Improving Neural Program Synthesis with Inferred Execution Traces

Richard Shin\textsuperscript{1}  Illia Polosukhin\textsuperscript{2}  Dawn Song\textsuperscript{1}

\textsuperscript{1} UC Berkeley
\textsuperscript{2} NEAR Protocol

Poster: Room 210 & 230 AB #31
For program synthesis from input-output examples, end-to-end neural networks have become popular. Current research trend: add better inductive bias to help model learn. Intuitively, execution traces are a great inductive bias for program synthesis.

**Encoder-decoder neural network**

**Desired program**

```python
def run():
    repeat(2):
        turnRight()
        move()
```

Background

- Program synthesis from **execution traces** should be an easier task:
  - Strict superset of information in input-output example
  - Contains detailed information about the desired program state at each step of execution
  - Greater supervision about the effects of each elementary operation

---

**Trace-based synthesizer**

```python
def run():
    repeat(2):
        turnRight()
        move()
```

---

**Improving Neural Program Synthesis with Inferred Execution Traces.** Richard Shin, Illia Polosukhin, Dawn Song.
Poster: Room 210 & 230 AB #31
Main question:
Given input-output examples, can we infer execution traces automatically and use the inferred traces to better synthesize programs?

Our findings:
Yes. On the Karel domain, we achieve state-of-the-art results, improving accuracy for both simple and complex programs.

Our hypothesis:
Adding an inductive bias in the form of explicit trace inference improves program synthesis.
Karel the Robot

Simple programming language designed for teaching programming.

An imperative program controls an agent (“Karel the Robot”) within a grid world.

Function:
def run():
    block

Conditional:
if (condition):
    block
if (condition):
    block
else:
    block

Loops:
for i in range(count):
    body
while (condition):
    body
while (not condition):
    body

Actions:
move()
turnLeft()
turnRight()
putMarker()
pickMarker()

Conditions:
frontIsClear()
leftIsClear()
rightIsClear()
markerPresent()
Summary of approach

**Step 1**
I/O → Trace Model

**Step 2**
Trace → Code Model

```python
def run():
    repeat(2):
        turnRight()
        move()
```

---

**Input**

**Output**

- Step 1
- Step 2
- Step 3
- Step 4
I/O → Trace

Execution trace predicted from I/O
Evaluation

- We used the same dataset as Bunel et al [1], consisting of
  - 1,116,854 training examples
  - 2,500 test examples

  Each example contains the ground truth program and 6 input-output pairs.

- To train the models:
  - We train the I/O $\rightarrow$ Trace model on 1,116,854 $\times$ 6 execution traces from the training set.
  - By running the trained I/O $\rightarrow$ Trace model over the training data, we obtain inferred traces for each example.
  - We train the Trace $\rightarrow$ Code model with the inferred traces from the I/O $\rightarrow$ Trace model.

- Model receives 5 input-output pairs; 6th is held out.


<table>
<thead>
<tr>
<th>Previous work</th>
<th>Top-1</th>
<th>Top-50</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Exact Match</td>
<td>Generalization</td>
</tr>
<tr>
<td>MLE (Bunel et al. 2018)</td>
<td>39.94%</td>
<td>71.91%</td>
</tr>
<tr>
<td>RL_beam_div_opt (Bunel et al. 2018)</td>
<td>32.71%</td>
<td>77.12%</td>
</tr>
<tr>
<td>I/O → Code, MLE (reimpl. of row 1)</td>
<td>40.1%</td>
<td>73.5%</td>
</tr>
<tr>
<td>I/O → Trace → Code, MLE</td>
<td><strong>42.8%</strong></td>
<td><strong>81.3%</strong></td>
</tr>
</tbody>
</table>

**Improving Neural Program Synthesis with Inferred Execution Traces.** Richard Shin, Illia Polosukhin, Dawn Song.

Poster: Room 210 & 230 AB #31
<table>
<thead>
<tr>
<th>Previous work</th>
<th>Top-1</th>
<th></th>
<th></th>
<th>Top-50</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Exact Match</td>
<td>Generalization</td>
<td>Guided Search</td>
<td>Generalization</td>
</tr>
<tr>
<td>MLE (Bunel et al. 2018)</td>
<td>39.94%</td>
<td>71.91%</td>
<td>—</td>
<td>86.37%</td>
</tr>
<tr>
<td>RL_beam_div_opt (Bunel et al. 2018)</td>
<td>32.71%</td>
<td>77.12%</td>
<td>—</td>
<td>85.38%</td>
</tr>
<tr>
<td>I/O → Code, MLE (reimpl. of row 1)</td>
<td>40.1%</td>
<td>73.5%</td>
<td>84.6%</td>
<td>85.8%</td>
</tr>
<tr>
<td>I/O → Trace → Code, MLE</td>
<td>42.8%</td>
<td>81.3%</td>
<td>88.8%</td>
<td>90.8%</td>
</tr>
</tbody>
</table>

inferring program textually matches the ground truth
<table>
<thead>
<tr>
<th>Previous work</th>
<th>Top-1</th>
<th>Top-50</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Exact Match</td>
<td>Generalization</td>
</tr>
<tr>
<td>MLE (Bunel et al. 2018)</td>
<td>39.94%</td>
<td>71.91%</td>
</tr>
<tr>
<td>RL_beam_div_opt (Bunel et al. 2018)</td>
<td>32.71%</td>
<td>77.12%</td>
</tr>
<tr>
<td>I/O → Code, MLE (reimpl. of row 1)</td>
<td>40.1%</td>
<td>73.5%</td>
</tr>
<tr>
<td>I/O → Trace → Code, MLE</td>
<td><strong>42.8%</strong></td>
<td><strong>81.3%</strong></td>
</tr>
</tbody>
</table>

Inferred program textually matches the ground truth.

Inferred program executes correctly on all 6 input-output pairs.
<table>
<thead>
<tr>
<th>Previous work</th>
<th>Top-1</th>
<th>Top-50</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Exact Match</td>
<td>Generalization</td>
</tr>
<tr>
<td>MLE (Bunel et al. 2018)</td>
<td>39.94%</td>
<td>71.91%</td>
</tr>
<tr>
<td>RL_beam_div_opt (Bunel et al. 2018)</td>
<td>32.71%</td>
<td>77.12%</td>
</tr>
<tr>
<td>I/O → Code, MLE (reimpl. of row 1)</td>
<td>40.1%</td>
<td>73.5%</td>
</tr>
<tr>
<td>I/O → Trace → Code, MLE</td>
<td><strong>42.8%</strong></td>
<td><strong>81.3%</strong></td>
</tr>
</tbody>
</table>

whether *any* of the 50 beam search outputs executes correctly on all 6 input-output pairs
1. Enumerate the top 50 program outputs in order using beam search
2. Test each candidate program on the 5 specifying input-output pairs
3. Given the first program correct on those 5 pairs, see if it works correctly on the **held-out 6th program**

---

**Improving Neural Program Synthesis with Inferred Execution Traces.** Richard Shin, Illia Polosukhin, Dawn Song. Poster: Room 210 & 230 AB #31
<table>
<thead>
<tr>
<th>Slice</th>
<th>% of dataset</th>
<th>I/O → Code</th>
<th>I/O → Trace → Code</th>
<th>Δ%</th>
</tr>
</thead>
<tbody>
<tr>
<td>No control flow</td>
<td>26.4%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>+0.0%</td>
</tr>
<tr>
<td>With conditionals</td>
<td>15.6%</td>
<td>87.4%</td>
<td>91.0%</td>
<td>+3.6%</td>
</tr>
<tr>
<td>With loops</td>
<td>29.9%</td>
<td>91.3%</td>
<td>94.3%</td>
<td>+3.0%</td>
</tr>
<tr>
<td>With conditionals and loops</td>
<td>73.6%</td>
<td>79.0%</td>
<td>84.8%</td>
<td>+5.8%</td>
</tr>
<tr>
<td>Program length 0–15</td>
<td>44.8%</td>
<td>99.5%</td>
<td>99.5%</td>
<td>+0.0%</td>
</tr>
<tr>
<td>Program length 15–30</td>
<td>40.7%</td>
<td>80.8%</td>
<td>86.9%</td>
<td>+6.1%</td>
</tr>
<tr>
<td>Program length 30+</td>
<td>14.5%</td>
<td>48.6%</td>
<td>61.0%</td>
<td>+12.4%</td>
</tr>
</tbody>
</table>

(all numbers are top-1 generalization)
Thanks for listening!

Come see our poster at
Room 210 & 230 AB #31

```python
def run():
    repeat(2):
        turnRight()
        move()
    turnRight()  # end
    turnRight()  # end
    move()      # end
    turnRight()  # end
    turnRight()  # end
    move()      # end
    turnRight()  # end
    turnRight()  # end
    move()      # end
```

---

<table>
<thead>
<tr>
<th>Step 1</th>
<th>Step 2</th>
<th>Step 3</th>
<th>Step 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>turnRight</td>
<td>turnRight</td>
<td>move</td>
<td>&lt;end&gt;</td>
</tr>
<tr>
<td>turnRight</td>
<td>turnRight</td>
<td>move</td>
<td>&lt;end&gt;</td>
</tr>
<tr>
<td>turnRight</td>
<td>turnRight</td>
<td>move</td>
<td>&lt;end&gt;</td>
</tr>
</tbody>
</table>

---

1. I/O → Trace Model

2. Trace → Code Model

---