Improving Neural Program Synthesis with Inferred Execution Traces

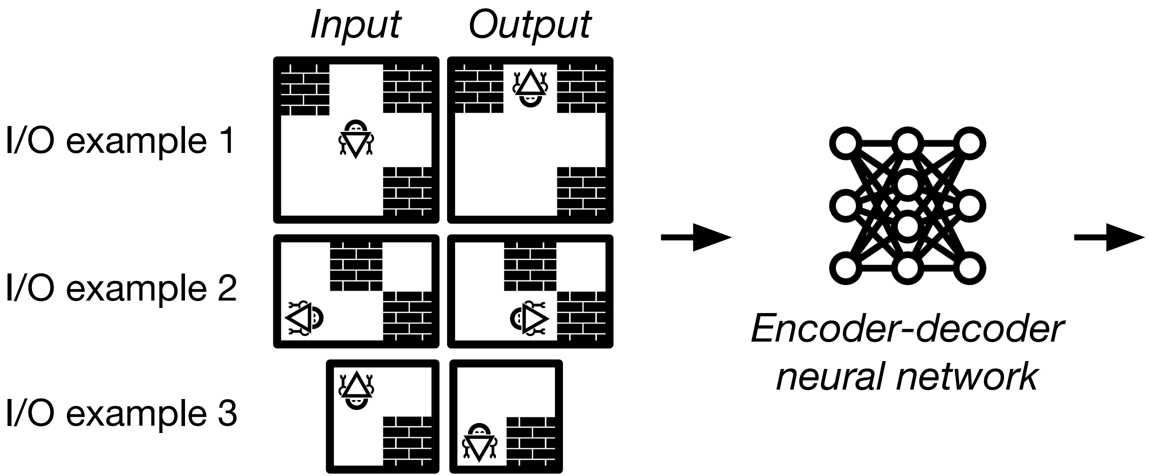
Richard Shin[1] Illia Polosukhin[2] Dawn Song[1]

[1] UC Berkeley
[2] NEAR Protocol

Poster: Room 210 & 230 AB #31

Background

- 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



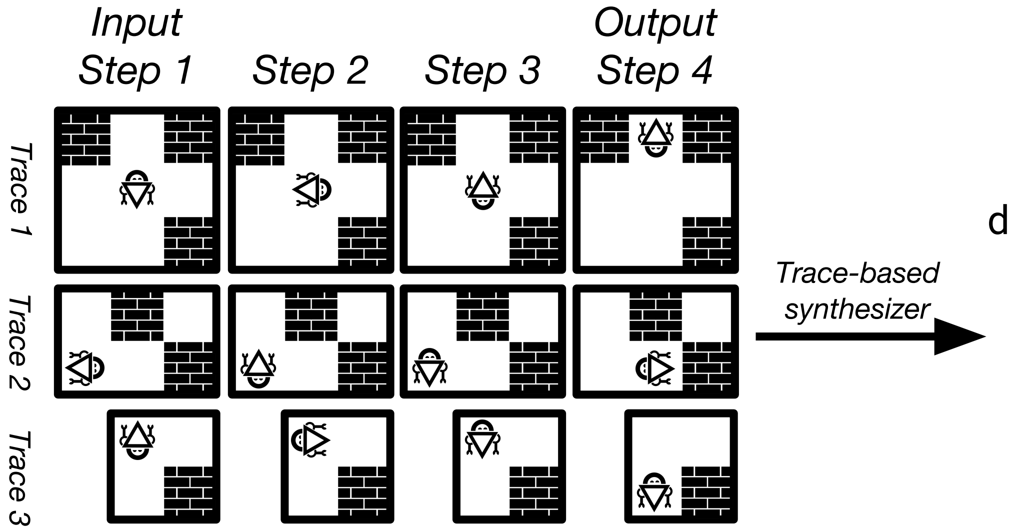
Desired program

```
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

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



Desired program

```
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?

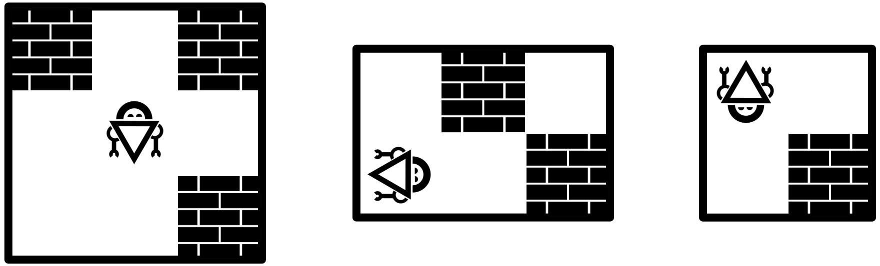
Our findings:
Yes. On the Karel domain, we achieve state-of-the-art results, improving accuracy for both simple and complex programs.

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:

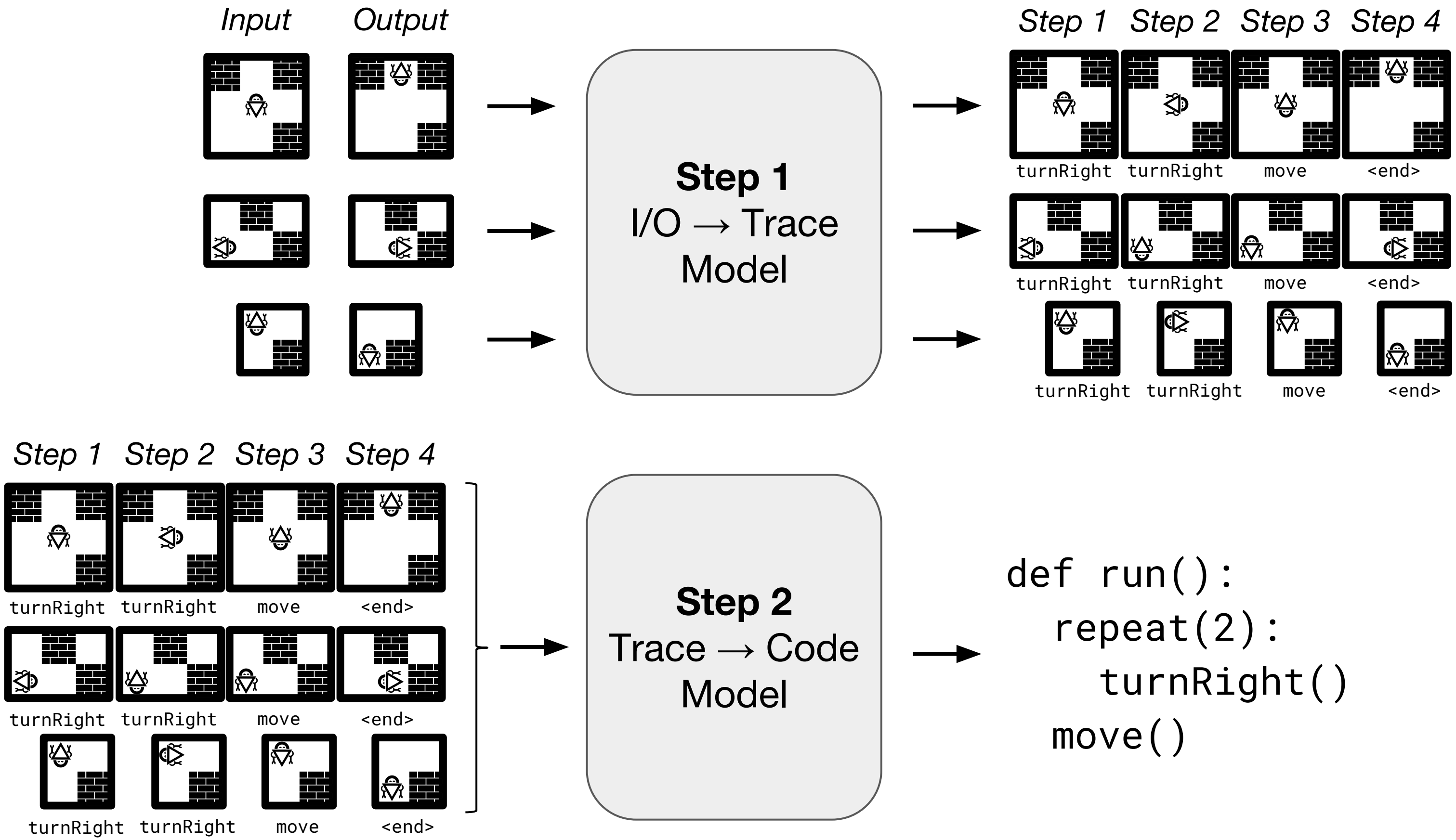
```
move()
turnLeft()
turnRight()
putMarker()
pickMarker()
```

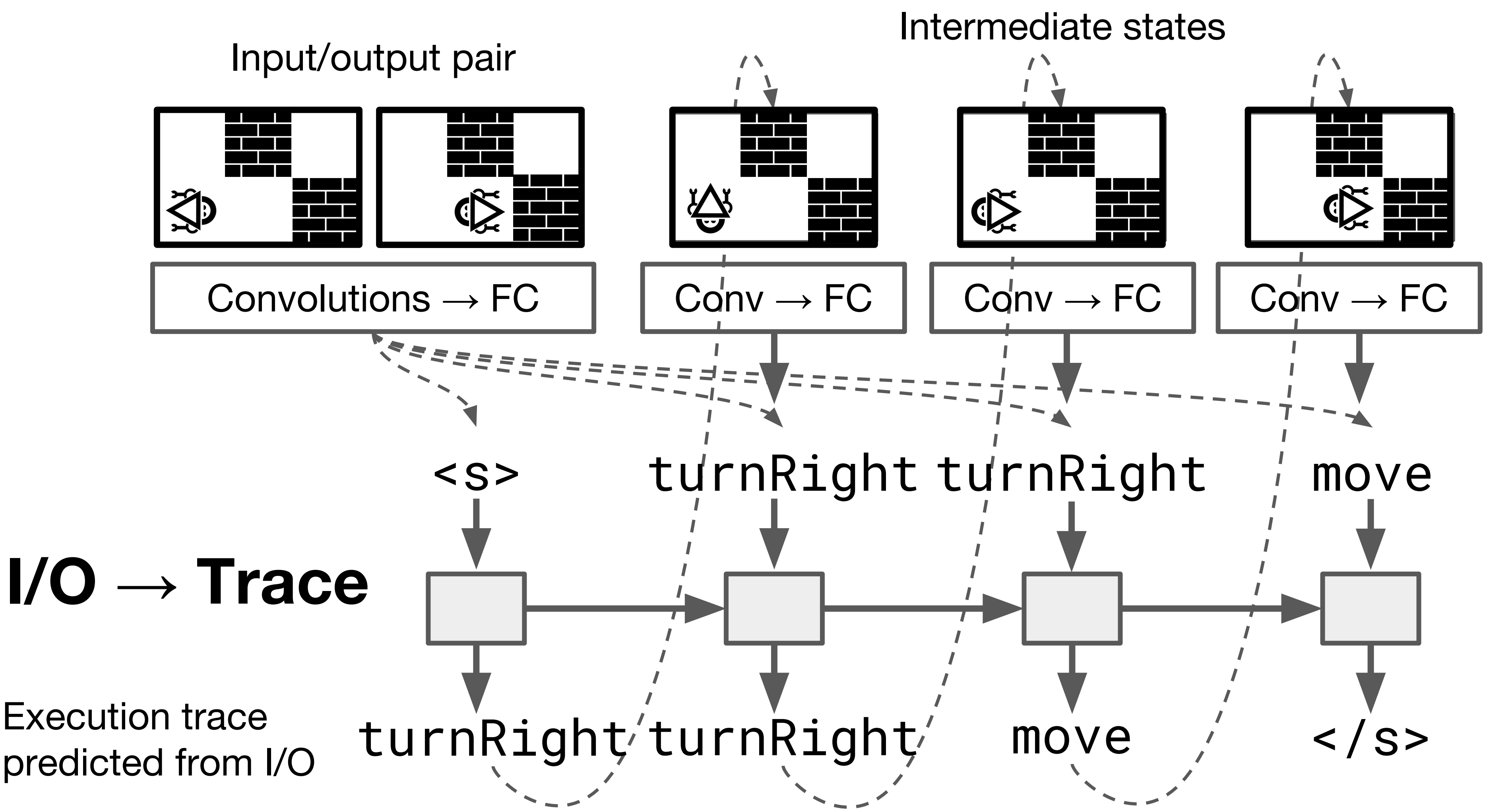
Conditions:

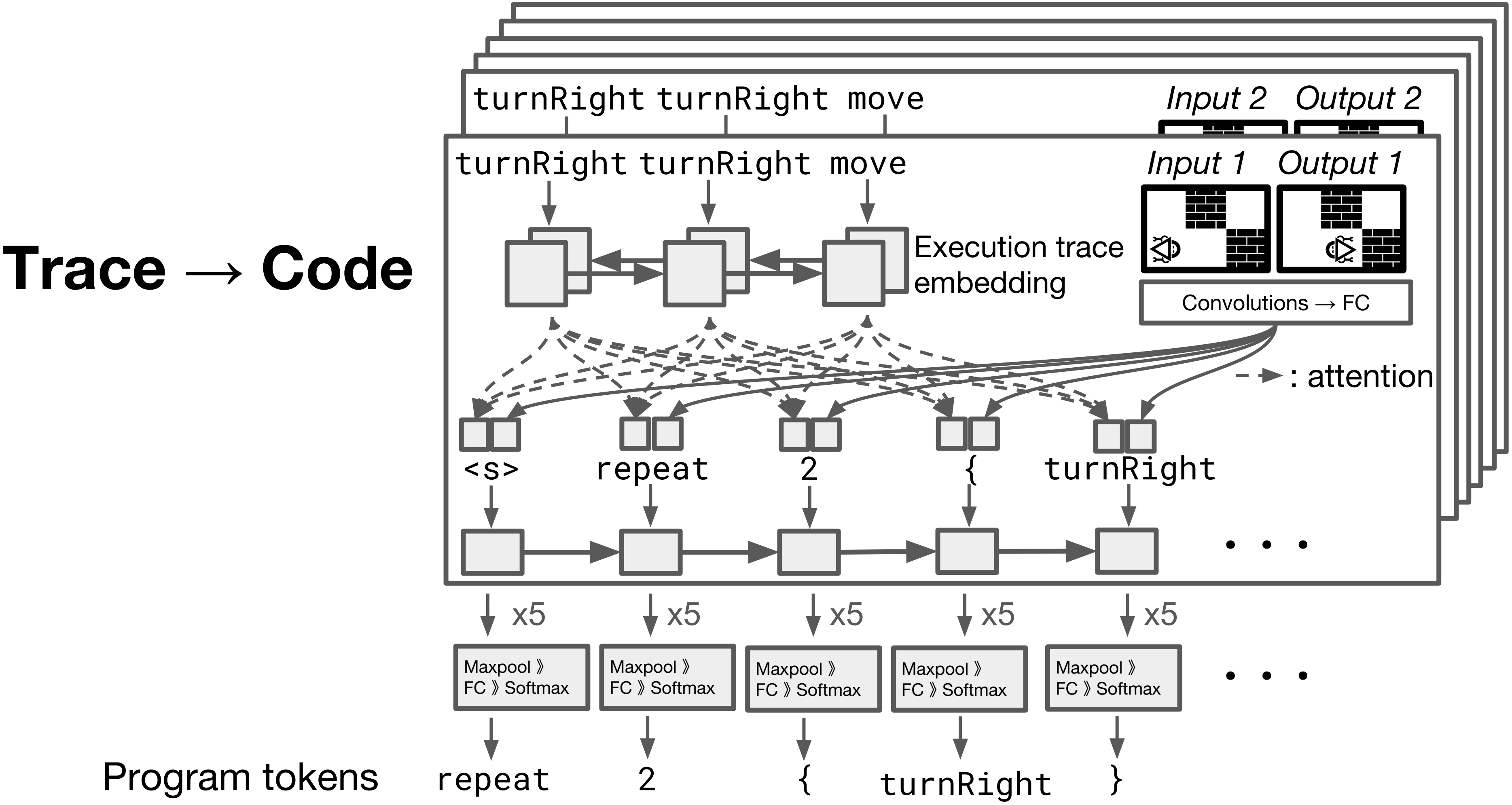
```
frontIsClear()
leftIsClear()
rightIsClear()
markerPresent()
```

Improving Neural Program Synthesis with Inferred Execution Traces. Richard Shin, Illia Polosukhin, Dawn Song.
Poster: Room 210 & 230 AB #31

Summary of approach




Intermediate states
Input/output pair
Convolutions → FC
Conv → FC
Conv → FC
Conv → FC
<s>
turnRight
turnRight
move
I/O → Trace
Execution trace predicted from I/O
turnRight
turnRight
move
</s>


Trace → Code
turnRight turnRight move
Input 2
Output 2
turnRight turnRight move
Input 1
Output 1
Execution trace embedding
Convolutions → FC
: attention
<s>
repeat
2
{
turnRight
x5
Maxpool 》 FC 》 Softmax
Program tokens
repeat
2
{
turnRight
}

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 → Trace model on 1,116,854 × 6 execution traces from the training set.
  - By running the trained I/O → Trace model over the training data, we obtain inferred traces for each example.
  - We train the Trace → Code model with the inferred traces from the I/O → Trace model.
- Model receives **5** input-output pairs; 6th is held out.

[1] Rudy Bunel, Matthew Hausknecht, Jacob Devlin, Rishabh Singh, and Pushmeet Kohli. *Leveraging Grammar and Reinforcement Learning for Neural Program Synthesis*. ICLR 2018.

		Top-1		Top-50	
		Exact Match	Generalization	Guided Search	Generalization
Previous work	MLE (Bunel et al. 2018)	39.94%	71.91%	—	86.37%
	RL_beam_div_opt (Bunel et al. 2018)	32.71%	77.12%	—	85.38%
	I/O → Code, MLE (reimpl. of row 1)	40.1%	73.5%	84.6%	85.8%
	I/O → Trace → Code, MLE	**42.8%**	**81.3%**	**88.8%**	**90.8%**

Improving Neural Program Synthesis with Inferred Execution Traces. Richard Shin, Illia Polosukhin, Dawn Song.
Poster: Room 210 & 230 AB #31

		Top-1		Top-50	
		Exact Match	Generalization	Guided Search	Generalization
Previous work	MLE (Bunel et al. 2018)	39.94%	71.91%	—	86.37%
	RL_beam_div_opt (Bunel et al. 2018)	32.71%	77.12%	—	85.38%
	I/O → Code, MLE (reimpl. of row 1)	40.1%	73.5%	84.6%	85.8%
	I/O → Trace → Code, MLE	**42.8%**	**81.3%**	**88.8%**	**90.8%**

inferred program
textually matches the
ground truth

Improving Neural Program Synthesis with Inferred Execution Traces. Richard Shin, Illia Polosukhin, Dawn Song.
Poster: Room 210 & 230 AB #31

		Top-1		Top-50	
		Exact Match	Generalization	Guided Search	Generalization
Previous work	MLE (Bunel et al. 2018)	39.94%	71.91%	—	86.37%
	RL_beam_div_opt (Bunel et al. 2018)	32.71%	77.12%	—	85.38%
	I/O → Code, MLE (reimpl. of row 1)	40.1%	73.5%	84.6%	85.8%
	I/O → Trace → Code, MLE	**42.8%**	**81.3%**	**88.8%**	**90.8%**

inferred program textually matches the ground truth

inferred program executes correctly on all **6** input-output pairs

Improving Neural Program Synthesis with Inferred Execution Traces. Richard Shin, Illia Polosukhin, Dawn Song.
Poster: Room 210 & 230 AB #31

		Top-1		Top-50	
		Exact Match	**Generalization**	**Guided Search**	**Generalization**
Previous work	MLE (Bunel et al. 2018)	39.94%	71.91%	—	86.37%
	RL_beam_div_opt (Bunel et al. 2018)	32.71%	77.12%	—	85.38%
	I/O → Code, MLE (reimpl. of row 1)	40.1%	73.5%	84.6%	85.8%
	I/O → Trace → Code, MLE	**42.8%**	**81.3%**	**88.8%**	**90.8%**

whether *any* of the 50 beam search outputs executes correctly on all **6** input-output pairs

	Top-1		Top-50	
	Exact Match	Generalization	Guided Search	Generalization
Previous work: MLE (Bunel et al. 2018)	39.94%	71.91%	—	86.37%
Previous work: RL_beam_div_opt (Bunel et al. 2018)	32.71%	77.12%	—	85.38%
I/O → Code, MLE (reimpl. of row 1)	40.1%	73.5%	84.6%	85.8%
I/O → Trace → Code, MLE	**42.8%**	**81.3%**	**88.8%**	**90.8%**

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

Slice	% of dataset	I/O → Code	I/O → Trace → Code	Δ%
No control flow	26.4%	100.0%	100.0%	**+0.0%**
With conditionals	15.6%	87.4%	91.0%	**+3.6%**
With loops	29.9%	91.3%	94.3%	**+3.0%**
With conditionals and loops	73.6%	79.0%	84.8%	**+5.8%**
Program length 0–15	44.8%	99.5%	99.5%	**+0.0%**
Program length 15–30	40.7%	80.8%	86.9%	**+6.1%**
Program length 30+	14.5%	48.6%	61.0%	**+12.4%**

(all numbers are top-1 generalization)

Thanks for listening!

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