Neural (Meta) Program Synthesis

Rishabh Singh, Google Brain

Great Collaborators!


Surya Bhupatiraju


Rudy Bunel


Jacob Devlin


Rasool Fakoor


Matthew Hausknecht

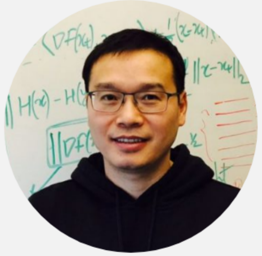

Pushmeet Kohli


Lihong Li


Abdel-rahman Mohamed


Emilio Parisotto


Jonathan Uesato


Denny Zhou

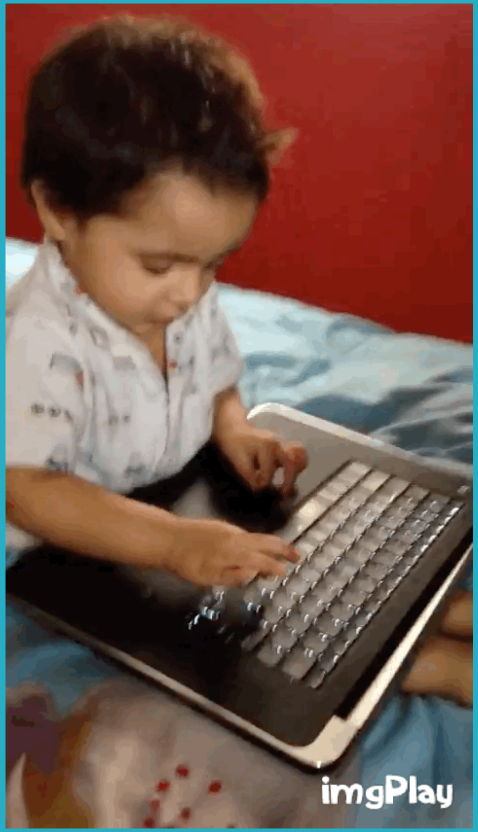
Deep Learning and Evolutionary Progression

Vision

Speech

Language

Programming



Perceptual Tasks

Algorithmic Tasks

Neural Program Learning

More Complex Tasks

Generalizability

Interpretability

Long term Vision

Agent to win programming contests [TopCoder]

Program Representations

- Program Repair (ICSE'18, ICLR'18)
- Fuzzing/Security Testing (ASE'17)
- Program Optimization

Neural Programmers

Spec

I/O Examples
Natural Language
Partial programs

```
def add5(x):
   return x+5

def dotwrite(ast):
   nodename = getNodename()
   label=symbol.sym_name.get(int(ast[0]),ast[0])
   print ' %s [label="%s' % (nodename, label),
   if isinstance(ast[1], str):
      if ast[1].strip():
         print '= %s"];' % ast[1]
      else:
         print '"]'
   else:
      print '"];'
      children = []
      for n, child in enumerate(ast[1:]):
         children.append(dotwrite(child))
      print ' %s -> {' % nodename,
      for name in children:
         print '%s' % name,
```

Logic

Basics

Experience

Samples

Neural Program Induction

Neural Programmer-Interpreters

Scott Reed & **Nando de Freitas**
Google DeepMind
London, UK
scott.ellison.reed@gmail.com
nandodefreitas@google.com

Inferring Algorithmic Patterns with Stack-Augmented Recurrent Nets

Armand Joulin
Facebook AI Research
770 Broadway, New York, USA.
ajoulin@fb.com

Tomas Mikolov
Facebook AI Research
770 Broadway, New York, USA.
tmikolov@fb.com

Neural Turing Machines

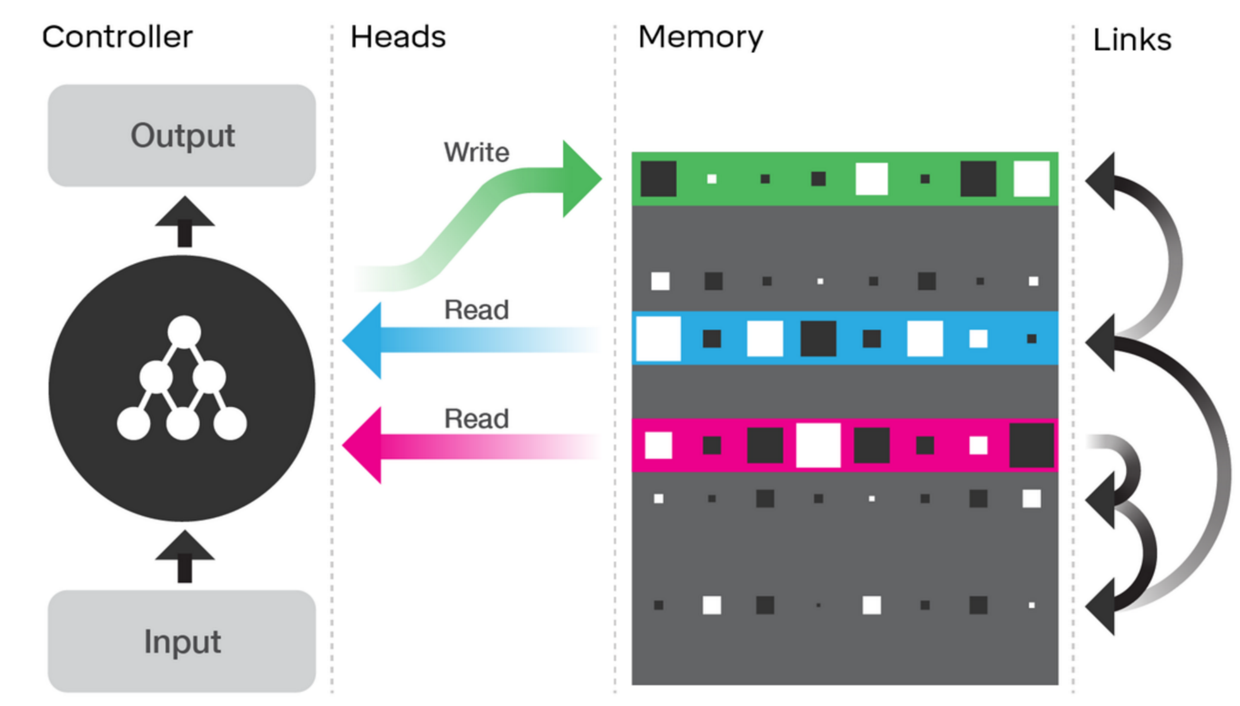
Alex Graves gravesa@google.com
Greg Wayne gregwayne@google.com
Ivo Danihelka danihelka@google.com

Google DeepMind, London, UK

Neural Random-Access Machines

Karol Kurach* & **Marcin Andrychowicz*** & **Ilya Sutskever**
Google
{kkurach,marcina,ilyasu}@google.com

Differentiable Neural Computer [Graves et al. Nature 2016]



Neural Program Induction

Differentiable memory, stack
Difficult to Generalize
Lots of Examples
Single-task learning
Non-Interpretable programs
Examples: NTM, DNC, etc.

Neural Program Synthesis

Functional Abstractions
Generalizes Better
Lots of Examples
Single-task learning
Interpretable programs
Examples: QuickSort

Meta- Neural Program Synthesis

Functional Abstractions
Strong Generalization
Few Examples
Multi-task learning
Interpretable programs

Neuro-Symbolic Program Synthesis (ICLR 2017)

Emilio Parisotto, Abdelrahman Mohamed, Rishabh Singh, Lihong Li, Dengyong Zhou, Pushmeet Kohli

FlashFill in Excel 2013

Email	First Name
Nancy.Freehafer@fourthcoffee.com	Nancy
Andrew.Cencini@northwindtraders.com	Andrew
Jan.Kotas@litwareinc.com	Jan
Mariya.Sergienko@graphicdesigninstitute.com	Mariya
Steven.Thorpe@northwindtraders.com	Steven
Michael.Neipper@northwindtraders.com	Michael
Robert.Zare@northwindtraders.com	Robert
Laura.Giussani@adventure-works.com	Laura
Anne.HL@northwindtraders.com	Anne
Alexander.David@contoso.com	Alexander
Kim.Shane@northwindtraders.com	Kim
Manish.Chopra@northwindtraders.com	Manish
Gerwald.Oberleitner@northwindtraders.com	Gerwald
Amr.Zaki@northwindtraders.com	
Yvnonne.McKay@northwindtraders.com	
Amanda.Pinto@northwindtraders.com	

Gulwani, Harris, Singh [CACM Research Highlight 2012]

FlashFill DSL

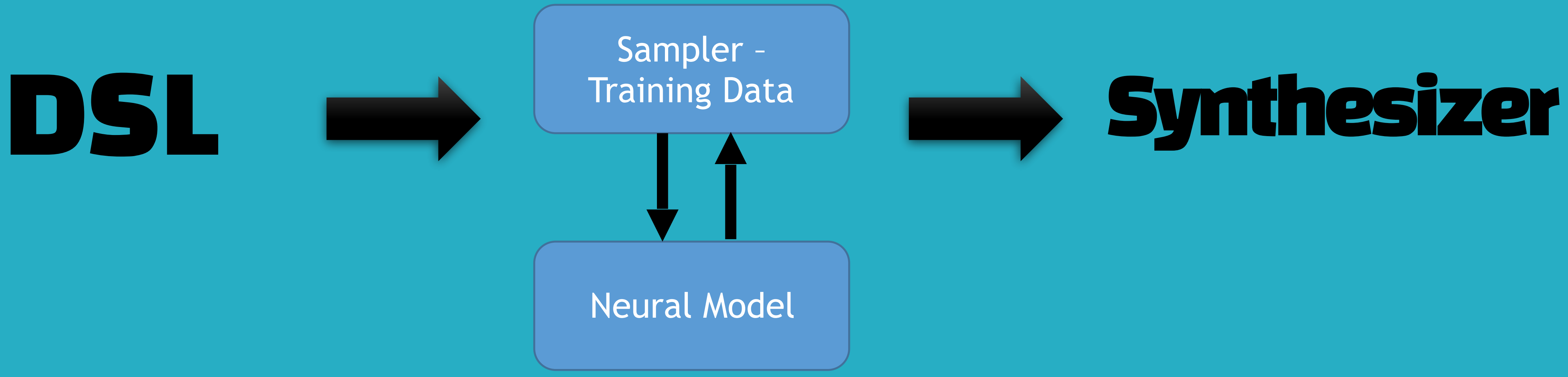
$$
\begin{array}{rcl}
\text{String } e & := & \texttt{Concat}(f_1, \cdots, f_n) \\
\text{Substring } f & := & \texttt{ConstStr}(s) \\
 & | & \texttt{SubStr}(v, p_l, p_r) \\
\text{Position } p & := & (r, k, \texttt{Dir}) \mid \texttt{ConstPos}(k) \\
\text{Direction } \texttt{Dir} & := & \texttt{Start} \mid \texttt{End} \\
\text{Regex } r & := & s \mid T_1 \ \cdots \mid T_n
\end{array}
$$

Example FlashFill Task

Input (v)	Output
William Henry Charles	Charles, W.
Larry Page	Page, L.
Sergey Brin	Brin, S.
Martha D. Saunders	Saunders, M.

```
Concat(f_1, ConstStr(", "), f_2, ConstStr("."))
f_1 = SubStr(v, (Word,-1,Start), (Word,-1,End))
f_2 = SubStr(v, CPos(0), CPos(1))
```

General Methodology



3 Key Properties

- Syntax
- Semantics
- Executable

Synthetic Training Data

Reference program: GetToken_Alphanum_3 | GetFrom_Colon_1 | GetFirst_Char_4	
Ud 9:25,JV3 Obb	2525,JV3 bbUd92
zLny xmHg 8:43 A44q	843 A44qzLny
A6 g45P 10:63 Jf	1063 JfA6g4
cuL.zF.dDX,12:31	dDX31cuLz
ZiG OE bj3u 7:11	bj3u11ZiGO

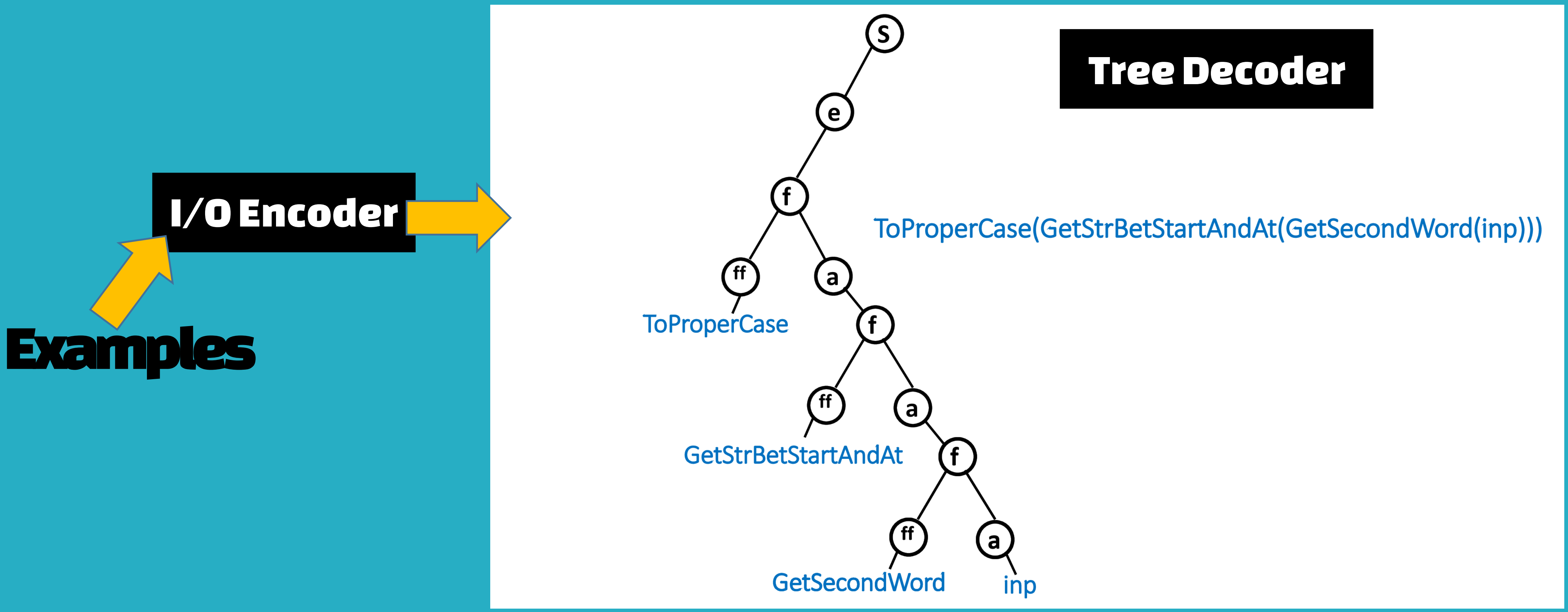
Reference program: GetToken_WS_-1(GetSpan(Number, 1 End, '/', 3, End)) | Const('R') | GetToken_Word_5 | Const('L') | Const(',') | ToProper(GetToken_Word_3) | GetToken_Alphanum_5 | EOS	
aC Ic 3.rFL JiW.MmB fzYoa TX oNpV fHm /ai WHGM Pgso.OXp VKW Jo R9 OJUF / / Xir	JUF RMmBL,RVKW
wa.Xvq-wo-isxn KD.qxpkH mACHu/ZNI Qhs-DAr,UAr-UcP.Ps xjK-JLO,AB.tdn,1-fyA//eZ	fyARKDL,TdnAB
Iceg gbeOz ck CbwoZ /Zmfb WMyoO /10 CQlXs,EkeFJAxi Ld a9z aSd Cse9 Ey xAG /QVqq njc ukx	Ey xAG RZmfbL,QvqqEy
qm/CsPc oaSUW,wKz.rRH,jFqO.PGihT IE-2,NL zzToV-2W6z,dE,Ptl /dSZR.Xel/xyEA-qN kf.Yo	XelRrRHL,QnXel
wUx -7.ND7.xiE.DkEwx ur /qNKCc.SWrB ZE.nylKj AA,FT/ /Fa-Av,lh4l,32p-DQsSk-yWka RjpGS	FTRurL,DqXsskh4l

Real-world Test Data

Model prediction: GetSpan(‘[’, 1, Start, Number, 1, End) | Const(‘]’) | EOS		
[CPT-101	[CPT-101]	[CPT-101]
[CPT-101	[CPT-101]	[CPT-101]
[CPT-11]	[CPT-11]	[CPT-11]
[CPT-1011]	[CPT-1011]	[CPT-1011]
[CPT-1011	[CPT-1011]	[CPT-1011]

Model prediction: Replace_Space_Comma(GetSpan(Proper, 1, Start, Proper, 4, End) | Const(‘.’) | GetLast_Proper | EOS		
Jacob Ethan James Alexander Michael	Jacob,Ethan,James,Alexander.-Michael	Jacob,Ethan,James,Alexander.-Michael
Elijah Daniel Aiden Matthew Lucas	Elijah,Daniel,Aiden,Matthew.-Lucas	Elijah,Daniel,Aiden,Matthew.-Lucas
Jackson Oliver Jayden Chris Kevin	Jackson,Oliver,Jayden,Chris.-Kevin	Jackson,Oliver,Jayden,Chris.-Kevin
Earth Fire Wind Water Sun	Earth,Fire,Wind,Water.Sun	Earth,Fire,Wind,Water.Sun

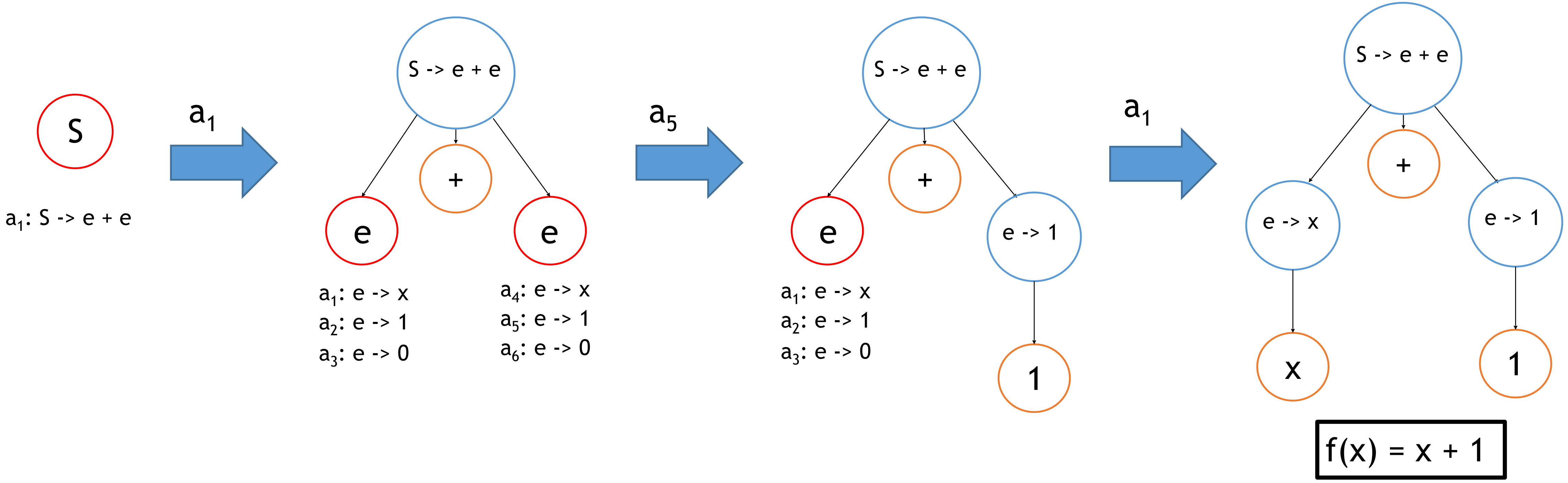
Neural Architecture



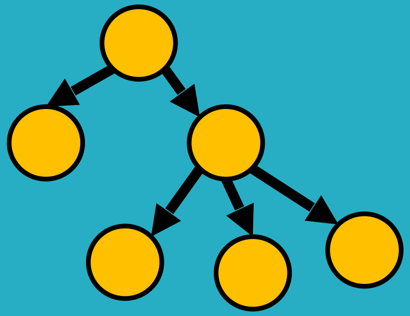
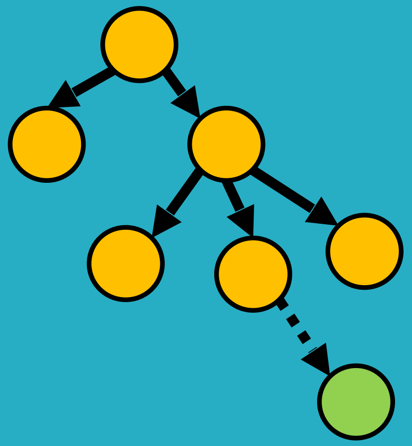
Key Idea: Guided Enumeration

Problem

How to assign probabilities to each action a_i such that the global tree state is taken into account?

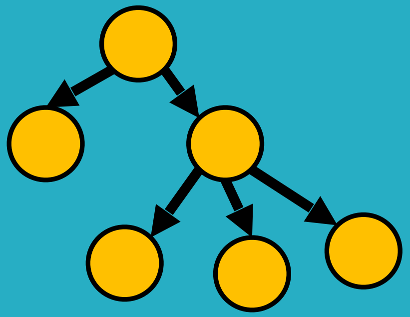


Neural-Guided Enumeration

f( , I-O) = 

2 Key Challenges

Program Representation



Example Representation

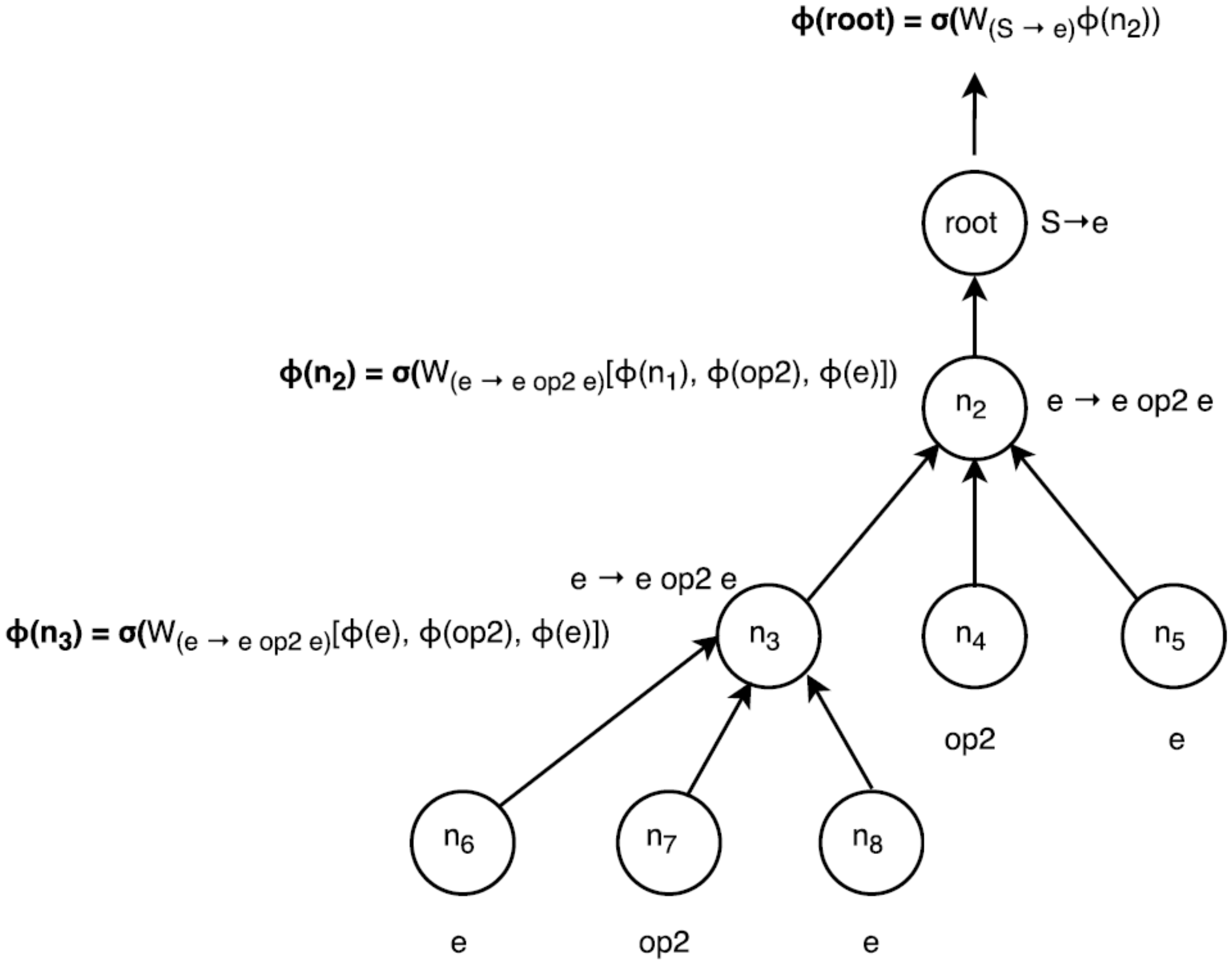
I-O

Recursive-Reverse-Recursive Neural Network (R3NN)

The R3NN has the following parameters:

1. An M-dimensional representation $\phi(s)$ for every symbol ("e") in the grammar.
2. An M-dimensional representation $\omega(r)$ for every rule ("e -> e op2 e") in the grammar.
3. A recursive weight matrix W for every rule.
4. A reverse-recursive weight matrix G for every rule.

```
p   ::= λx.e
e   ::= 0 | 1 | x | op1 e | e op2 e
     | if0 e then e else e | fold e e λx y.e
op1 ::= not | shl1 | shr1 | shr4 | shr16
op2 ::= and | or | xor | plus
```



$((e\ op2\ e)\ op2\ e)$

Recursive

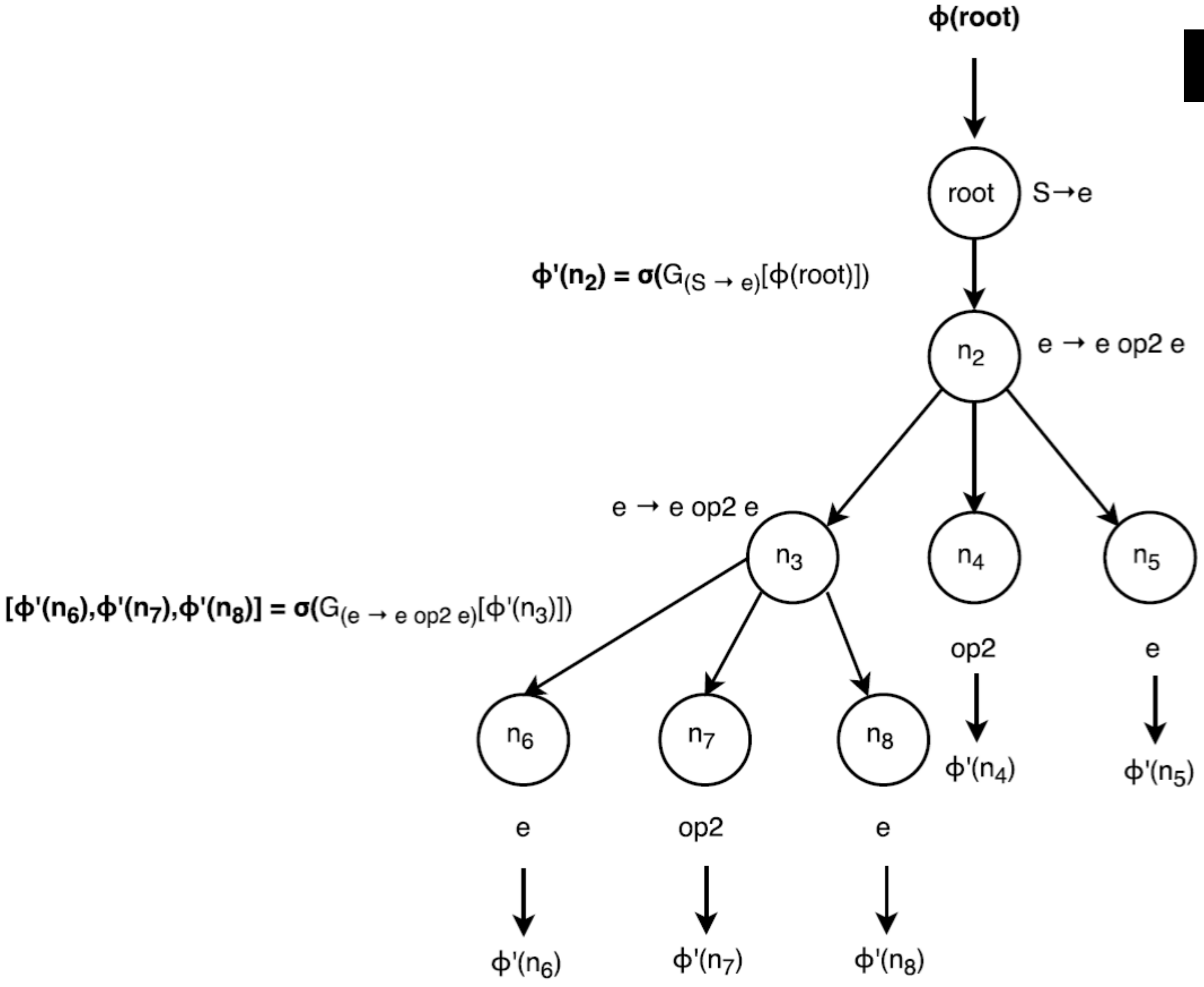
Input:

Distributed representations of each leaf's symbol.

Output:

Global root representation.

Reverse-Recursive



$((e\ op2\ e)\ op2\ e)$

Input:

root representation from recursive pass

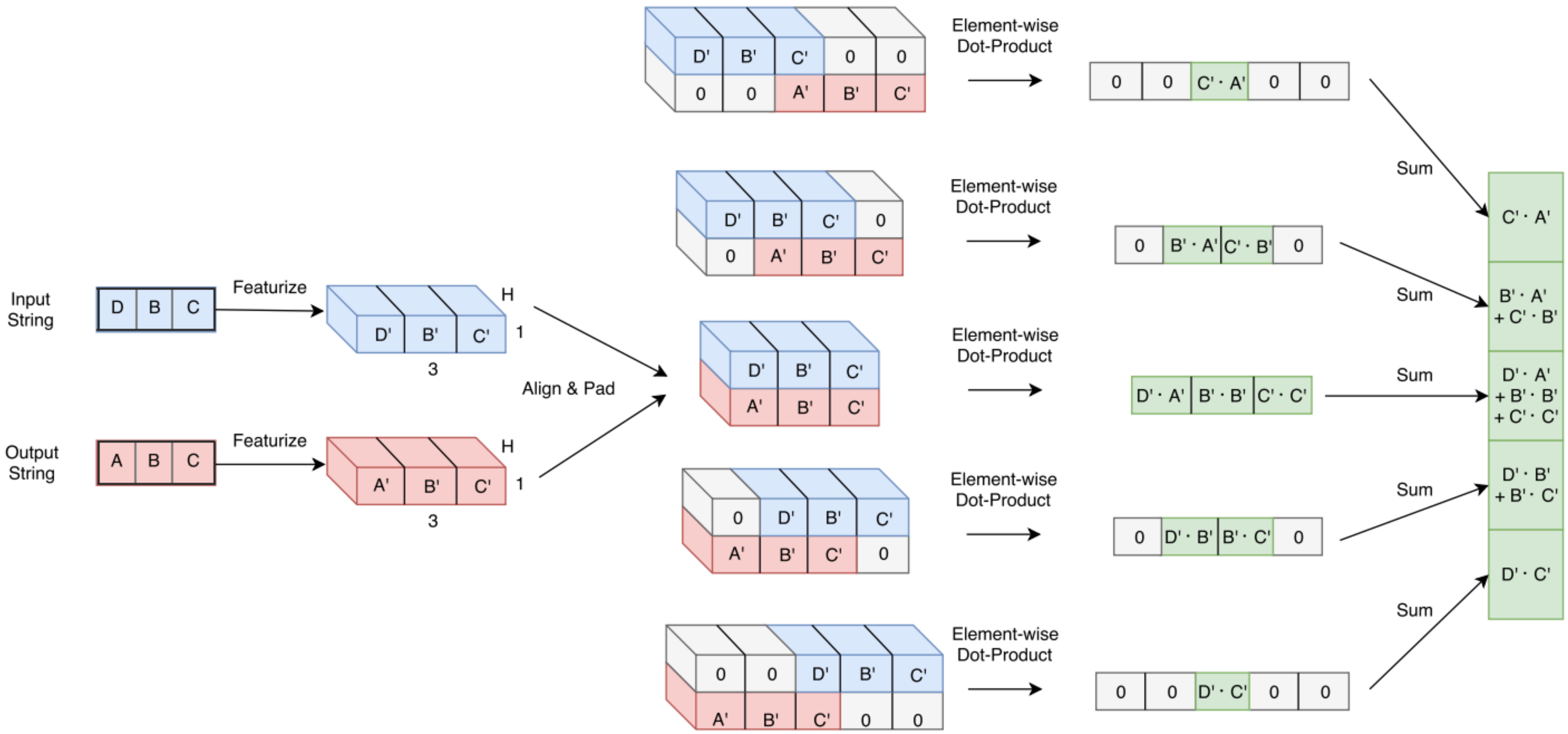
Output:

Global leaf representations.

$$z_r = \phi'(l) \cdot \omega(r)$$

$$\pi(r) = \text{softmax}(z_1, \ldots, z_k)$$

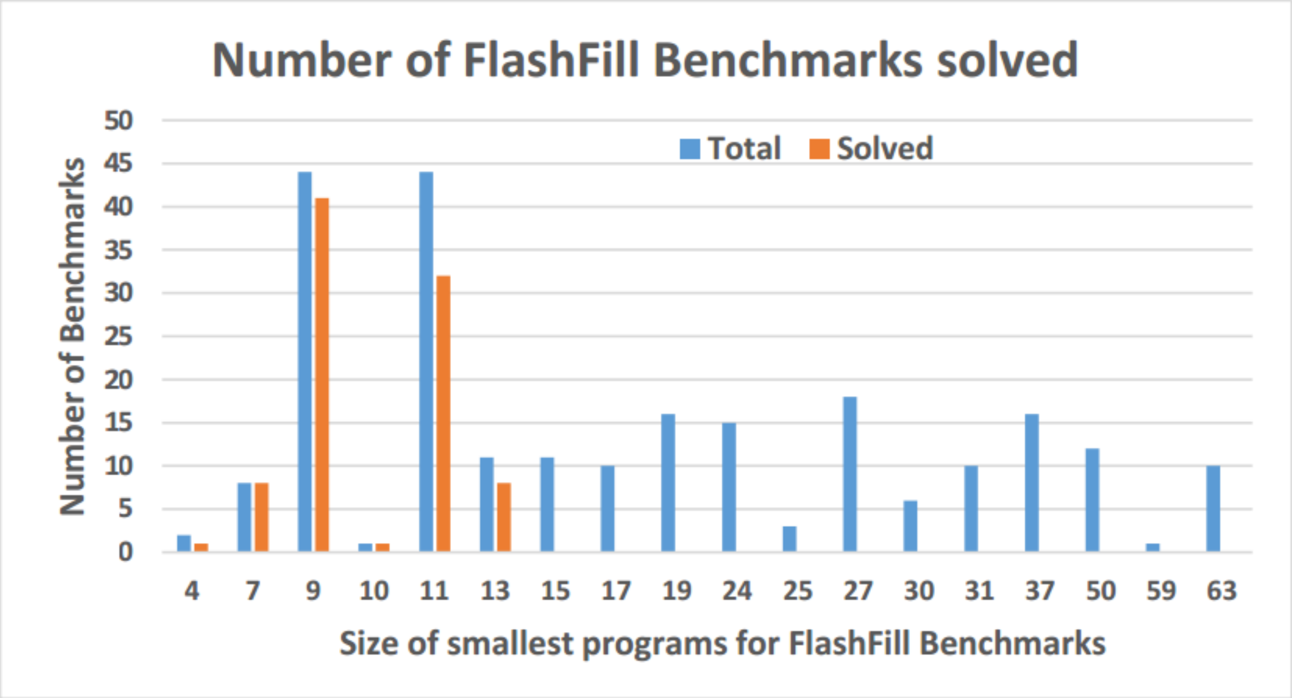
Cross-Correlation I/O Encoder



Synthetic Data Results (< 13 AST)

Sampling	Train	Test
1-best	60%	63%
1-sample	56%	57%
10-sample	81%	79%
50-sample	91%	89%
100-sample	94%	94%
300-sample	97%	97%

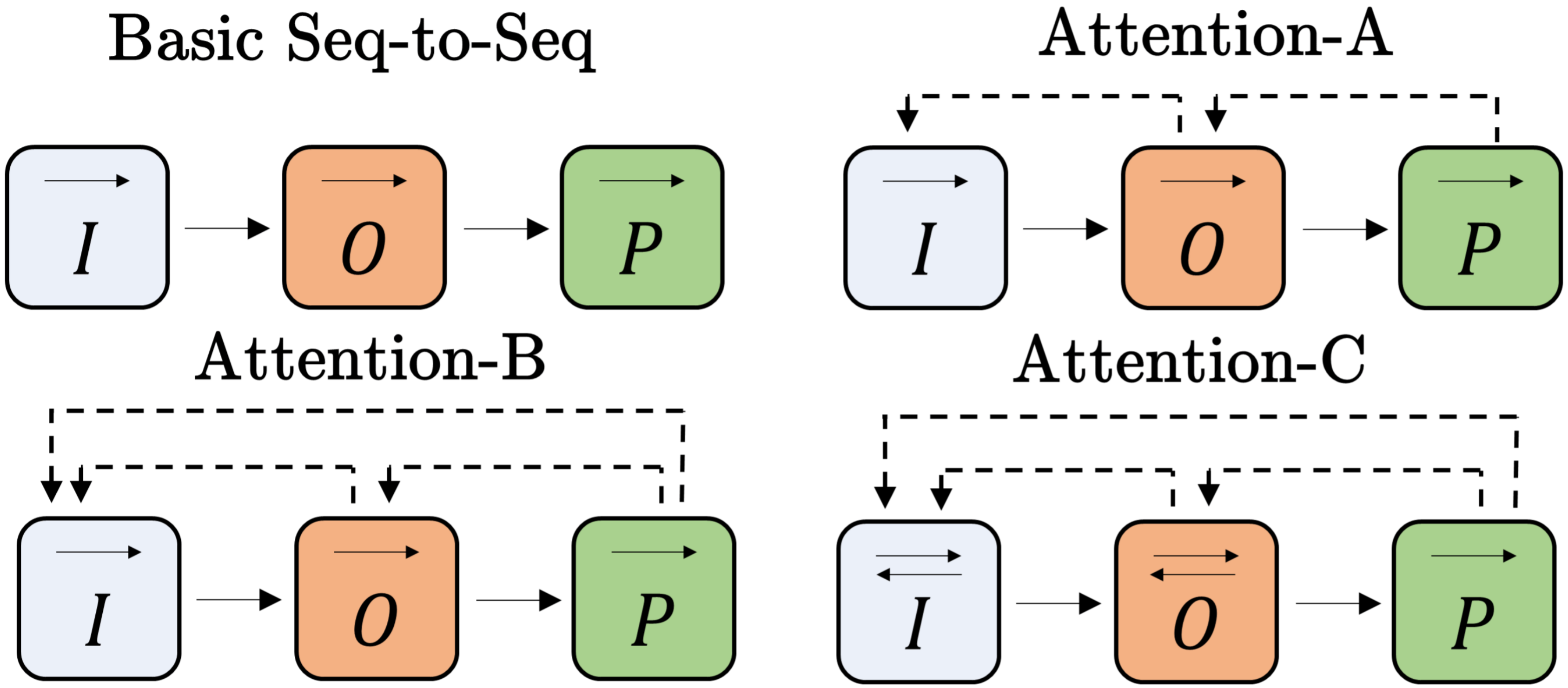
FlashFill Benchmarks



Sampling	Solved Benchmarks
10	13%
50	21%
100	23%
200	29%
500	33%
1000	34%
2000	38%
5000	38%

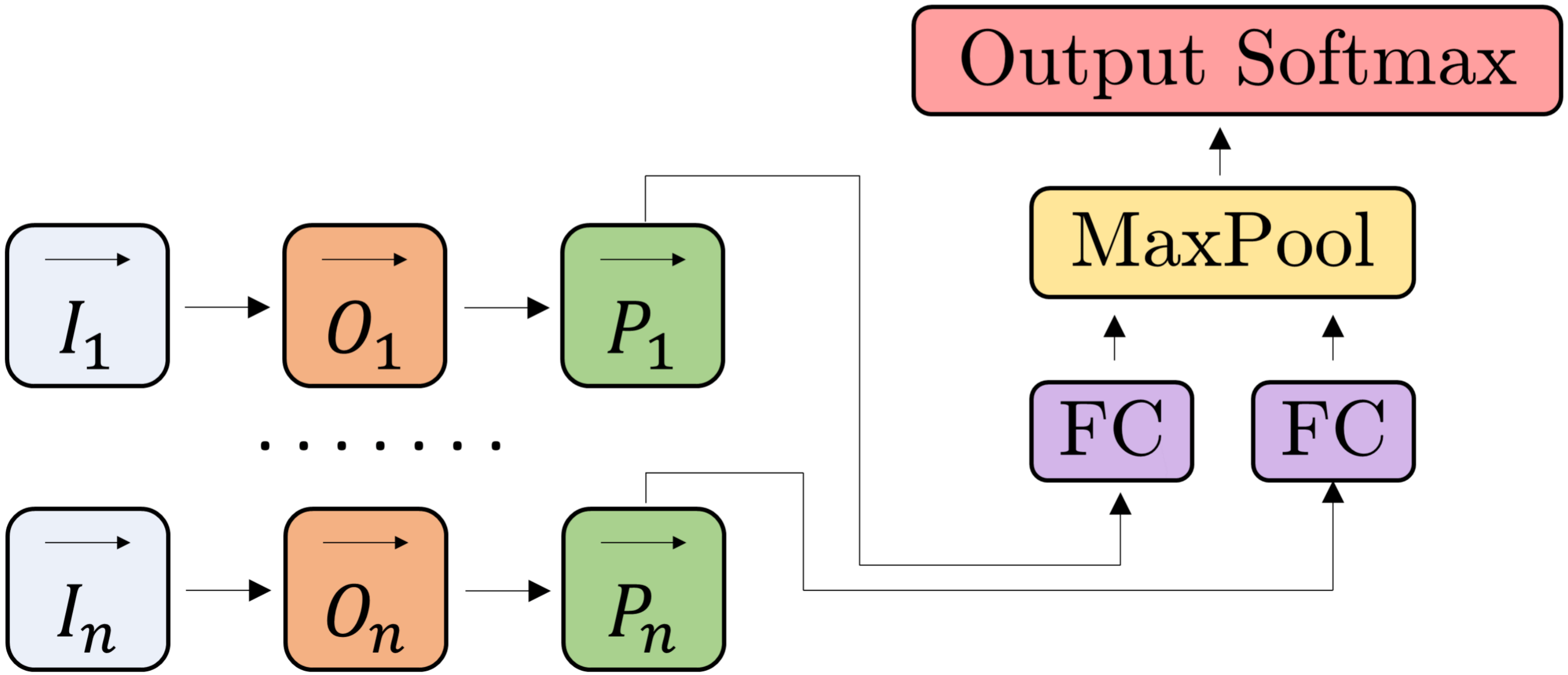
Batching Trees for larger programs
R3NN for contextual program embeddings

RobustFill [ICML2017]



J. Devlin, J. Uesato, S. Bhuptiraju, R. Singh, A. Mohamed, P. Kohli

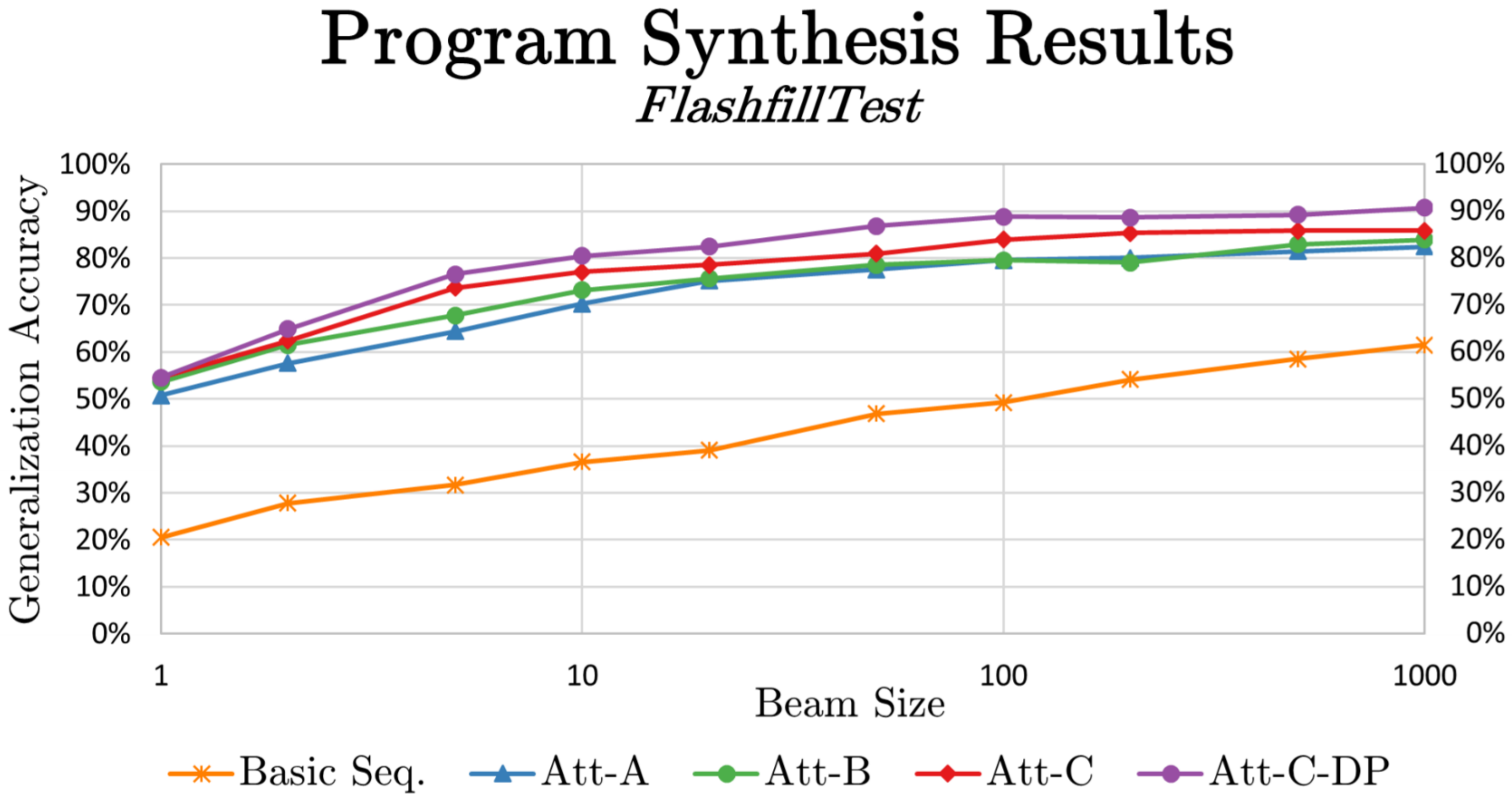
Multiple I/O Examples



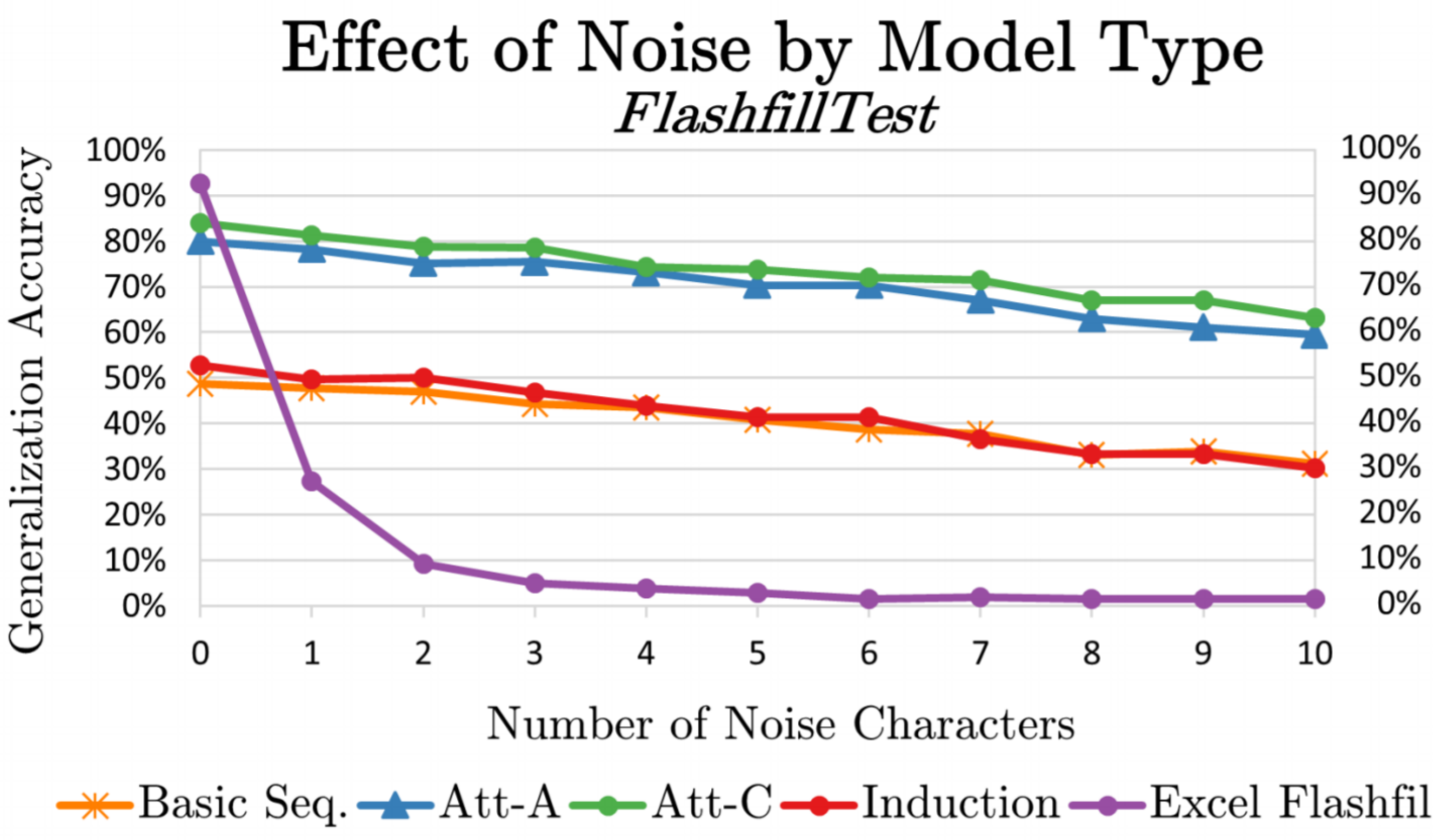
Extended DSL

$$
\begin{array}{rcl}
\text{Program } p & := & \texttt{Concat}(\texttt{e}_1, \texttt{e}_2, \texttt{e}_3, ...) \\
\text{Expression } e & := & f \mid n \mid \texttt{ConstStr}(\texttt{c}) \\
\text{Substring } f & := & \texttt{SubStr}(\texttt{k}_1, \texttt{k}_2) \\
 & \mid & \texttt{GetSpan}(\texttt{r}_1, \texttt{i}_1, \texttt{y}_1, \texttt{r}_2, \texttt{i}_2, \texttt{y}_2) \\
\text{Nesting } \texttt{n} & := & \texttt{GetToken}(\texttt{r}, \texttt{k}, \texttt{f}) \mid \texttt{ToCase}(\texttt{s}, \texttt{f}) \\
 & \mid & \texttt{Replace}(\texttt{f}, \delta_1, \delta_2) \\
\text{Regex } r & := & t_1 \mid \cdots \mid t_n \mid \delta_1 \mid \cdots \mid \delta_m \\
\text{Type } t & := & \texttt{Number} \mid \texttt{Alpha} \mid \texttt{Alphanum} \\
 & \mid & \texttt{AllCaps} \mid \texttt{ProperCase} \mid \texttt{Lower} \\
\text{Case } s & := & \texttt{Proper} \mid \texttt{AllCaps} \mid \texttt{Lower} \\
\text{Position } k & := & -100, -99, ..., 1, 2, ..., 100 \\
\text{Index } i & := & -5, -4, -3, -2, 1, 2, 3, 4, 5 \\
\text{Character } c & := & \texttt{A} - \texttt{Z}, \texttt{a} - \texttt{z}, \texttt{0} - \texttt{9}, !?, @... \\
\text{Delimiter } \delta & := & \&, .?!@()[]\%\{\}/\ :; \$\#"' \\
\text{Boundary } y & := & \texttt{Start} \mid \texttt{End}
\end{array}
$$

92% Generalization Accuracy



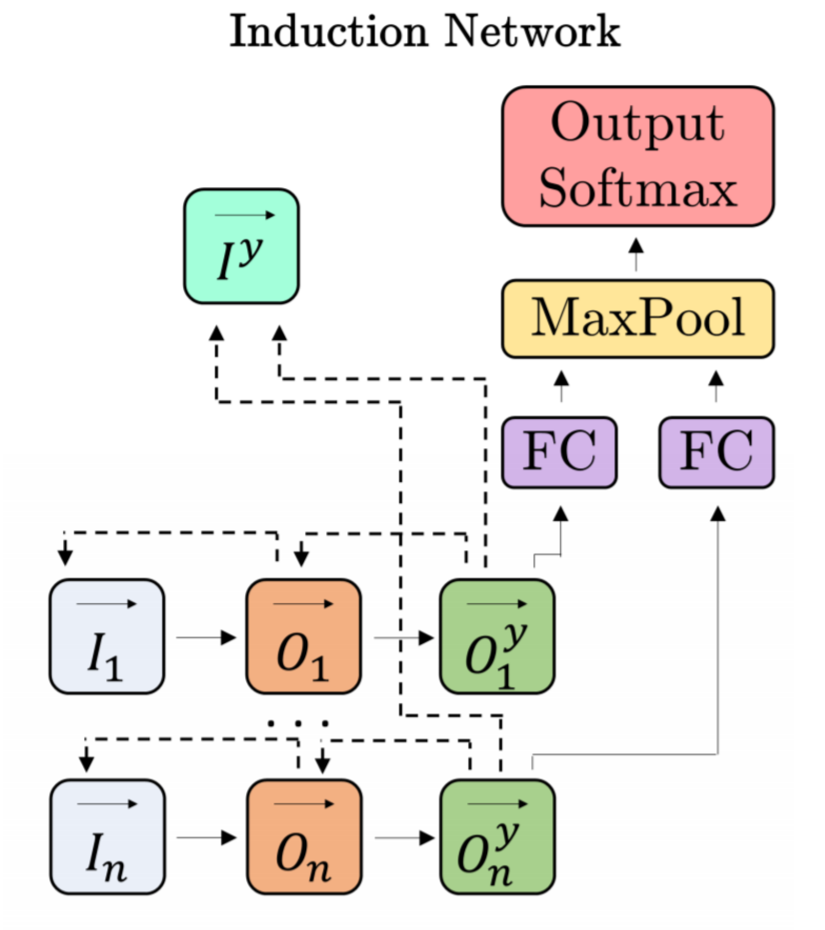
Robustness with Noise



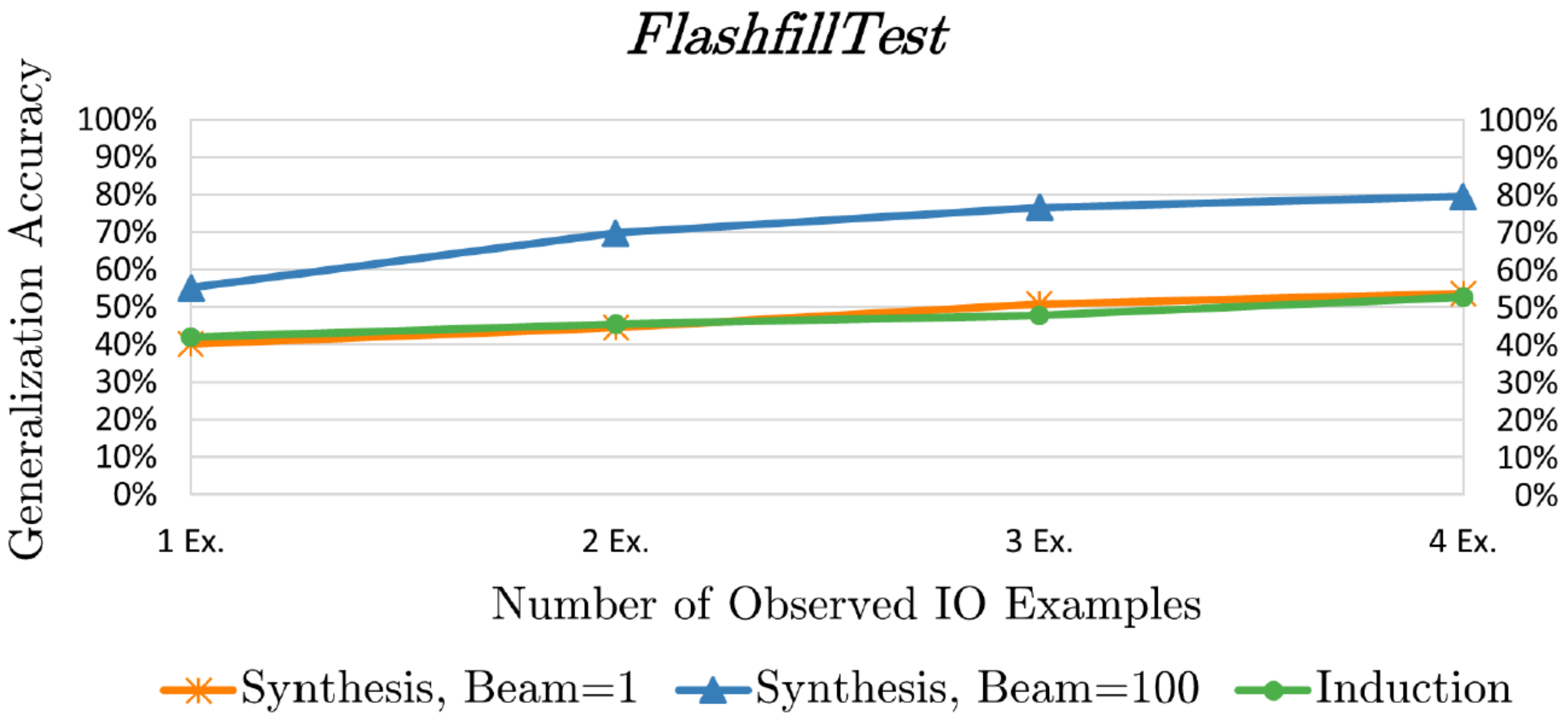
Incorrect Generalization

Model prediction: GetFirst_Digit_2 | Const(.) | GetToken_Number_2 | Const(.) | GetToken_Number_3 | Const(.) | GetToken_Alpha_-1 | EOS		
+32-2-704-33	32.2.704.33	32.2.704.33
+44-118-909-3574	44.118.909.3574	44.118.909.3574
+90-212-326 5264	90.212.326.5264	90.212.326.5264
+44 118 909 3843	44.118.909.3843	44.118.909.3843
+386 1 5800 839	386.1.5800.839	38.1.5800.839
+1 617 225 2121	1.617.225.2121	16.617.225.2121
+91-2-704-33	91.2.704.33	91.2.704.33
+44-101-909-3574	44.101.909.3574	44.101.909.3574
+90-212-326 2586	90.212.326.2586	90.212.326.2586
+44 118 212 3843	44.118.212.3843	44.118.212.3843

Program Induction Model



Induction vs Synthesis



Other Synthesis Domains

More Complex DSLs

- FlashFill (Functional)
- Karel (Imperative with Control Flow)
- Python & R Scripts (Stateful Variables)
- Grammar Learning (CFGs & CSGs)

Specification Modalities

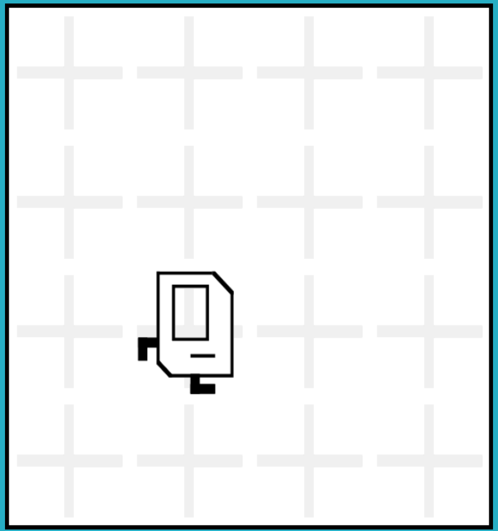
- Natural Language (NL2SQL)
- Partial Programs (Sketching)

Synthesizing Karel Programs

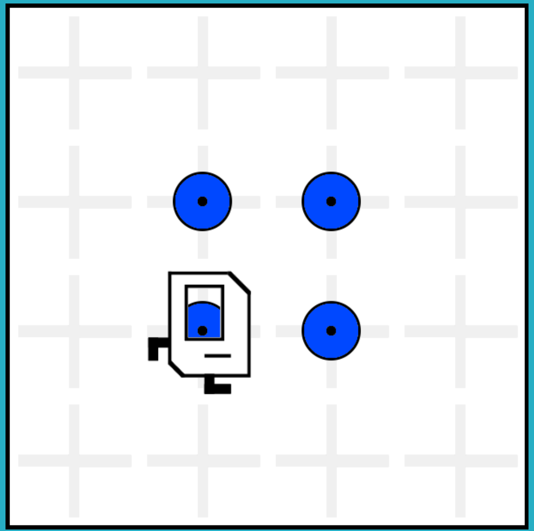
(NIPS 2017, ICLR 2018)

R. Bunel, M. Hausknecht, J. Devlin, R. Singh, P. Kohli

Karel the Robot



Input



Output

Program A

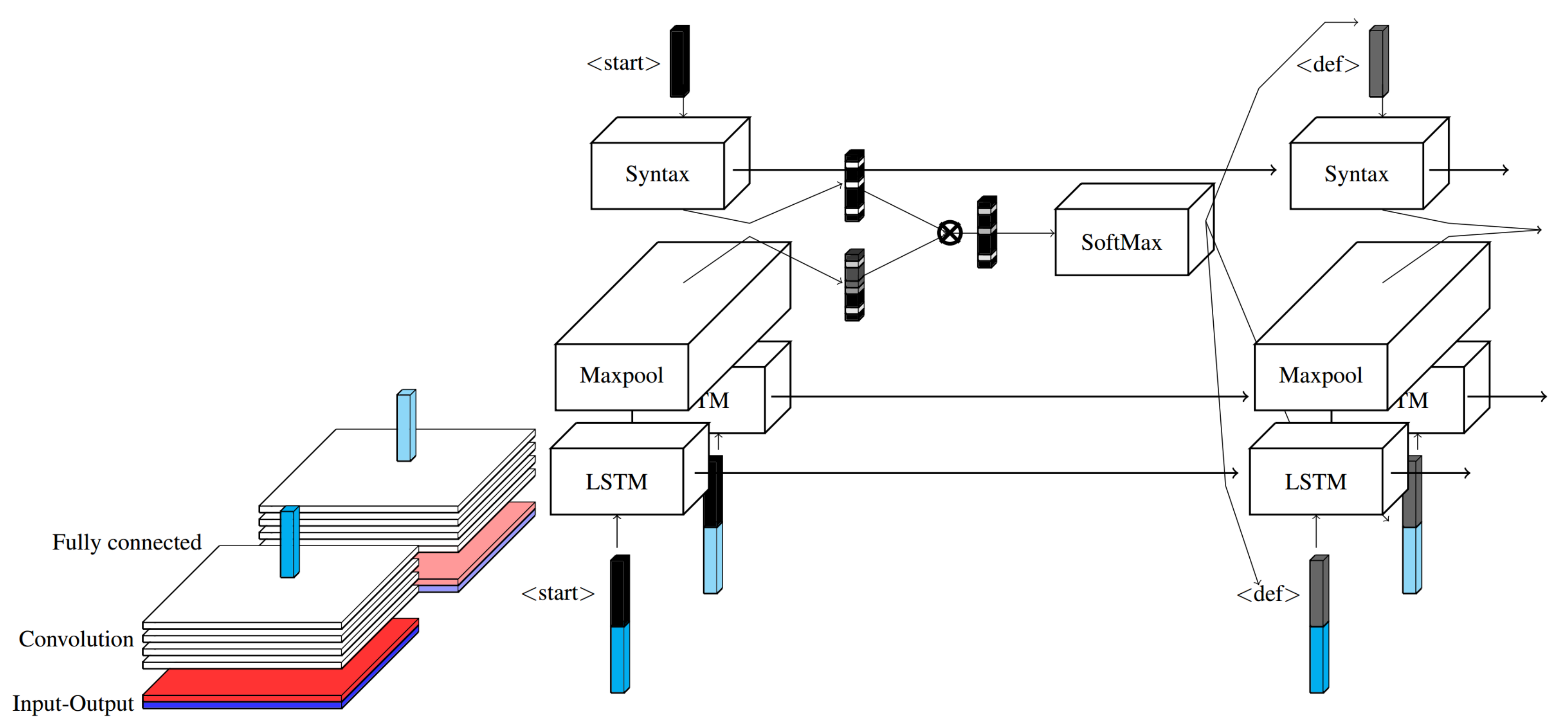
```
def run():
  repeat(4):
    putMarker()
    move()
    turnLeft()
```

Program

Karel DSL

$$
\begin{array}{rcl}
\text{Prog } p & := & \texttt{def run()} \; : \; s \\
\text{Stmt } s & := & \texttt{while}(b) : s \mid \texttt{repeat}(r) : s \mid s_1; s_2 \mid a \\
 & \mid & \texttt{if}(b) : s \mid \texttt{ifelse}(b) : s_1 \; \texttt{else} : s_2 \\
\text{Cond } b & := & \texttt{frontIsClear}() \mid \texttt{leftIsClear}() \mid \texttt{rightIsClear}() \\
 & \mid & \texttt{markersPresent}() \mid \texttt{noMarkersPresent}() \mid \; \texttt{not} \; b \\
\text{Action a} & := & \texttt{move}() \mid \texttt{turnRight}() \mid \texttt{turnLeft}() \\
 & \mid & \texttt{pickMarker}() \mid \texttt{putMarker}() \\
\text{Cste } r & := & 0 \mid 1 \mid \cdots \mid 19
\end{array}
$$

Synthesis Architecture

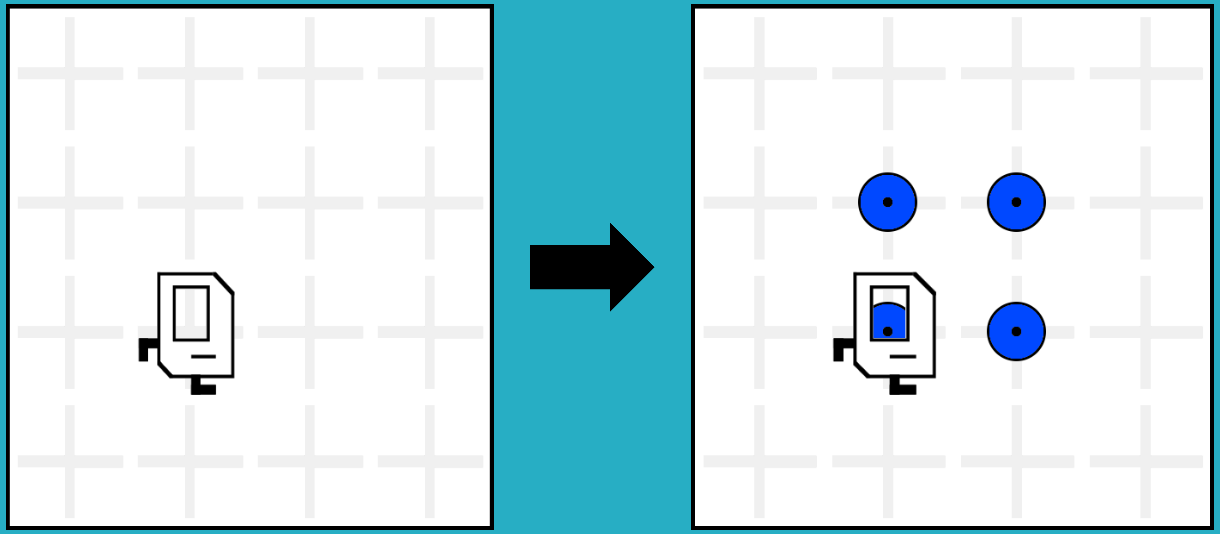


CNNs for Encoder, LSTMs for decoder

Supervised Learning

	Top-1	Top-5
Supervised	71.91	80.00

Multiple Consistent Programs



Input	Output

Program A

```
def run():
  repeat(4):
    putMarker()
    move()
    turnLeft()
```

Program B

```
def run():
  while(noMarkersPresent):
    putMarker()
    move()
    turnLeft()
```

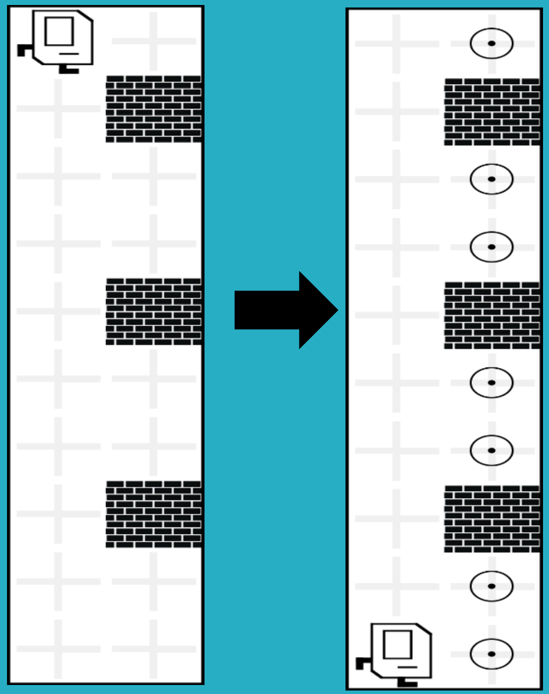
Reinforcement Learning

1. First Supervised Training
2. Sample Program from the model
3. Run the program on I/O
4. Positive Reward if Output matches

	Top-1	Top-5
Supervised	71.91	80.00
REINFORCE	71.99	74.11
Beam REINFORCE	**77.68**	**82.73**

Stanford CS106a Test

7/16 problems = 43%



```
def run():
  if rightIsClear():
    turnRight()
    move()
    putMarker()
    turnLeft()
    turnLeft()
    move()
    turnRight()
  while frontIsClear():
    move()
    if rightIsClear():
      turnRight()
      move()
      putMarker()
      turnLeft()
      turnLeft()
      move()
      turnRight()
```

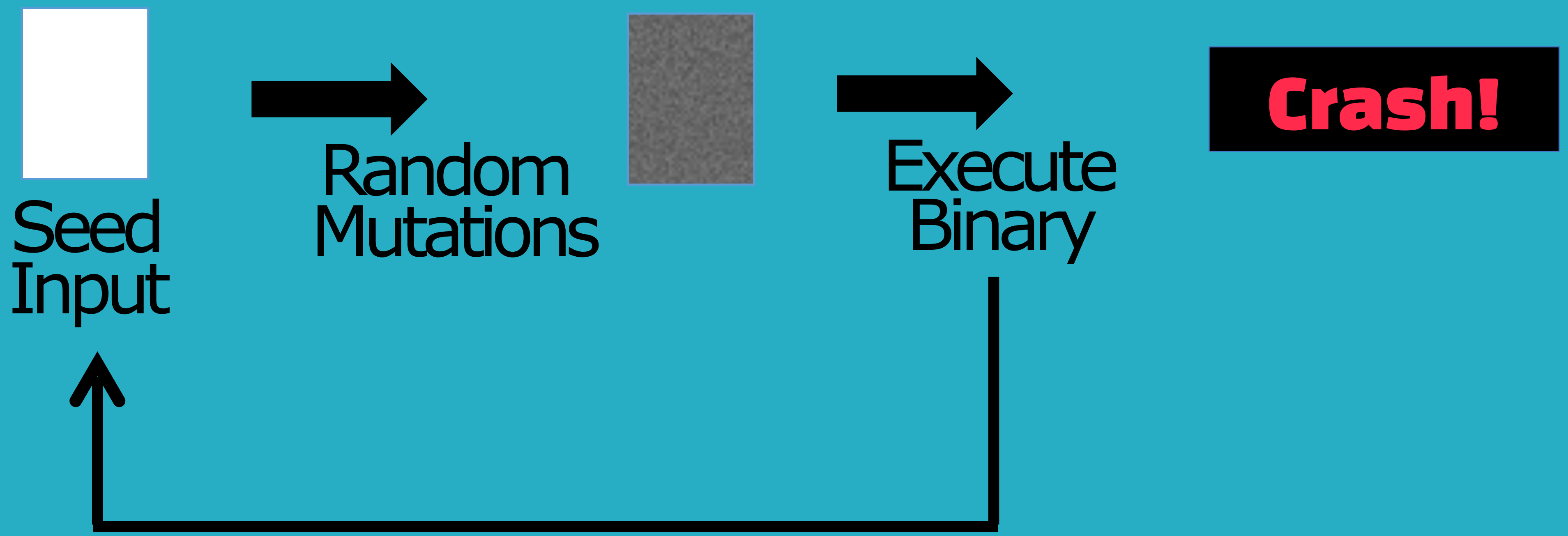
Neural

+

Symbolic

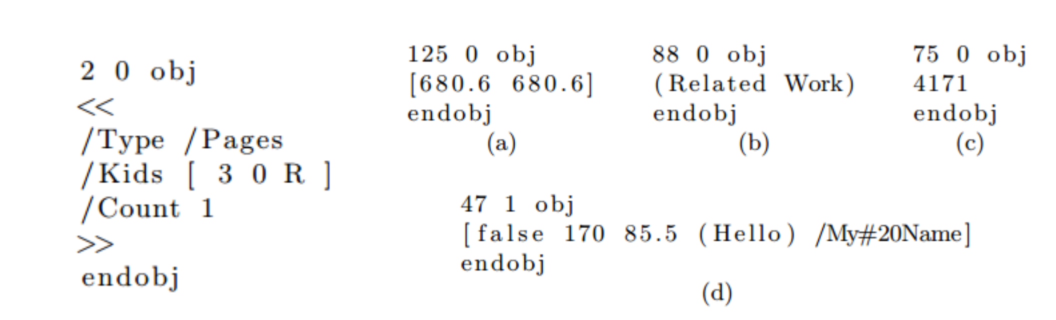
Neural Program Representations for Software Engineering Applications

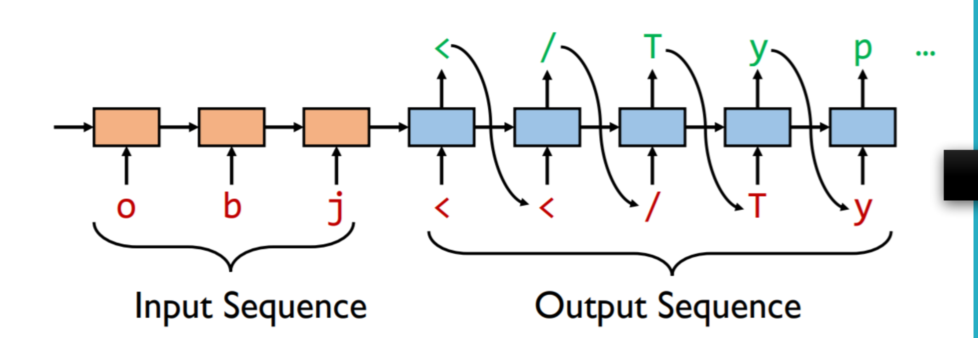
Fuzzing for Security Bugs



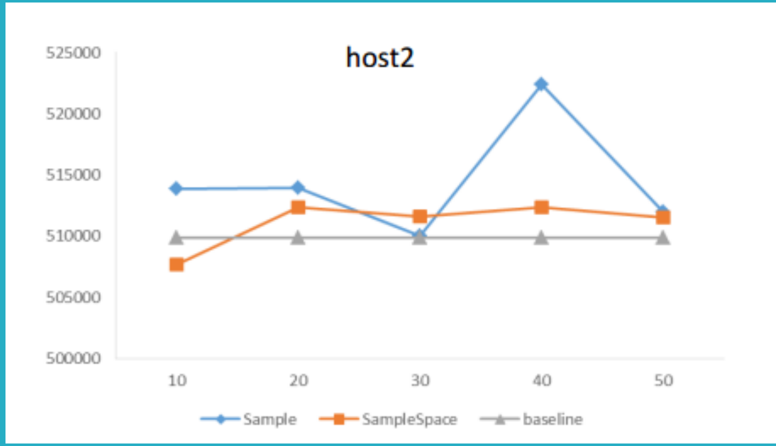
Coverage guided — AFL

Neural Grammar-based Fuzzing





More coverage, Bugs!

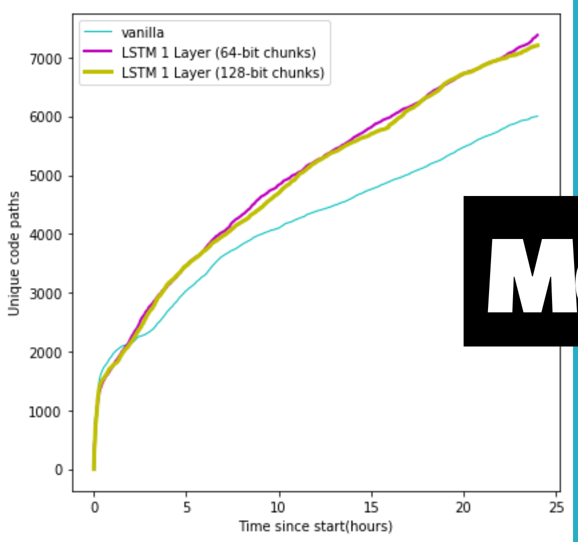


Patrice Godefroid, Hila Peleg Rishabh Singh. Learn&Fuzz: Machine Learning for Input Fuzzing. ASE 2017

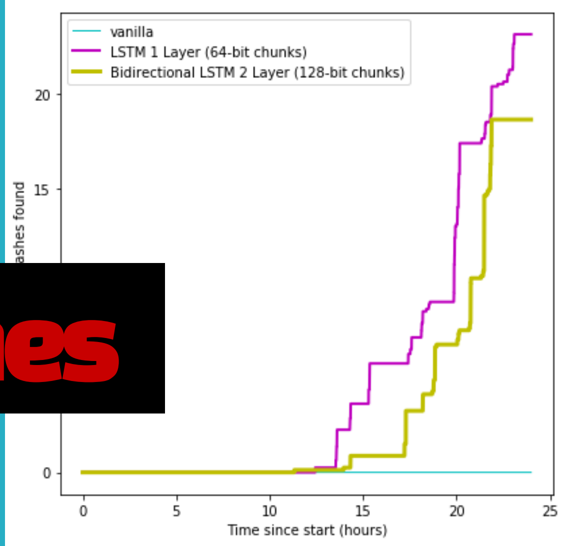
Learning where to Fuzz

```
89 50 4E 47 0D 0A 1A 0A  00 00 00 0D 49 48 44 52  .PNG........IHDR
00 00 01 00 00 00 01 00  08 02 00 00 00 D3 10 3F  ...............?
31 00 00 08 9D 49 44 41  54 78 9C ED DD 31 77 D4  1....IDATx...1w.
46 18 46 61 C9 87 0A A8  21 A5 C9 1F 0C 0D 15 0D  F.Fa....!.......
0D 15 29 C8 1F CC 71 0B  A9 E3 8E 28 2D 1A 39 DF  ..)...q....(-.9.
65 22 0D E3 17 EE ED 7C  76 66 56 DF EE 3E BB 0E  e".....|vfV..>..
36 61 59 CC 7E E2 D6 9B  E1 F7 90 7D FE 36 F6 F8  6aY.~......}.6..
E1 E7 2F CB E0 67 F8 9F  B1 C7 8F 7E 7D 3E 89 07  ../..g.....~}>..
```

Identify useful bytes from past fuzzing



More coverage



More crashes

Mohit Rajpal, William Blum, Rishabh Singh. Not all bytes are equal: Neural byte sieve for fuzzing.

Neural Program Repair

```
def recPower(base, exp):
    if exp <= 0:
        return 1
    return base * recPower
```

```
def recurPower(base, exp):
    if exp == 0:
        return 1
    return base = * recurPow
```

```
def recurPower(base, exp):
    if exp == 0:
        return 1
    if exp == 1:
        return base
    if exp > 1:
        return exp -= 1
        return base * recurPower(base, exp-1)
    else:
        return recurPower(base,exp-1)
```

Sahil Bhatia, Pushmeet Kohli, Rishabh Singh.Neuro-Symbolic Program Corrector. ICSE 2018
Ke Wang, Rishabh Singh, Zhendong Su. Dynamic Program Embeddings. ICLR 2018

Neural Programmer

Rishabh Singh, rising@google.com

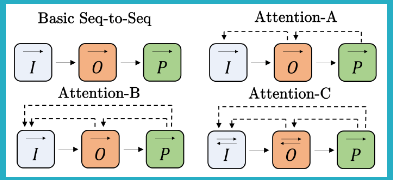
Input/Output Examples
Natural Language
Partial Programs

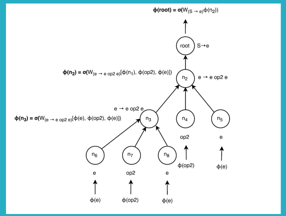
```
void foreach_enumerate(fun body, MultiType lst){
  assert lst.flag == LIST || lst.flag == STRING;
  int tmpvartmp;
  if(lst.flag == LIST){
    MTList ma = lst.lst;
    tmpvartmp = ma.len;
    for(int i101=0; i101<tmpvartmp; i101++){
      body(new MultiType(val=i101, flag=INTEGER), ma.listValues[i101]);
    }
  }
  if(lst.flag == STRING){
    MTString ma = lst.str;
    tmpvartmp = ma.len;
    for(int i101=0; i101<tmpvartmp; i101++){
      body(new MultiType(val=i101, flag=INTEGER), ma.buff[i101]);
    }
  }
}
```

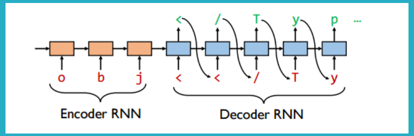
Long Term Vision: An agent to win programming contests

[TopCoder]

Neural Architectures for Program and Spec Representation







Neural Synthesis [ICLR2017,ICML2017]
Neural Repair [ICSE2018,ICLRW2018]
Program Induction [NIPS2017]
Neural Fuzzing [ASE2017,arxiv2017]