Neural-Guided Deductive Search for Real-Time Program Synthesis from Examples

Ashwin Kalyan
ashwink@gatech.edu
Abhishek Mohta
t-abhoht@microsoft.com
Alex Polozov
polozov@microsoft.com
Dhruv Batra
dbatra@gatech.edu
Prateek Jain
prajain@microsoft.com
Sumit Gulwani
sumitg@microsoft.com

PROBLEM

- Program synthesis from input-output examples
- Example domain: string → string transformations for data wrangling

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Hello&quot;</td>
<td>&quot;World&quot;</td>
</tr>
</tbody>
</table>

- Domain-specific language (excerpt):

```java
String transform := atom | Concat(atom, transform); 
String atom := ConstStr(s) | Substring(pos, pos); 
Pos := AbsolutePosition(\#) | RegexPosition(r, \#) | RelativeOffsetPosition(A); 
String s := \#; int k; Regex r; 
```

- Desiderata: correct, generalizable, and fast

APPROACH

Key idea: deductive search guided by an offline-trained neural model.

Unify the strengths of symbolic and neural program synthesis:

- √ Programs that produce correct computation
- √ Generalizes from a single example in 68% cases
- √ Real-time performance of < 0.1 sec
- √ Trainable with SGD avoids engineering effort

Symbolic: deductive search

- Derive the desired program top-down
- Logically reduce the synthesis problem into smaller subproblems
- Ensure that the derived programs satisfy the examples

Neural: branch prioritization

- A model to eliminate most unproductive search branches a priori
- Incorporates statistical insight about real-life programs
- Improves search performance up to 12x

TRAINING

- Deductive search generates independent subproblems
- The search space without neural guidance is exponential
- Thus, 375 real-world tasks ⇒ over 600,000 subproblems:
  (Branch \( \Gamma \), subproblem spec \( \varphi \), a posteriori best program score)
- Fully supervised regression training, squared error loss

EVALUATION

Industrial setup: generalize from a single input-output example.

Goal: match or exceed PROSE accuracy − significantly reduce its time.

Baselines:

- √ PROSE [Polozov & Gulwani 2015], purely symbolic
- √ RobustFill [Devlin et al. 2017] with 1-3 examples, purely neural
- DeepCoder [Balog et al. 2017] with 1-3 examples, a shallow neuro-symbolic hybrid

Method | Accuracy % | Speed-up ÷ PROSE |
--- | --- | --- |
PROSE | 67.12 | 1.00 |
DeepCoder, 1 ex. | 35.81 | 1.82 |
DeepCoder, 2 ex. | 47.38 | 1.53 |
DeepCoder, 3 ex. | 62.92 | 1.42 |
RobustFill, 1 ex. | 24.53 | 0.25 |
RobustFill, 2 ex. | 39.72 | 0.27 |
RobustFill, 3 ex. | 56.41 | 0.30 |
NGDS | 68.49 | 1.67 |

Purposely neural or purely symbolic systems suffer in accuracy and/or speed.

Successful scenario (2.7 × speed-up)

Input | Output
--- | ---
41.7114830017 | 41.7114830017
41.6076278685 | 41.6379013671