<td><img src="image" alt="Function 3" /></td>
<td>$f_3(x) = (f_0 (* x (f_2 x)))$</td>
</tr>
<tr>
<td><img src="image" alt="Function 4" /></td>
<td>$f_4(x) = (f_0 (* x (f_3 x)))$</td>
</tr>
<tr>
<td><img src="image" alt="Function 5" /></td>
<td>$f_5(x) = (/ \text{ real } (f_0 x))$</td>
</tr>
<tr>
<td>$f(x) = (f_5 \ x)$</td>
<td>($f_5$: rational function)</td>
</tr>
<tr>
<td><img src="image" alt="Function 6" /></td>
<td>($f_4$: 4th order polynomial)</td>
</tr>
</tbody>
</table>
New domain: Generative graphics programs (Turtle/LOGO)

Training tasks:
Generative graphics programs: Samples from DSL

DSL samples before learning

DSL samples after learning
Generative graphics programs: Learned library contains parametric drawing routines

- Semicircle:
- Greek Spiral:
- Polygons & Stars:
- Spiral:
- S-Curves:
- Circles:
Learning to program: Poster AB #24

\[
f_2(p, f, n, x) = (\text{if } (p \ x) \ \text{nil} \\
\quad \text{cons} \ (f \ x) \ (f_2 \ (n \ x))) \\
(f_2: \text{ unfold})
\]

\[
f_3(i, l) = (\text{if } (= i 0) \ (\text{car} \ l) \\
\quad (f_3 \ (f_1 \ i) \ (\text{cdr} \ l))) \\
(f_3: \text{ index})
\]

\[
f_4(f, l, x) = (\text{if } (\text{empty?} \ l) \ x \\
\quad (f \ (\text{car} \ l) \ (f_4 \ (\text{cdr} \ l)))) \\
(f_4: \text{ fold})
\]

\[
f_5(f, l) = (\text{if } (\text{empty?} \ l) \ \text{nil} \\
\quad \text{cons} \ (f \ (\text{car} \ l)) \ (f_5 \ (\text{cdr} \ l))) \\
(f_5: \text{ map})
\]

Symbolic Regression

<table>
<thead>
<tr>
<th>Symbolic Function</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>(f(x) = (f_1 \times))</td>
<td><img src="https://example.com/image1.png" alt="Image 1" /></td>
</tr>
<tr>
<td>(f(x) = (f_6 \times))</td>
<td><img src="https://example.com/image2.png" alt="Image 2" /></td>
</tr>
</tbody>
</table>

\[
f_0(x) = (+ \ x \ \text{real})
\]

\[
f_1(x) = (f_0 \ (* \ \text{real} \ x))
\]

\[
f_2(x) = (f_1 \ (* \ x \ (f_0 \ x)))
\]

\[
f_3(x) = (f_0 \ (* \ x \ (f_2 \ x)))
\]

\[
f_4(x) = (f_0 \ (* \ x \ (f_3 \ x)))
\]

\(f_4: 4\text{th order polynomial}\)

\[
f_5(x) = (/ \ \text{real} \ x)
\]

\[
f_6(x) = (f_5 \ (f_0 \ x))
\]

\(f_6: \text{ rational function}\)