Teach Neural Networks to Learn Rules

An Overview on Neural Program Synthesis and Neural Program Induction

---

Shao-Hua Sun

Graduating Ph.D. candidate in Computer Science
at University of Southern California (USC)

&

Incoming Assistant Professor in Electrical Engineering
at National Taiwan University (NTU)

Outline

- Recent Advance in Deep Learning
- Teach Neural Networks to Learn Rules
  - An Overview
  - Neural Program Induction
  - Neural Program Synthesis

Outline

- Recent Advance in Deep Learning
- Teach Neural Networks to Learn Rules
  - An Overview
  - Neural Program Induction
  - Neural Program Synthesis

Image Classification

Model	image size	# parameters	Mult-Adds	Top 1 Acc. (%)	Top 5 Acc. (%)
Inception V2 [29]	224×224	11.2 M	1.94 B	74.8	92.2
NASNet-A (5 @ 1538)	**299×299**	**10.9 M**	**2.35 B**	**78.6**	**94.2**
Inception V3 [60]	299×299	23.8 M	5.72 B	78.8	94.4
Xception [9]	299×299	22.8 M	8.38 B	79.0	94.5
Inception ResNet V2 [58]	299×299	55.8 M	13.2 B	80.1	95.1
NASNet-A (7 @ 1920)	**299×299**	**22.6 M**	**4.93 B**	**80.8**	**95.3**
ResNeXt-101 (64 x 4d) [68]	320×320	83.6 M	31.5 B	80.9	95.6
PolyNet [69]	331×331	92 M	34.7 B	81.3	95.8
DPN-131 [8]	320×320	79.5 M	32.0 B	81.5	95.8
SENet [25]	**320×320**	**145.8 M**	**42.3 B**	**82.7**	**96.2**
NASNet-A (6 @ 4032)	**331×331**	**88.9 M**	**23.8 B**	**82.7**	**96.2**

Table 2. Performance of architecture search and other published state-of-the-art models on ImageNet classification. Mult-Adds indicate the number of composite multiply-accumulate operations for a single image. Note that the composite multiple-accumulate operations are calculated for the image size reported in the table. Model size for [25] calculated from open-source implementation.

Zoph *et al.* Learning Transferable Architectures for Scalable Image Recognition

Instance Segmentation

Figure 5. More results of **Mask R-CNN** on COCO test images, using ResNet-101-FPN and running at 5 fps, with 35.7 mask AP (Table 1).

He *et al.* Mask R-CNN

Visual Question Answering

What is on the coffee table ?

candles

What color is the hydrant ?

black and yellow

What is on the bed ?

books

What is the long stick for ?

whipping

Method	VQA v2 test-dev All	Yes/no	Numb.	Other	VQA v2 test-std All	Yes/no	Numb.	Other
Prior (most common answer in training set) [14]	–	–	–	–	25.98	61.20	0.36	1.17
LSTM Language only (blind model) [14]	–	–	–	–	44.26	67.01	31.55	27.37
Deeper LSTM Q norm. I [**?**] as reported in [14]	–	–	–	–	54.22	73.46	35.18	41.83
MCB [13] as reported in [14]	–	–	–	–	62.27	78.82	38.28	53.36
UPMC-LIP6 [7]	–	–	–	–	65.71	82.07	41.06	57.12
Athena	–	–	–	–	67.59	82.50	44.19	59.97
LV-NUS	–	–	–	–	66.77	81.89	46.29	58.30
HDU-USYD-UNCC	–	–	–	–	68.09	84.50	45.39	59.01
Proposed model								
ResNet features 7×7, single network	62.07	79.20	39.46	52.62	62.27	79.32	39.77	52.59
Image features from bottom-up attention, adaptive K, single network	65.32	81.82	44.21	56.05	65.67	82.20	43.90	56.26
ResNet features 7×7, ensemble	66.34	83.38	43.17	57.10	66.73	83.71	43.77	57.20
Image features from bottom-up attention, adaptive K, ensemble	**69.87**	**86.08**	**48.99**	**60.80**	**70.34**	**86.60**	**48.64**	**61.15**

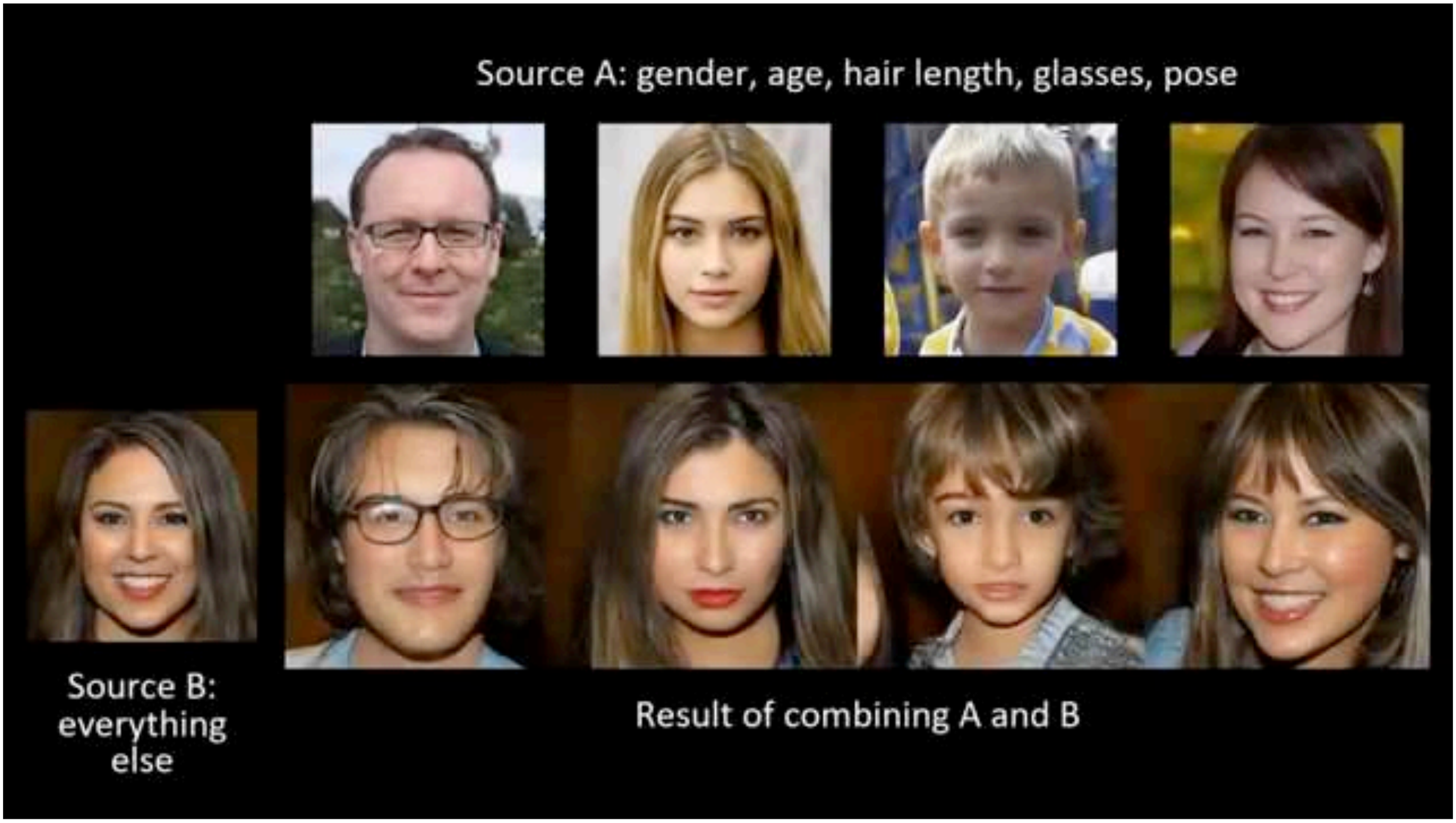
Table 3. Comparison of our best model with competing methods. Excerpt from the official *VQA v2 Leaderboard* [1].

Teney *et al.* Tips and Tricks for Visual Question Answering: Learnings from the 2017 Challenge

Face Image Generation

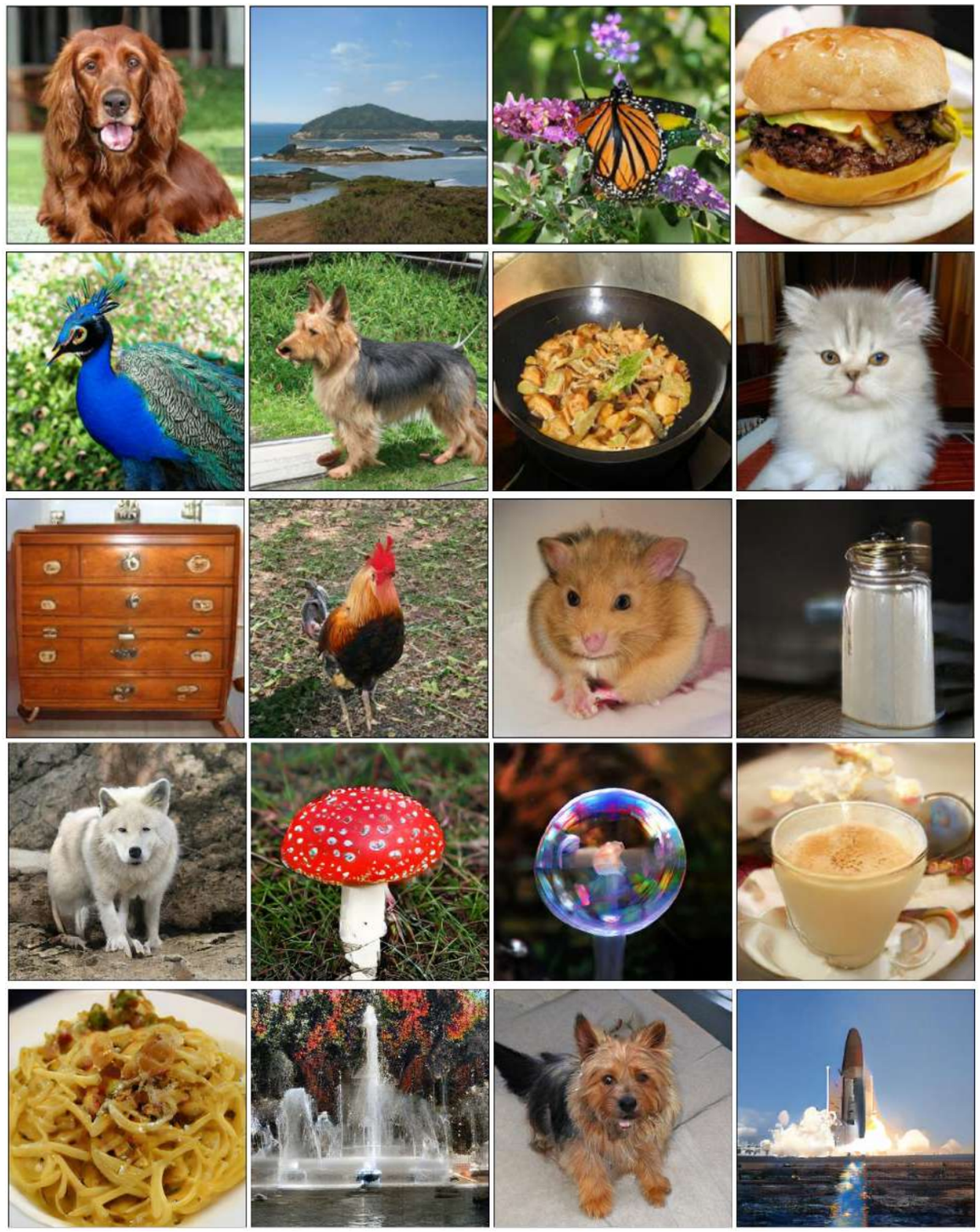
Kerras *et al.* Progressive Growing of GANs for Improved Quality, Stability, and Variation

Face Image Generation with Style



Kerras *et al.* A Style-Based Generator Architecture for Generative Adversarial Networks

Natural Image Generation



Brock *et al.* Large Scale GAN Training for High Fidelity Natural Image Synthesis

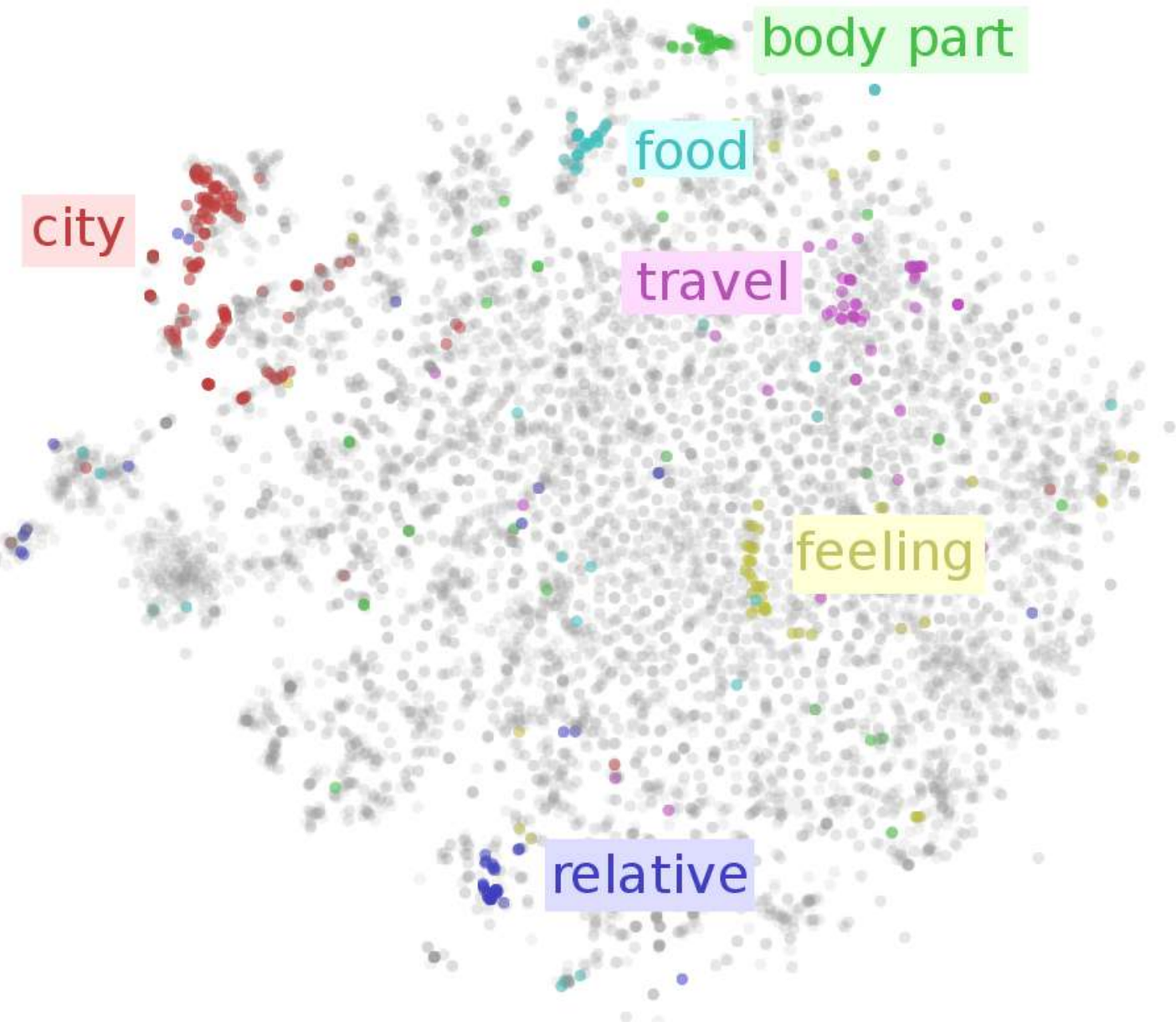
Natural Image Generation

generated image

generated image

Brock *et al.* Large Scale GAN Training for High Fidelity Natural Image Synthesis

Word Embeddings



On word embeddings - Part 1 by Ruder

Machine Translation

Source	"The reason Boeing are doing this is to cram more seats in to make their plane more competitive with our products," said Kevin Keniston, head of passenger comfort at Europe's Airbus.	
PBMT	"La raison pour laquelle Boeing sont en train de faire, c'est de concentrer davantage de sièges pour prendre leur avion plus compétitive avec nos produits", a déclaré Kevin M. Keniston, chef du confort des passagers de l'Airbus de l'Europe.	3.0
GNMT	"La raison pour laquelle Boeing fait cela est de créer plus de sièges pour rendre son avion plus compétitif avec nos produits", a déclaré Kevin Keniston, chef du confort des passagers chez Airbus.	6.0
Human	"Boeing fait ça pour pouvoir caser plus de sièges et rendre ses avions plus compétitifs par rapports à nos produits", a déclaré Kevin Keniston, directeur de Confort Passager chez l'avionneur européen Airbus.	6.0
Source	When asked about this, an official of the American administration replied: "The United States is not conducting electronic surveillance aimed at offices of the World Bank and IMF in Washington."	
PBMT	Interrogé à ce sujet, un responsable de l'administration américaine a répondu : "Les Etats-Unis n'est pas effectuer une surveillance électronique destiné aux bureaux de la Banque mondiale et du FMI à Washington".	3.0
GNMT	Interrogé à ce sujet, un fonctionnaire de l'administration américaine a répondu: "Les États-Unis n'effectuent pas de surveillance électronique à l'intention des bureaux de la Banque mondiale et du FMI à Washington".	6.0
Human	Interrogé sur le sujet, un responsable de l'administration américaine a répondu: "les Etats-Unis ne mènent pas de surveillance électronique visant les sièges de la Banque mondiale et du FMI à Washington".	6.0

Wu *et al.* Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation

Named Entity Recognition

contentSkip to site indexPoliticsSubscribeLog InSubscribeLog InToday's PaperAdvertisementSupported ORG byF.B.I. Agent Peter Strzok PERSON , Who Criticized Trump PERSON in Texts, Is FiredImagePeter Strzok, a top F.B.I. GPE counterintelligence agent who was taken off the special counsel investigation after his disparaging texts about President Trump PERSON were uncovered, was fired. CreditT.J. Kirkpatrick PERSON for The New York TimesBy Adam Goldman ORG and Michael S. SchmidtAug PERSON . 13 CARDINAL , 2018WASHINGTON CARDINAL — Peter Strzok PERSON , the F.B.I. GPE senior counterintelligence agent who disparaged President Trump PERSON in inflammatory text messages and helped oversee the Hillary Clinton PERSON email and Russia GPE investigations, has been fired for violating bureau policies, Mr. Strzok PERSON 's lawyer said Monday DATE .Mr. Trump and his allies seized on the texts — exchanged during the 2016 DATE campaign with a former F.B.I. GPE lawyer, Lisa Page — in PERSON assailing the Russia GPE investigation as an illegitimate "witch hunt." Mr. Strzok PERSON , who rose over 20 years DATE at the F.B.I. GPE to become one of its most experienced counterintelligence agents, was a key figure in the early months DATE of the inquiry.Along with writing the texts, Mr. Strzok PERSON was accused of sending a highly sensitive search warrant to his personal email account.The F.B.I. GPE had been under immense political pressure by Mr. Trump PERSON to dismiss Mr. Strzok PERSON , who was removed last summer DATE from the staff of the special counsel, Robert S. Mueller III PERSON . The president has repeatedly denounced Mr. Strzok PERSON in posts on

Named Entity Recognition and Classification with Scikit-Learn by Susan Li

Esteves *et al.* Named Entity Recognition in Twitter using Images and Text

Question Answering

- Input Question:

  `Where do water droplets collide with ice crystals to form precipitation?`

- Input Paragraph:

  `... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals within a cloud. ...`

- Output Answer:

  `within a cloud`

System	Dev EM	Dev F1	Test EM	Test F1
Leaderboard (Oct 8th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
#1 Single - nlnet	-	-	83.5	90.1
#2 Single - QANet	-	-	82.5	89.3
Published				
BiDAF+ELMo (Single)	-	85.8	-	-
R.M. Reader (Single)	78.9	86.3	79.5	86.6
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
$\text{BERT}_{\text{BASE}}$ (Single)	80.8	88.5	-	-
$\text{BERT}_{\text{LARGE}}$ (Single)	84.1	90.9	-	-
$\text{BERT}_{\text{LARGE}}$ (Ensemble)	85.8	91.8	-	-
$\text{BERT}_{\text{LARGE}}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
$\text{BERT}_{\text{LARGE}}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**

Table 2: SQuAD results. The BERT ensemble is 7x systems which use different pre-training checkpoints and fine-tuning seeds.

Devlin *et al.* BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

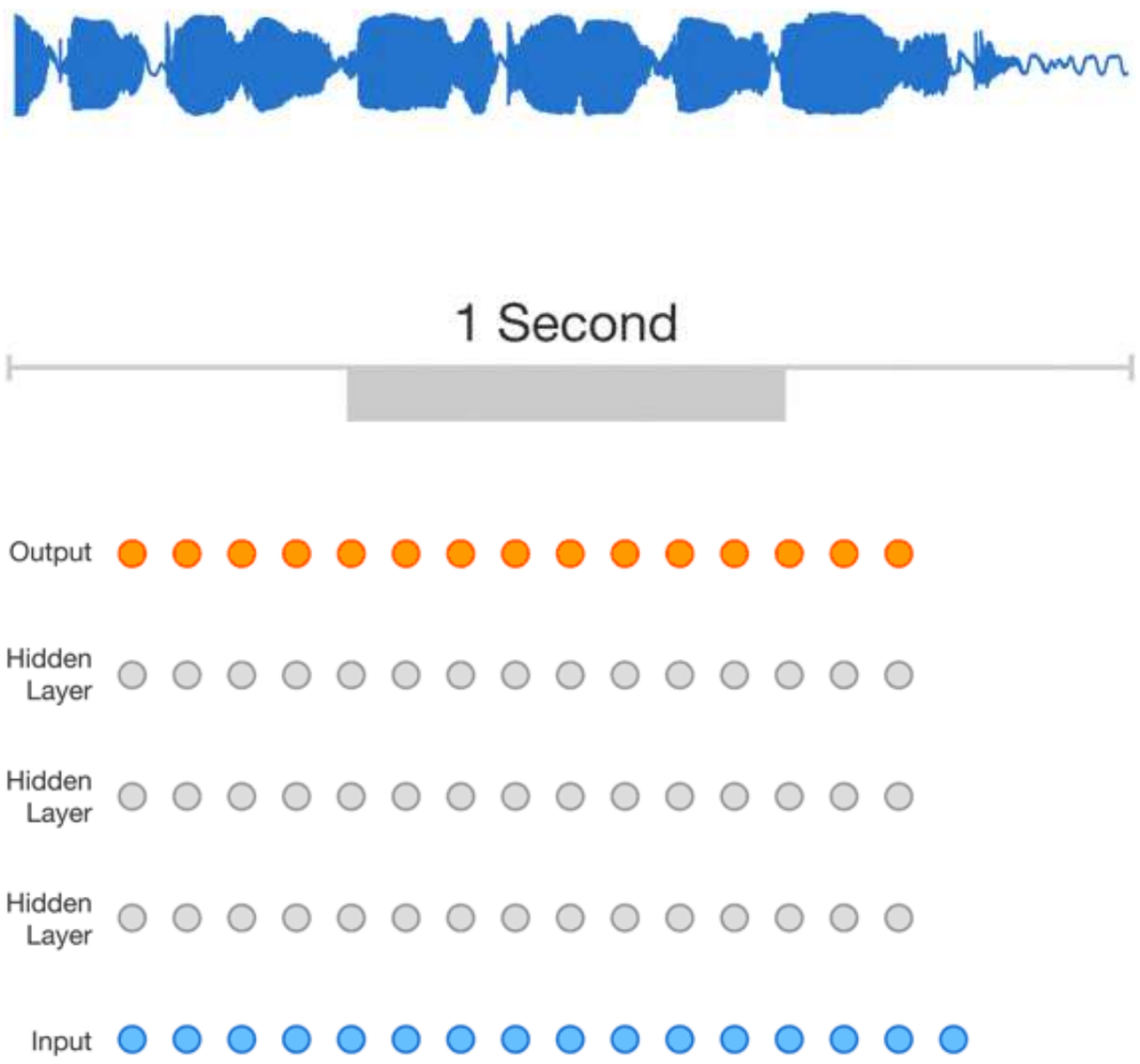
Speech Recognition

Word Error Rate

Senone set	Model/combination step	WER devset	WER test	WER devset	WER test
		ngram-LM		LSTM-LMs	
9k	BLSTM	11.5	8.3	9.2	6.3
27k	BLSTM	11.4	8.0	9.3	6.3
27k-puhpum	BLSTM	11.3	8.0	9.2	6.3
9k	BLSTM+ResNet+LACE+CNN-BLSTM	9.6	7.2	7.7	5.4
9k-puhpum	BLSTM+ResNet+LACE	9.7	7.4	7.8	5.4
9k-puhpum	BLSTM+ResNet+LACE+CNN-BLSTM	9.7	7.3	7.8	5.5
27k	BLSTM+ResNet+LACE	10.0	7.5	8.0	5.8
-	Confusion network combination			7.4	5.2
-	+ LSTM rescoring			7.3	5.2
-	+ ngram rescoring			7.2	5.2
-	+ backchannel penalty			7.2	5.1

Xiong *et al.* The Microsoft 2017 Conversational Speech Recognition System

Speech Synthesis (text-to-speech)



Van Den Oord *et al.* WaveNet: A Generative Model for Raw Audio

Strategy Board Game

AlphaGo

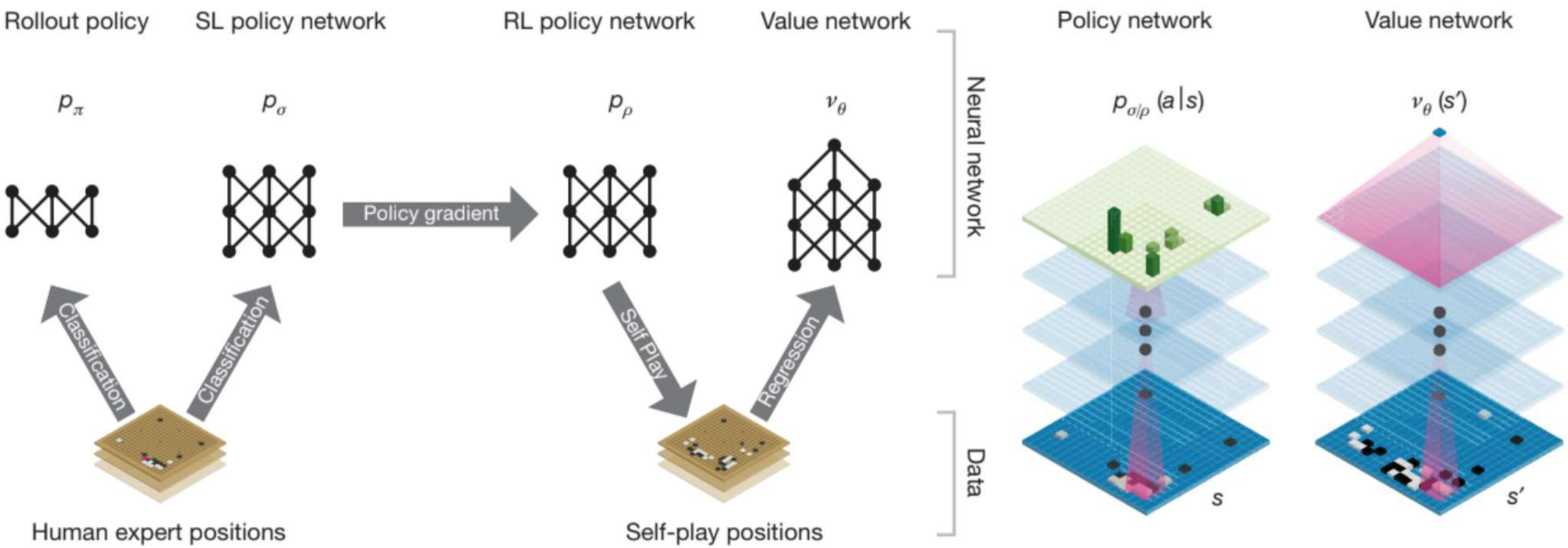
AlphaGo vs Lee Sedol

AlphaGo Master vs Ke Jie

Silver *et al.* Mastering the game of Go with deep neural networks and tree search
AlphaGo
AlphaGo: How it works technically? by Jonathan Hui

Strategy Board Game

AlphaGo



Supervised Learning + Reinforcement Learning + Monte Carlo Tree Search

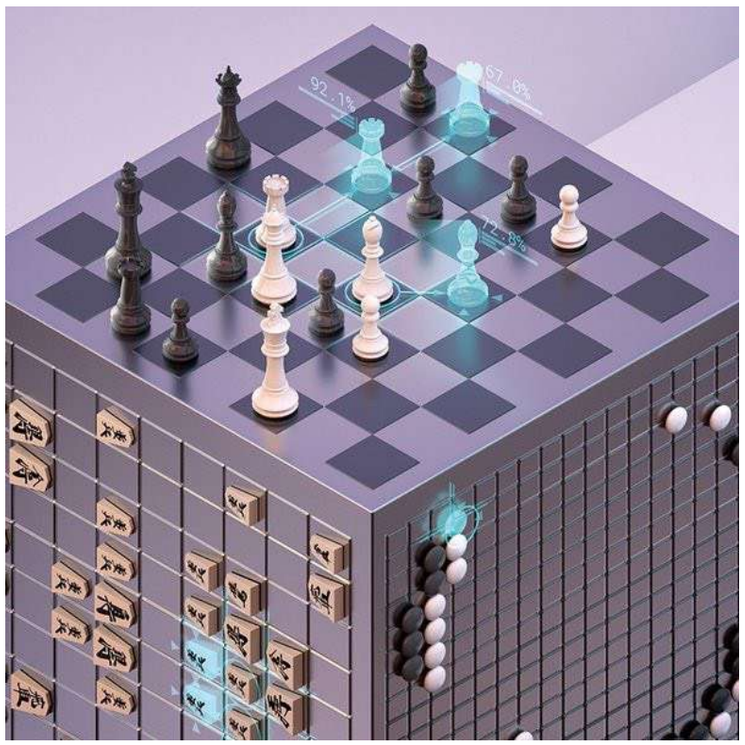
Silver *et al.* Mastering the game of Go with deep neural networks and tree search

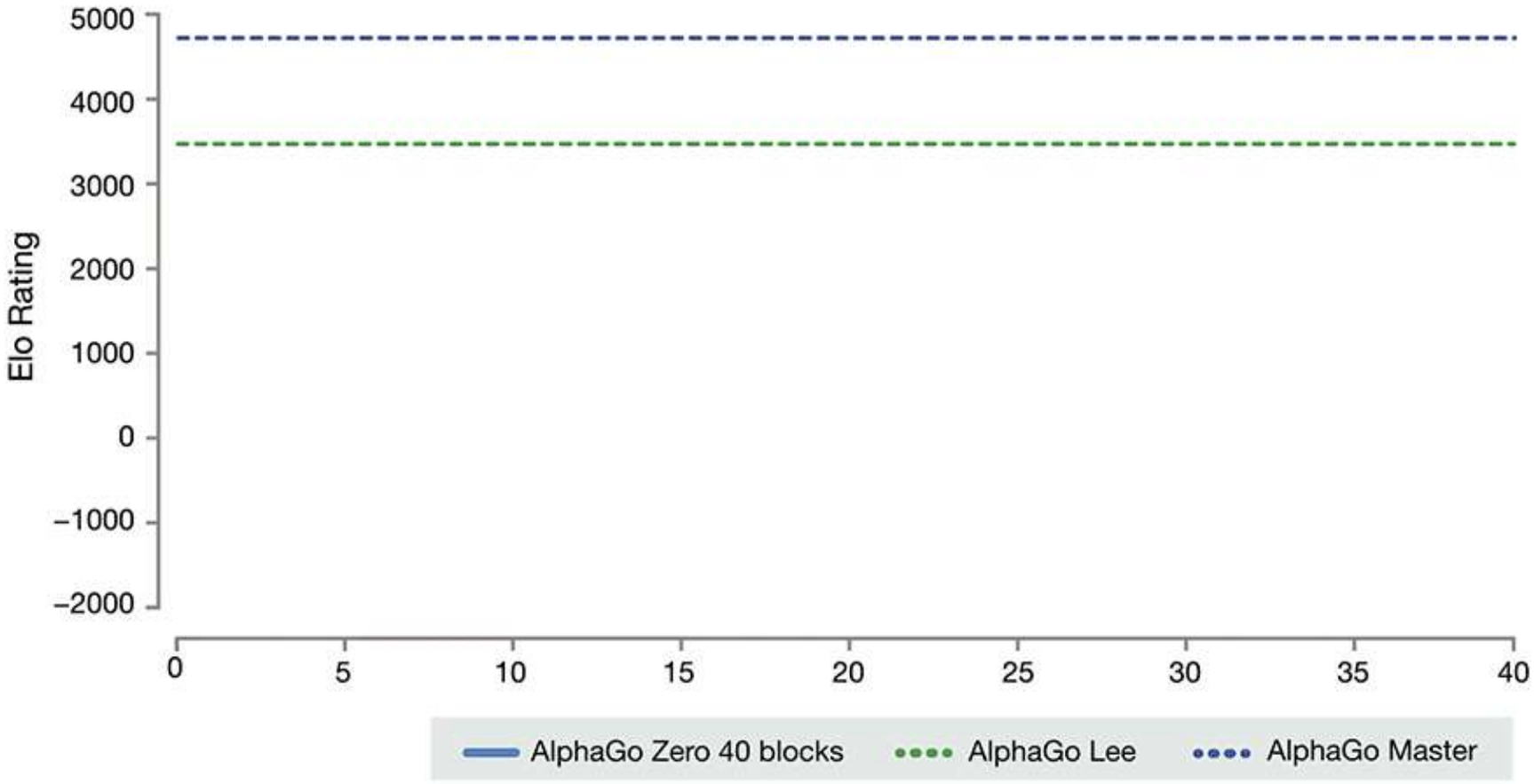
AlphaGo

AlphaGo: How it works technically? by Jonathan Hui

Strategy Board Game

AlphaGo Zero: Learning from scratch





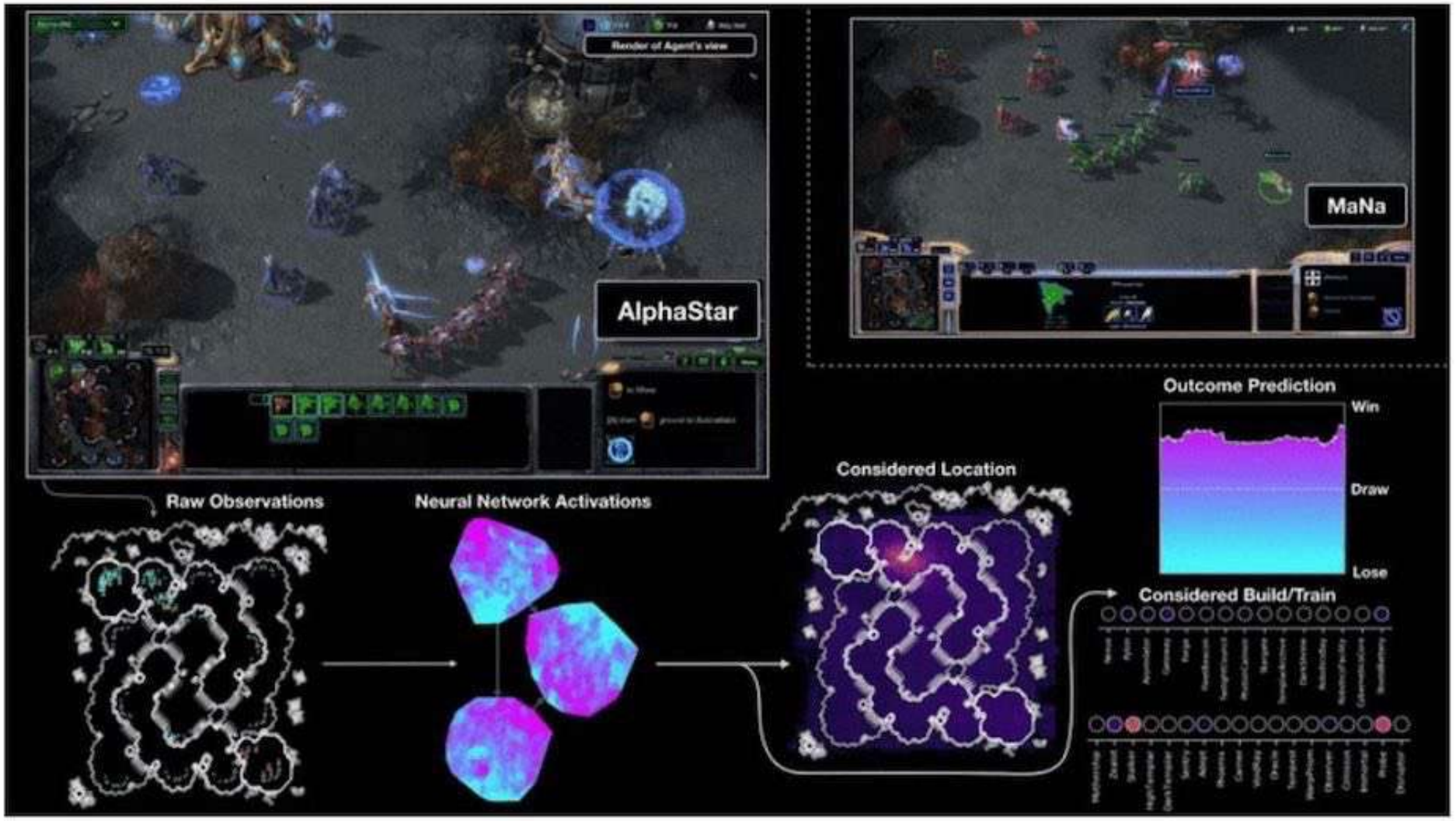
Reinforcement Learning + Monte Carlo Tree Search

Elo ratings: a measure of the relative skill levels of players in competitive games such as Go - show how AlphaGo has become progressively stronger during its development

Silver *et al.* Mastering the Game of Go without Human Knowledge

Game AI

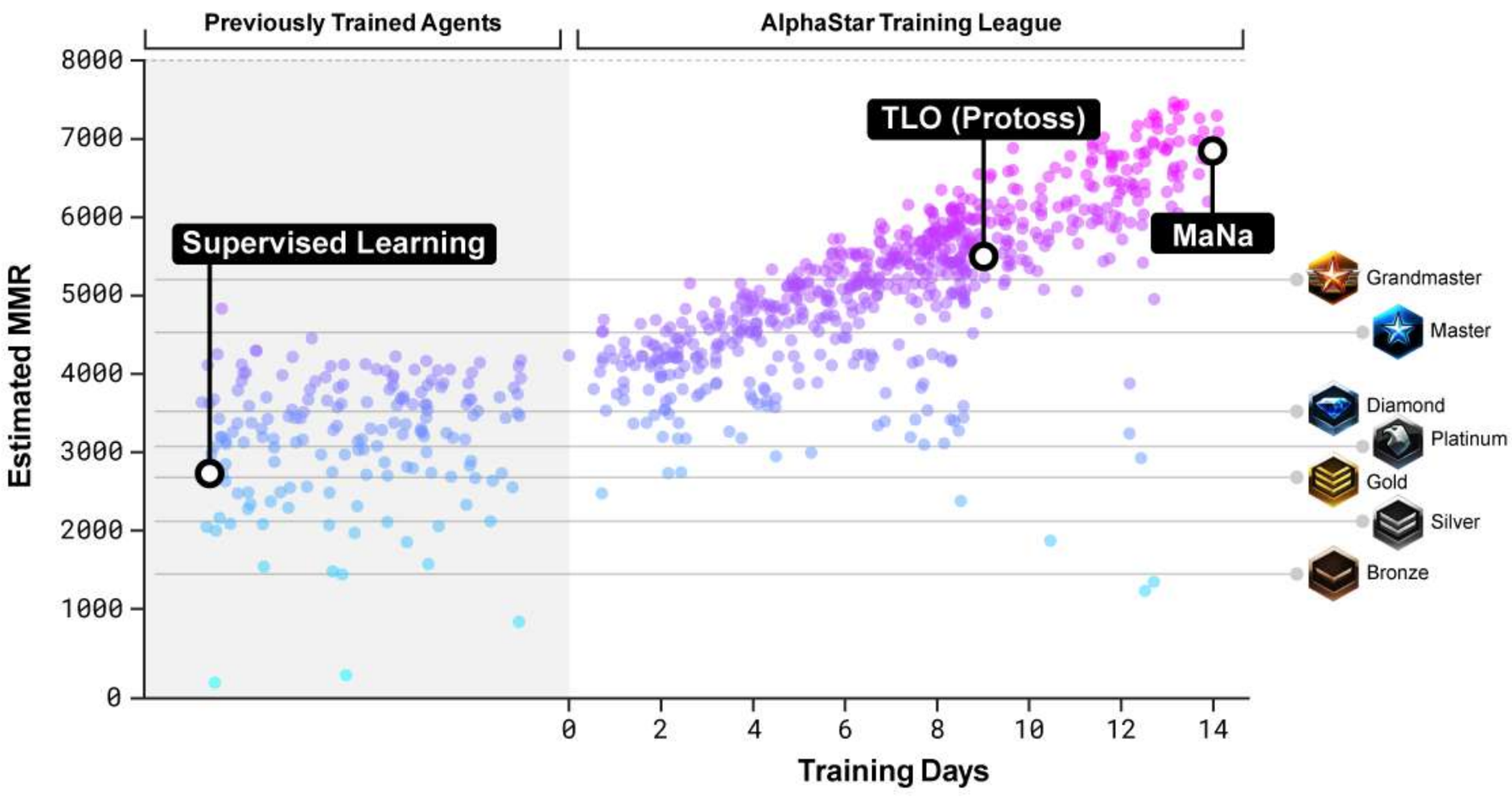
AlphaStar: StarCraft II



AlphaStar

Game AI

AlphaStar: StarCraft II

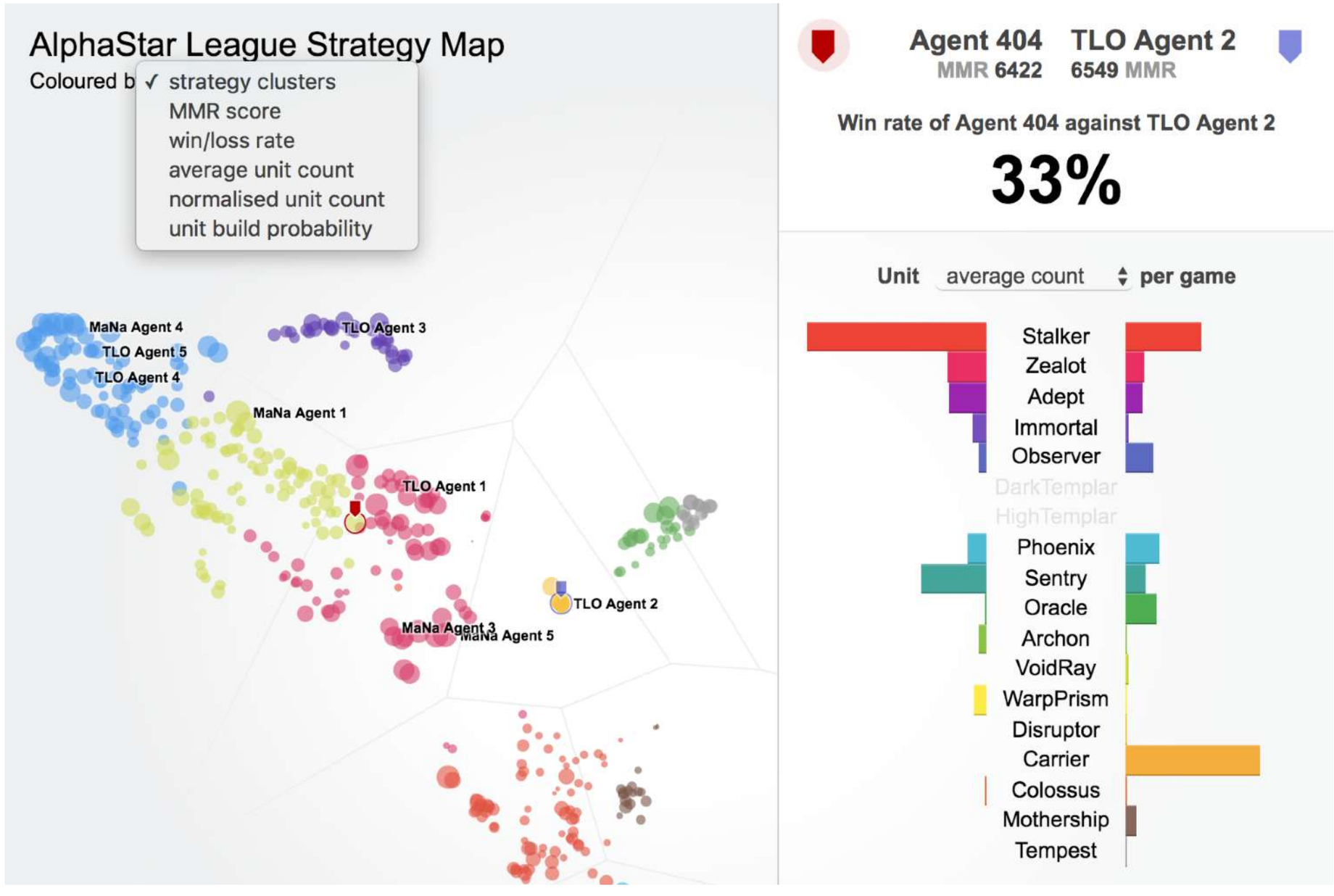


Supervised Learning + Reinforcement Learning

AlphaStar

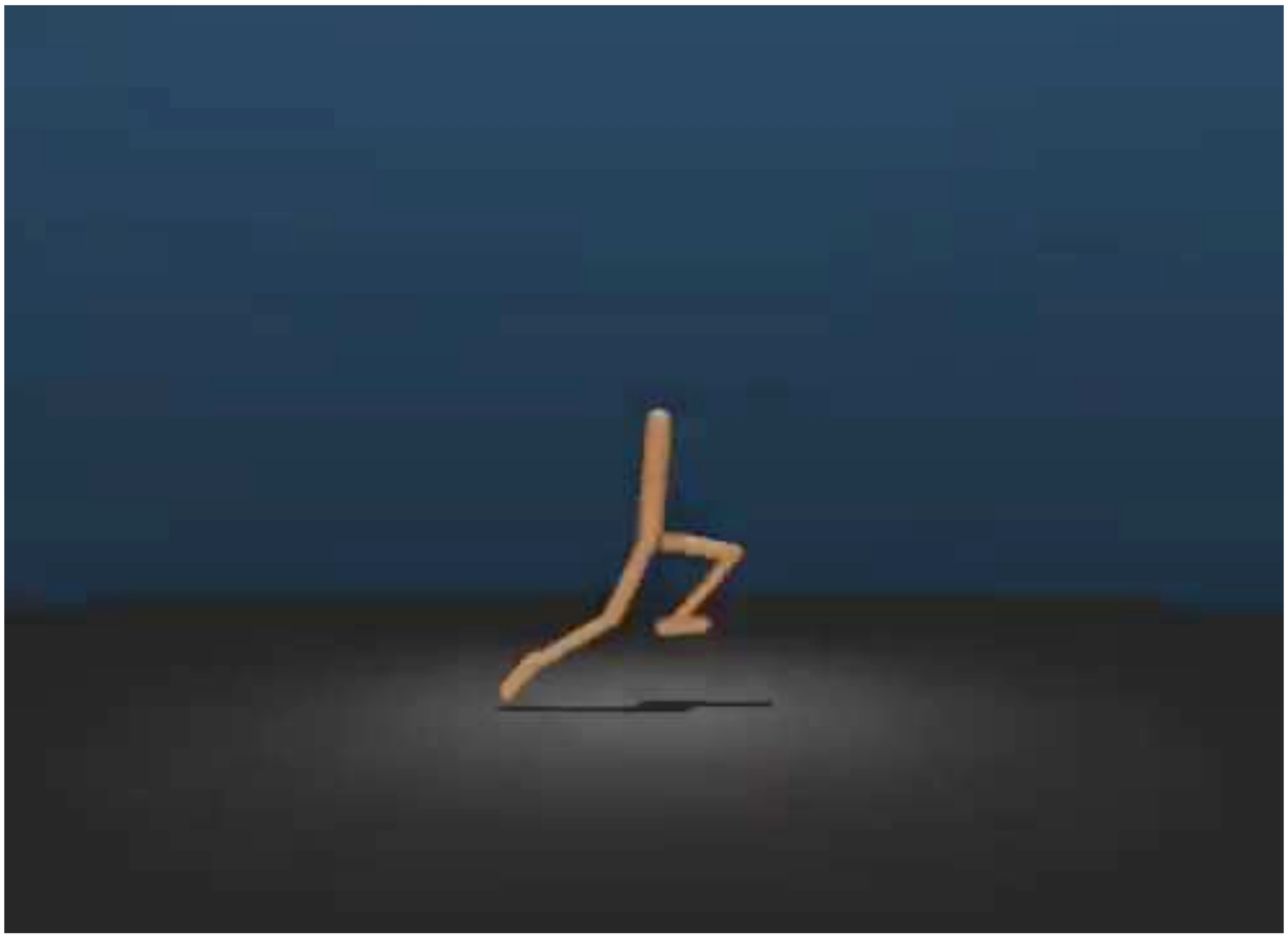
Game AI

AlphaStar: StarCraft II



AlphaStar

Locomotion



Heess *et al*. Emergence of Locomotion Behaviours in Rich Environments

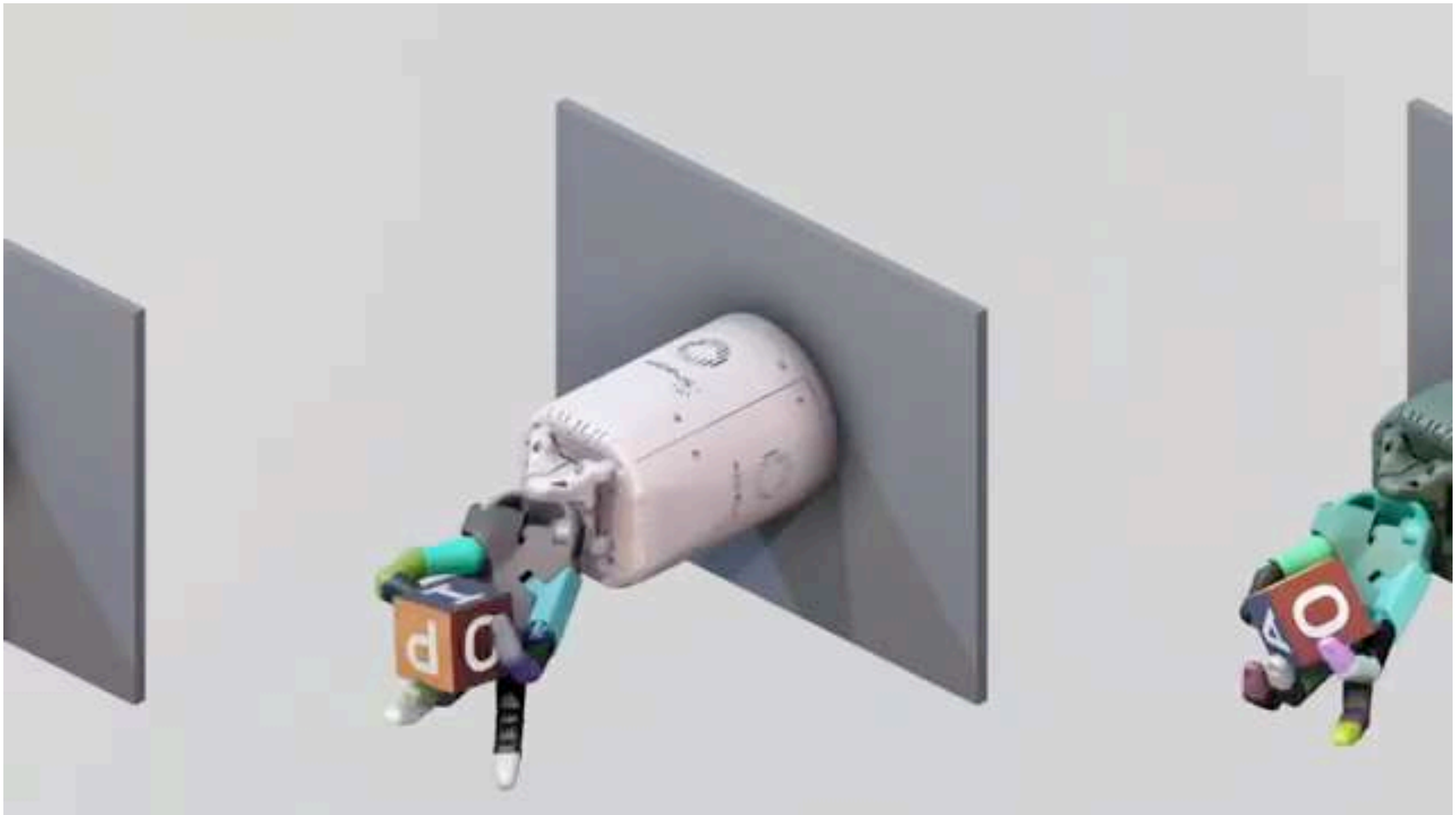
Character Control

Holden *et al*. Phase-Functioned Neural Networks for Character Control

Robot Manipulation: Grasping

Kalashnikov *et al*. QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation

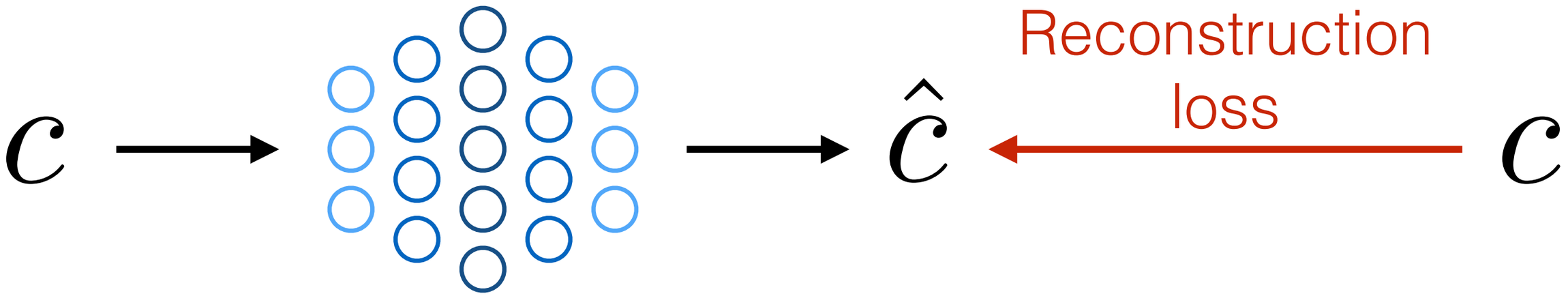
Robot Manipulation: Dexterity

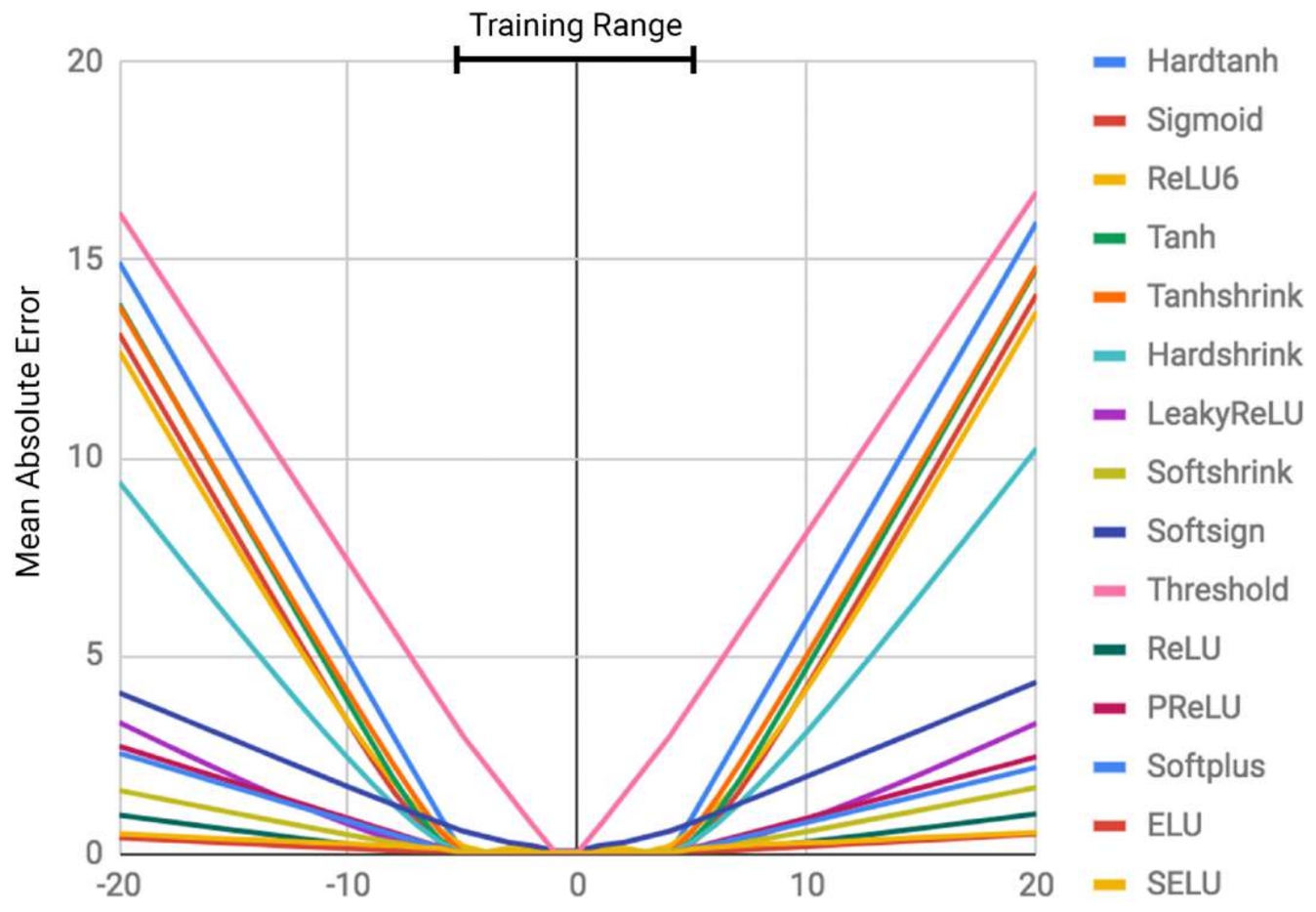


Learning Dexterous In-Hand Manipulation, OpenAI

Can Neural Networks Generalize Well?

Learning an Identity Mapping





Trask *et al*. Neural Arithmetic Logic Units

Neural Networks Do Not Learn Rules

Outline

- Recent Advance in Deep Learning
- Teach Neural Networks to Learn Rules
  - An Overview
  - Neural Program Induction
  - Neural Program Synthesis

Can We Teach Neural Networks to Learn Rules?

Digit Addition

carry	0	1			0	1	0	1	1		
	1	2	8		1	2	8	1	2	3	
+	5	1	2	+	5	1	2	7	8	9	
	6	4	0		6	4	0	9	1	2	

Can neural networks learn the rules of **digit addition** and generalize to numbers with more digits?

Outline

- Recent Advance in Deep Learning
- Teach Neural Networks to Learn Rules
- An Overview
- Neural Program Induction
- Neural Program Synthesis

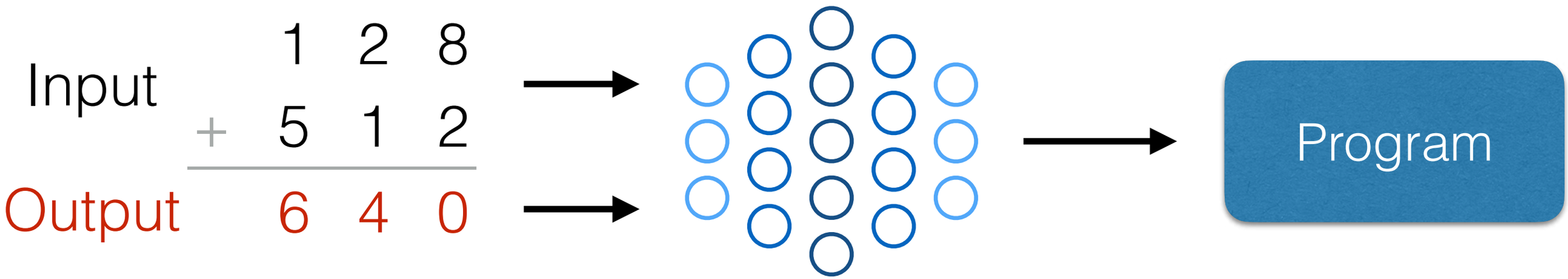
Neural Program Synthesis

&

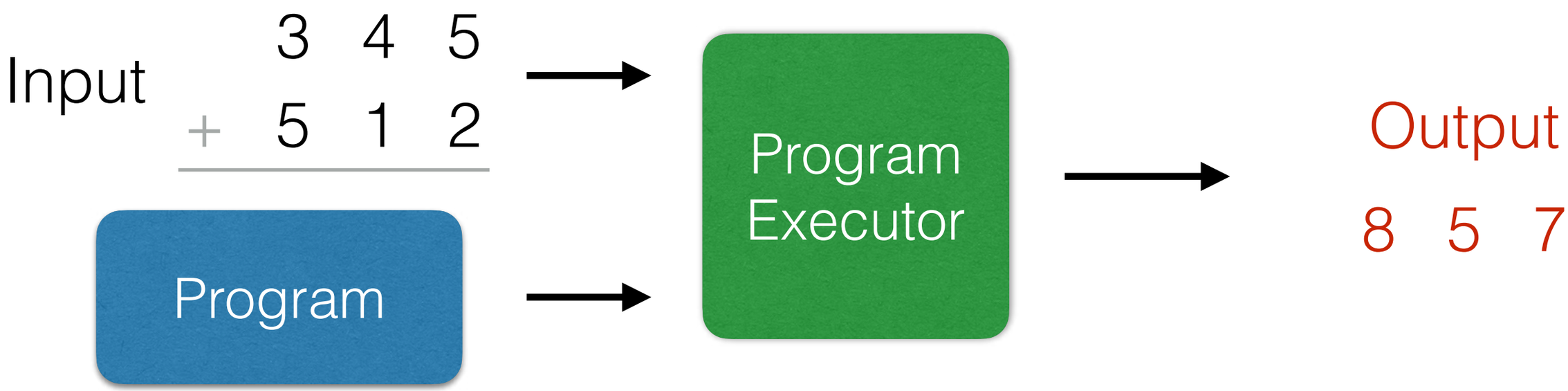
Neural Program Induction

Neural Program Synthesis

1. Infer the underlying program that can solve the task



2. Solve the task by executing the program



Neural Program Induction

Learn to mimic a program

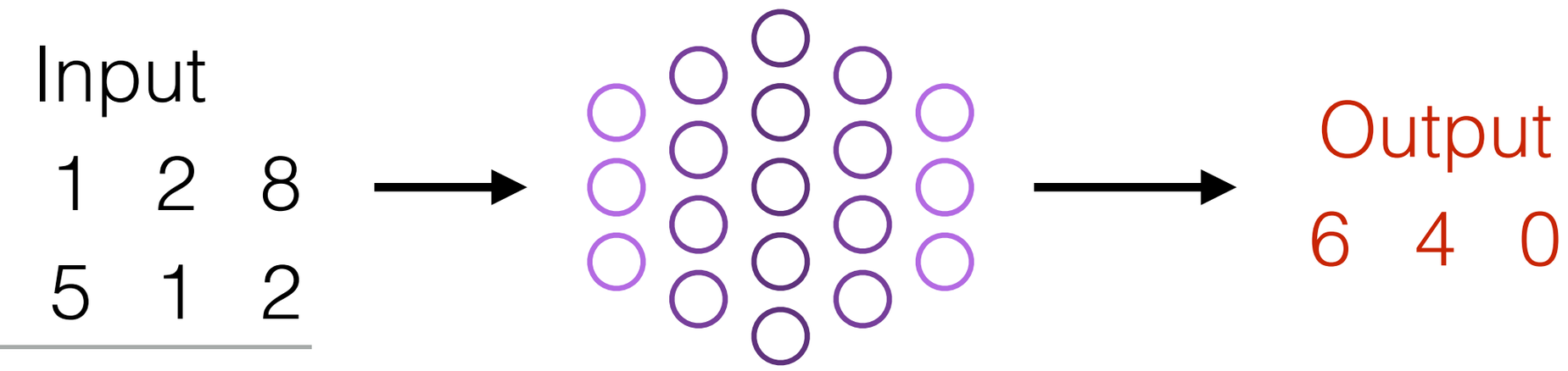
- Induce a latent representation of the program

Training

Input

1 2 8

\+ 5 1 2



Output

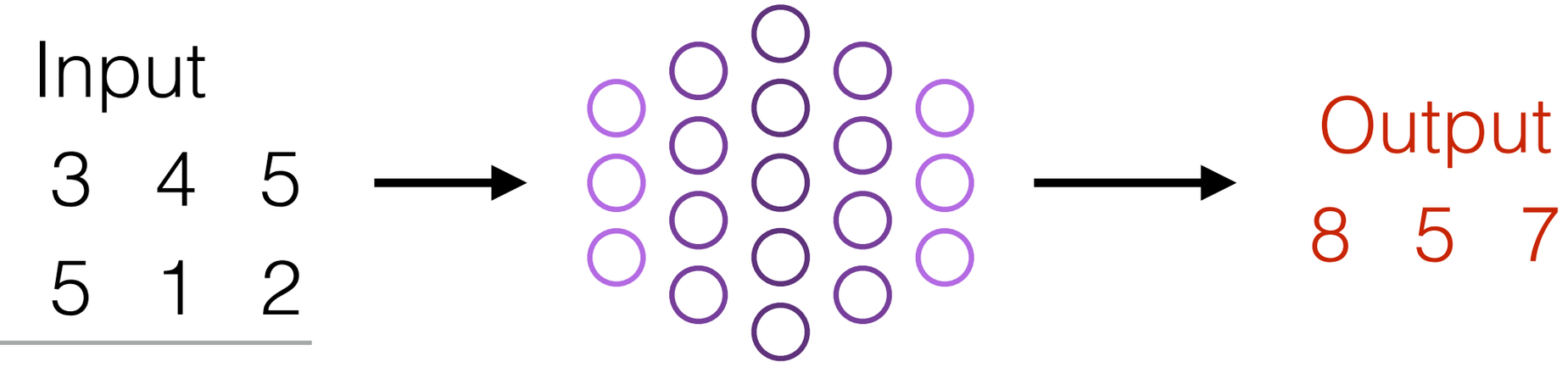
6 4 0

Testing

Input

3 4 5

\+ 5 1 2



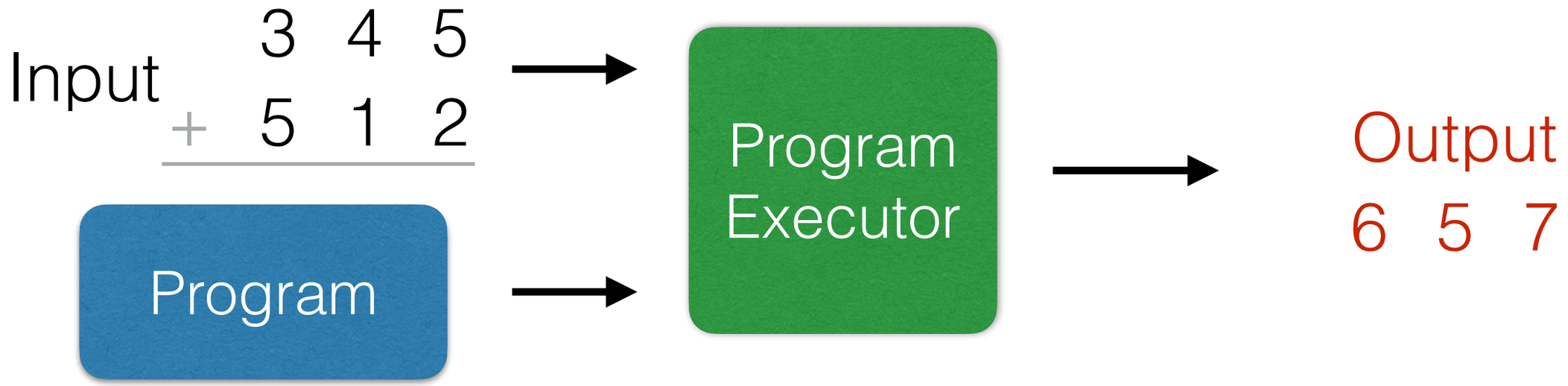
Output

8 5 7

Recap: Neural Program Synthesis/Induction

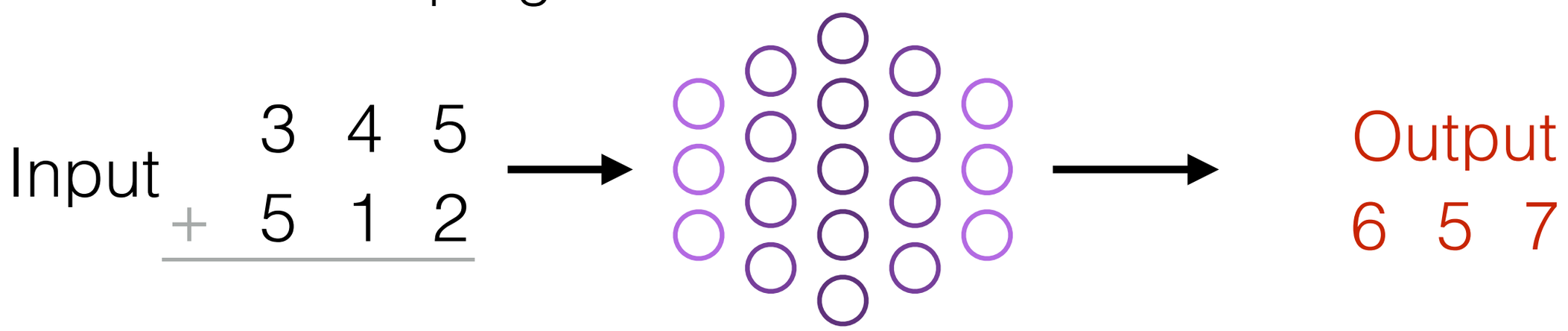
Neural program synthesis

- Infer and execute the underlying program to solve the task



Neural program induction

- Learn to mimic a program



Recap: Neural Program Synthesis/Induction

Neural program synthesis

- Infer and execute the underlying program to solve the task

- A predefined Domain Specific Language (DSL) is given
- A program executor is required to execute inferred programs
- Underlying programs are explicitly predicted

Neural program induction

- Learn to mimic a program

- A program predictor and executor combined
- Underlying programs are not explicitly predicted

A Case Study

RobustFill: Neural Program Learning under Noisy I/O

Jacob Devlin, Jonathan Uesato, Surya Bhupatiraju, Rishabh Singh,
Abdel-rahman Mohamed, Pushmeet Kohli

ICML 2017

Motivation & Problem Formulation

Microsoft wants to make their Excel more powerful

$I_1 = \texttt{January}$	$O_1 = \texttt{jan}$
$I_2 = \texttt{February}$	$O_2 = \texttt{feb}$
$I_3 = \texttt{March}$	$O_3 = \texttt{mar}$
$I_1^y = \texttt{April}$	$O_1^y = \texttt{apr}$
$I_2^y = \texttt{May}$	$O_2^y = \texttt{may}$
$P = \texttt{ToCase(Lower, SubStr(1,3))}$	

Neural Program Induction

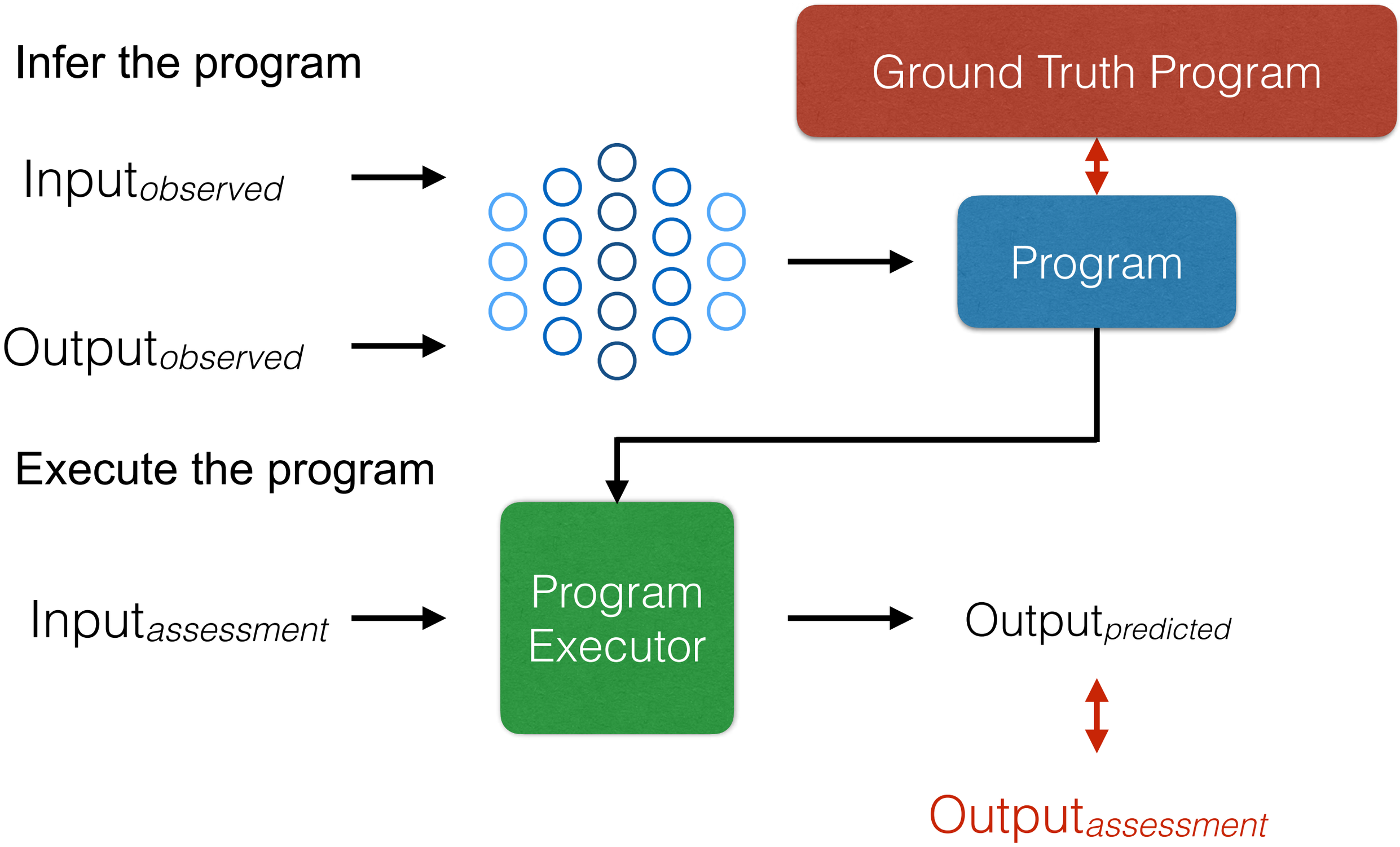
Neural Program Synthesis

Problem Formulation

Input*observed*	Output*observed*
$I_1 = \texttt{January}$	$O_1 = \texttt{jan}$
$I_2 = \texttt{February}$	$O_2 = \texttt{feb}$
$I_3 = \texttt{March}$	$O_3 = \texttt{mar}$
$I_1^y = \texttt{April}$	$O_1^y = \texttt{apr}$
$I_2^y = \texttt{May}$	$O_2^y = \texttt{may}$

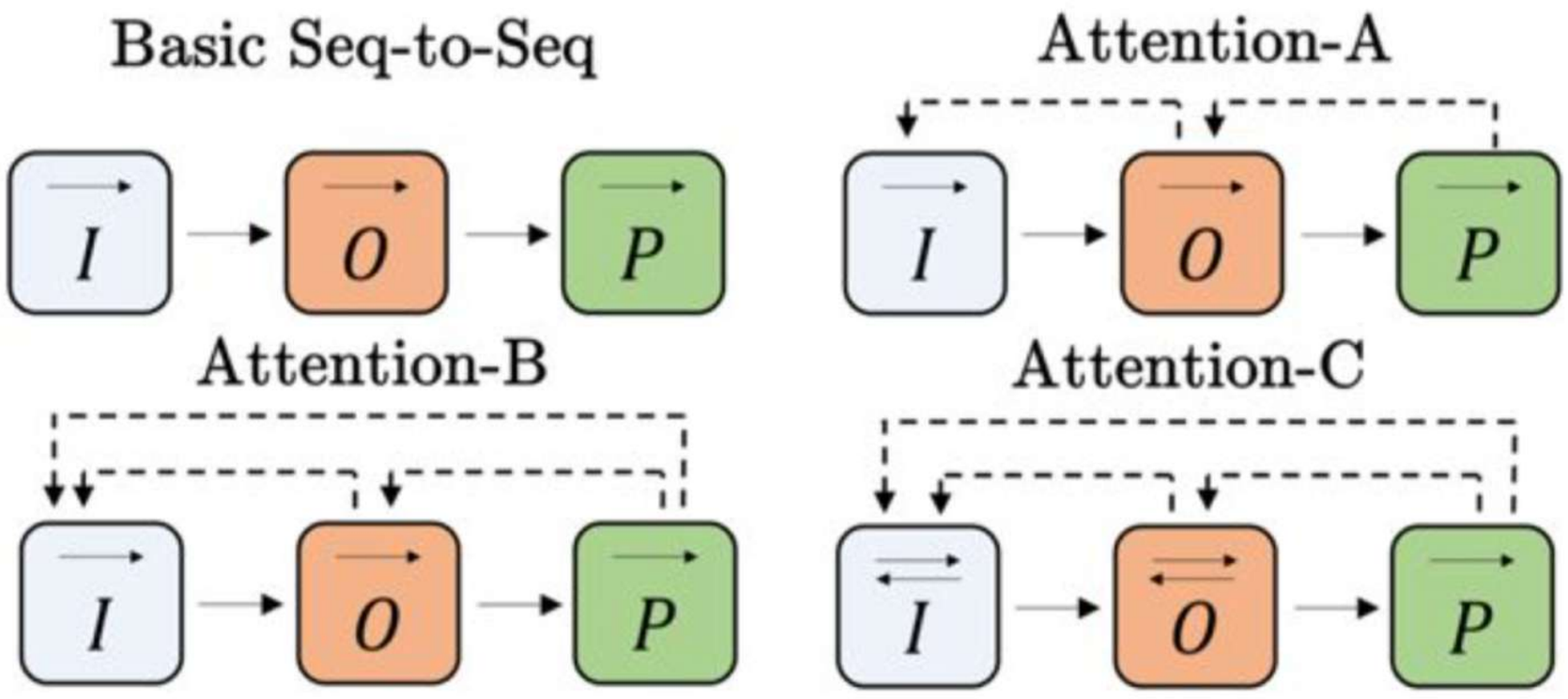
Input*assessment* Output*assessment*

Problem Formulation: Neural Program Synthesis



Synthesis: Model Architecture

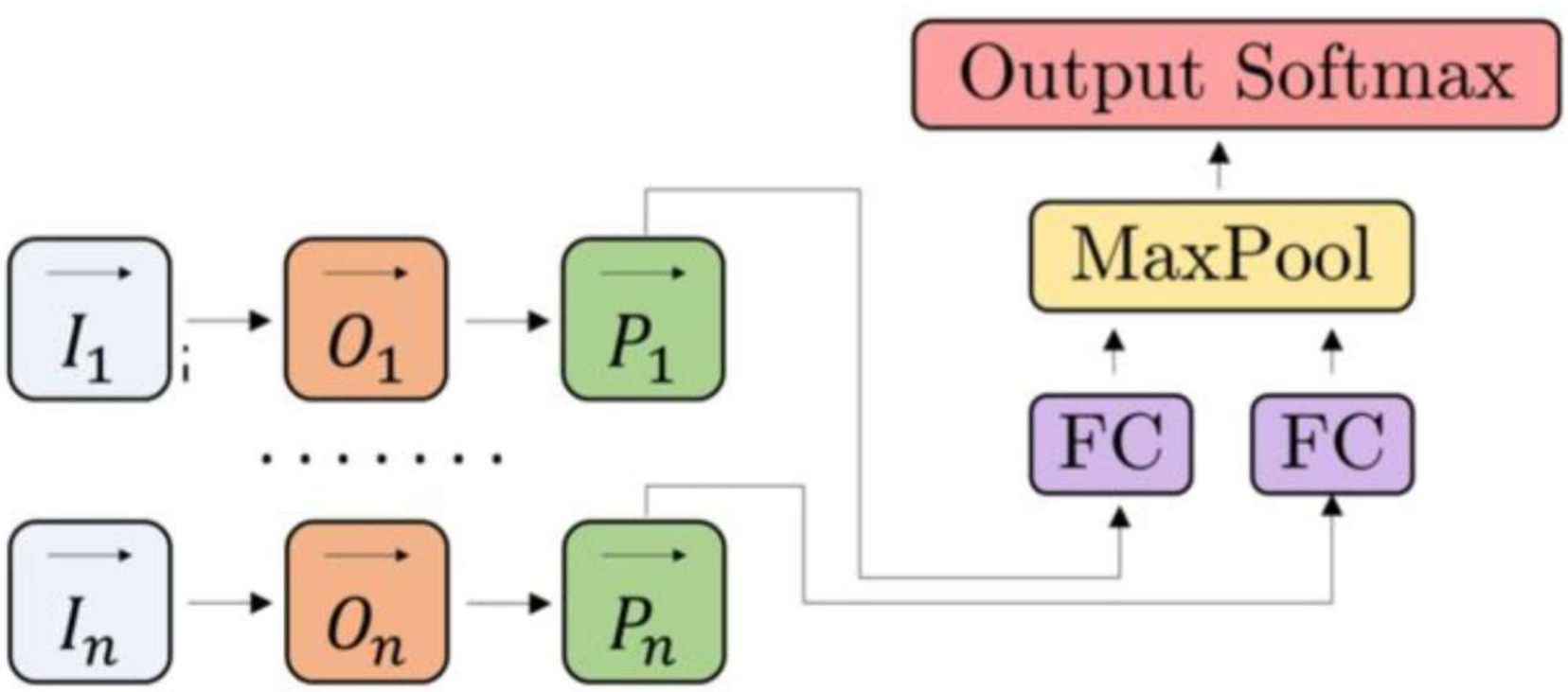
- Basic Seq2seq
- Attention A: O attends I and P attends O
- Attention-B: O attends I and P attends I and O
- Attention-C: O and I are bidirectional LSTMs



Synthesis: Model Architecture

Multi-example pooling

- Handle multiple example I/O pairs
- Order invariant



Synthesis: Domain Specific Language

$$
\begin{aligned}
[\![\texttt{Concat}(e_1, e_2, e_3, ...)]\!]_v &= \texttt{Concat}([\![e_1]\!]_v, [\![e_2]\!]_v, [\![e_3]\!]_v, ...) \\
[\![n_1(n_2)]\!]_v &= [\![n_1]\!]_{v_1}, \text{ where } v_1 = [\![n_2]\!]_v \\
[\![n(f)]\!]_v &= [\![n]\!]_{v_1}, \text{ where } v_1 = [\![f]\!]_v \\
[\![\texttt{ConstStr}(c)]\!]_v &= c \\
[\![\texttt{SubStr}(k_1, k_2)]\!]_v &= v[p_1..p_2], \text{ where} \\
& \quad p_1 = k_1 > 0 \; ? \; k_1 \; : \; \texttt{len(v)} + k_1 \\
& \quad p_2 = k_2 > 0 \; ? \; k_2 \; : \; \texttt{len(v)} + k_2 \\
[\![\texttt{GetSpan}(r_1, i_1, y_1, r_2, i_2, y_2)]\!]_v &= v[p_1..p_2] \text{ ,where} \\
& \quad p_1 = y_1 \text{(Start or End) of } |i_1|^{\texttt{th}} \text{ match of } r_1 \text{ in } v \text{ from beginning (end if } i_i < 0) \\
& \quad p_2 = y_2 \text{(Start or End) of } |i_2|^{\texttt{th}} \text{ match of } r_2 \text{ in } v \text{ from beginning (end if } i_2 < 0) \\
[\![\texttt{GetToken(t, i)}]\!]_{\texttt{v}} &= |i|^{\texttt{th}} \text{ match of } t \text{ in } v \text{ from beginning (end if } i < 0) \\
[\![\texttt{GetUpto(r)}]\!]_{\texttt{v}} &= v[0..i], \text{ where } i \text{ is the index of end of first match of } r \text{ in } v \text{ from beginning} \\
[\![\texttt{GetFrom(r)}]\!]_{\texttt{v}} &= v[j..len(v)], \text{ where j is the end of last match of } r \text{ in } v \text{ from end} \\
[\![\texttt{GetFirst(t, i)}]\!]_{\texttt{v}} &= \texttt{Concat}(s_1, \cdots, s_i), \text{ where } s_j \text{ denotes the } j^{\texttt{th}} \text{ match of } t \text{ in } v \\
[\![\texttt{GetAll(t)}]\!]_{\texttt{v}} &= \texttt{Concat}(s_1, \cdots, s_m), \text{ where } s_i \text{ denotes the } i^{\texttt{th}} \text{ match of } t \text{ in } v \text{ and } m \text{ denotes the total matches} \\
[\![\texttt{ToCase(s)}]\!]_{\texttt{v}} &= \texttt{ToCase}(s, v) \\
[\![\texttt{Trim()}]\!]_{\texttt{v}} &= \texttt{Trim}(v) \\
[\![\texttt{Replace}(\delta_1, \delta_2)]\!]_{\texttt{v}} &= \texttt{Replace}(v, \delta_1, \delta_2)
\end{aligned}
$$

Synthesis: Training Data

Reference program: GetToken_Alphanum_3 | GetFrom_Colon | GetFirst_Char_4	
Ud 9:25,JV3 Obb	2525,JV3 ObbUd92
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: Get_Word_-1(GetSpan(Word, 1, Start, '(', 5, Start)) | GetToken_Number_-5 | GetAll_Proper | SubStr(-24, -14) | GetToken_Alphanum_-2 | EOS	
4 Kw ( )SrK (11 (3 CHA xVf)4)8 Qagimg) ( )(vs	Qagimg4Kw Sr Vf QagimgVf)4)8 QaQagimg
iY))hspA.5 ( )8,ZsLL (nZk.6 (E4w)2(Hpprsqr)2(Z	Hpgjprsqr8Zs Zk Hpprsqrk.6 (E4w)22
Cqg)) ((1005 (( )VCE hz) (10 Hadj)zg Tqwpaxft-7 5 6	hz10005Cqg Hadj Tqwpaxft Hadj)zg T5
JvY) (Ihitux)) ((6 SFl (7 XLTD sfs))11,lU7 (6 9	lU7Jv Ihitux Frl XLTD sfs)6
NjtT(D7QV (4 (yPuY)8.sa ( ))6 aX 4)DXR (@6) Ztje	DXR4Njt Pu Ztje)6 aX 4)DX6

Reference program: GetToken_AllCaps_-2(GetSpan(AllCaps, 1, Start, AllCaps, 5, Start)) | EOS	
YDXJZ @ZYUD Wc-YKT GTIL BNX	W
JUGRB.MPKA.MTHV,tEczT-GZJ.MFT	MTHV
VXO.OMQDK.JC-OAR,HZGH-DJKC	JC
HCUD-WDOC,RTTRQ-KVETK-whx-DIKDI	RTTRQ
JFNB.Avj,ODZBT-XHV,KYB @,RHVVW	ODZBT

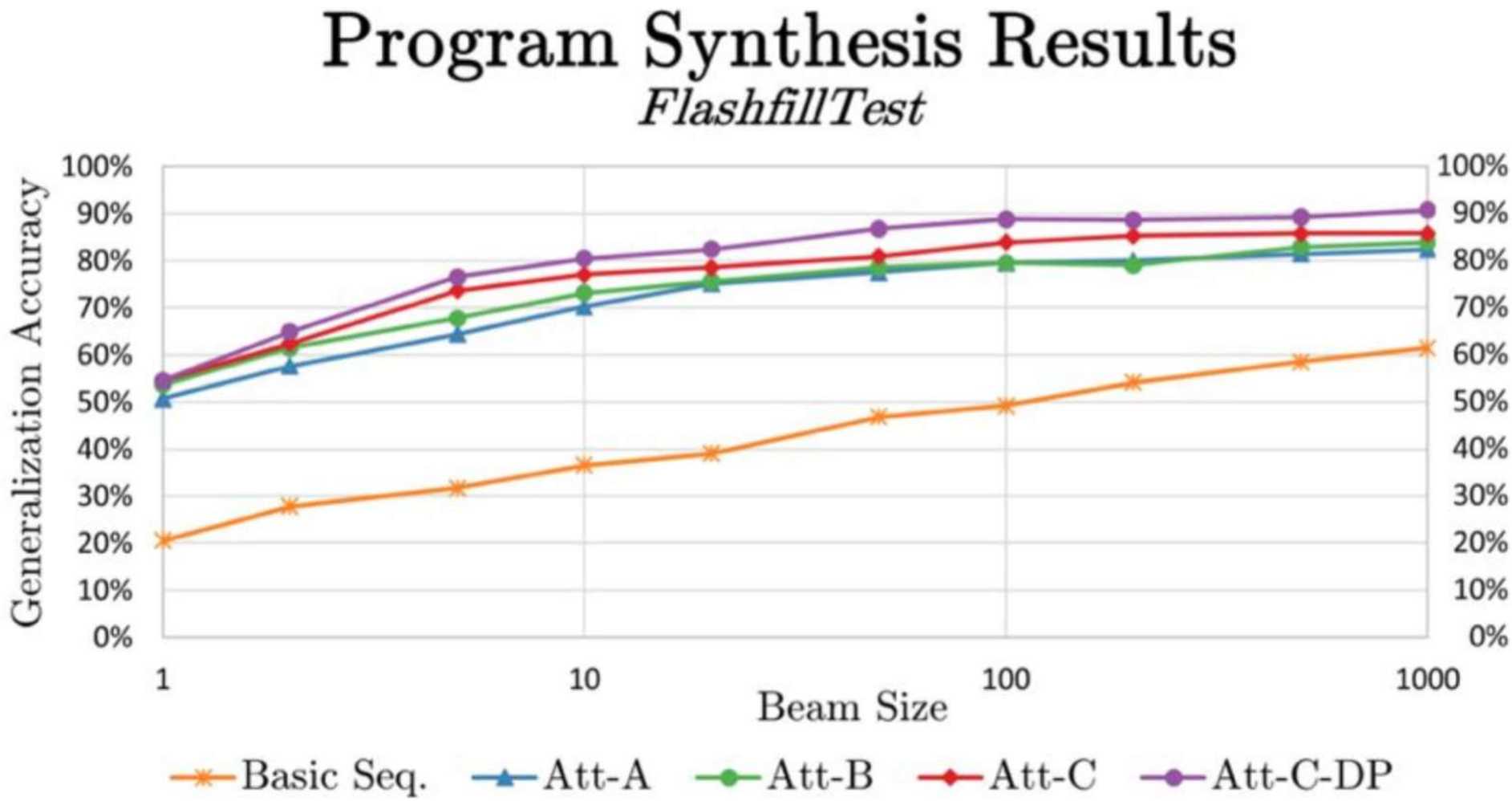
Testing Data

FlashFillTest

- 205 instances
  - Each instance
    - 4 observed I/O examples
    - 6 assessment I/O examples

Mason Smith	Mason.S@
Lucas Janckle	Lucas.J@
Emily Jacobnette	Emily.B@
Charlotte Ford	Charlotte.F@
Harper Underwood	Harper.U@
Emma Stevens	Emma.S@
Chris Charles	Chris.C@
Liam Lewis	Liam.L@
Olivia Berglun	Olivia.B@
Abigail Jones	Abigail.J@

Synthesis: Results

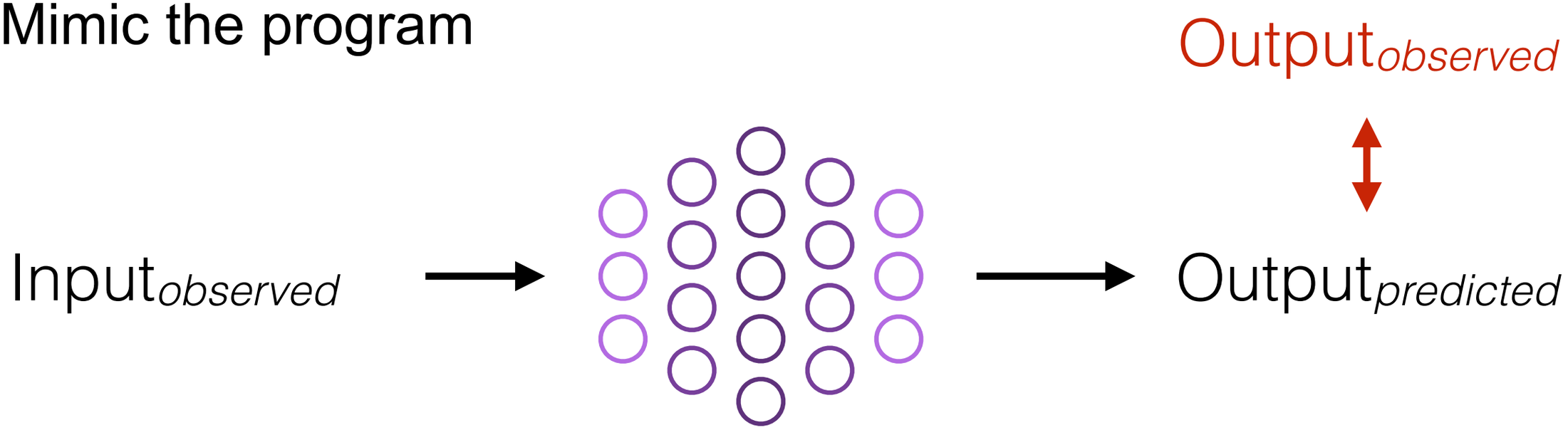


Synthesis: Results

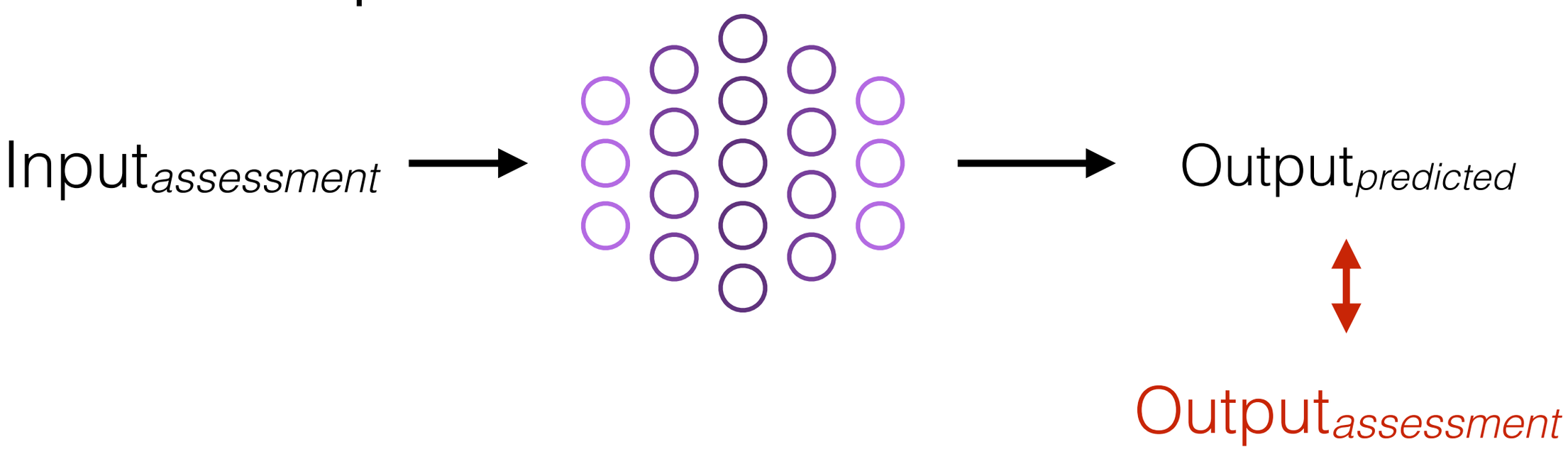
Model prediction: GetToken_Proper_1 | Const(.) | GetToken_Char_1(GetToken_Proper_-1) | Const(@) | EOS		
Mason Smith	Mason.S@	Mason.S@
Lucas Janckle	Lucas.J@	Lucas.J@
Emily Jacobnette	Emily.B@	Emily.B@
Charlotte Ford	Charlotte.F@	Charlotte.F@
Harper Underwood	Harper.U@	Harper.U@
Emma Stevens	Emma.S@	Emma.S@
Chris Charles	Chris.C@	Chris.C@
Liam Lewis	Liam.L@	Liam.L@
Olivia Berglun	Olivia.B@	Olivia.B@
Abigail Jones	Abigail.J@	Abigail.J@

Problem Formulation: Neural Program Induction

Mimic the program



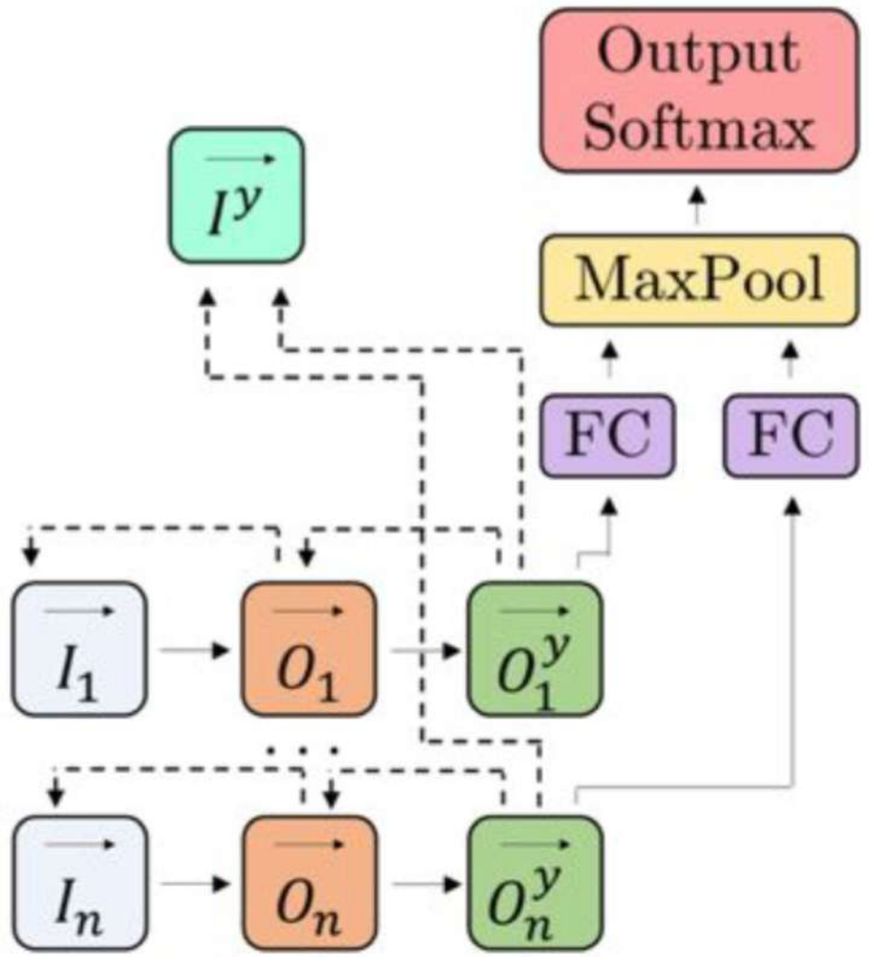
Predict the output



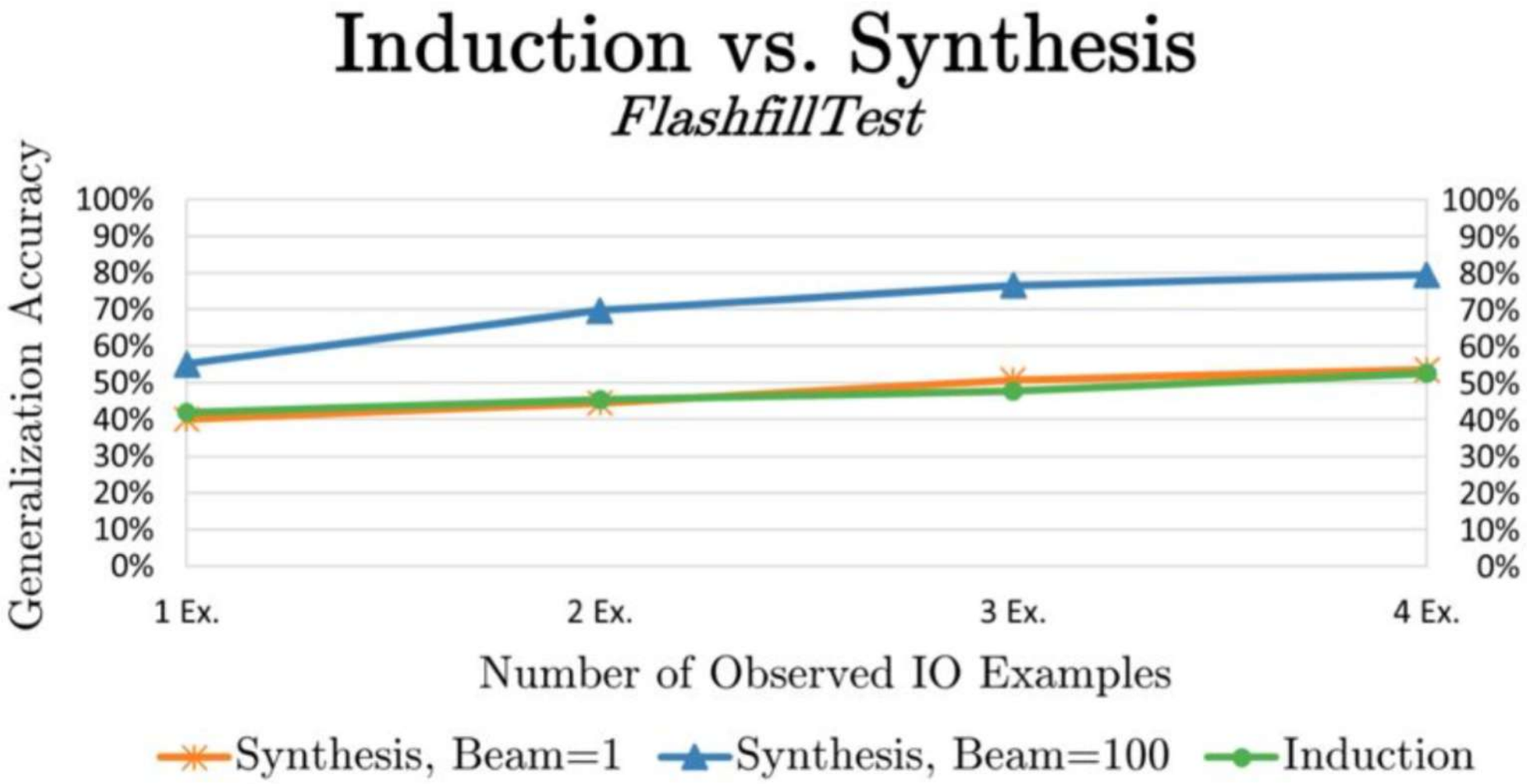
Induction: Model Architecture

Similar to the program synthesis models

- But it is directly trained to predict the output O



Results


Induction vs. Synthesis
FlashfillTest
Generalization Accuracy
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
1 Ex.
2 Ex.
3 Ex.
4 Ex.
Number of Observed IO Examples
Synthesis, Beam=1
Synthesis, Beam=100
Induction

Accuracies

All-example accuracy

- All generated assessment outputs need to be correct

Average-example accuracy

- 5-out-of-6 assessment examples accumulates more credit than 0

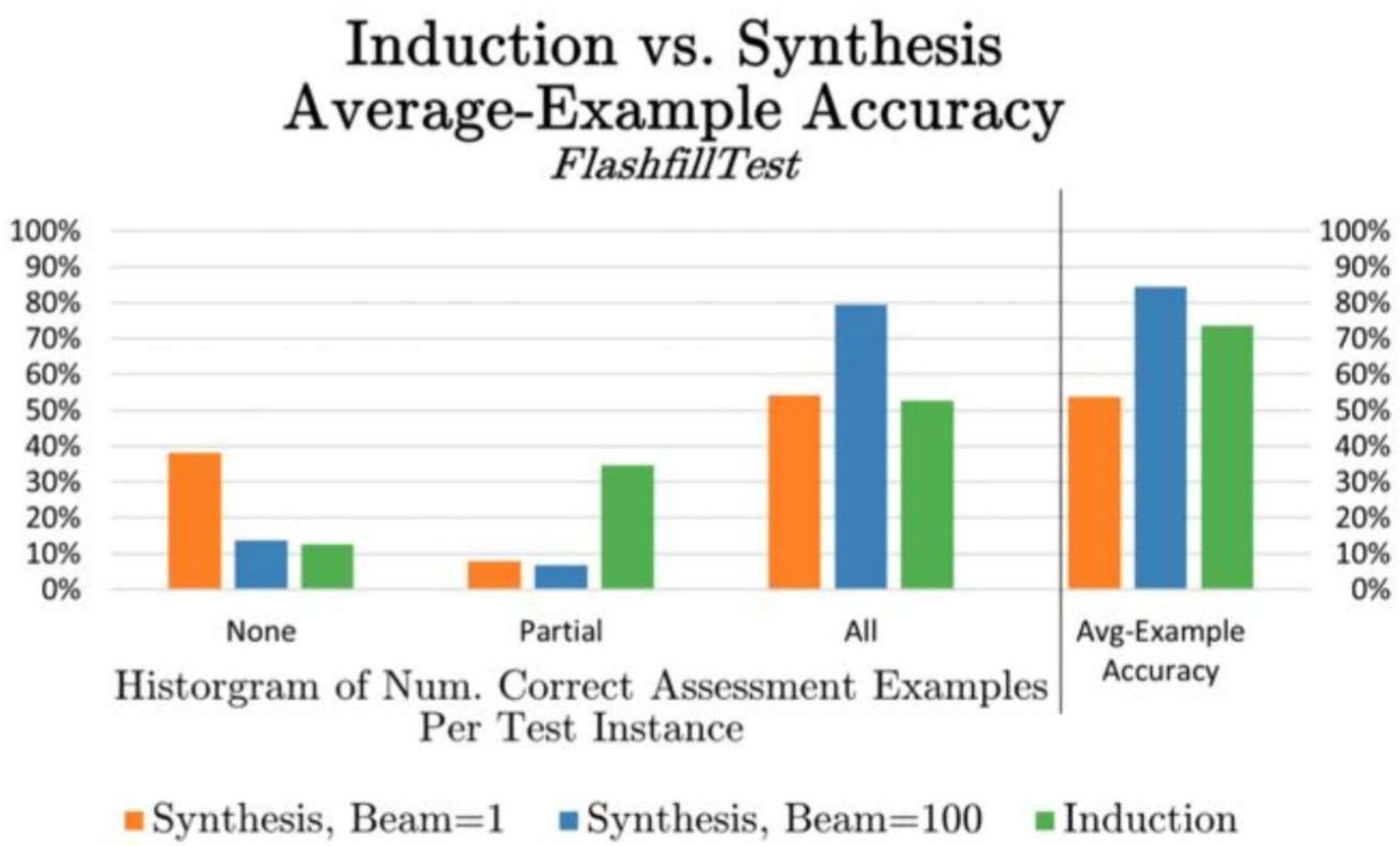
Results

Synthesis

- All or nothing

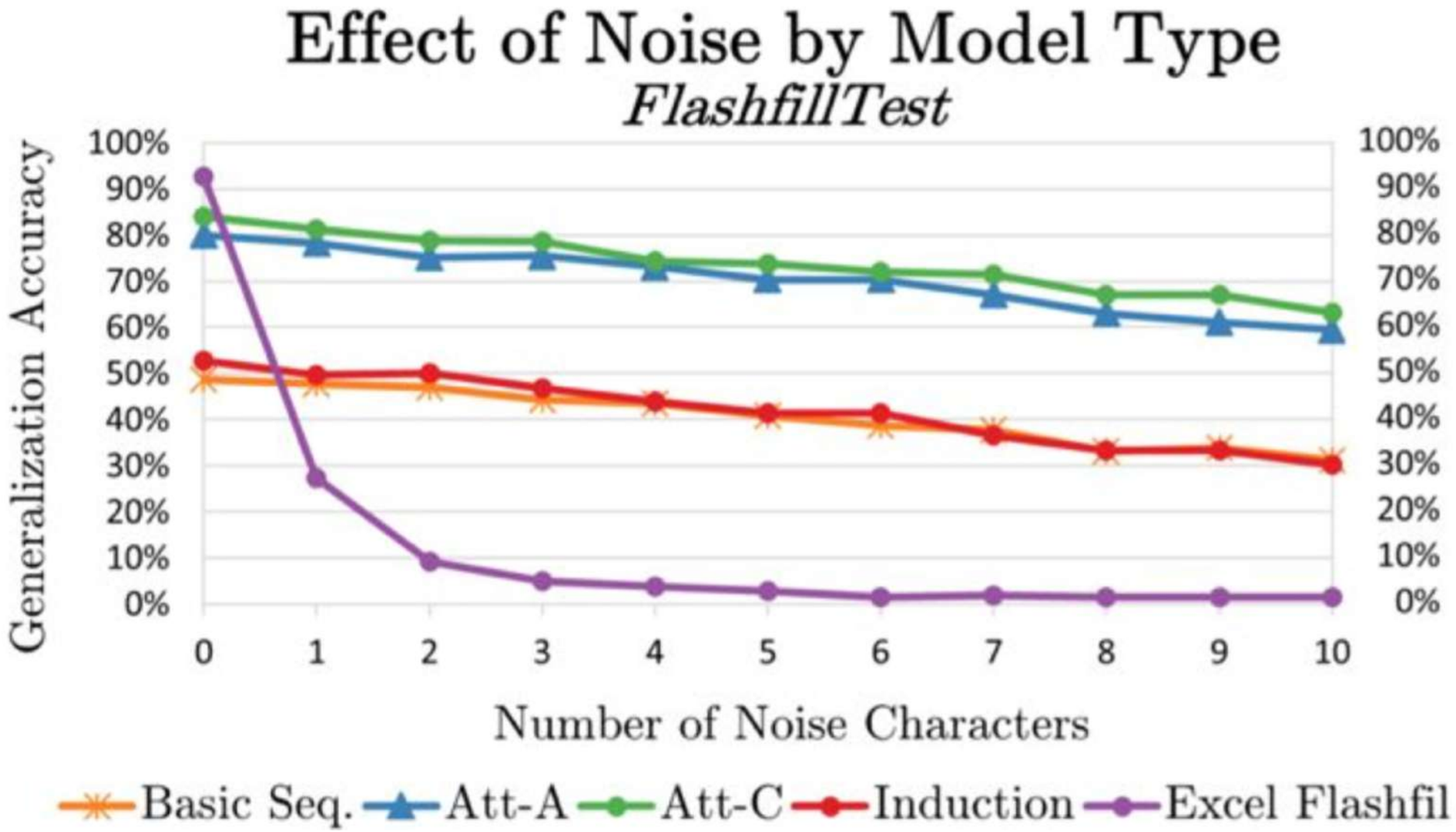
Induction

- Produce partially correct outputs



Noisy I/O

Learning-based models are more robust to noise



Outline

- Recent Advance in Deep Learning
- Teach Neural Networks to Learn Rules
- An Overview
- Neural Program Induction
- Neural Program Synthesis

Neural Program Induction

Learn to mimic a program

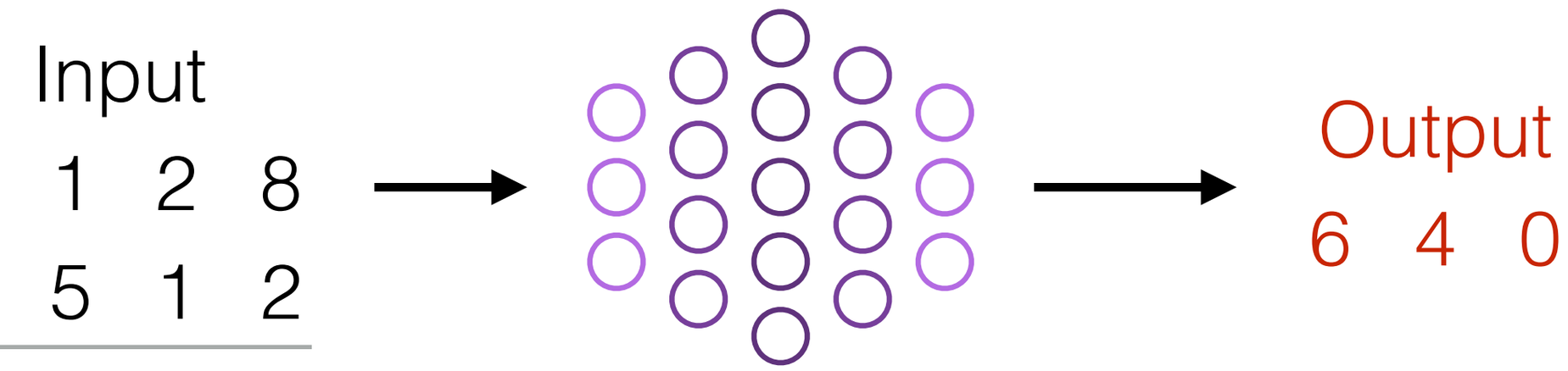
- Induce a latent representation of the program

Training

Input

1 2 8

\+ 5 1 2



Output

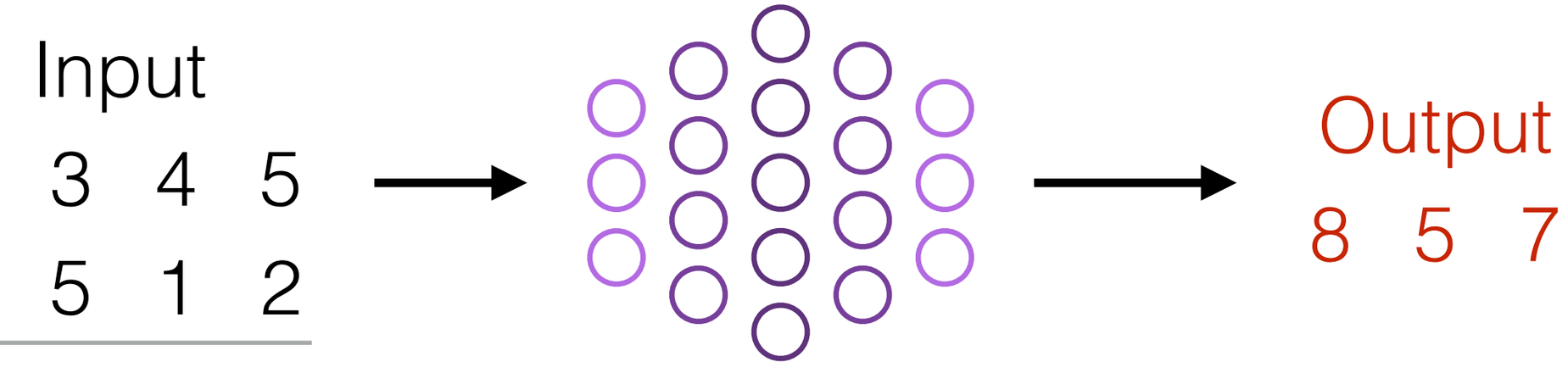
6 4 0

Testing

Input

3 4 5

\+ 5 1 2



Output

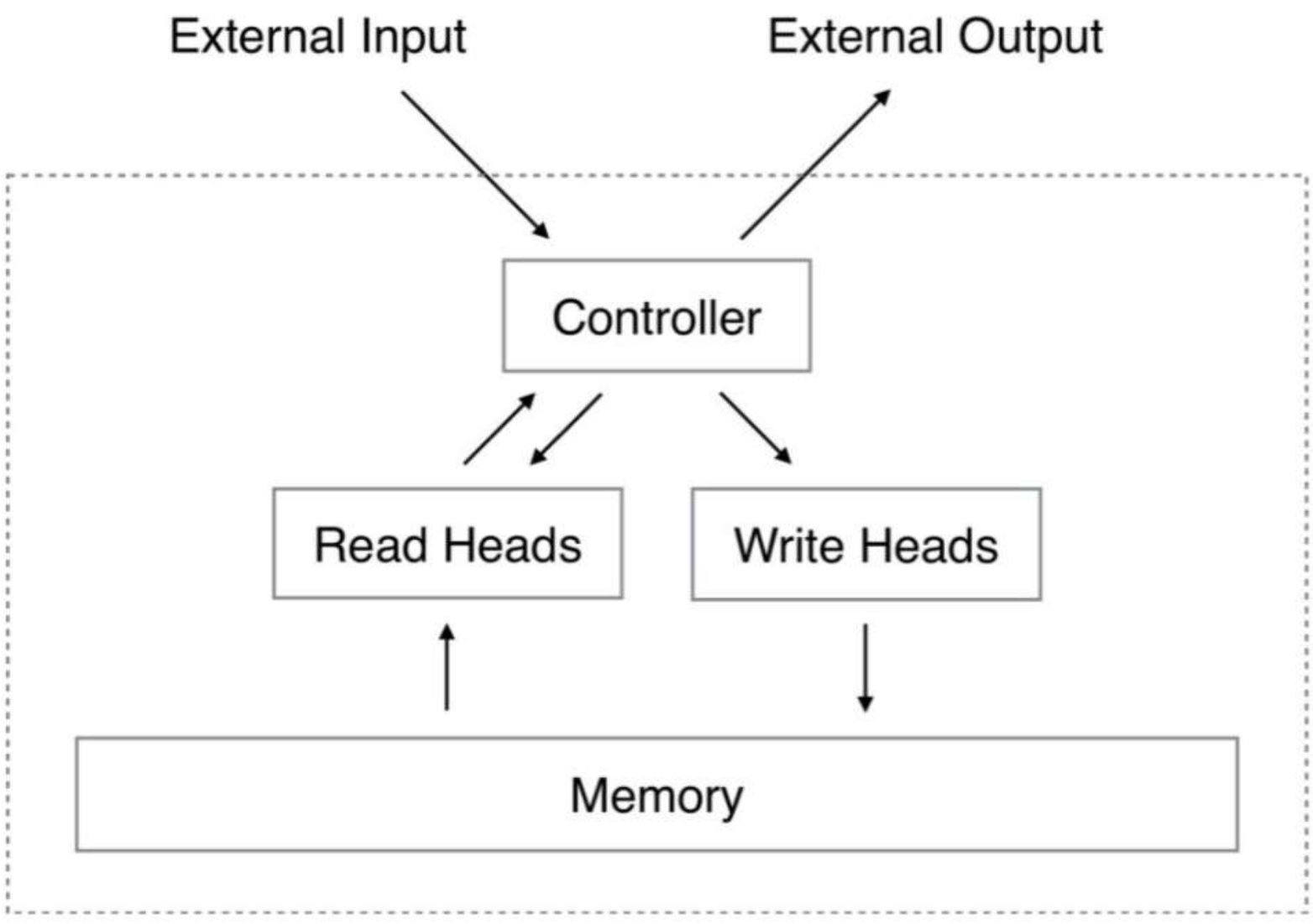
8 5 7

Neural Turing Machines

Alex Graves, Greg Wayne, Ivo Danihelka

arXiv 2014

Model


External Input
External Output
Controller
Read Heads
Write Heads
Memory

Results: Copy

Store and recall a long sequence of arbitrary information

LSTM

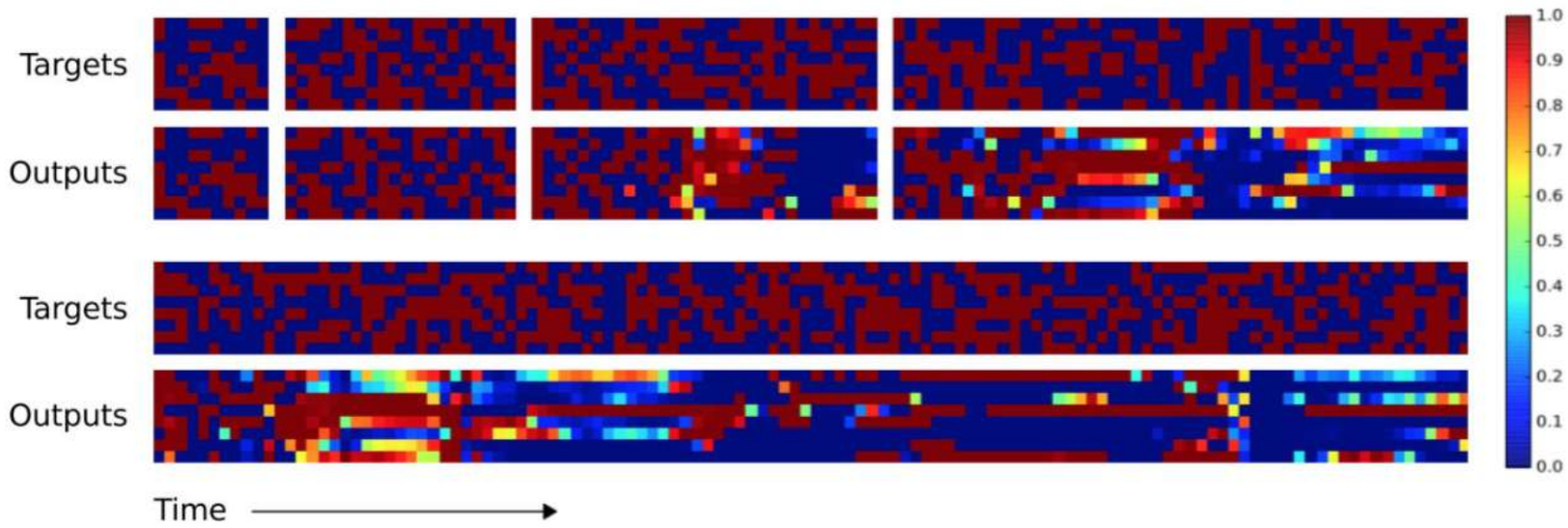


Figure 5: LSTM Generalisation on the Copy Task. The plots show inputs and outputs for the same sequence lengths as Figure 4. Like NTM, LSTM learns to reproduce sequences of up to length 20 almost perfectly. However it clearly fails to generalise to longer sequences. Also note that the length of the accurate prefix decreases as the sequence length increases, suggesting that the network has trouble retaining information for long periods.

Results: Copy

Store and recall a long sequence of arbitrary information

NTM

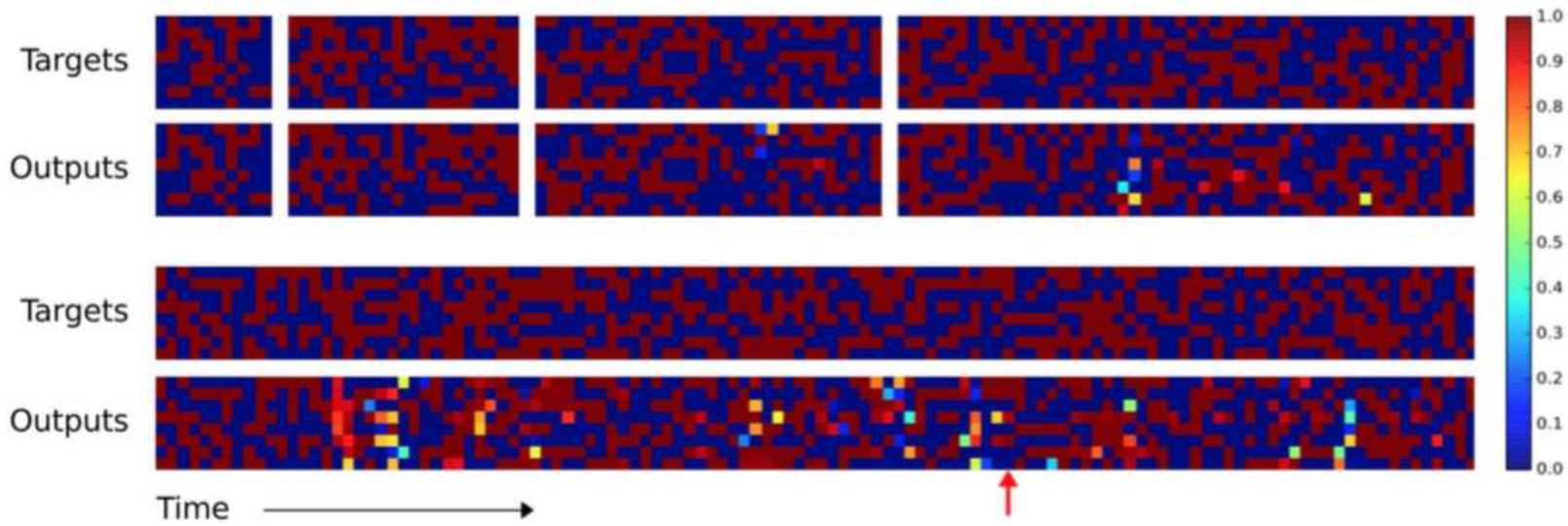
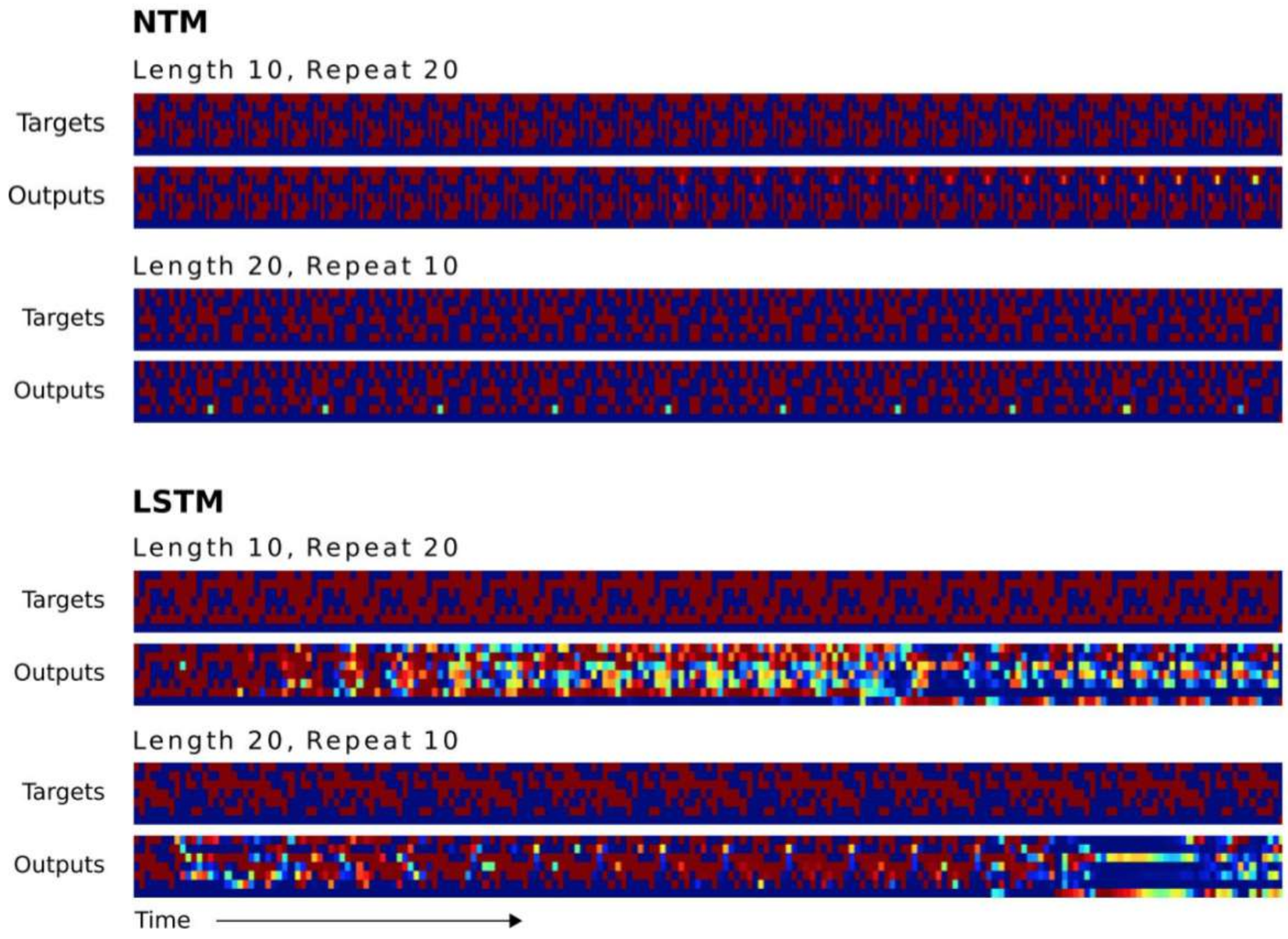


Figure 4: NTM Generalisation on the Copy Task. The four pairs of plots in the top row depict network outputs and corresponding copy targets for test sequences of length 10, 20, 30, and 50, respectively. The plots in the bottom row are for a length 120 sequence. The network was only trained on sequences of up to length 20. The first four sequences are reproduced with high confidence and very few mistakes. The longest one has a few more local errors and one global error: at the point indicated by the red arrow at the bottom, a single vector is duplicated, pushing all subsequent vectors one step back. Despite being subjectively close to a correct copy, this leads to a high loss.

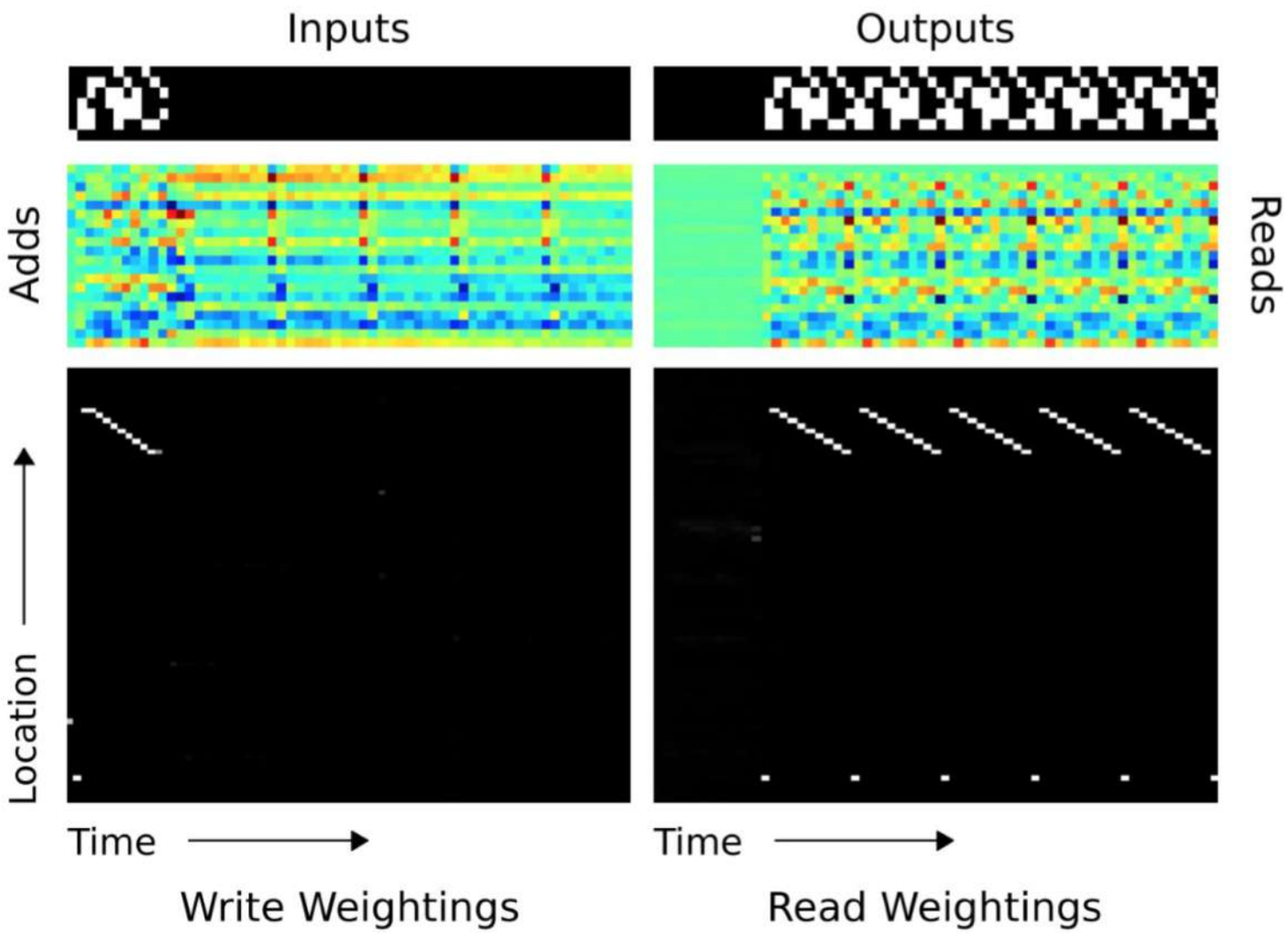
Results: Repeat Copy

Output the copied sequence a specified number of times and then emit an end-of-sequence marker



Results: Repeat Copy

Output the copied sequence a specified number of times and then emit an end-of-sequence marker

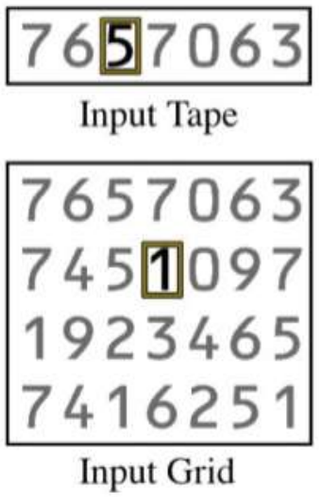


Learning Simple Algorithms from Examples

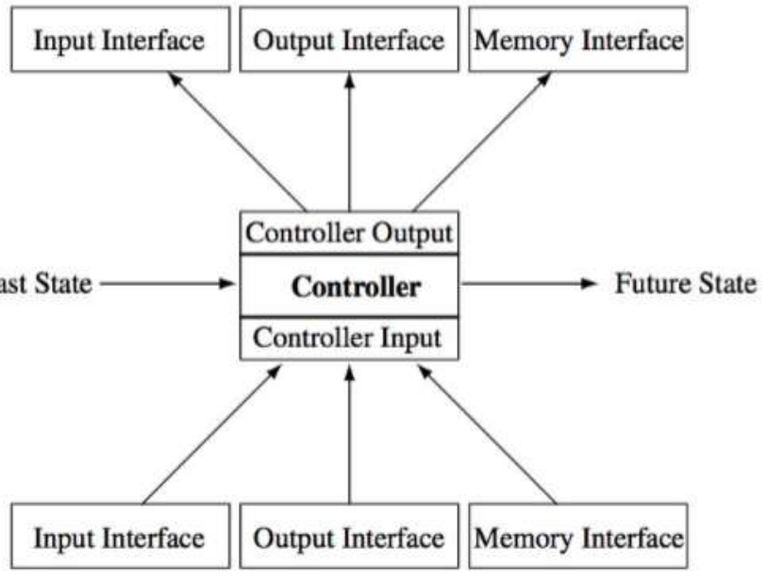
Wojciech Zaremba, Tomas Mikolov, Armand Joulin, Rob Fergus

ICML 2016

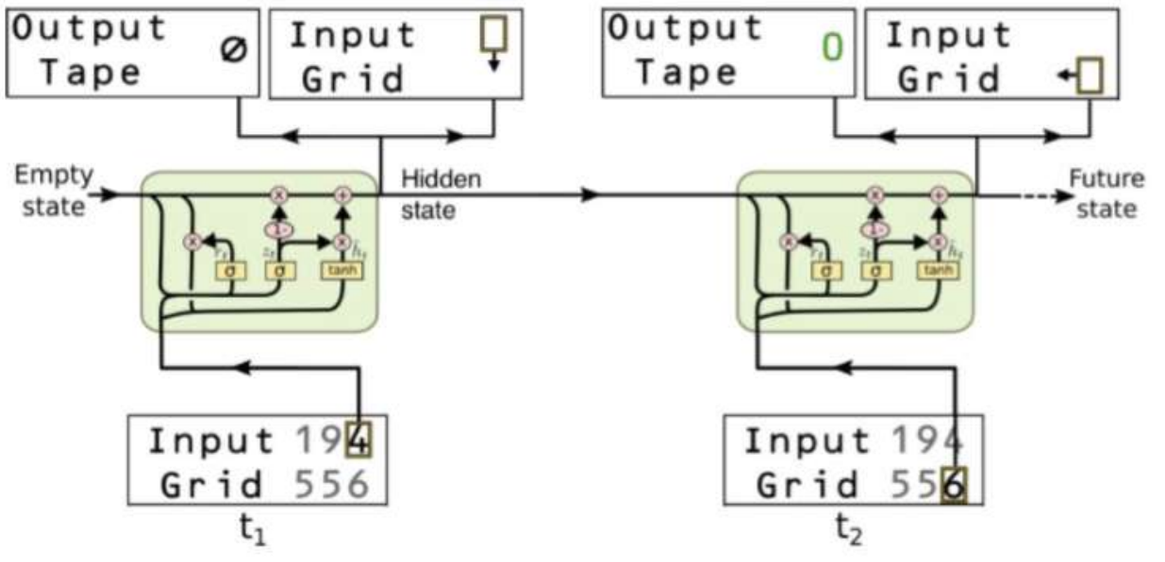
Model & Input/Output Interface



(a)



(b)



(c)

Tasks

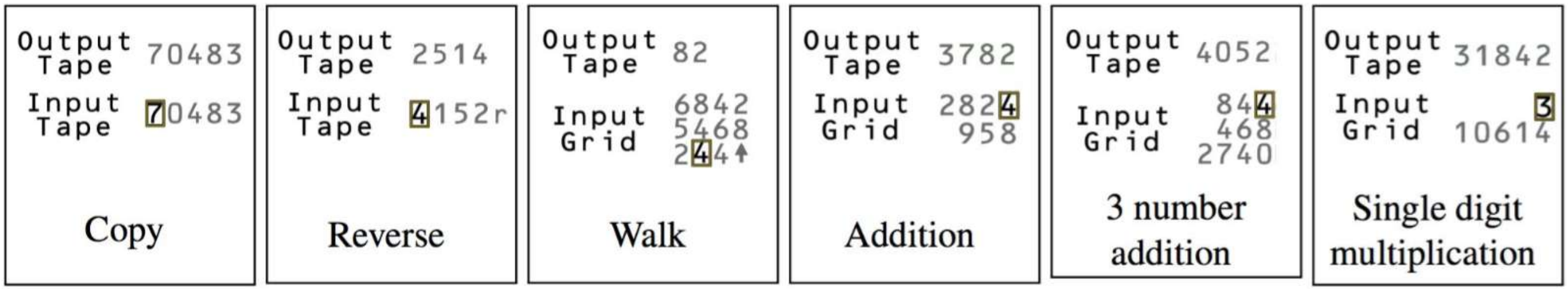


Figure 2: Examples of the six tasks, presented in their initial state. The yellow box indicates the starting position of the read head on the Input Interface. The gray characters on the Output Tape are **target** symbols used in training.

Results

Green: feed-forward
Red: GRU controller
Yellow: LSTM controller

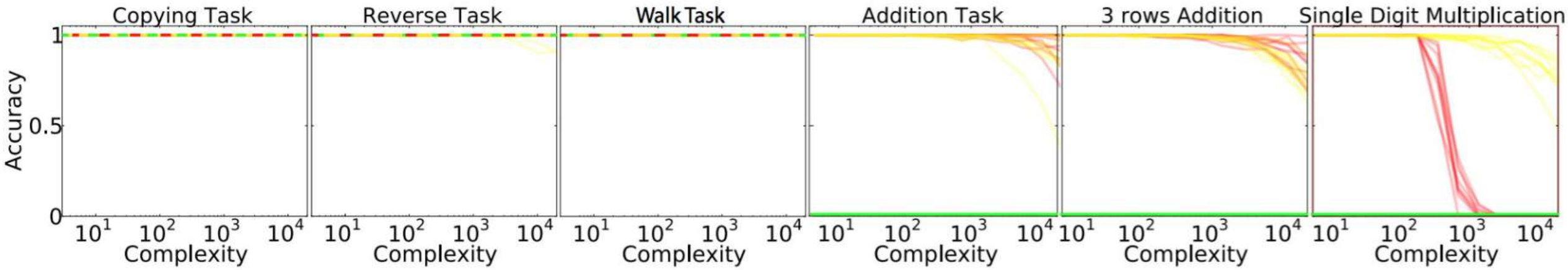


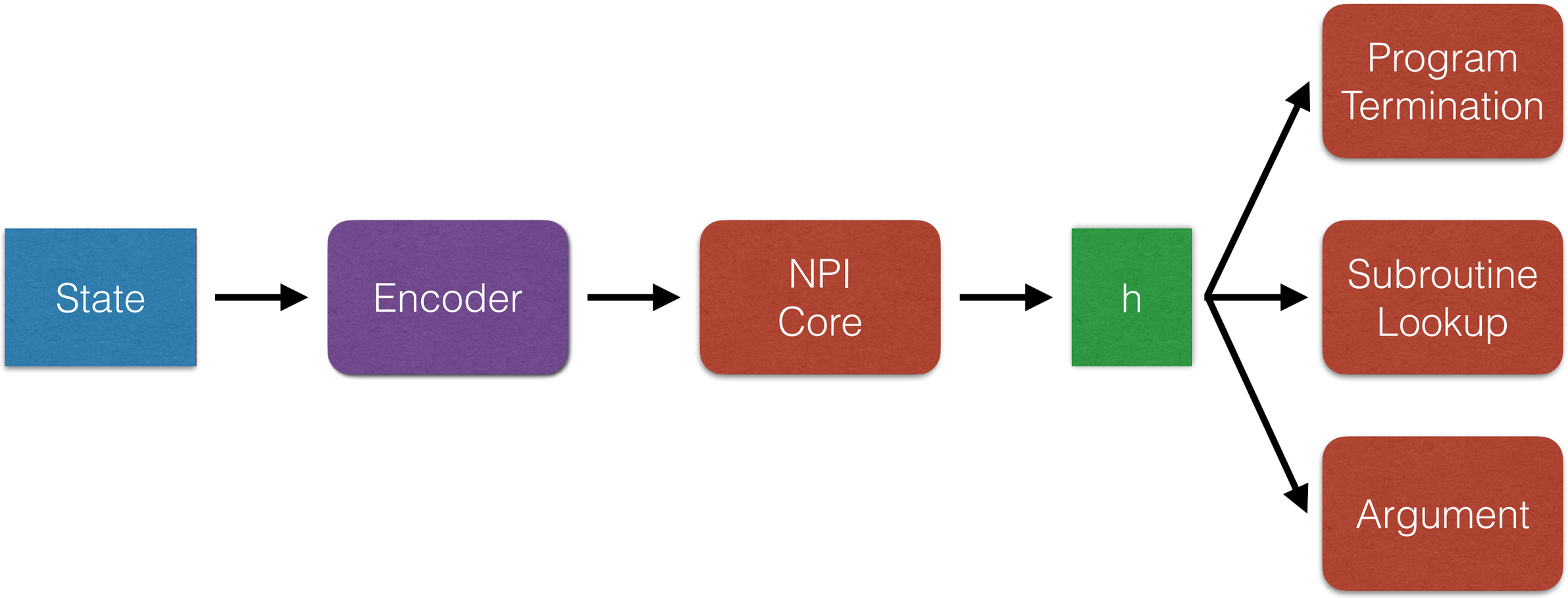
Figure 3: Test accuracy for all tasks with **supervised** actions over 10 runs for feed-forward (green), GRU (red) and LSTM (yellow) controllers. In this setting the optimal policy is provided. Complexity is the number of time steps required to compute the solution. Every task has slightly different conversion factor between complexity and the sequence length: a complexity of 10^4 for copy and walk would mean 10^4 input symbols; for reverse would correspond to $\frac{10^4}{2}$ input symbols; for addition would involve two $\frac{10^4}{2}$ long numbers; for 3 row addition would involve three $\frac{10^4}{3}$ long numbers and for single digit multiplication would involve a single 10^4 long number.

Neural Programmer-Interpreters

Scott Reed, Nando de Freitas

ICLR 2016

Model


State
Encoder
NPI Core
h
Program Termination
Subroutine Lookup
Argument

Task: Canonicalizing 3D Car Models

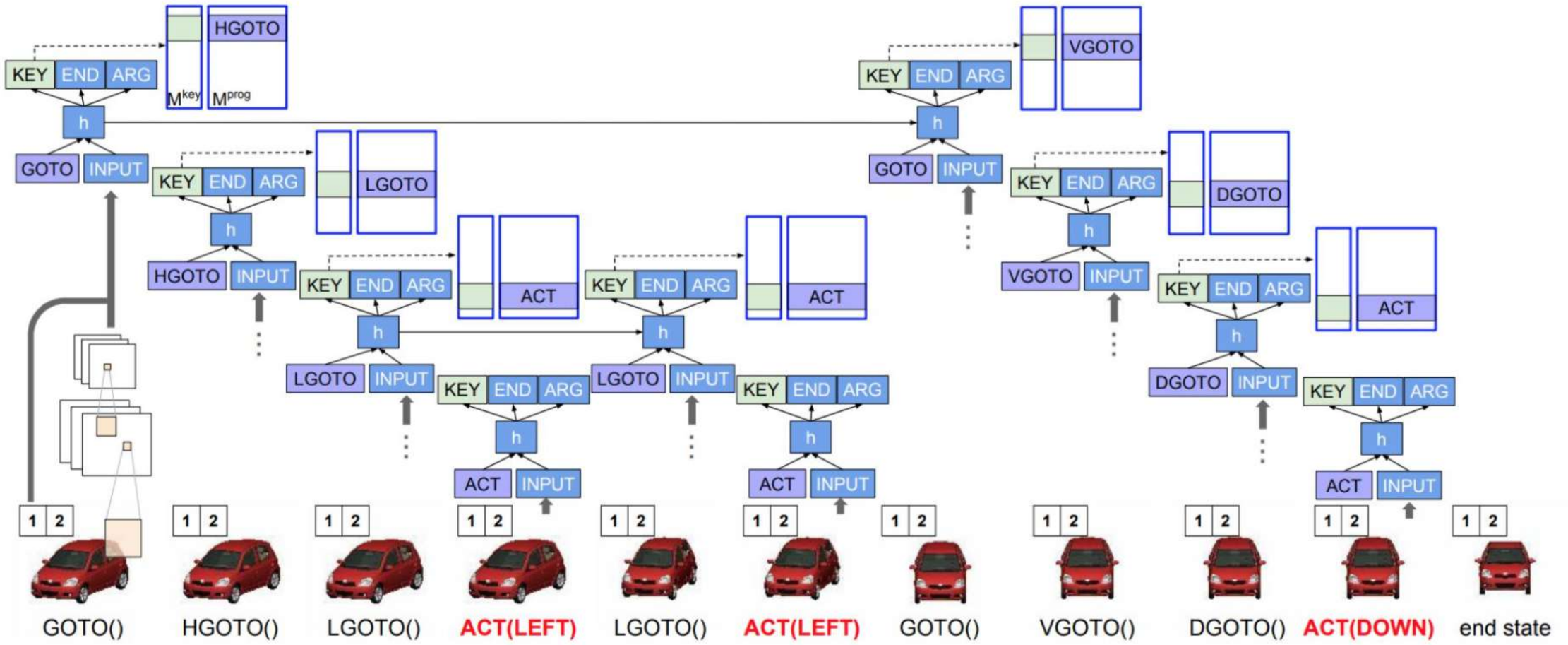


Figure 1: Example execution of canonicalizing 3D car models. The task is to move the camera such that a target angle and elevation are reached. There is a read-only scratch pad containing the target (angle 1, elevation 2 here). The image encoder is a convnet trained from scratch on pixels.

Task: Digit Addition

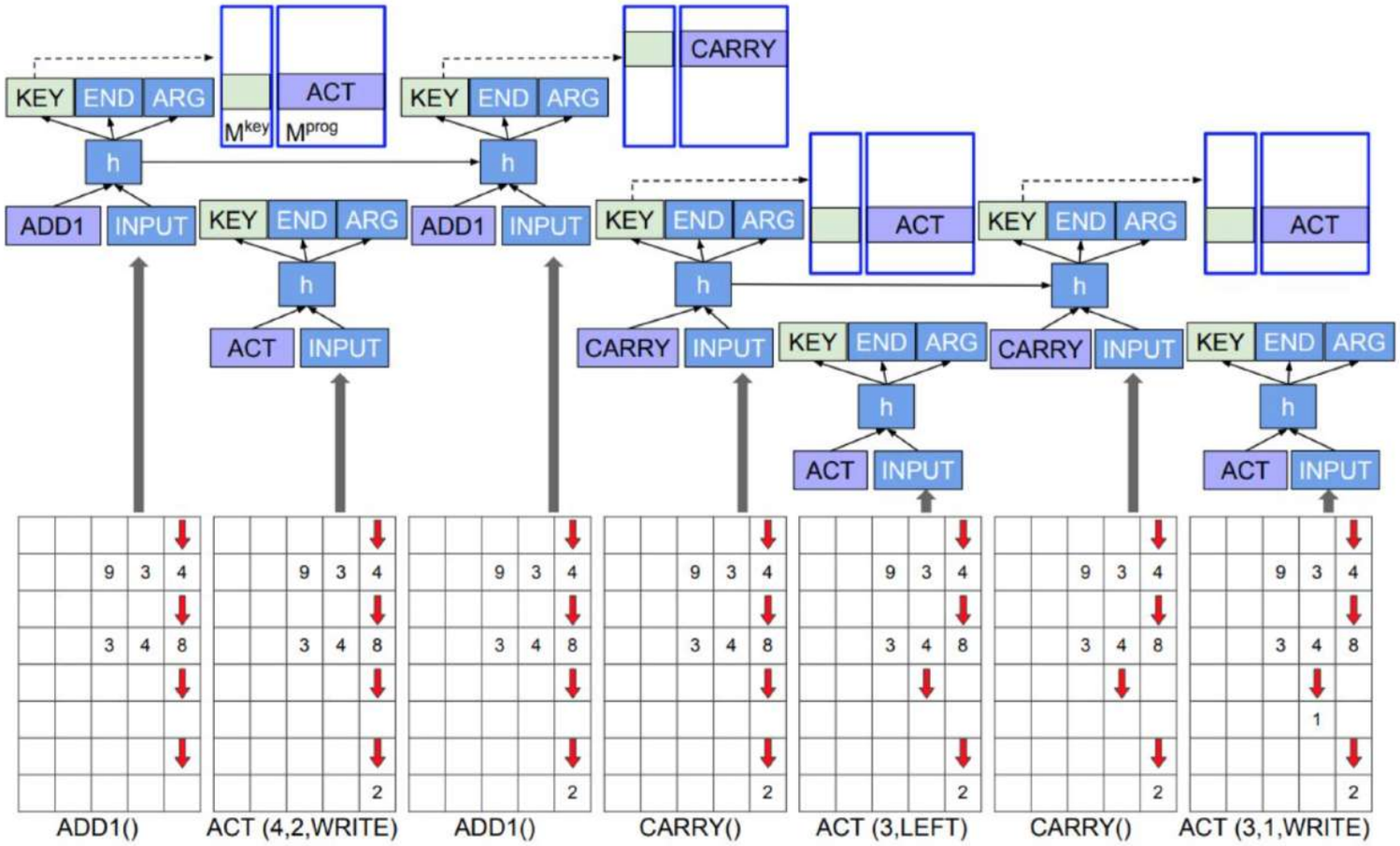


Figure 2: Example execution trace of single-digit addition. The task is to perform a single-digit add on the numbers at pointer locations in the first two rows. The carry (row 3) and output (row 4) should be updated to reflect the addition. At each time step, an observation of the environment (viewed from each pointer on a scratch pad) is encoded into a fixed-length vector.

Results

Require less data

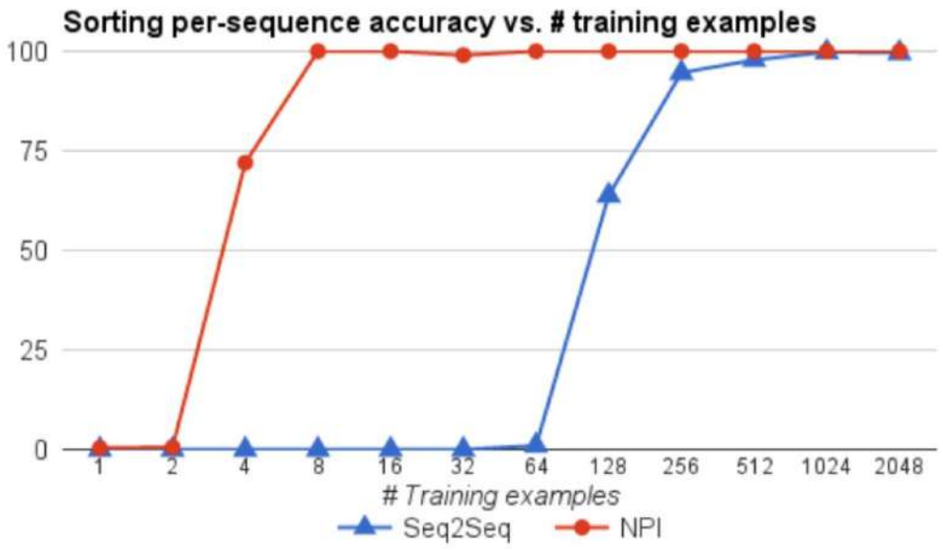


Figure 5: **Sample complexity**. Test accuracy of sequence-to-sequence LSTM versus NPI on length-20 arrays of single-digit numbers. Note that NPI is able to mine and train on subprogram traces from each bubblesort example.

Generalize to longer sequences

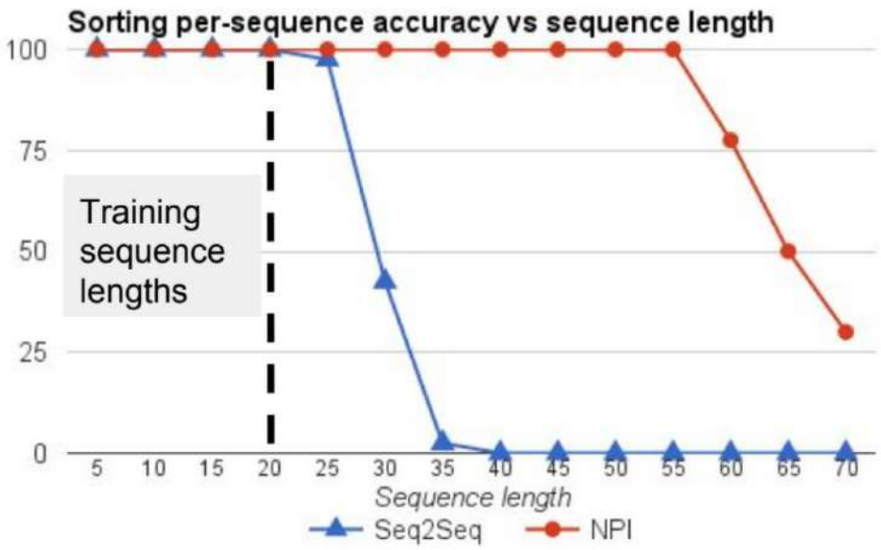


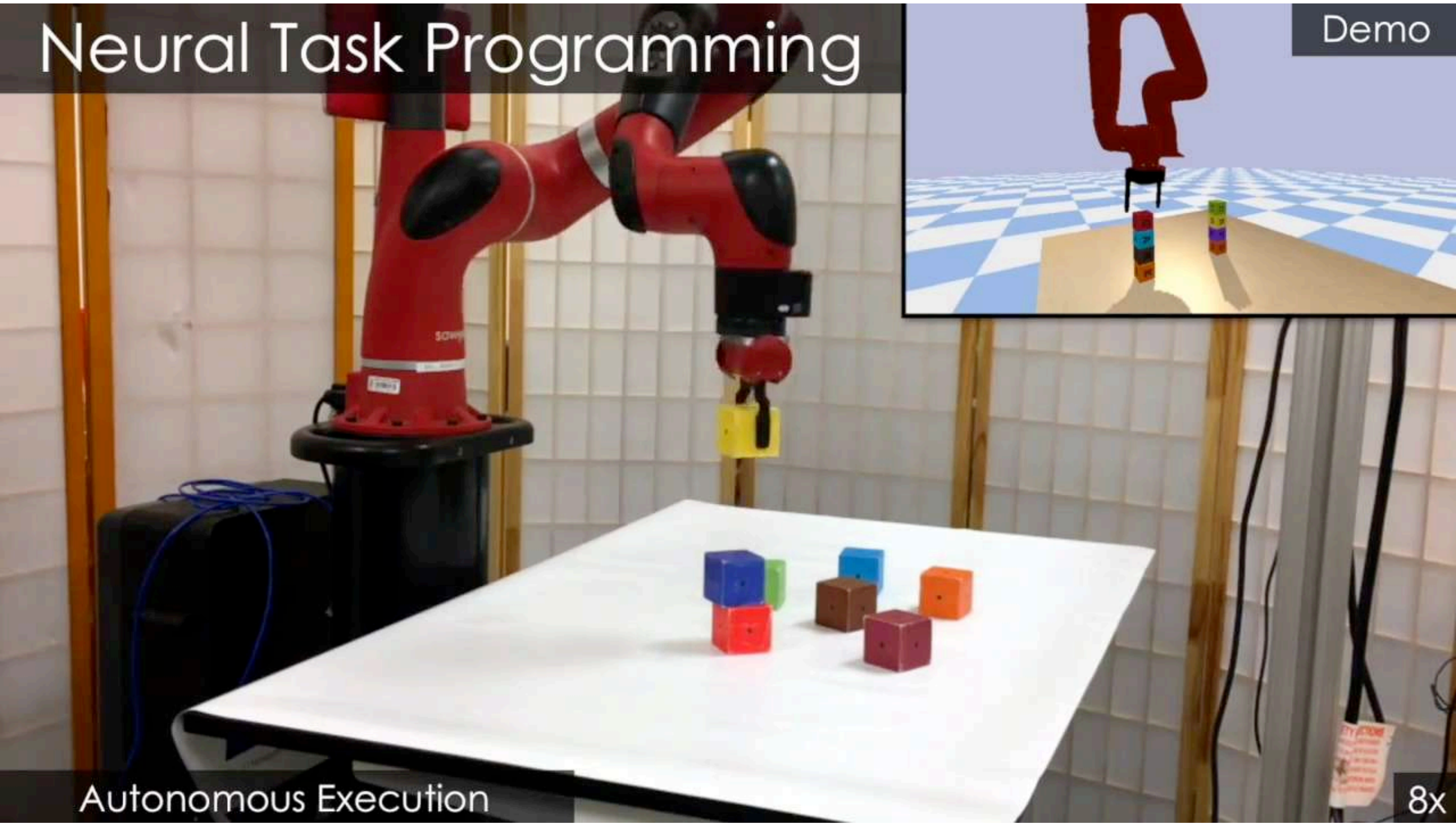
Figure 6: **Strong vs. weak generalization**. Test accuracy of sequence-to-sequence LSTM versus NPI on varying-length arrays of single-digit numbers. Both models were trained on arrays of single-digit numbers up to length 20.

Neural Task Programming: Learning to Generalize Across Hierarchical Tasks

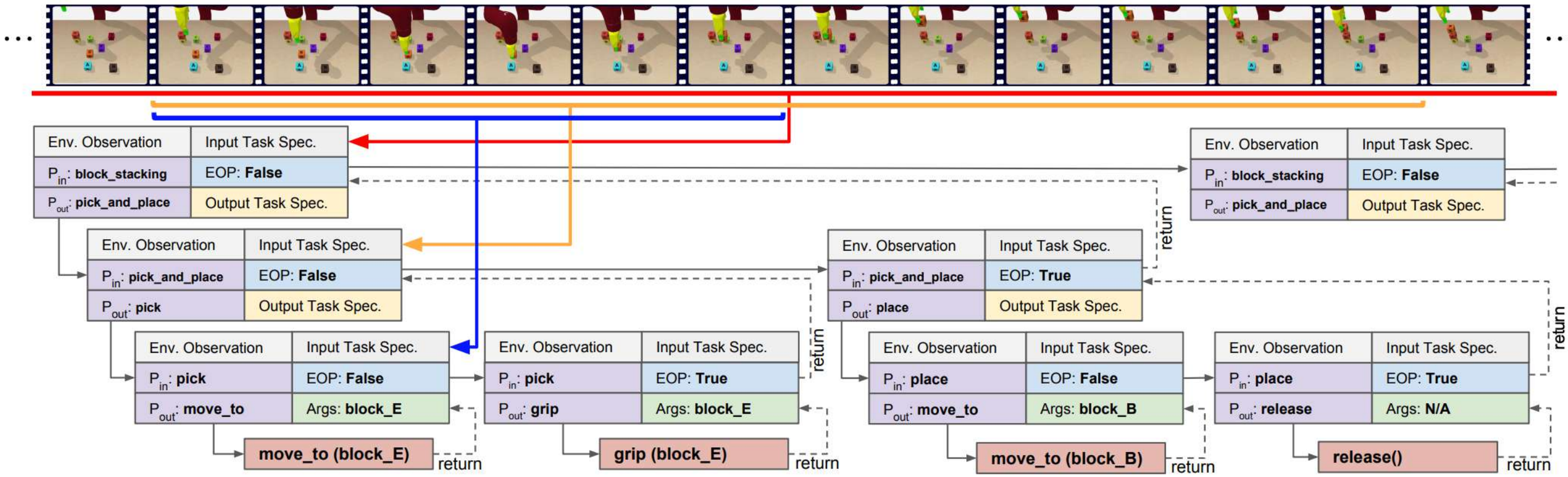
Danfei Xu, Suraj Nair, Yuke Zhu, Julian Gao,
Animesh Garg, Li Fei-Fei, Silvio Savarese

ICRA 2018

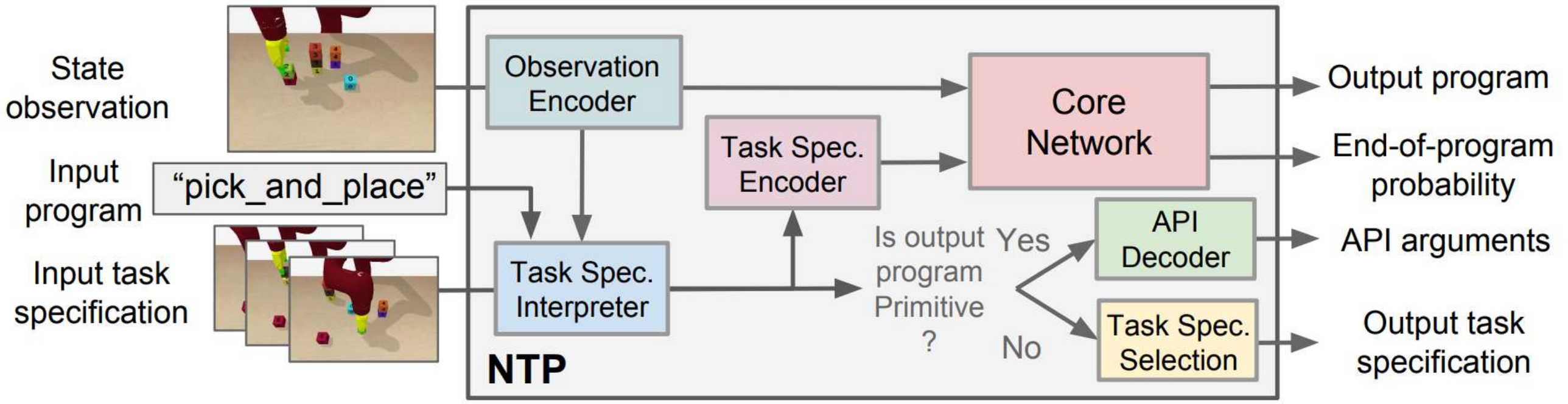
Problem



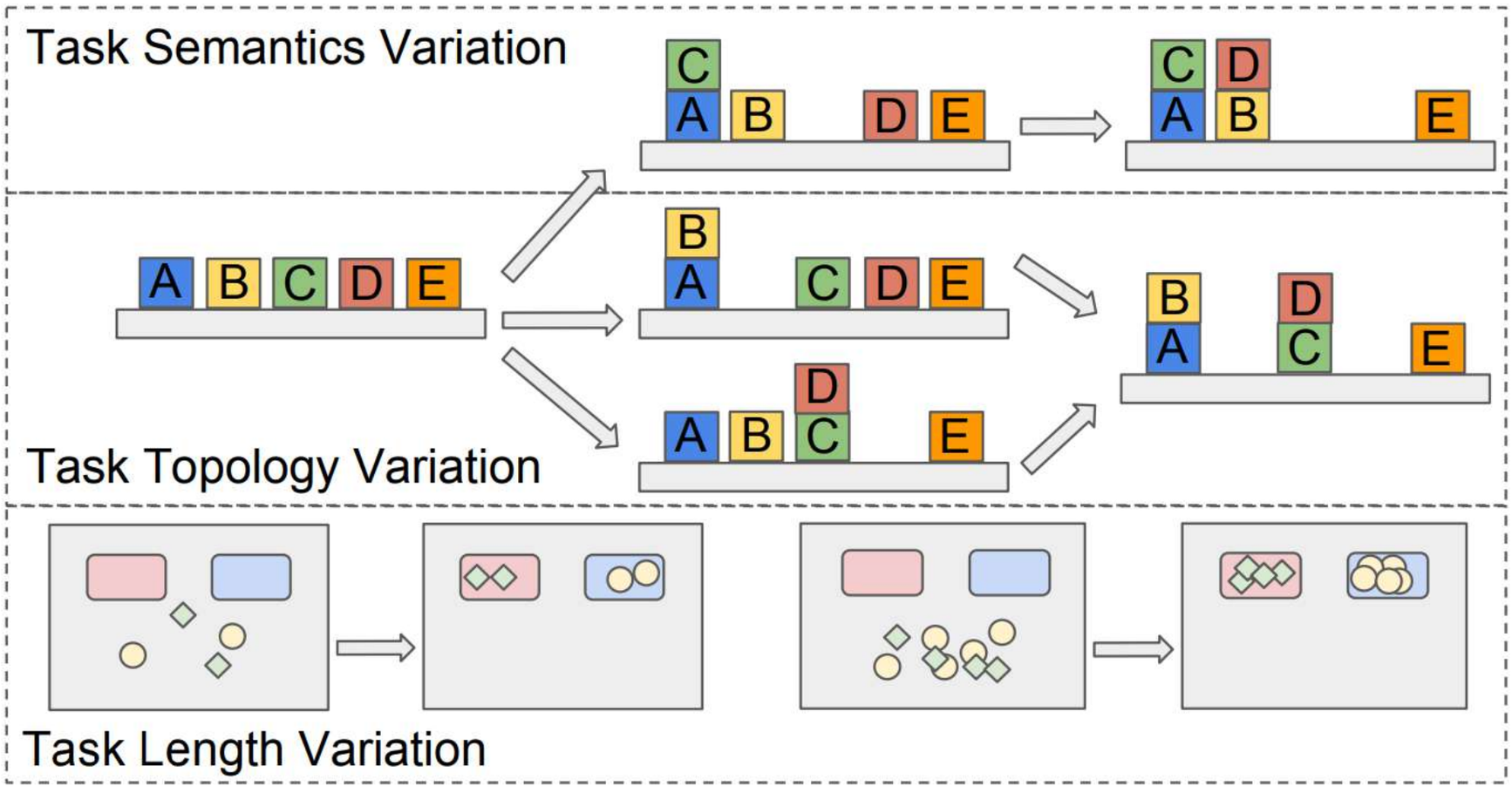
Task Decomposition



Model


State observation
Input program
"pick_and_place"
Input task specification
Observation Encoder
Task Spec. Encoder
Task Spec. Interpreter
Core Network
Is output program Primitive ?
Yes
No
API Decoder
Task Spec. Selection
Output program
End-of-program probability
API arguments
Output task specification
NTP

Task Variations



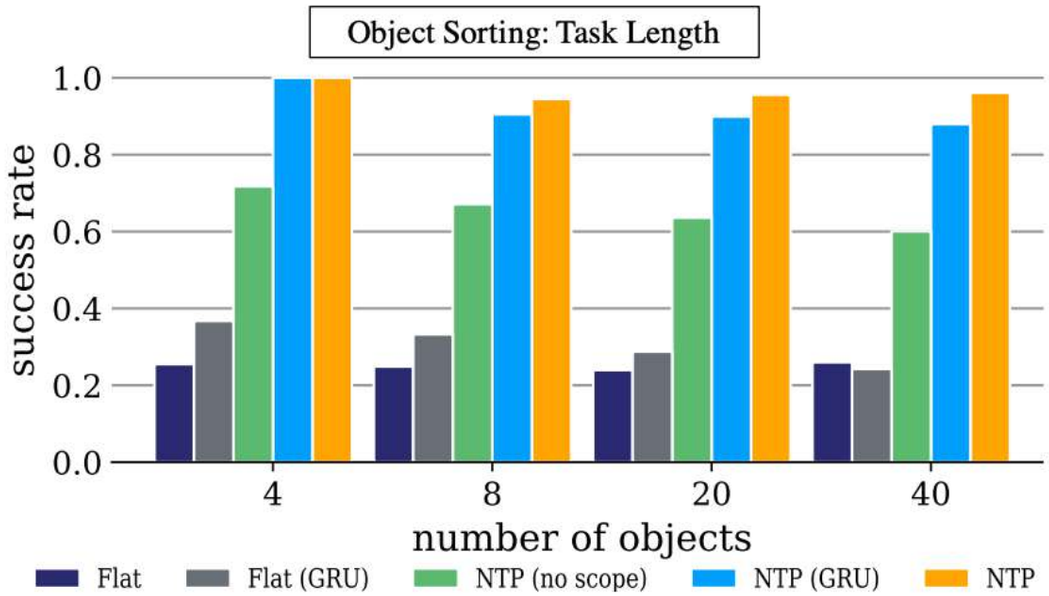
Different final states

Different orders of intermediate states

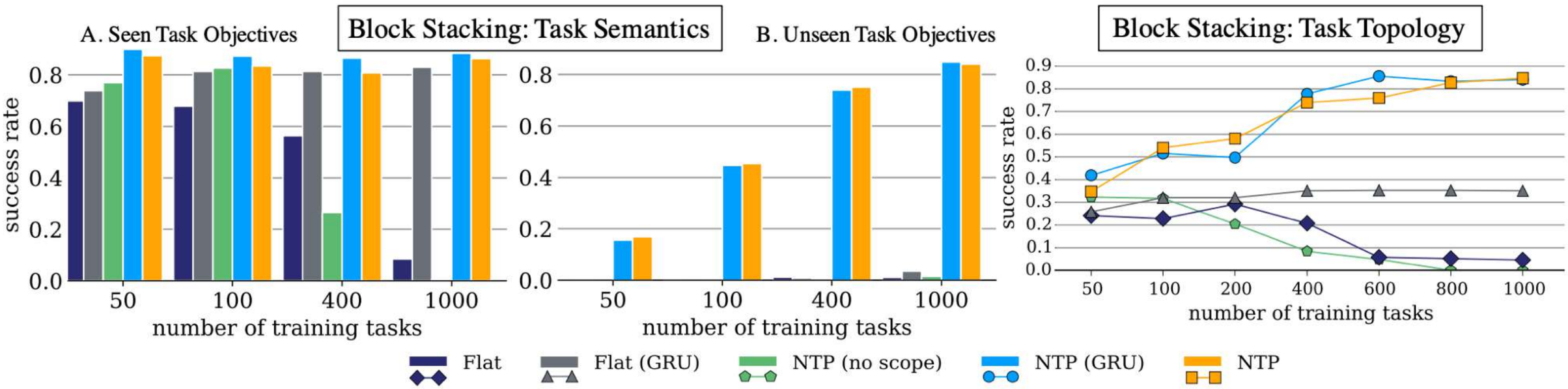
Different lengths

Results

Object Sorting



Block Stacking



Summary - Neural Program Induction

1. Design special network architectures

- Neural Turing Machines
- Learning Simple Algorithms from Examples

2. Provide heavy supervision to modular networks

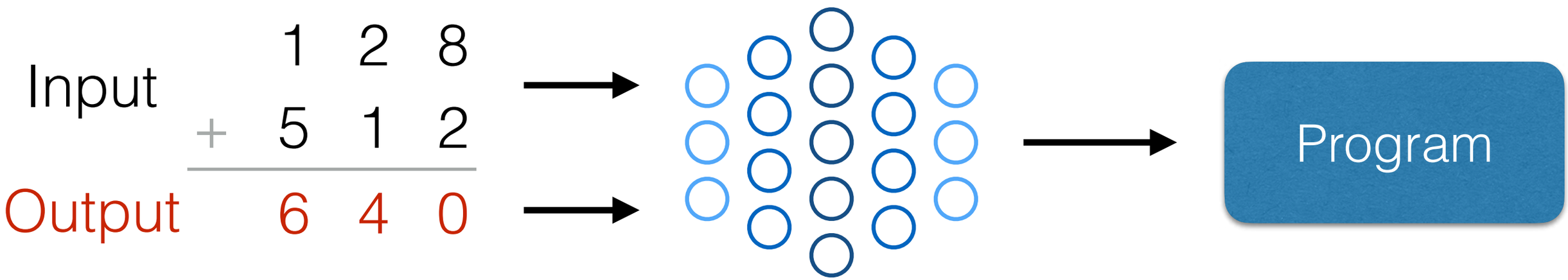
- Neural Programmer-Interpreters
- Neural Task Programming: Learning to Generalize Across Hierarchical Tasks

Outline

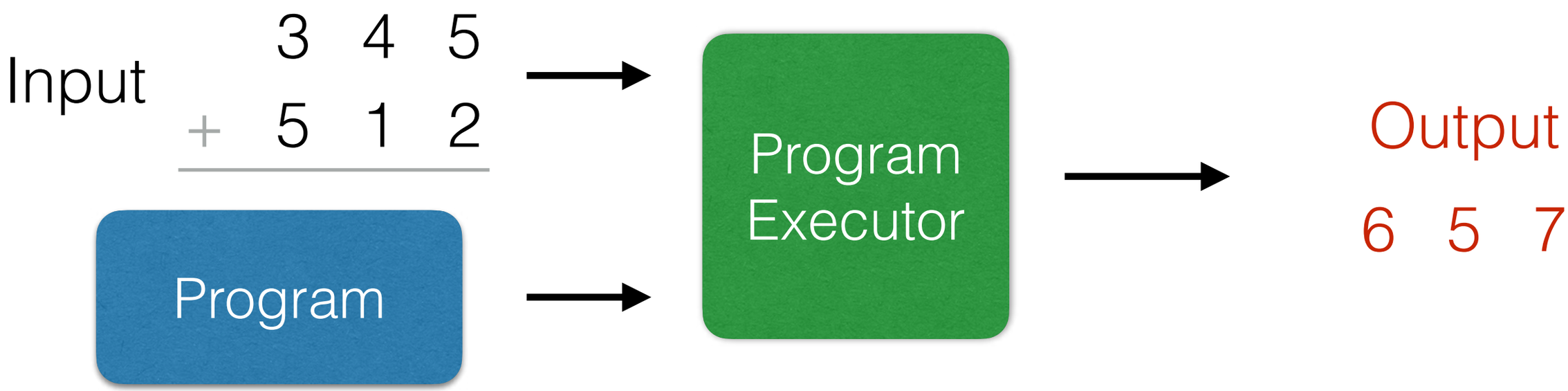
- Recent Advance in Deep Learning
- Teach Neural Networks to Learn Rules
- An Overview
- Neural Program Induction
- Neural Program Synthesis

Neural Program Synthesis

1. Infer the underlying program that can solve the task

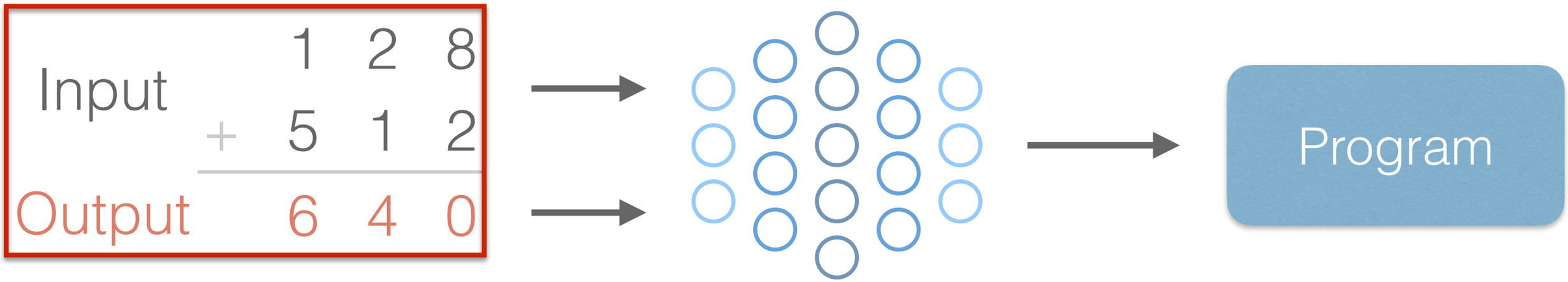


2. Solve the task by executing the program

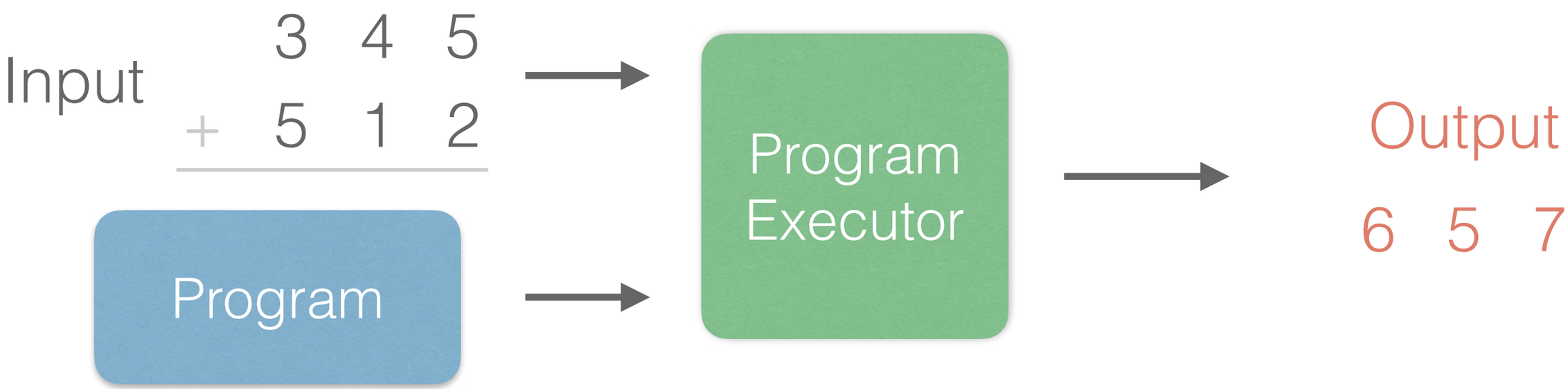


Task Specifications

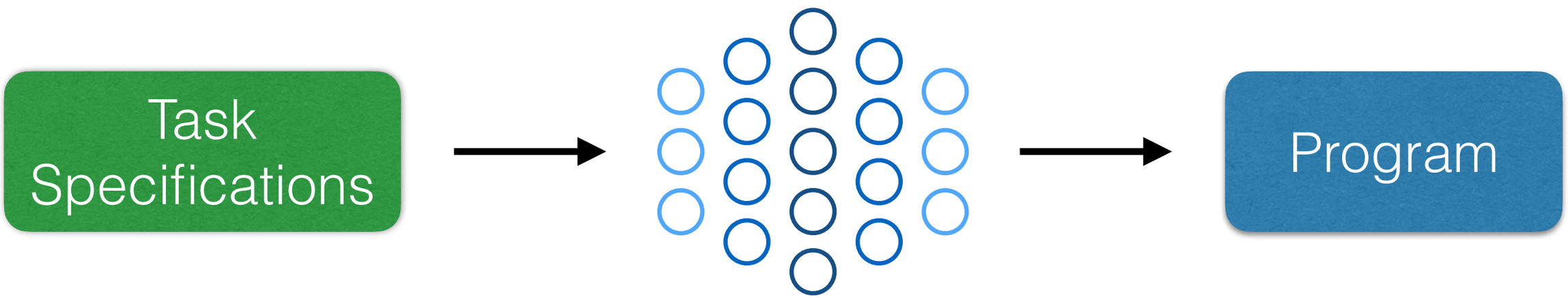
1. Infer the underlying program that can solve the task



2. Solve the task by executing the program

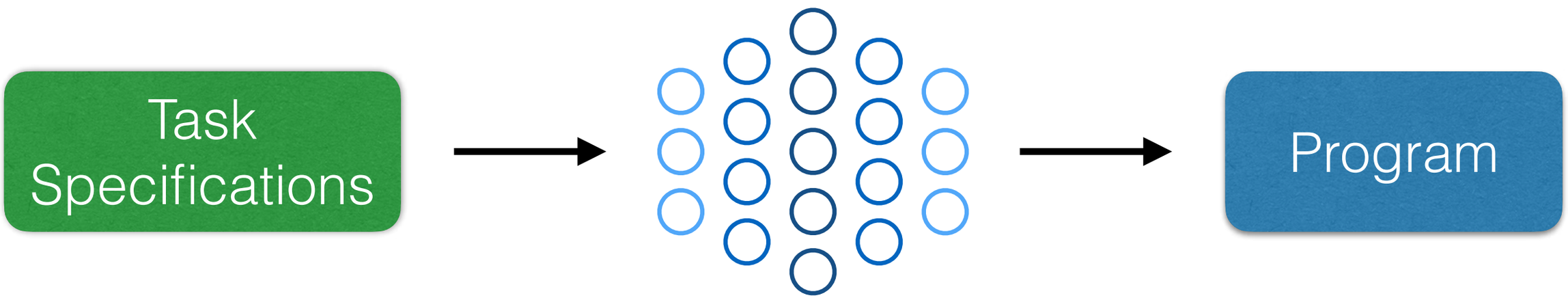


Task Specifications



- **I/O pairs**: I/O strings, I/O images
- **Execution traces**: demonstration videos
- **States**: images, voxels
- **Language descriptions**
- **Reward function**

Task Specifications



- **I/O pairs**: I/O strings, I/O images
- Execution traces: demonstration videos
- States: images, voxels
- Language descriptions
- Reward function

RobustFill: Neural Program Learning under Noisy I/O

Jacob Devlin, Jonathan Uesato, Surya Bhupatiraju, Rishabh Singh,
Abdel-rahman Mohamed, Pushmeet Kohli

ICML 2017

Motivation & Problem Formulation

Microsoft wants to make their Excel more powerful

$I_1 = \texttt{January}$	$O_1 = \texttt{jan}$
$I_2 = \texttt{February}$	$O_2 = \texttt{feb}$
$I_3 = \texttt{March}$	$O_3 = \texttt{mar}$
$I_1^y = \texttt{April}$ $I_2^y = \texttt{May}$	$O_1^y = \texttt{apr}$ $O_2^y = \texttt{may}$
$P = \texttt{ToCase(Lower, SubStr(1,3))}$	

← Neural Program Synthesis

DeepCoder: Learning to Write Programs

Matej Balog, Alexander L. Gaunt, Marc Brockschmidt, Sebastian Nowozin, Daniel Tarlow

ICLR 2017

Learning + Search-based Program Synthesis

Example

```
a ← [int]                    An input-output example:
b ← FILTER (<0) a            Input:
c ← MAP (*4) b               [-17, -3, 4, 11, 0, -5, -9, 13, 6, 6, -8, 11]
d ← SORT c                   Output:
e ← REVERSE d                [-12, -20, -32, -36, -68]
```

Neural network predicts the probability of each function appearing in the source code (multi-label classification under a rank loss)

Predicted Attributes

`(*4), (>0), MAP, FLITER, SORT, REVERSE`

×

Search Algorithm

DFS, Sketch, etc.

→

Program

Search Speedup

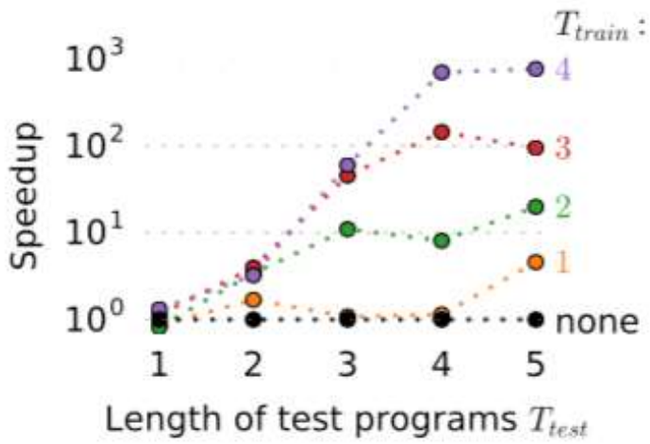
Program length = 3

Timeout needed	DFS			Enumeration			λ^2			Sketch		Beam
to solve	20%	40%	60%	20%	40%	60%	20%	40%	60%	20%	40%	20%
Baseline	$41ms$	$126ms$	$314ms$	$80ms$	$335ms$	$861ms$	$18.9s$	$49.6s$	$84.2s$	$>10^3s$	$>10^3s$	$>10^3s$
DeepCoder	$2.7ms$	$33ms$	$110ms$	$1.3ms$	$6.1ms$	$27ms$	$0.23s$	$0.52s$	$13.5s$	$2.13s$	$455s$	$292s$
Speedup	**15.2×**	**3.9×**	**2.9×**	**62.2×**	**54.6×**	**31.5×**	**80.4×**	**94.6×**	**6.2×**	**>467×**	**>2.2×**	**>3.4×**

Program length = 5

Timeout needed	DFS			Enumeration			λ^2
to solve	20%	40%	60%	20%	40%	60%	20%
Baseline	$163s$	$2887s$	$6832s$	$8181s$	$>10^4s$	$>10^4s$	$463s$
DeepCoder	$24s$	$514s$	$2654s$	$9s$	$264s$	$4640s$	$48s$
Speedup	**6.8×**	**5.6×**	**2.6×**	**907×**	**>37×**	**>2×**	**9.6×**

(a)



(b)

Leveraging Grammar and Reinforcement Learning for Neural Program Synthesis

Rudy Bunel, Matthew Hausknecht, Jacob Devlin, Rishabh Singh, Pushmeet Kohli

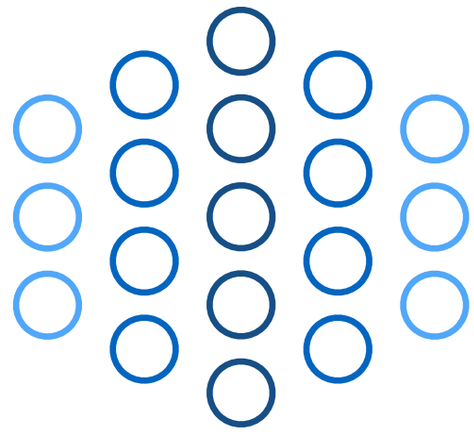
ICLR 2018

Motivation & Problem Formulation

Infer the program

Input$_{observed}$

Output$_{observed}$



Ground Truth Program

Program

Execute the program

Input$_{assessment}$

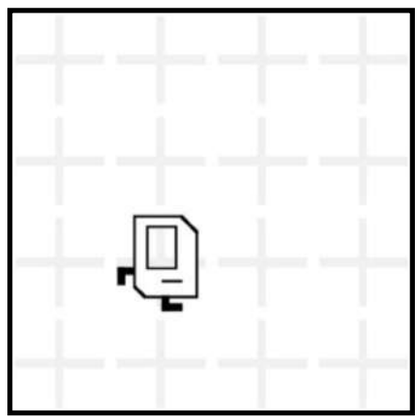
Program Executor

Output$_{predicted}$

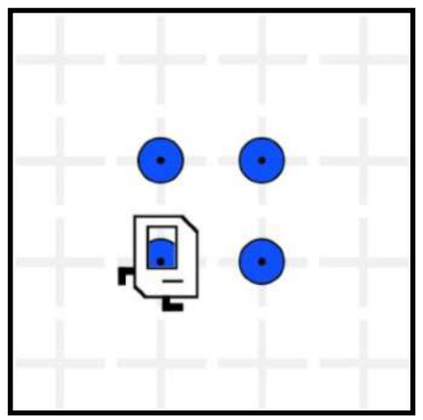
Output$_{assessment}$

Motivation - Program Aliasing

Input state



Output state



Program A

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

Program B

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

Ground Truth Program

Synthesized Program

Model Architecture

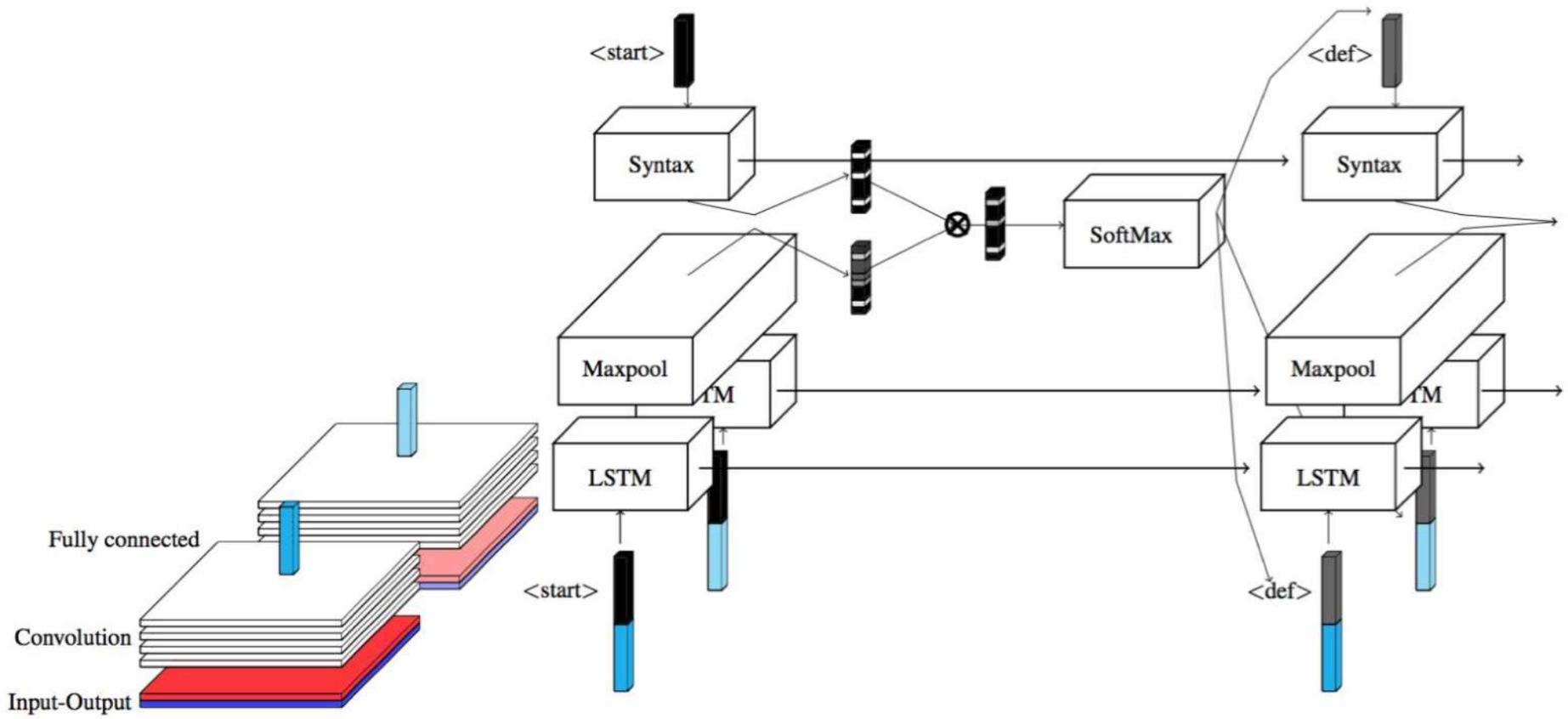
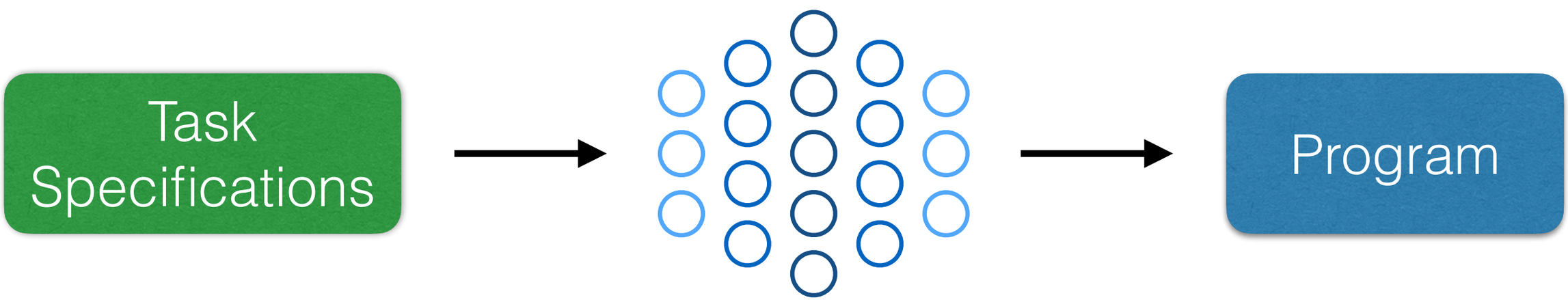


Figure 2: Architecture of our model. Each pair of Input-Output is embedded jointly by a CNN. One decoder LSTM is run for each example, getting fed in a concatenation of the previous token and the IO pair embedding (constant across timestep). Results of all the decoders are maxpooled and the prediction is modulated by the mask generated by the syntax model. The probability over the next token is then obtained by a Softmax transformation.

Fine-tune the Model using RL

	Full Dataset		Small Dataset	
Top-1	**Generalization**	**Exact Match**	**Generalization**	**Exact Match**
MLE	71.91	**39.94**	12.58	8.93
RL	68.39	34.74	0	0
RL_beam	75.72	8.21	**25.28**	**17.63**
RL_beam_div	76.20	31.25	23.72	16.31
RL_beam_div_opt	**77.12**	32.17	24.24	16.63

Task Specifications



- **I/O pairs**: I/O strings, I/O images
- **Execution traces**: demonstration videos
- **States**: images, voxels
- **Language descriptions**
- **Reward function**

Neural Program Synthesis from Diverse Demonstration Videos

Shao-Hua Sun, Hyeonwoo Noh, Sriram Somasundaram, Joseph J. Lim

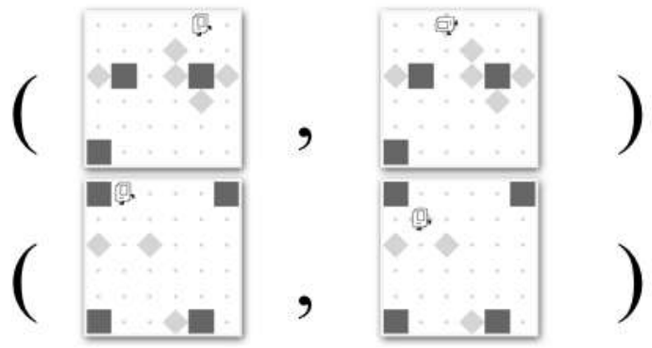
ICML 2018

Program Synthesis from I/O Pairs

Task specification

Program

Input/output pairs



```
DEF run()
    if isFontClear():
        move
    else:
        turnLeft
    move
    turnLeft
    repeat(2):
        turnRight
        putMarker
```


Devlin et al. "*Robustfill: Neural program learning under noisy i/o.*" ICML 2017
Balog, et al. "Deepcoder: Learning to write programs." ICLR 2017
Rudy R et al. "*Leveraging grammar and reinforcement learning for neural program synthesis.*" ICLR 2018

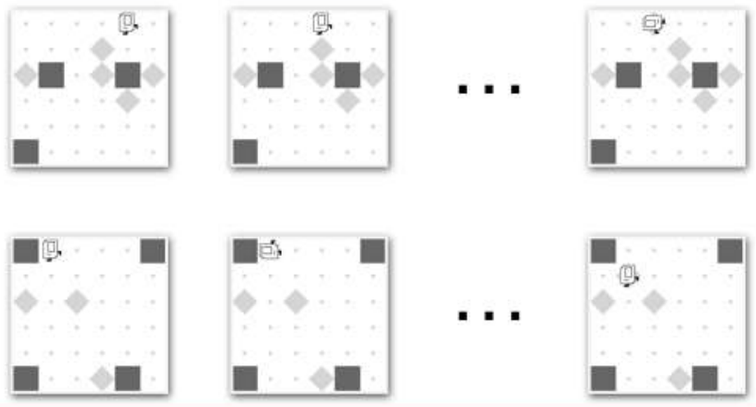
Program Synthesis from Demos

Task specification

Program

Input/output pairs

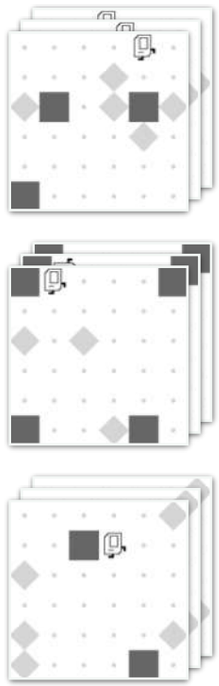
Demonstration Sequences



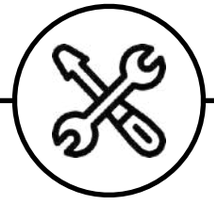
```
DEF run()
    if isFontClear():
        move
    else:
        turnLeft
    move
    turnLeft
    repeat(2):
        turnRight
        putMarker
```

Imitation Learning

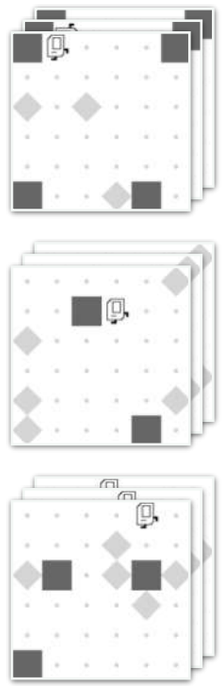
Demonstrations



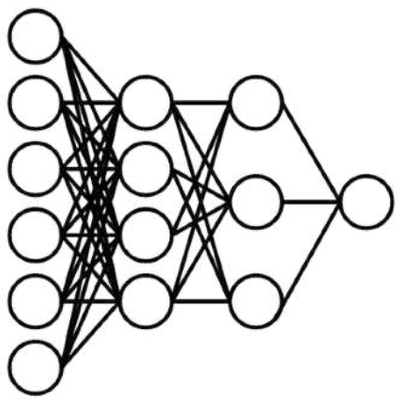
Imitate



Execution

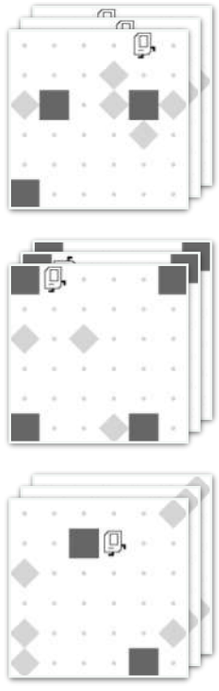


Neural Network
Policy



Imitation Learning by Synthesizing Programs

Demonstrations

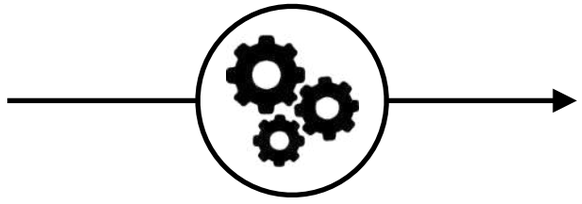


Synthesize

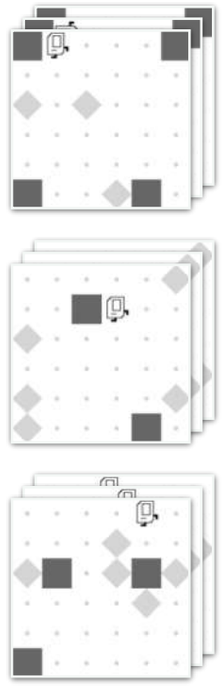
Program Policy

```
DEF run()
    if isFontClear():
        move
    else:
        turnLeft
    move
    turnLeft
    repeat(2):
        turnRight
        putMarker
```

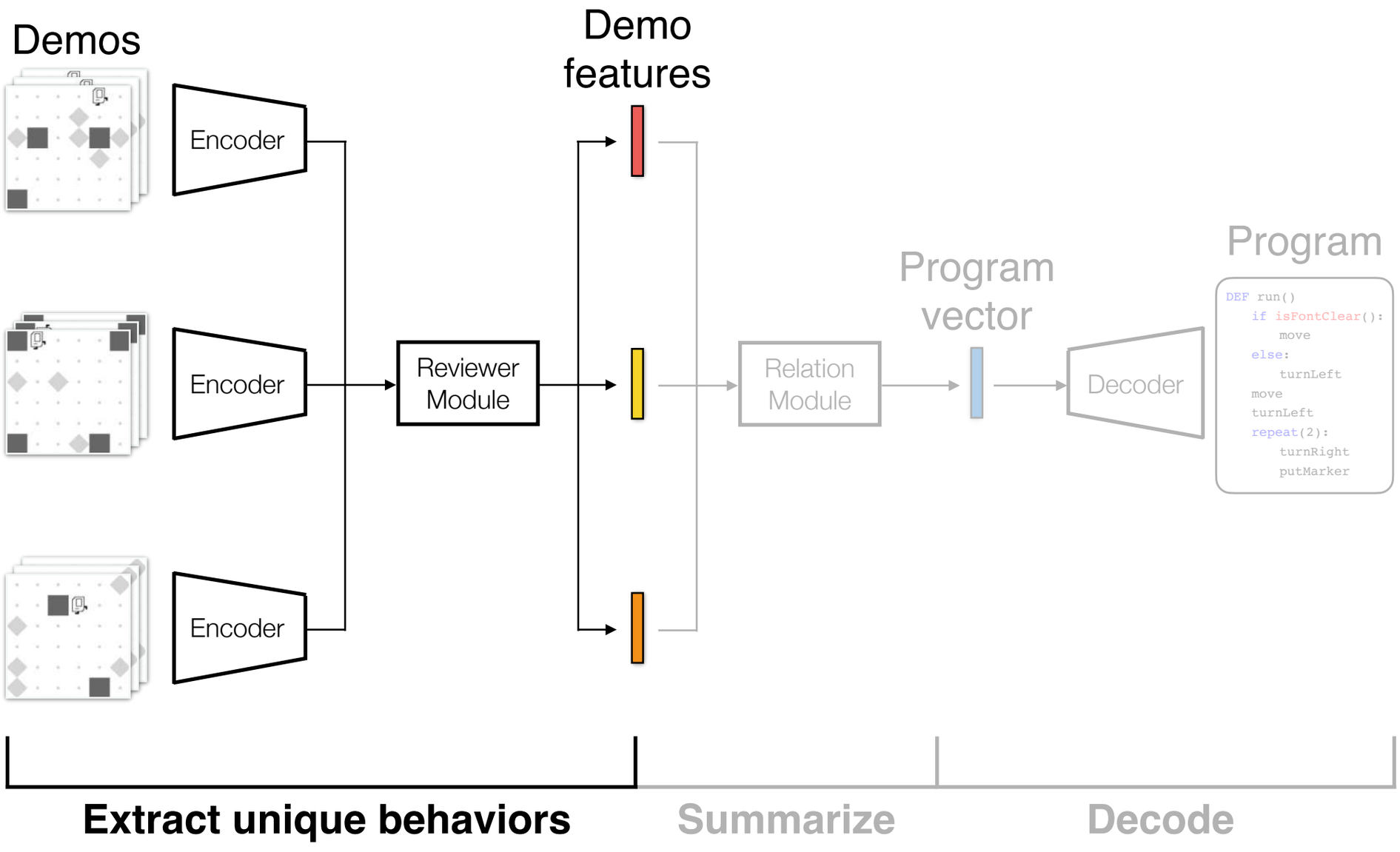
Execute



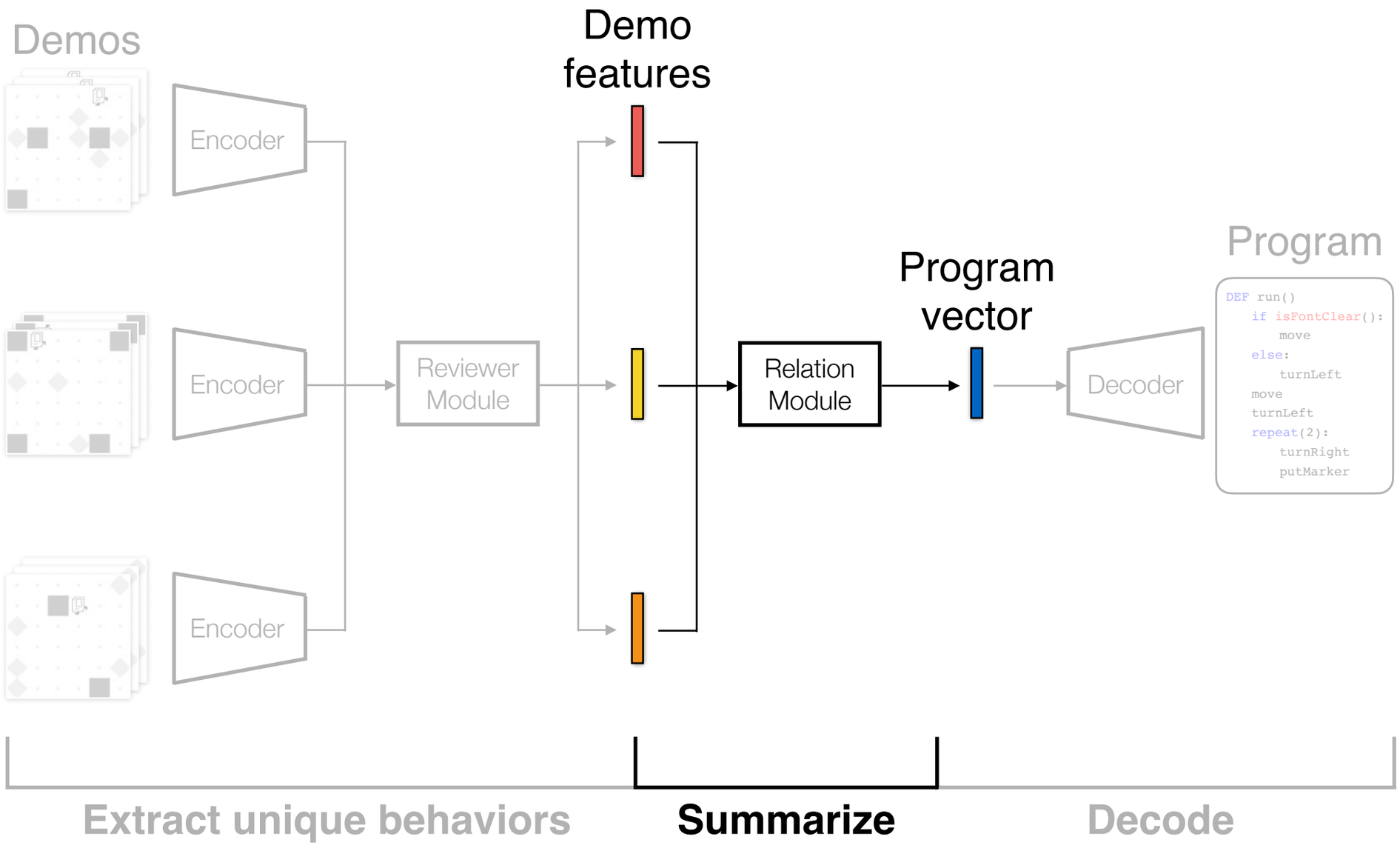
Execution



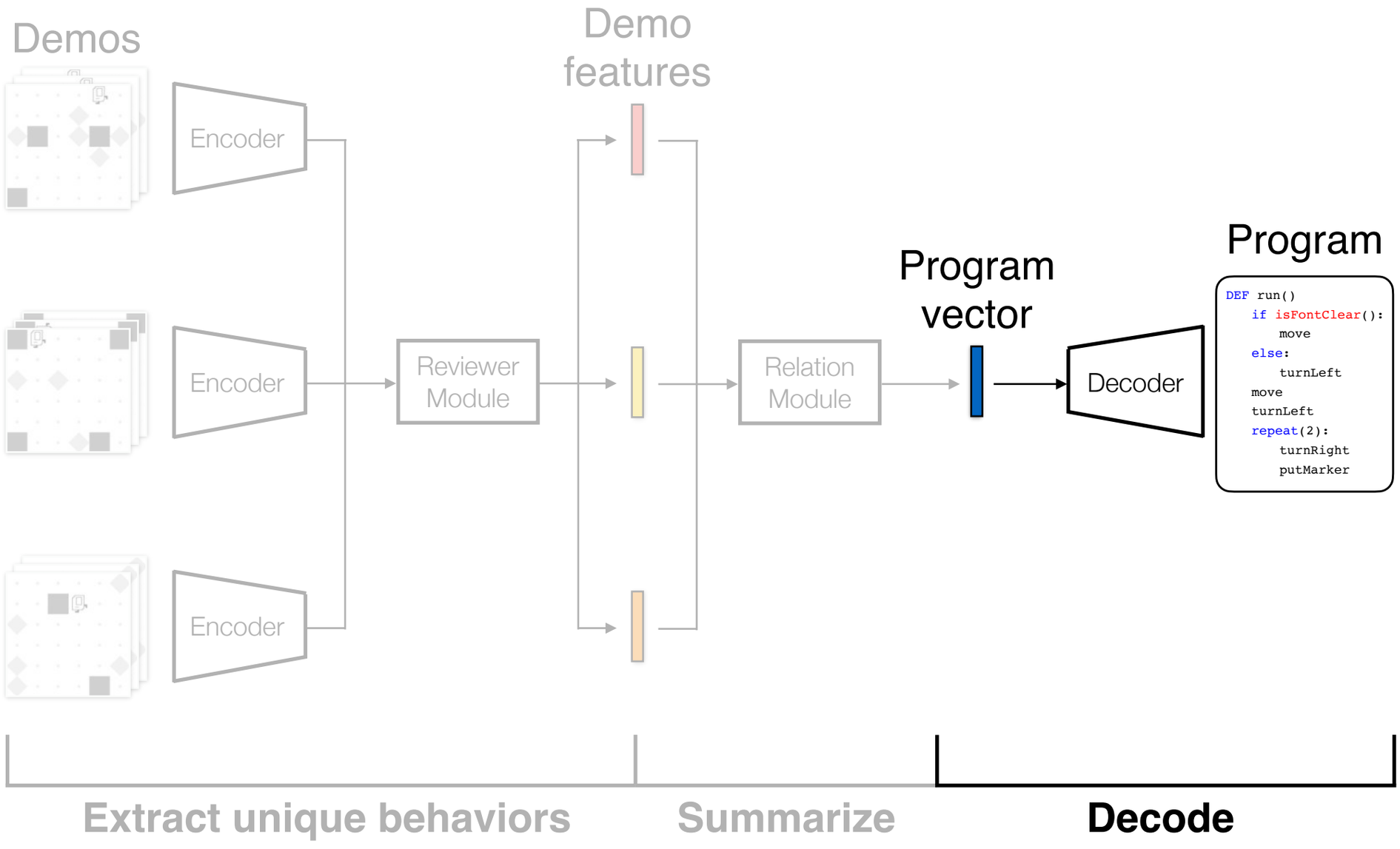
Model Overview



Model Overview



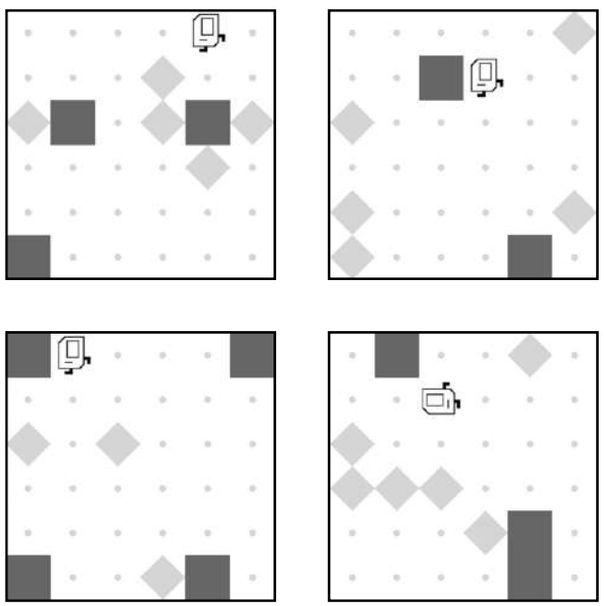
Model Overview



Environments

Karel

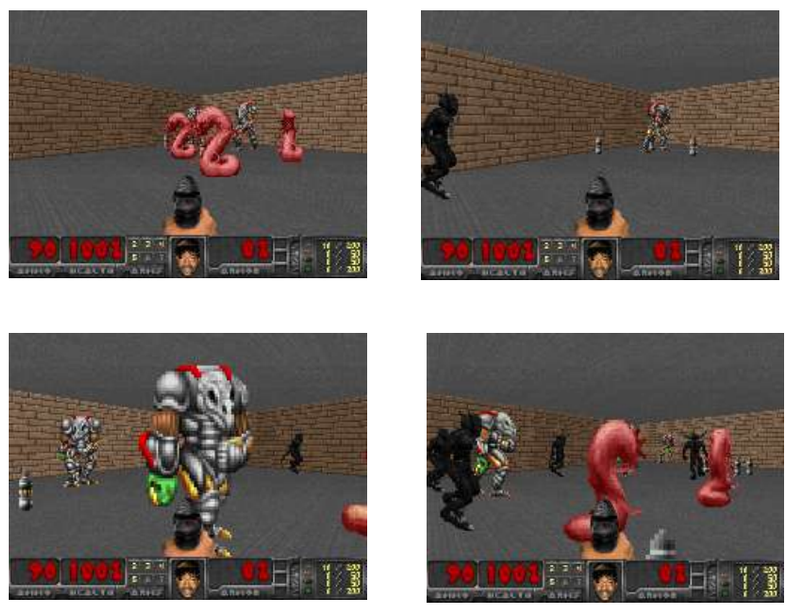
```
DEF run()
   if isFontClear():
      move
   else:
      turnLeft
   move
   turnLeft
   repeat(2):
      turnRight
      putMarker
```



Richard E Pattis. “Karel the robot: a gentle introduction to the art of programming.” John Wiley & Sons, Inc., 1981

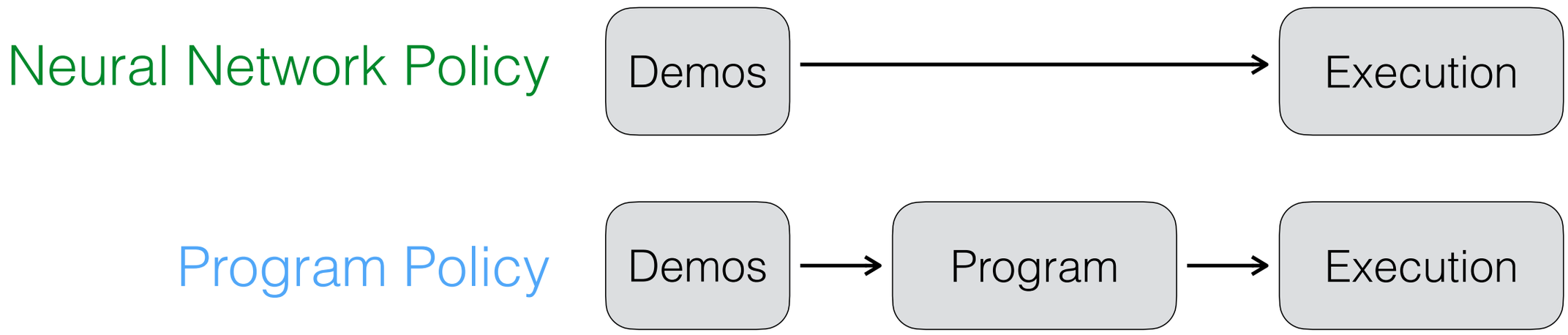
ViZDoom

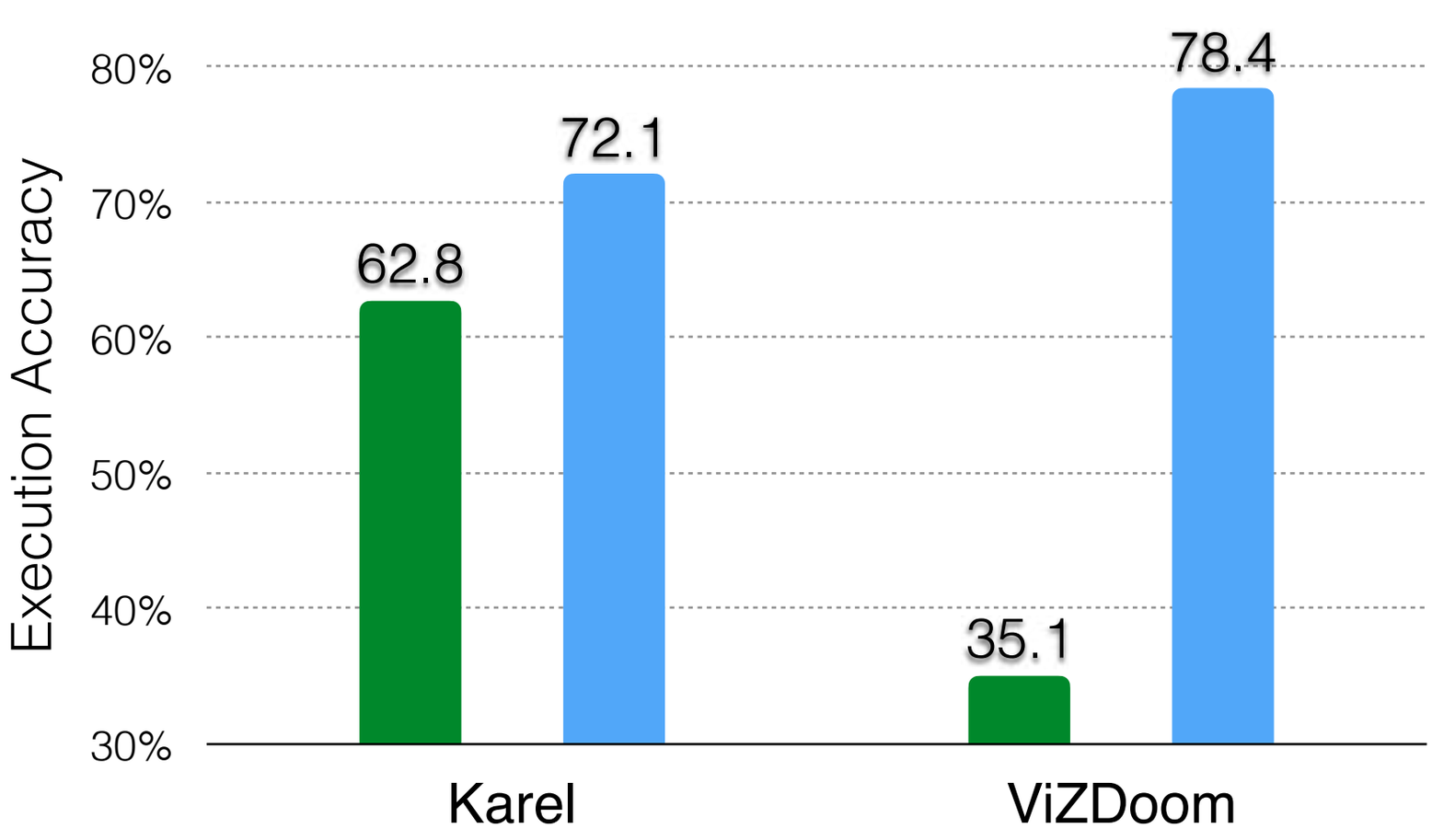
```
DEF run()
   while isFontClear(HellKnight):
      attack
      moveForward
   if isThere(Demon):
      moveRight
   else:
      moveLeft
      moveBackward
```



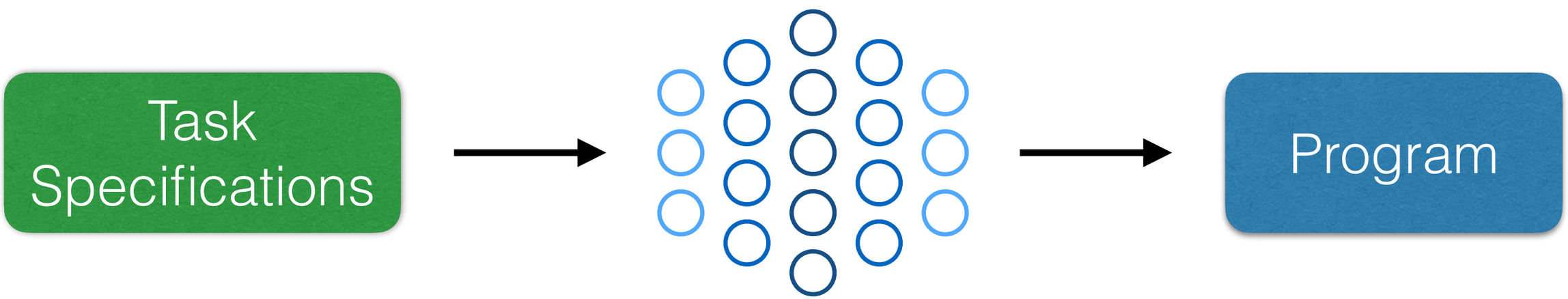
Kempka et al., “Vizdoom: A doom-based ai research platform for visual reinforcement learning.” In CIG, 2016

Quantitative Results





Task Specifications



- **I/O pairs**: I/O strings, I/O images
- **Execution traces**: demonstration videos
- **States**: images, voxels
- **Language descriptions**
- **Reward function**

Improving Neural Program Synthesis with Inferred Execution Traces

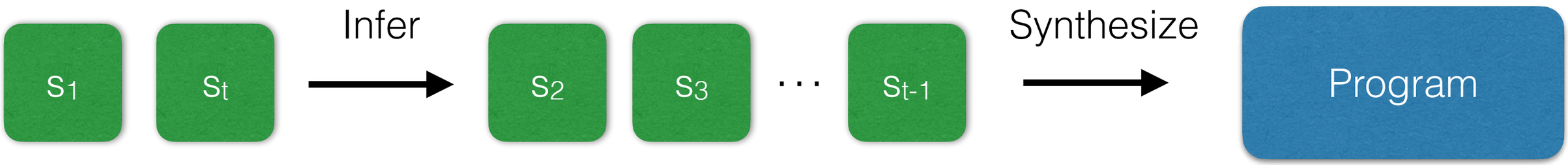
Richard Shin, Illia Polosukhin, Dawn Song

Neural Abstract Machines & Program Induction Workshop at ICML 2018

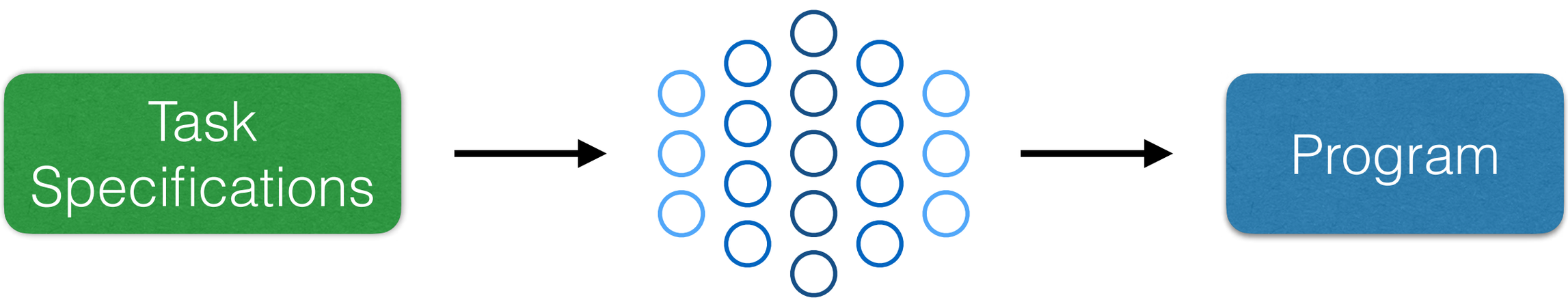
Execution-Guided Neural Program Synthesis

Xinyun Chen, Chang Liu, Dawn Song

ICLR 2019



Task Specifications



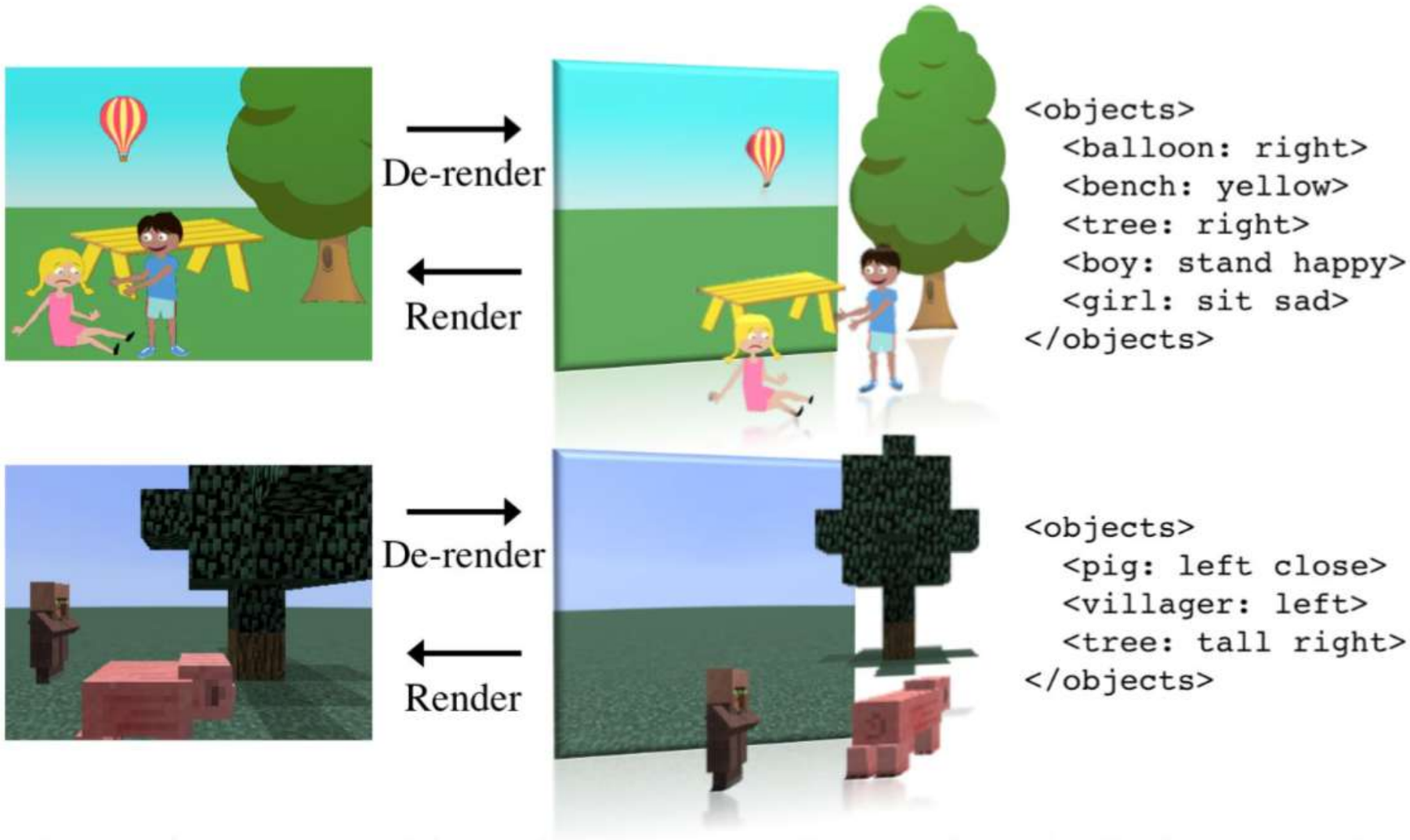
- **I/O pairs**: I/O strings, I/O images
- **Execution traces**: demonstration videos
- **States**: images, voxels
- **Language descriptions**
- **Reward function**

Neural Scene De-rendering

Jiajun Wu, Joshua B. Tenenbaum, and Pushmeet Kohli

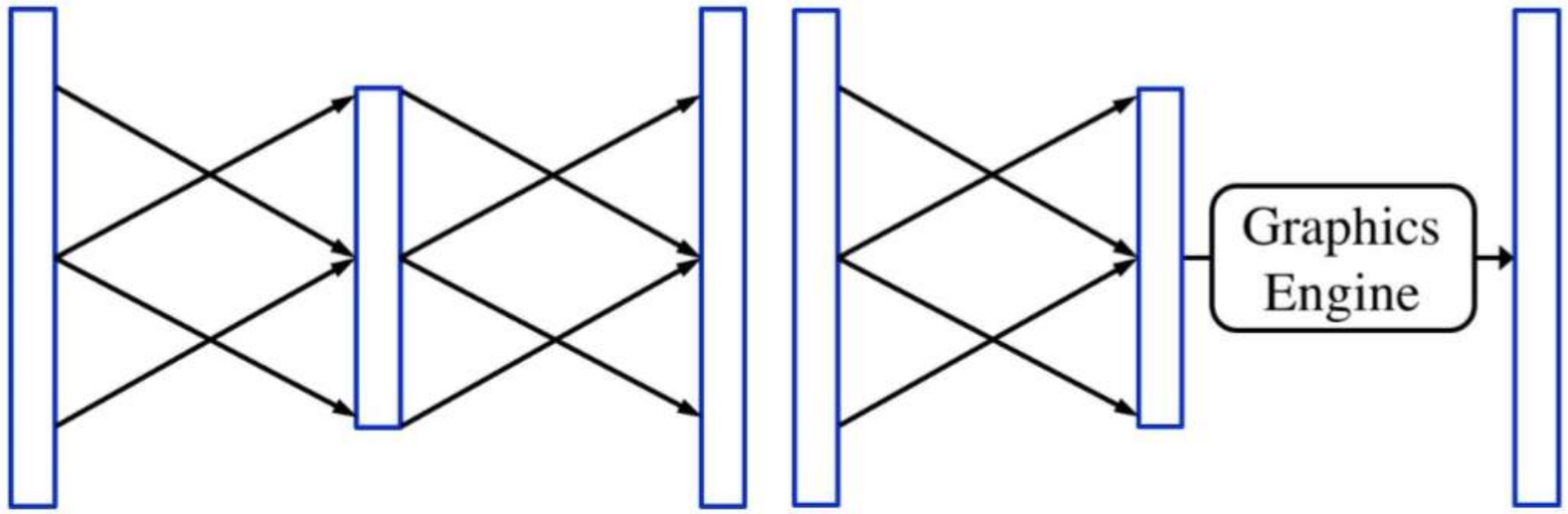
CVPR 2017

Motivation & Problem Formulation



Motivation & Problem Formulation

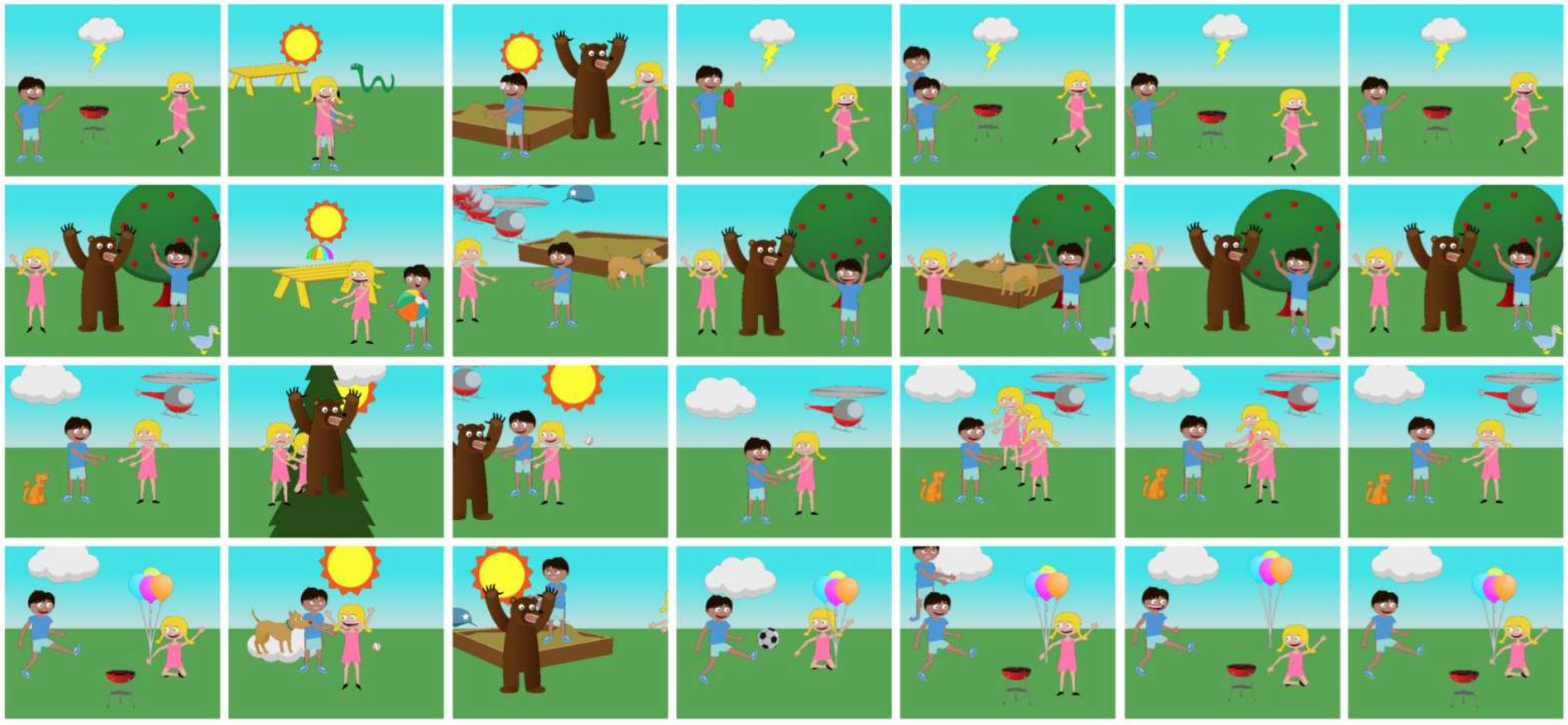
Maybe autoencoders are not the best model to learn good latent representations



(a) A standard autoencoder (b) A generalized autoencoder

Figure 2: **Generalized encoding-decoding structure.** Different from a standard autoencoder (a), our generalized structure (b) uses a graphics engine as the decoder, which by nature takes an interpretable and disentangled representation as input, and renders a high quality image.

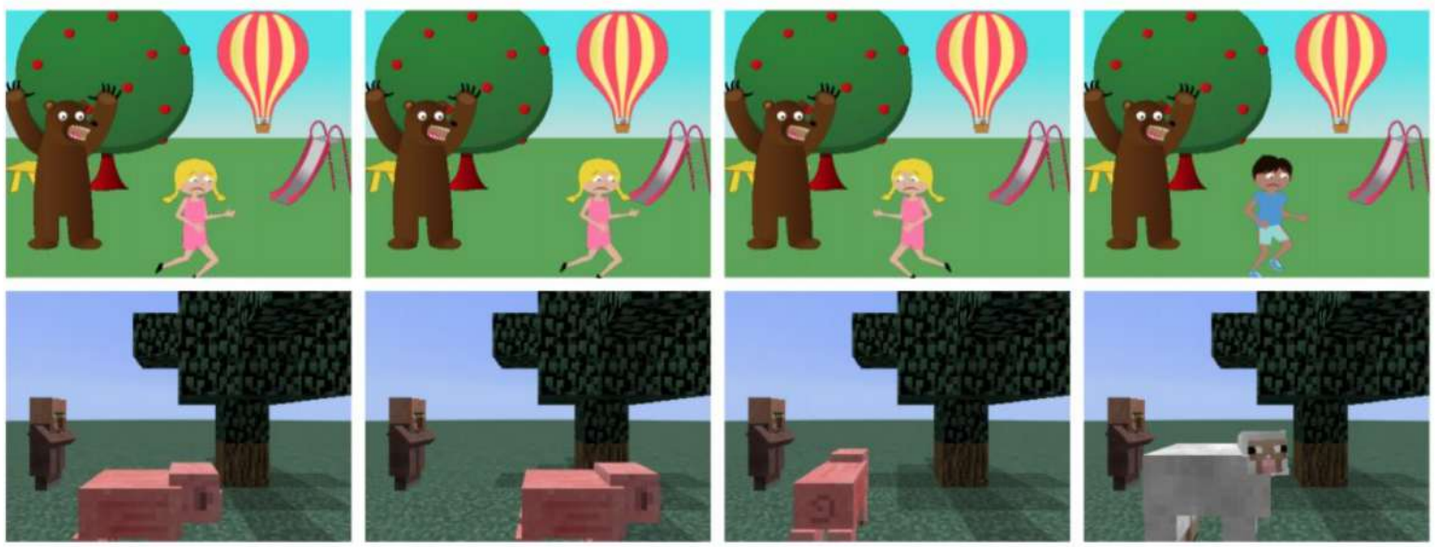
Reconstructed Results



(a) Input image (b) CNN (c) CNN+LSTM (d) NSD (box) (e) NSD (seg) (f) NSD (seg+) (g) NSD (full)

Figure 8: **Results on the Abstract Scene dataset.** From left to right: (a) input images, and results of (b) the CNN model, (c) the CNN+LSTM model, (d) our de-rendering framework with box proposals, (e) our framework with segment proposals, (f) same as (e) but trained with REINFORCE, and (g) our full model with analysis-by-synthesis refinement on top of (f). See Section 4.1 for details of these methods and Section 4.2 analyses of the results.

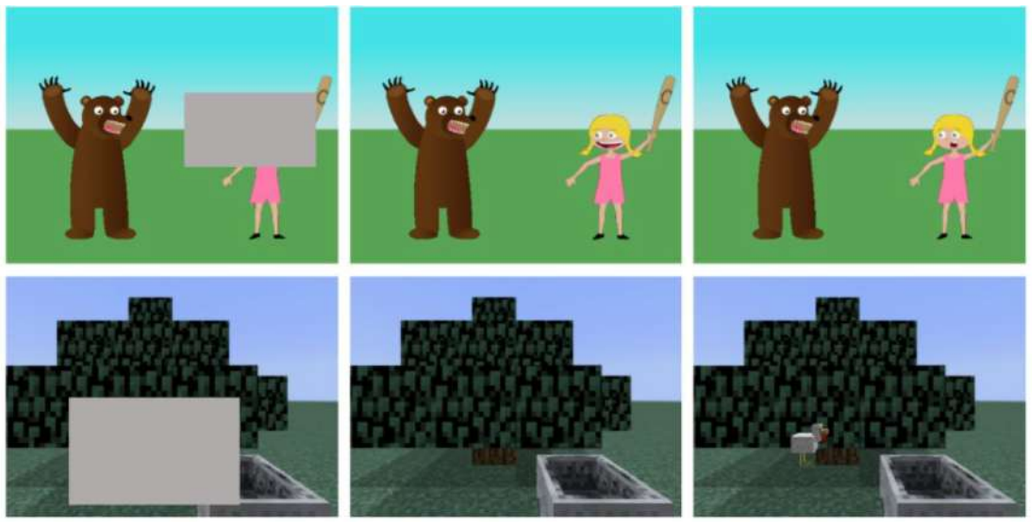
Applications



(a) Original image (b) New position (c) New pose (d) New category

Figure 10: **Results on image editing.** Given an image, we can modify the position, pose, and category of objects with the inferred representation and the graphics engine.

Applications



(a) Corrupted input (b) Reconstruction (c) Original image

Figure 11: **Results on inpainting.** Our framework performs well, though it fails for almost fully occluded objects or parts. In the future, we may include context modeling to possibly correct some of the errors (*e.g.*, the girl in the first row, when facing a bear, should be surprised or afraid, not happy).

Applications

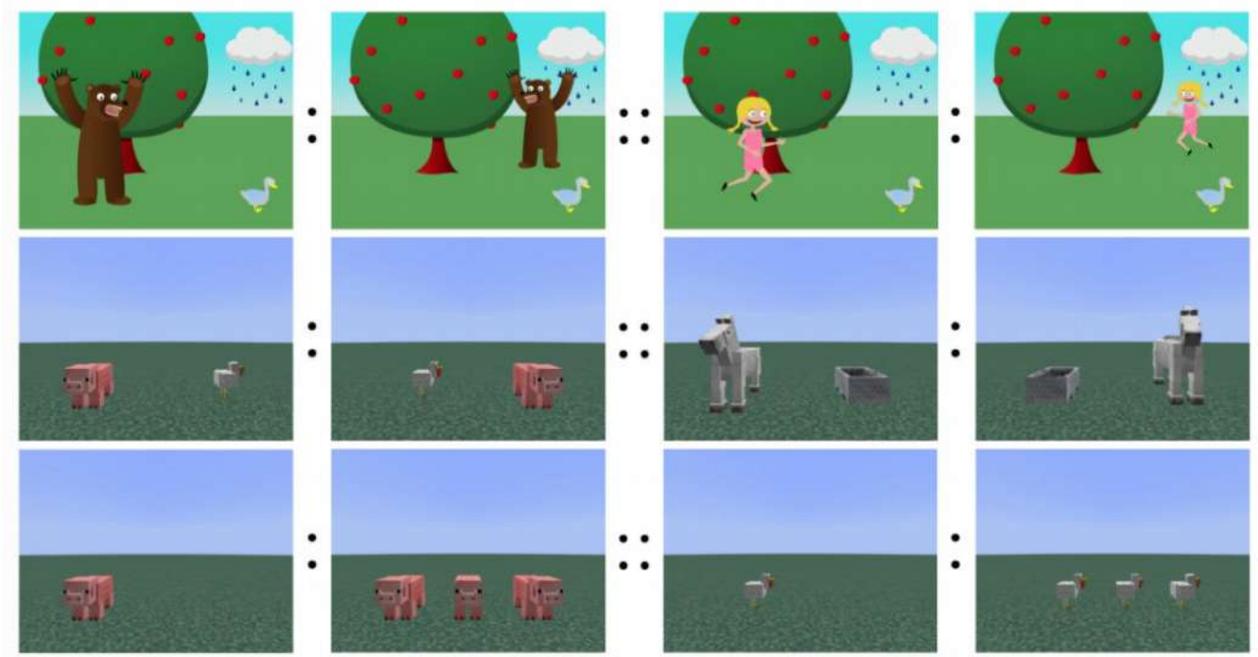


Figure 12: **Results on visual analogy-making.** Given a pair of reference images and a query, our framework can make analogies based on the position and pose of an object (top), and on the number of objects (bottom). See text for details.

Applications

Raw (LSTM)	jenny gets in the sandbox
NSD (LSTM)	jenny and mike both are both playing while the football sits in the sandbox
Raw (NN)	mike and jenny tired of playing frisbee decide to fly Jenny's new kite instead.
NSD (NN)	jenny and mike are having fun in the sandbox unaware of the storm that's coming their way

Raw (LSTM)	a picnic table while a snake and mike on it
NSD (LSTM)	jenny is angry by a snake while she and mike are running towards the park
Raw (NN)	a sad mike is hitting a baseball to an angry Jenny.
NSD (NN)	jenny is scared of a snake at their campsite but mike wants to go catch it.

Figure 13: **Results on image captioning.** Both LSTM and the nearest neighbor method work better using the de-rendered representations, compared to using raw pixels.

Learning to Describe Scenes with Programs

Yunchao Liu, Zheng Wu, Daniel Ritchie,
William T. Freeman, Joshua B. Tenenbaum, Jiajun Wu

ICLR 2019

Model

Mask R-CNN

Pairwise Binary Classification

Program Synthesis

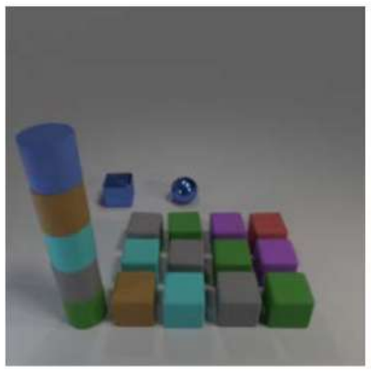
Image → Objects → Objects & Groups → Program

```
(sphere, small, metal,
white, x=2, y=3, z=0)

(sphere, small, metal,
green, x=2, y=2, z=0)

(sphere, small, metal,
purple, x=2, y=1, z=0)

⋮
```

```
for(i<3)
 for(j<3-i)
  sphere(pos=(1+i,1+j,0),
         color=6-j)
for(i<4)
  cylinder(pos=(3,3,i),
         color=7)
cylinder(pos=(4,2,0),
         color=3)
```

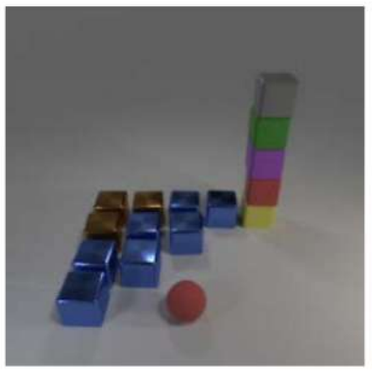
Results

(a)



```
for(i<5)
 cylinder(pos=(0,0,i),
          color=5-i)
for(i<4)
  for(j<3)
    cube(pos=(1+i,j,0),
         color=2+i+j)
cube(pos=(0,4,0),
     color=1)
sphere(pos=(2,4,0),
     color=1)
```

```
for(i<5)
 cylinder(pos=(4,0,i)
          color=7-i)
rotate(i<4,start=0,
       center=(2,1,0))
  sphere(pos=(1,1,0),
         color=5)
  sphere(pos=(2,1,0),
         color=5)
cylinder(pos=(0,2,0),
         color=2)
cube(pos=(0,0,0),color=3)
```



```
for(i<4)
 for(j<4-i)
  cube(pos=(i,i+j,0),
         color=1+j/2)
sphere(pos=(2,0,0),
       color=7)
for(i<5)
  cube(pos=(4,3,i),
       color=8-i)
```

(b)

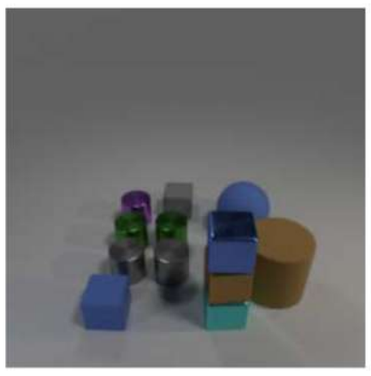
```
for(i<4)
  sphere(pos=(0,i,0),
         color=8)
for(i<4)
  cylinder(pos=(1,2,i),
           color=5+i/2)
for(i<3)
  for(j<2+i)
    cube(pos=(2+i,1+j,0),
         color=(6-j))
```

```
for(i<3)
  for(j<2+i)
    cylinder(pos=(2+i,2,j),
             color=4+j)
rotate(i<4,start=0,
       center=(0,1,0))
  cylinder(pos=(0,1,0),
           color=1+i)
for(i<4)
  cylinder(pos=(4,0,i),
           color=4-2*(i/2))
```

```
rotate(i<4,start=0,
       center=(0,0,0))
  cylinder(pos=(0,0,0),
           color=4-i/2)
for(i<3)
  for(j<4-i)
    cube(pos=(2+i,i+j,0),
         color=6+j/2)
for(i<3)
  cylinder(pos=(0,3,i),
           color=3+i)
```

(c)

```
cylinder(pos=(0.2,0.2,0),
         color=6)
for(i<3)
  for(j<2)
    cube(pos=(0.3+i,2.2-
    i+j,0),color=5-2i)
for(i<4)
  cylinder(pos=(2.4,3.8,i)
         color=5-2*(i%2))
sphere(pos=(3.3,1.8,0),
       color=3)
```



```
cube(pos=(0.4,0,0),
         color=1)
for(i<2)
  for(j<3-i)
    cylinder(
      pos=(0.7+i,1.1+j,
      0),color=4+j)
for(i<3)
  cube(pos=(2.8,0,i)
         color=3-i)
...
```

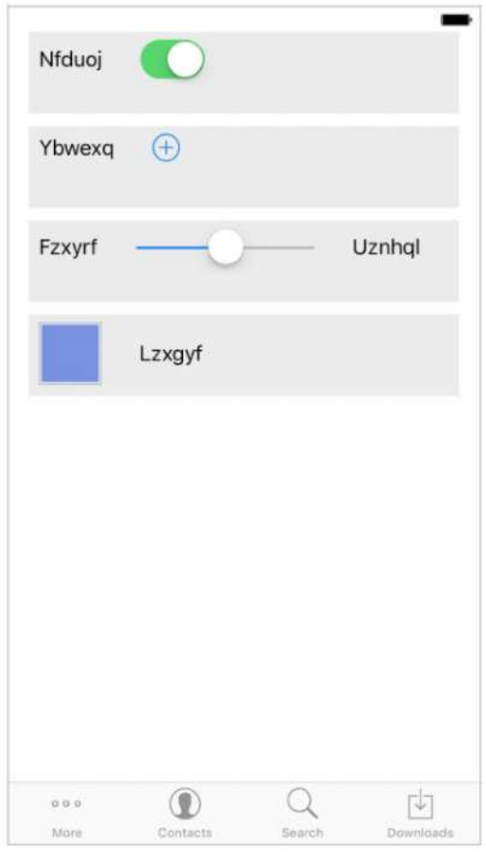
```
cylinder(pos=(0.3,2.4,0),
         color=7)
cube(pos=(0.5,0.2,0),
         color=5)
cube(pos=(1.0,1.2,0),
         color=8)
for(i<3)
  for(j<i+1)
    sphere(pos=(1.4+i,4.0-
    i-j,0),color=3-j)
...
```

pix2code: Generating Code from a Graphical User Interface Screenshot

Tony Beltramelli

arXiv 2017

Motivation & Problem Formulation



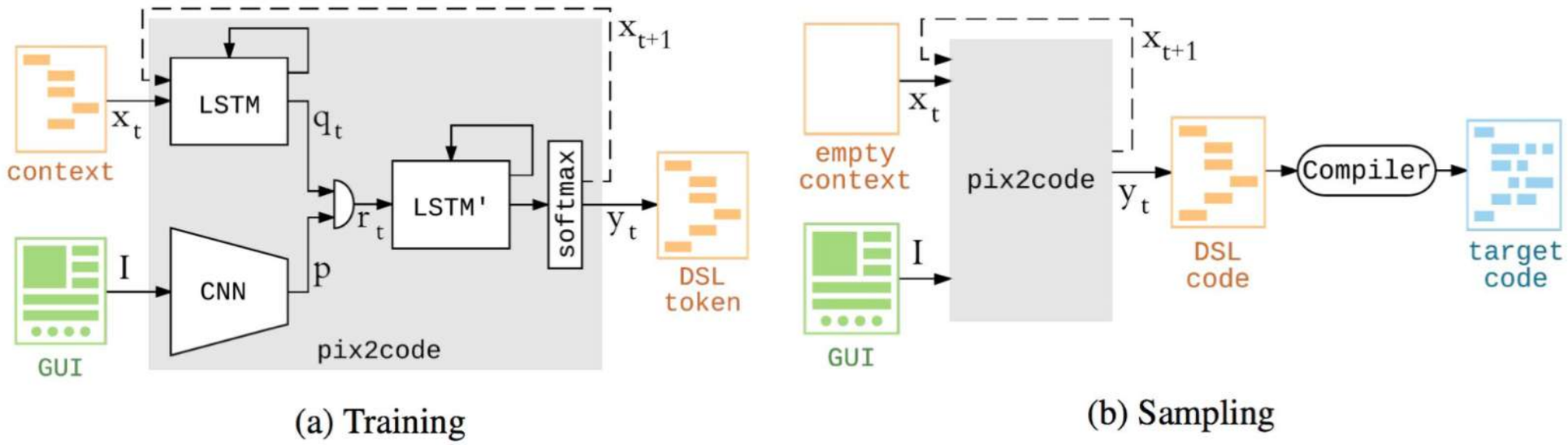
(a) iOS GUI screenshot

```
stack {
    row {
        label, switch
    }
    row {
        label, btn-add
    }
    row {
        label, slider, label
    }
    row {
        img, label
    }
}
footer {
    btn-more, btn-contact, btn-search, btn-download
}
```

(b) Code describing the GUI written in our DSL

Figure 2: An example of a native iOS GUI written in our markup-like DSL.

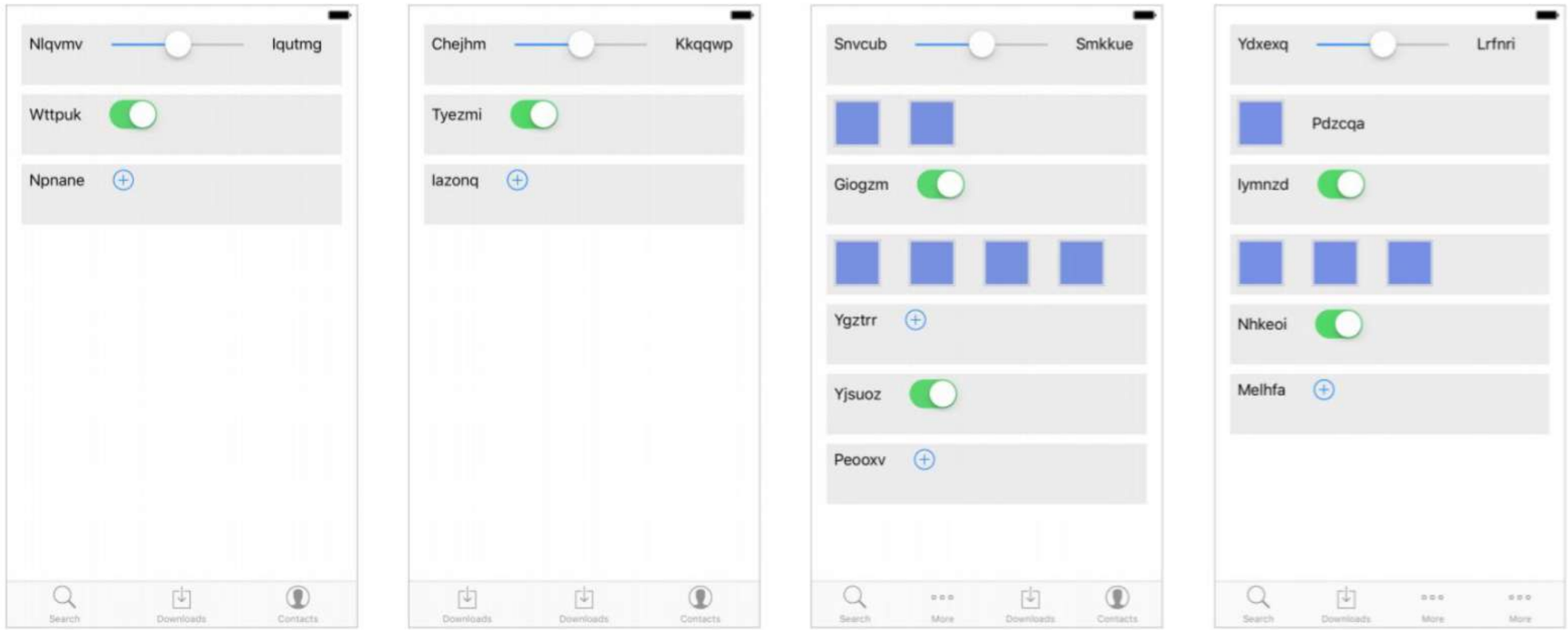
Model Architecture



(a) Training (b) Sampling

Figure 1: Overview of the *pix2code* model architecture. During training, the GUI image is encoded by a CNN-based vision model; the context (i.e. a sequence of one-hot encoded tokens corresponding to DSL code) is encoded by a language model consisting of a stack of LSTM layers. The two resulting feature vectors are then concatenated and fed into a second stack of LSTM layers acting as a decoder. Finally, a *softmax* layer is used to sample one token at a time; the output size of the *softmax* layer corresponding to the DSL vocabulary size. Given an image and a sequence of tokens, the model (i.e. contained in the gray box) is differentiable and can thus be optimized end-to-end through gradient descent to predict the next token in the sequence. During sampling, the input context is updated for each prediction to contain the last predicted token. The resulting sequence of DSL tokens is compiled to the desired target language using traditional compiler design techniques.

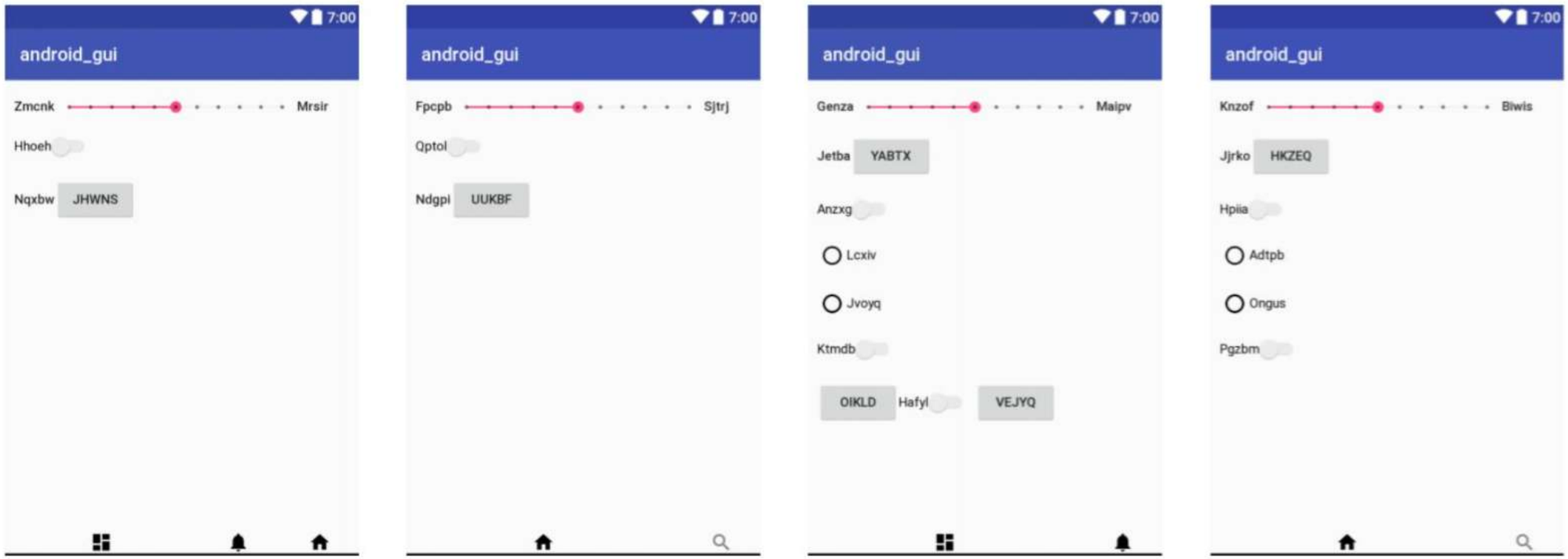
Results



(a) Groundtruth GUI 1 (b) Generated GUI 1 (c) Groundtruth GUI 2 (d) Generated GUI 2

Figure 4: Experiment samples for the iOS GUI dataset.

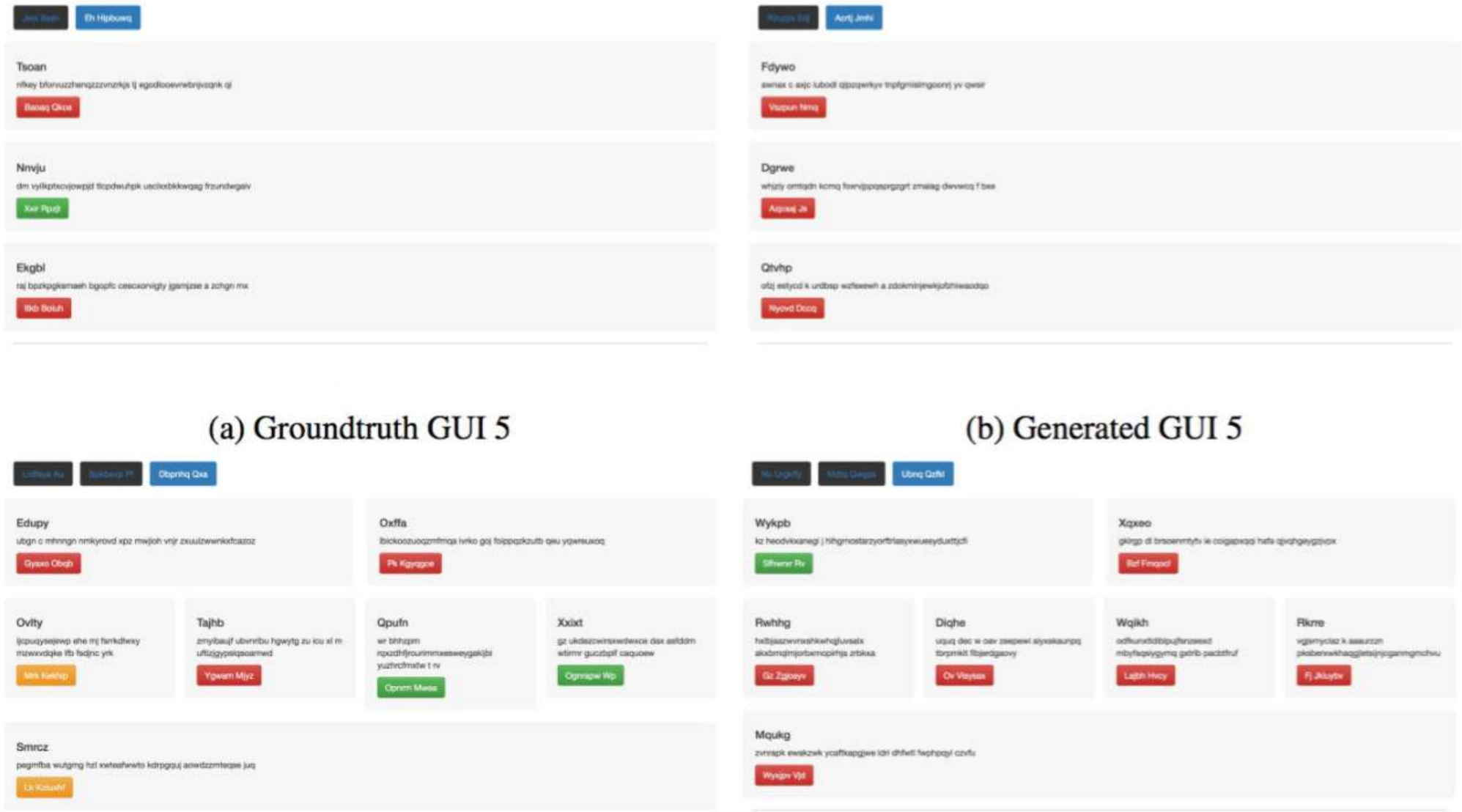
Results



(a) Groundtruth GUI 3 (b) Generated GUI 3 (c) Groundtruth GUI 4 (d) Generated GUI 4

Figure 6: Experiment samples from the Android GUI dataset.

Results



(a) Groundtruth GUI 5 (b) Generated GUI 5

(c) Groundtruth GUI 6 (d) Generated GUI 6

Figure 5: Experiment samples from the web-based GUI dataset.

Learning to Infer and Execute 3D Shape Programs

Yonglong Tian, Andrew Luo, Xingyuan Sun, Kevin Ellis, William T. Freeman, Joshua B. Tenenbaum, Jiajun Wu

ICLR 2019

Motivation & Problem Formulation

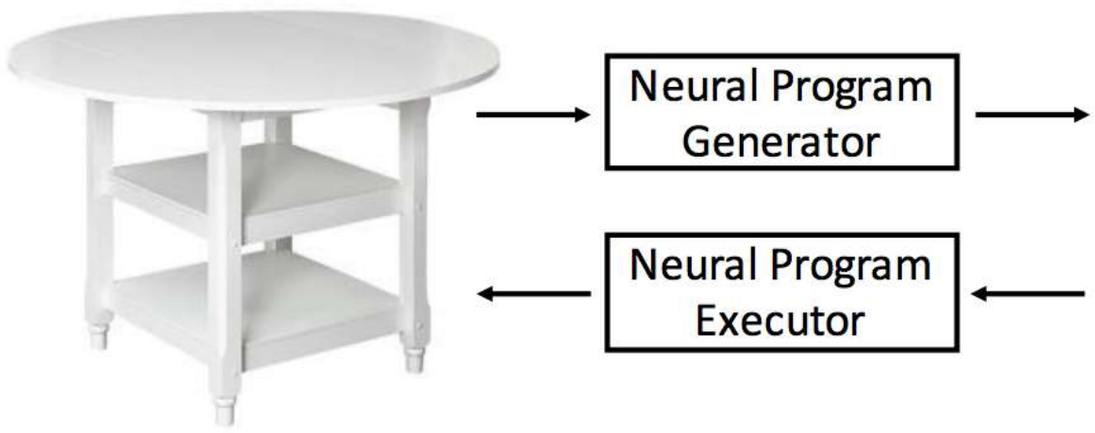
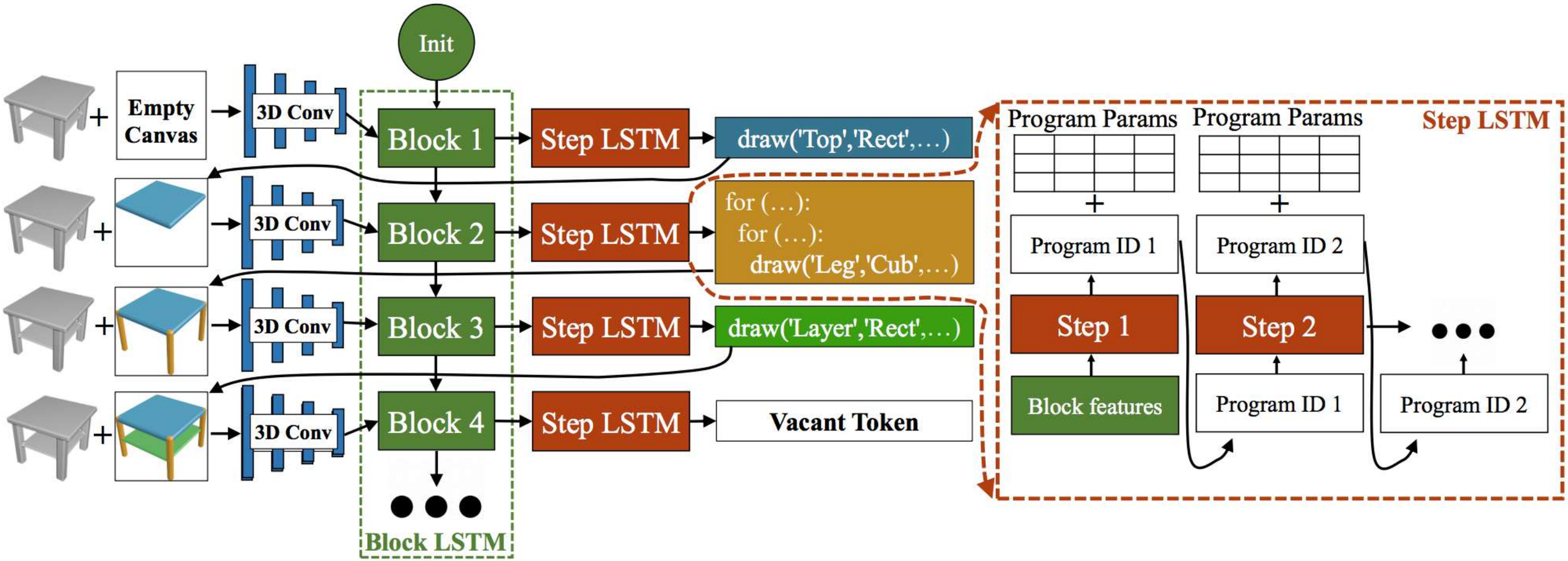


Figure 1: A 3D shape can be represented by a program via a program generator. This program can be executed by a neural program executor to produce the corresponding 3D shape.

Model Architecture



Results

Reconstruction before adaption

```
draw('Top','Cir',P=(4,0,0),G=(1,7))

draw('Support','Cyl',P=(-9,0,0),G=(15,3))

for(i<4,Rot(θrot=90,ax=(-9,1,0)))
 draw('Base','Line',P=(-9,1,0),
   G=(-9,-6,-5),θrot×i, ax)

draw('Layer','Rect',P=(-3,0,0),G=(2,4,6))
```

Input

Reconstruction after adaption

```
draw('Top','Cir',(P=(0,0,0),G=(2,6)))

draw('Support','Cyl',P=(-11,0,0), G=(13,1))

for(i<5,'Rot',θrot=72,ax=(-10,0,0))
 draw('Base','Line',P=(-10,0,0),
      G=(-11,-6,-3),θrot×i, ax)

draw('TiltBack','Cub',P=(3,2,-5),G=(8,2,9,7))

for(i<2,'Trans',u1=(0,0,11))
 for(j<2,'Trans',u2=(0,4,0))
  draw('ChairBeam','Cub',P=(2,-4,-6)
   +(j×u2)+(i×u1),G=(3,1,2))

for(i<2,'Trans',u=(0,0,10))
 draw('HoriBar','Cub',P=(4,-4,-6)
   +(i×u),G=(1,5,2))
```

(a)

Reconstruction before adaption

```
draw('Top',`Sqr',P=(-4,-1,0),G=(4,9))

draw('Support','Cyl',P=(-11,-1,0),G=(12,3))

draw('BackSupp','Cub',P=(0,7,-3),G=(3,2,7))

draw('TiltBack','Cub',P=(4,6,-10),
   G=(8,3,19,20))

for(i<2,'Trans',u=(0,0,19))
 draw('Sideboard', 'Rect',P=(1,-2,-10)
   +(i×u),G=(6,6,1))
```

Input

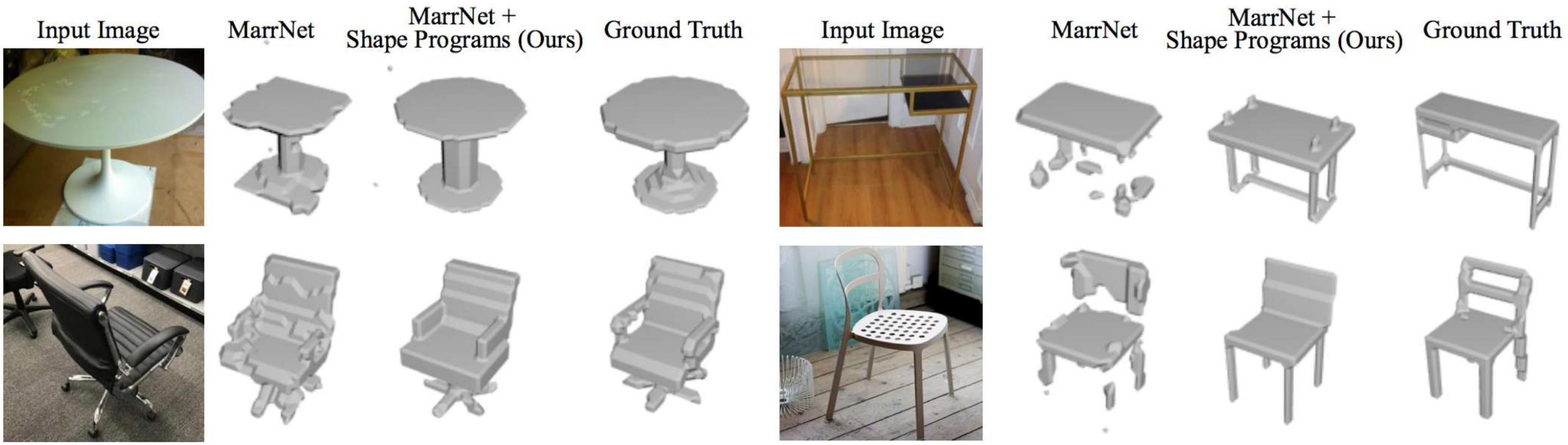
Reconstruction after adaption

```
draw('Top',`Rect',P=(-8,-1,0),G=(9,10,11))

for(i<2,'Trans',u1=(0,0,17))
 for(j<2,'Trans',u2=(0,17,0))
  draw('Leg','Cub',P=(-11,-10,-10)
   +(j×u2)+(i×u1),G=(12,2,3))

draw('TiltBack','Cub',P=(0,4,-10),
   G=(10,4,21,11))

for(i<2,'Trans',u=(0,0,19))
 draw('Sideboard', 'Rect',P=(0,-2,-12)
   +(i×u),G=(6,9,4))
```

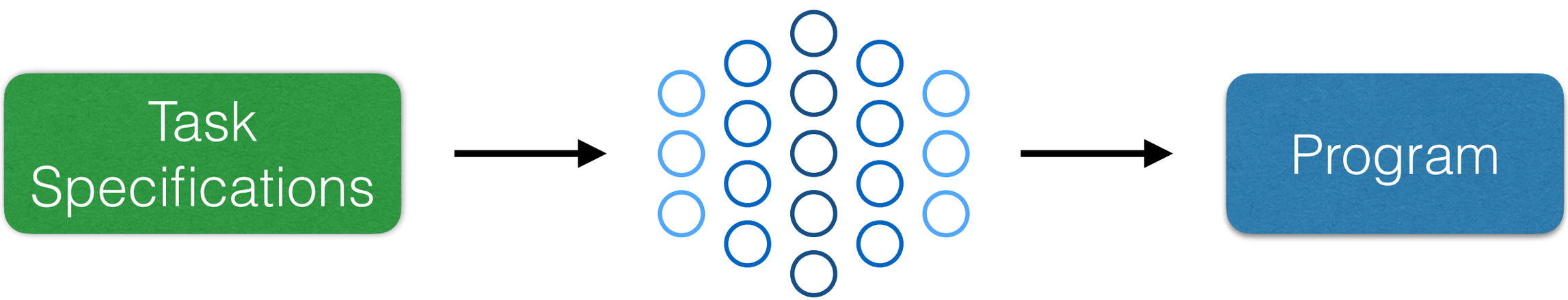
(b)

Results

Image → Predicted voxel → Predicted program → Rendered voxel



Task Specifications



- **I/O pairs**: I/O strings, I/O images
- **Execution traces**: demonstration videos
- **States**: images, voxels
- Language descriptions
- Reward function

NL2Bash: A Corpus and Semantic Parser for Natural Language Interface to the Linux Operating System

Xi Victoria Lin, Chenglong Wang, Luke Zettlemoyer, Michael D. Ernst

LREC 2018

Motivation & Problem Formulation

Natural Language	Bash Command(s)

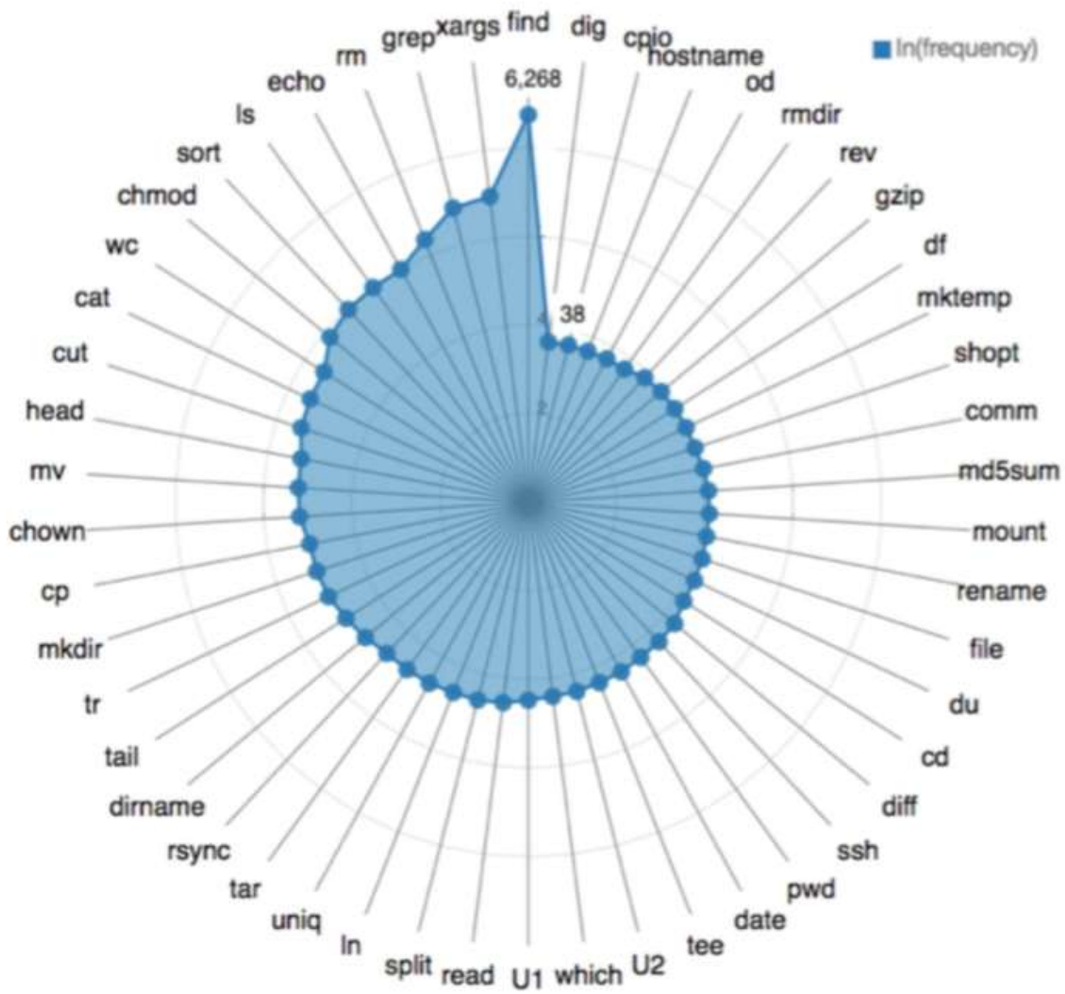
find .java files in the current directory tree that contain the pattern 'TODO' and print their names	`grep -l "TODO" *.java`
	`find . -name "*.java" -exec grep -il "TODO" {} \;`
	`find . -name "*.java" | xargs -I {} grep -l "TODO" {}`

display the 5 largest files in the current directory and its sub-directories	`find . -type f | sort -nk 5,5 | tail -5`
	`du -a . | sort -rh | head -n5`
	`find . -type f -printf '%s %p\n' | sort -rn | head -n5`

search for all jpg images on the system and archive them to tar ball "images.tar"	`tar -cvf images.tar $(find / -type f -name *.jpg)`
	`tar -rvf images.tar $(find / -type f -name *.jpg)`
	`find / -type f -name "*.jpg" -exec tar -cvf images.tar {} \;`

Dataset

Split	Train	Dev	Test
# pairs	8,090	609	606
# unique NL	7,340	549	547
# unique command	6,400	599	XX
# unique command template	4,002	509	XX



50 most frequent Bash utilities
(frequency in log scale)

Results

Model		Acc^1_F	Acc^3_F	Acc^1_T	Acc^3_T
Seq2Seq	Char	0.24	0.27	0.35	0.38
	Token	0.10	0.12	**0.53**	0.59
	Sub-token	0.19	0.27	0.41	0.53
CopyNet	Char	0.25	0.31	0.34	0.41
	Token	0.21	0.34	0.47	**0.61**
	Sub-token	**0.31**	**0.40**	0.44	0.53
Tellina		0.29	0.32	0.51	0.58

Table 8: Translation accuracies of the baseline systems on 100 instances sampled from the dev set.

Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning

Victor Zhong, Caiming Xiong, Richard Socher

arXiv 2017

Motivation & Problem Formulation

Table: CFLDraft

Pick #	CFL Team	Player	Position	College
27	Hamilton Tiger-Cats	Connor Healy	DB	Wilfrid Laurier
28	Calgary Stampeders	Anthony Forgone	OL	York
29	Ottawa Renegades	L.P. Ladouceur	DT	California
30	Toronto Argonauts	Frank Hoffman	DL	York
...	...	...	...	...

Question:

How many CFL teams are from York College?

SQL:

```
SELECT COUNT CFL Team FROM
CFLDraft WHERE College = "York"
```

Result:

2

Figure 2: An example in WikiSQL. The inputs consist of a table and a question. The outputs consist of a ground truth SQL query and the corresponding result from execution.

Model

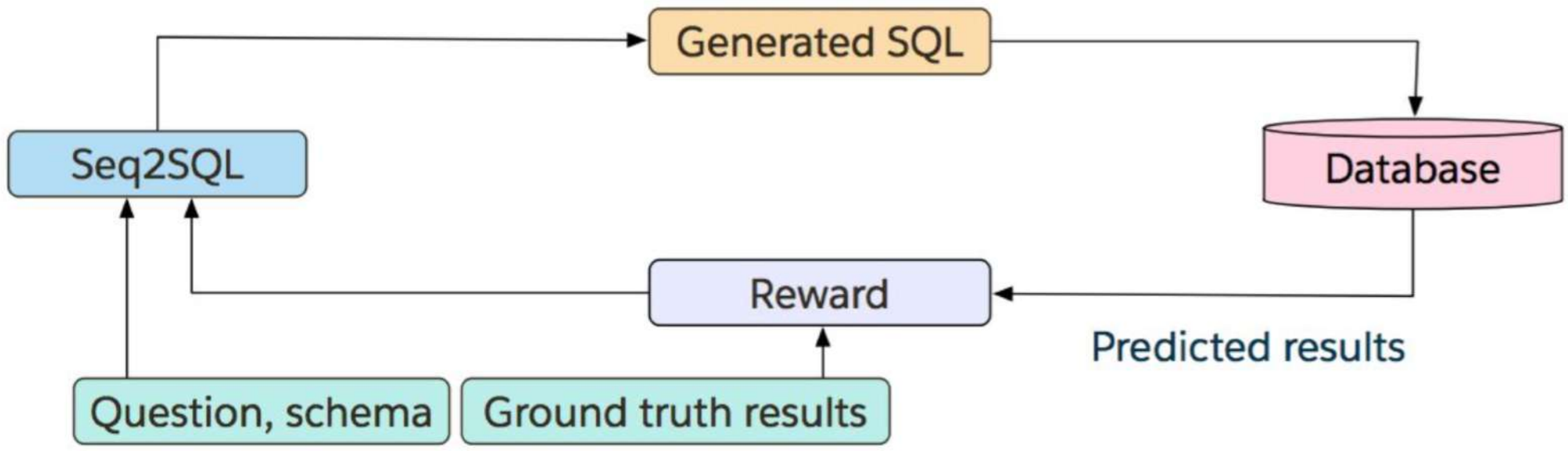


Figure 1: Seq2SQL takes as input a question and the columns of a table. It generates the corresponding SQL query, which, during training, is executed against a database. The result of the execution is utilized as the reward to train the reinforcement learning algorithm.

Model

Why using reinforcement learning?

Q1

```
SELECT name FROM insurance WHERE age > 18 AND gender = "male"
```

Q2

```
SELECT name FROM insurance WHERE gender = "male" AND age > 18
```

Program aliasing

Dataset

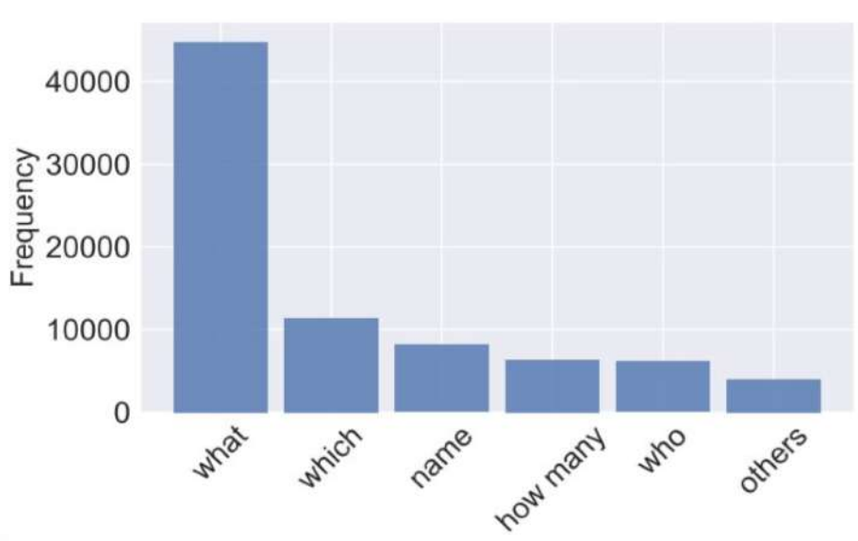


Figure 4: Distribution of questions in WikiSQL.

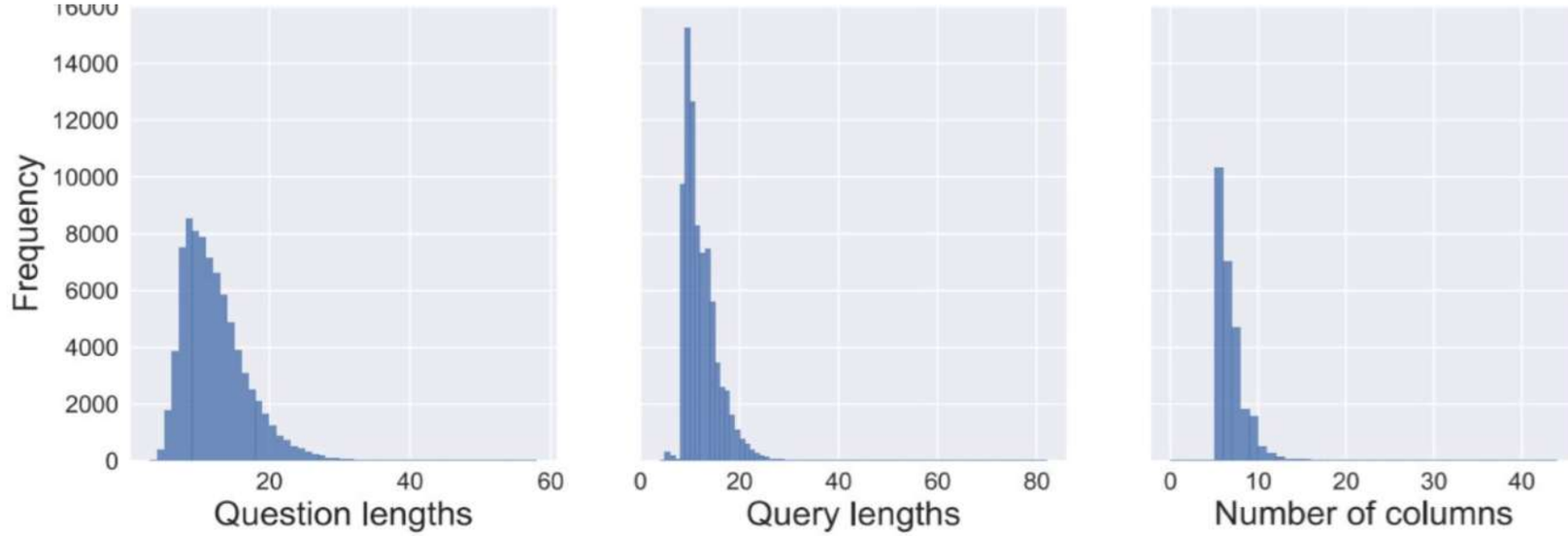


Figure 5: Distribution of table, question, query sizes in WikiSQL.

Results

RL helps

Model	Dev Acc_{lf}	Dev Acc_{ex}	Test Acc_{lf}	Test Acc_{ex}
Baseline (Dong & Lapata, 2016)	23.3%	37.0%	23.4%	35.9%
Aug Ptr Network	44.1%	53.8%	43.3%	53.3%
Seq2SQL (no RL)	48.2%	58.1%	47.4%	57.1%
Seq2SQL	**49.5%**	**60.8%**	**48.3%**	**59.4%**

Table 2: Performance on WikiSQL. Both metrics are defined in Section 3.1. For Seq2SQL (no RL), the `WHERE` clause is supervised via teacher forcing as opposed to reinforcement learning.

Ain't Nobody Got Time For Coding: Structure-Aware Program Synthesis From Natural Language

Jakub Bednarek, Karol Piaskowski, Krzysztof Krawiec

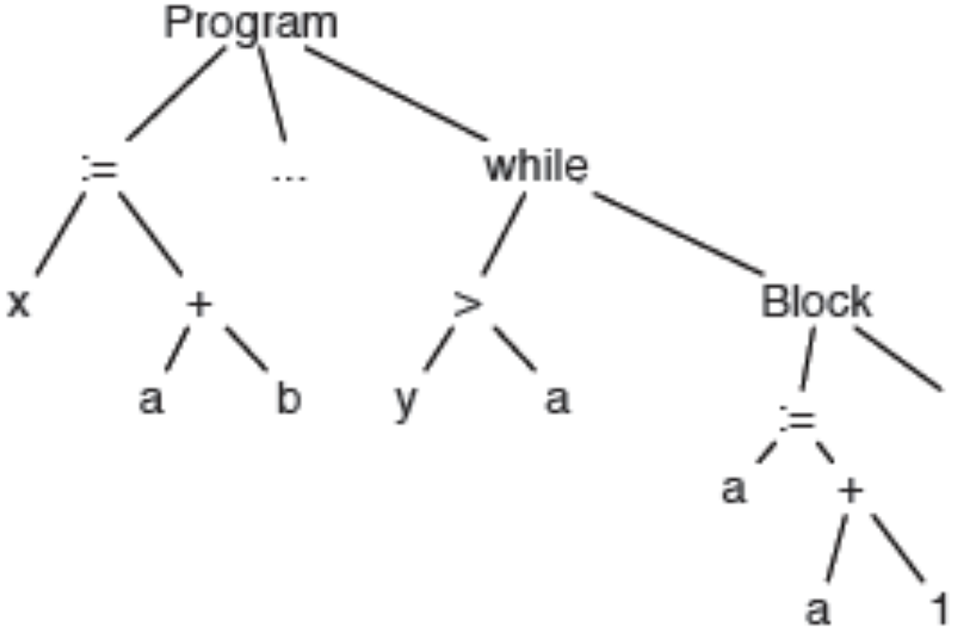
arXiv 2018

Problem Formulation

Description

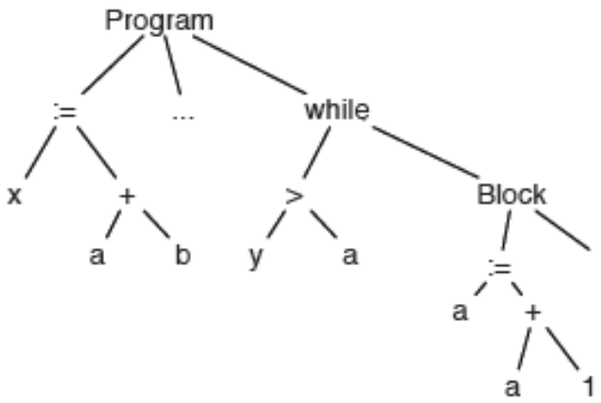
you are given numbers a and b, your task is to find a + b
you given numbers a b, your is find a + b
*given **a numbers b**, find a + b*
given a numbers b, a + b
a b, a + b

Program

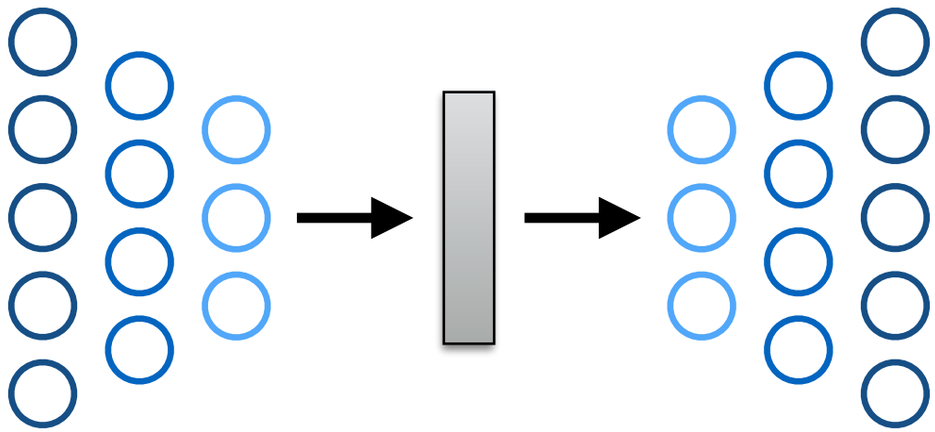


Model

Program

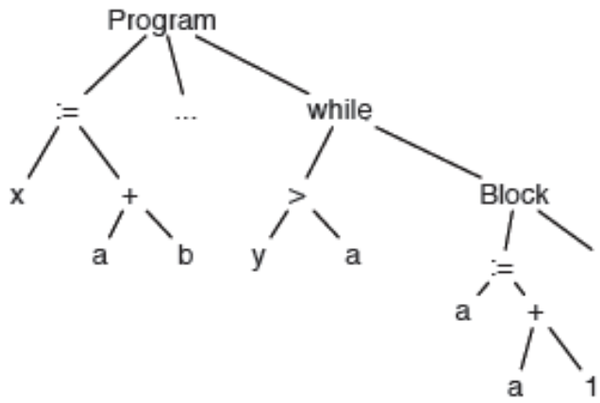


Recurrent TreeLSTM



Double RNN

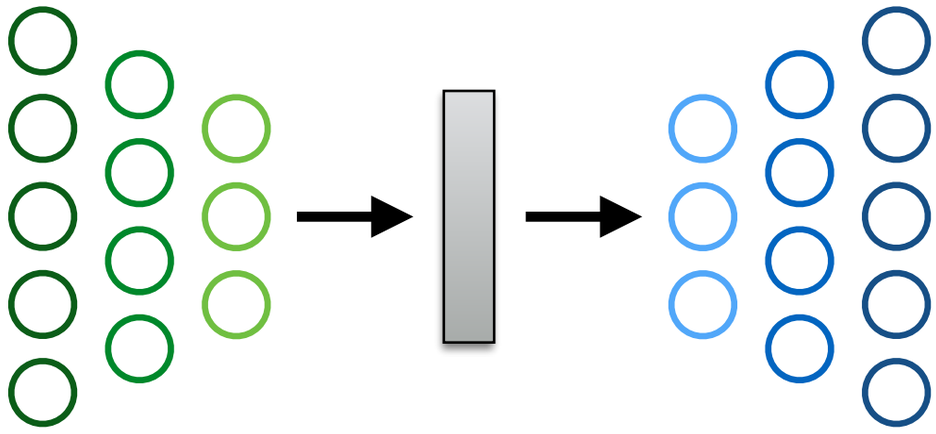
Reconstructed Program



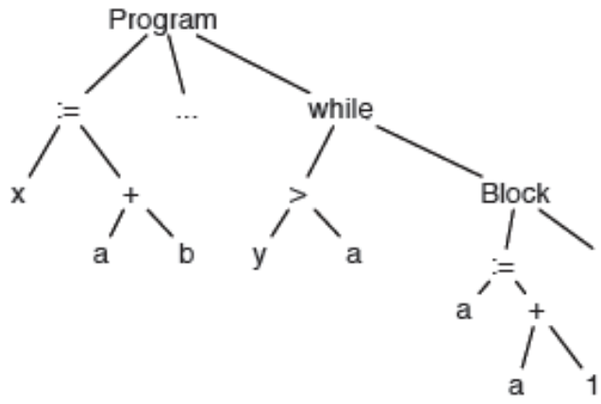
Description

you are given numbers a and b, your task is to find a + b
you given numbers a b, your is find a + b
*given **a numbers b**, find a + b*
given a numbers b, a + b
a b, a + b

LSTM



Predicted Program



Results

Specification	Synthesized program
you are given numbers a and b, your task is to find a + b	`(+, a, b)`
you given numbers a b, your is find a + b	`(+, a, b)`
*given **a numbers b**, find a + b*	`(+, a, b)`
given a numbers b, a + b	`(+, a, b)`
a b, a + b	`(+, (+, a, b), c)`
*you are given numbers a and b, your task is to find **a multiplied by b***	`(*, a, b)`
*you are given numbers a and b, your task is to find **minimum a and b***	`(min, a, b)`
*given a number a and an array of numbers b, find the length of the longest subsequence of range from **0 to a inclusive** that is a prefix of b*	`(reduce, (range, 0, (+, a, 1)), 0, (lambda2, (if, (==, arg2, (if, (<, arg1, (len, b)), (deref, b, arg1), 0)), (+, arg1, 1), arg1)))`
*given a number a and an array of numbers b, find the length of the longest subsequence of range from **1 to a exclusive** that is a prefix of b*	`(reduce, (range, 1, a), 0, (lambda2, (if, (==, arg2, (if, (<, arg1, (len, b)), (deref, b, arg1), 0)), (+, arg1, 1), arg1)))`
given an array of numbers a, find median of values in a after only keeping first half	`(deref, (sort, (slice, a, 0, (/, (len, a), 2))), (/, (len, (slice, a, 0, (/, (len, a), 2))), 2))`
given an array of numbers a, find mean of values in a after only keeping second half	`(/, (reduce, (slice, a, (/, (len, a), 2), (len, a)), 0, +), (len, (slice, a, (/, (len, a), 2), (len, a))))`

Competitive programming with AlphaCode

Li et. al. at DeepMind

preprint 2022

Competitive Programming

Problem Description

Backspace

You are given two strings s and t, both consisting of lowercase English letters. You are going to type the string s character by character, from the first character to the last one.

When typing a character, instead of pressing the button corresponding to it, you can press the "Backspace" button. It deletes the last character you have typed among those that aren't deleted yet (or does nothing if there are no characters in the current string). For example, if s is "abcbd" and you press Backspace instead of typing the first and the fourth characters, you will get the string "bd" (the first press of Backspace deletes no character, and the second press deletes the character 'c'). Another example, if s is "abcaa" and you press Backspace instead of the last two letters, then the resulting text is "a".

Your task is to determine whether you can obtain the string t, if you type the string s and press "Backspace" instead of typing several (maybe zero) characters of s.

Input

The first line contains a single integer q ($1 \le q \le 10^5$) the number of test cases.
The first line of each test case contains the string s ($1 \le |s| \le 10^5$). Each character of s is a lowercase English letter.
The second line of each test case contains the string t ($1 \le |t| \le 10^5$). Each character of t is a lowercase English letter.
It is guaranteed that the total number of characters in the strings over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print "YES" if you can obtain the string t by typing the string s and replacing some characters with presses of "Backspace" button, or "NO" if you cannot.
You may print each letter in any case (YES, yes, Yes will all be recognized as positive answer, NO, no and nO will all be recognized as negative answer).

Example Input & Output

Example Input

```
4
ababa
ba
ababa
bb
aaa
aaaa
aababa
ababa
```

Example Output

```
YES
NO
NO
YES
```

Problem Formulation

Problem Description

Backspace

You are given two strings s and t, both consisting of lowercase English letters. You are going to type the string s character by character, from the first character to the last one.

When typing a character, instead of pressing the button corresponding to it, you can press the "Backspace" button. It deletes the last character you have typed among those that aren't deleted yet (or does nothing if there are no characters in the current string). For example, if s is "abcbd" and you press Backspace instead of typing the first and the fourth characters, you will get the string "bd" (the first press of Backspace deletes no character, and the second press deletes the character 'c'). Another example, if s is "abcaa" and you press Backspace instead of the last two letters, then the resulting text is "a".

Your task is to determine whether you can obtain the string t, if you type the string s and press "Backspace" instead of typing several (maybe zero) characters of s.

Input

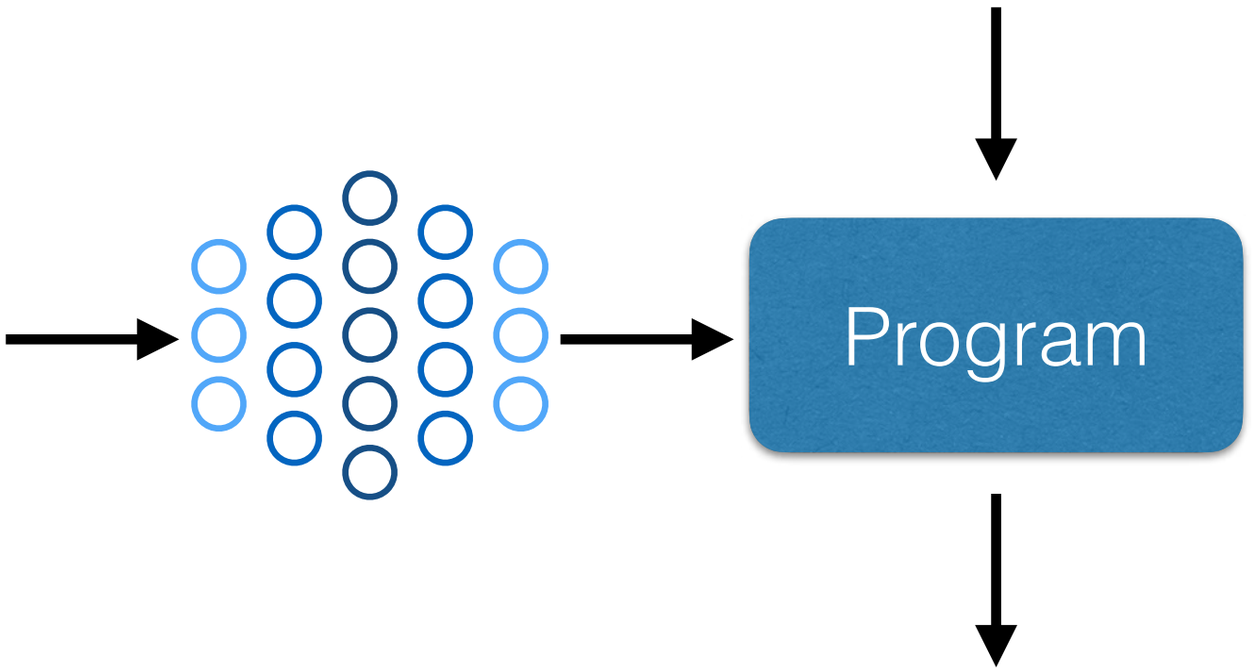
The first line contains a single integer q ($1 \le q \le 10^5$) the number of test cases.
The first line of each test case contains the string s ($1 \le |s| \le 10^5$). Each character of s is a lowercase English letter.
The second line of each test case contains the string t ($1 \le |t| \le 10^5$). Each character of t is a lowercase English letter.
It is guaranteed that the total number of characters in the strings over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print "YES" if you can obtain the string t by typing the string s and replacing some characters with presses of "Backspace" button, or "NO" if you cannot.
You may print each letter in any case (YES, yes, Yes will all be recognized as positive answer, NO, no and nO will all be recognized as negative answer).

Example Input

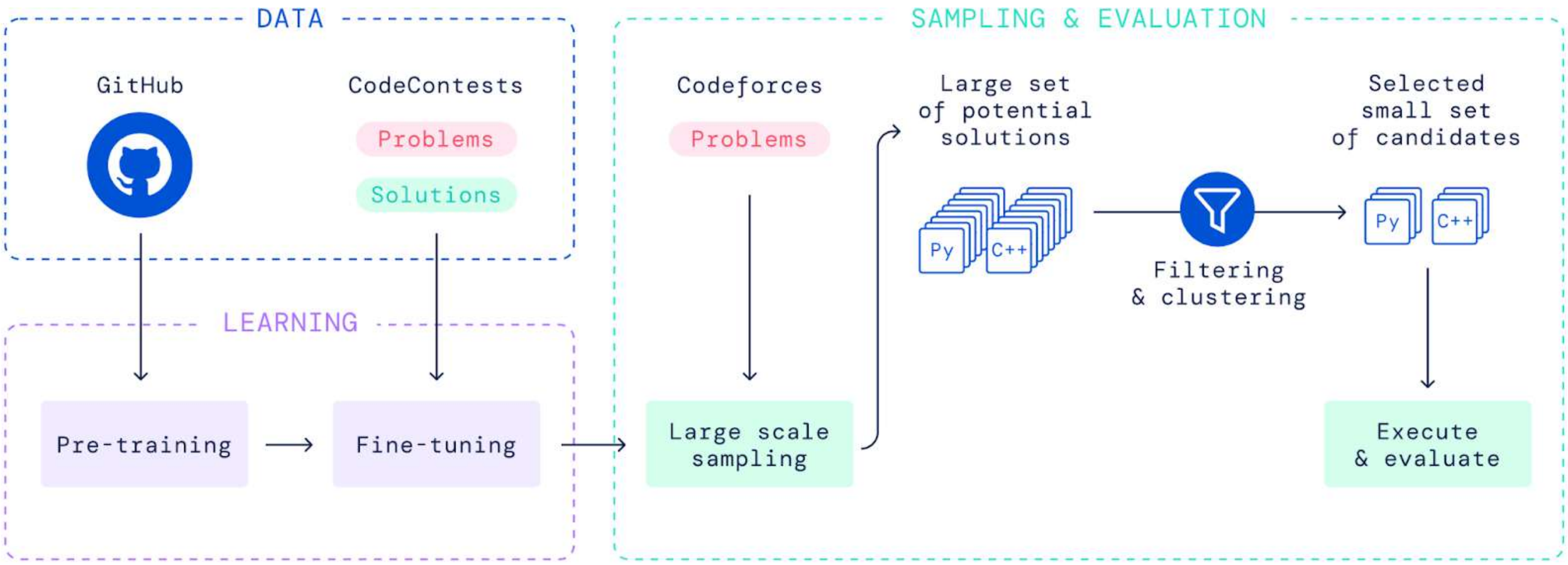
```
4
ababa
ba
ababa
bb
aaa
aaaa
aababa
ababa
```



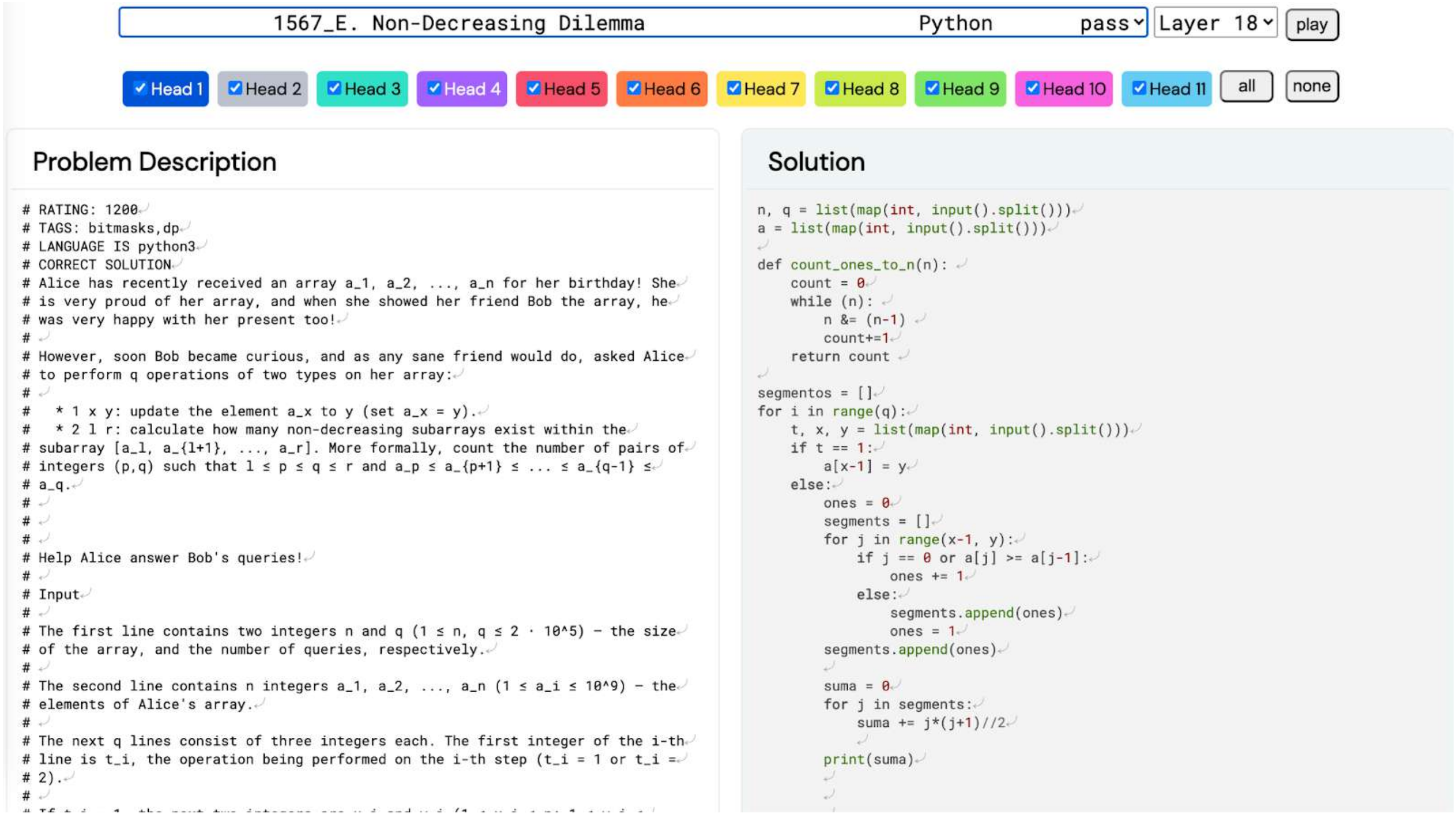
Example Output

```
YES
NO
NO
YES
```

Model Overview



Results

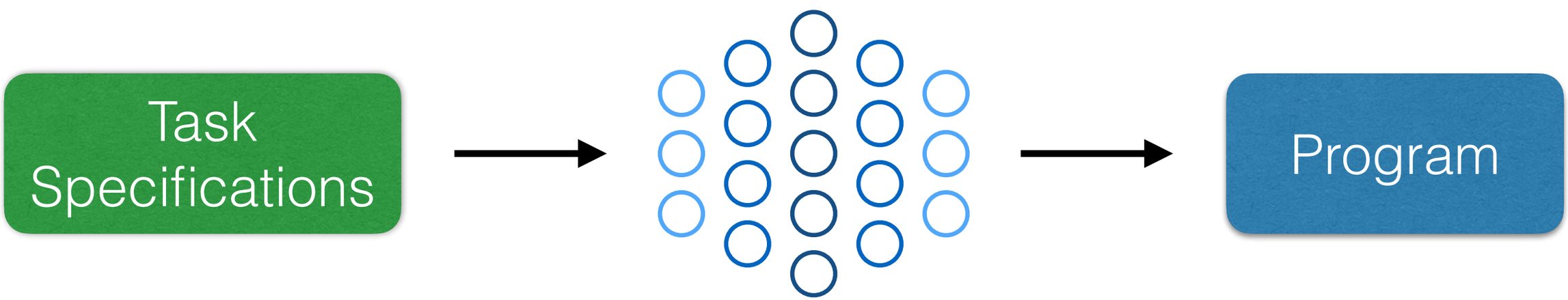

1567_E. Non-Decreasing Dilemma
Python
pass
Layer 18
play
Head 1
Head 2
Head 3
Head 4
Head 5
Head 6
Head 7
Head 8
Head 9
Head 10
Head 11
all
none
Problem Description
RATING: 1200
TAGS: bitmasks,dp
LANGUAGE IS python3
CORRECT SOLUTION
Alice has recently received an array a_1, a_2, ..., a_n for her birthday! She
is very proud of her array, and when she showed her friend Bob the array, he
was very happy with her present too!
#
However, soon Bob became curious, and as any sane friend would do, asked Alice
to perform q operations of two types on her array:
#
* 1 x y: update the element a_x to y (set a_x = y).
* 2 l r: calculate how many non-decreasing subarrays exist within the
subarray [a_l, a_{l+1}, ..., a_r]. More formally, count the number of pairs of
integers (p,q) such that l ≤ p ≤ q ≤ r and a_p ≤ a_{p+1} ≤ ... ≤ a_{q-1} ≤
a_q.
#
#
#
Help Alice answer Bob's queries!
#
Input
#
The first line contains two integers n and q (1 ≤ n, q ≤ 2 · 10^5) – the size
of the array, and the number of queries, respectively.
#
The second line contains n integers a_1, a_2, ..., a_n (1 ≤ a_i ≤ 10^9) – the
elements of Alice's array.
#
The next q lines consist of three integers each. The first integer of the i-th
line is t_i, the operation being performed on the i-th step (t_i = 1 or t_i =
2).
#
Solution
n, q = list(map(int, input().split()))
a = list(map(int, input().split()))
def count_ones_to_n(n):
count = 0
while (n):
n &= (n-1)
count+=1
return count
segmentos = []
for i in range(q):
t, x, y = list(map(int, input().split()))
if t == 1:
a[x-1] = y
else:
ones = 0
segments = []
for j in range(x-1, y):
if j == 0 or a[j] >= a[j-1]:
ones += 1
else:
segments.append(ones)
ones = 1
segments.append(ones)
suma = 0
for j in segments:
suma += j*(j+1)//2
print(suma)

Results

Contest ID	1591	1608	1613	1615	1617	1618	1619	1620	1622	1623	Average
Best	43.5%	43.6%	59.8%	60.5%	65.1%	32.2%	47.1%	54.0%	57.5%	20.6%	**48.4%**
Estimated	44.3%	46.3%	66.1%	62.4%	73.9%	52.2%	47.3%	63.3%	66.2%	20.9%	**54.3%**
Worst	74.5%	95.7%	75.0%	90.4%	82.3%	53.5%	88.1%	75.1%	81.6%	55.3%	**77.2%**

Table 4 | **Estimated percent ranking of our system in 10 Codeforces competitions (lower is better).** For each contest, we show ranking using simulated time and incorrect submission penalties (Estimated), as well as the best and worst possible rankings using minimum and maximum possible time penalties as estimates, averaged over 3 evaluations. Percents are how many users performed better than AlphaCode. Our system achieved an overall ranking of top 54.3% averaged across the 10 contests.

Task Specifications



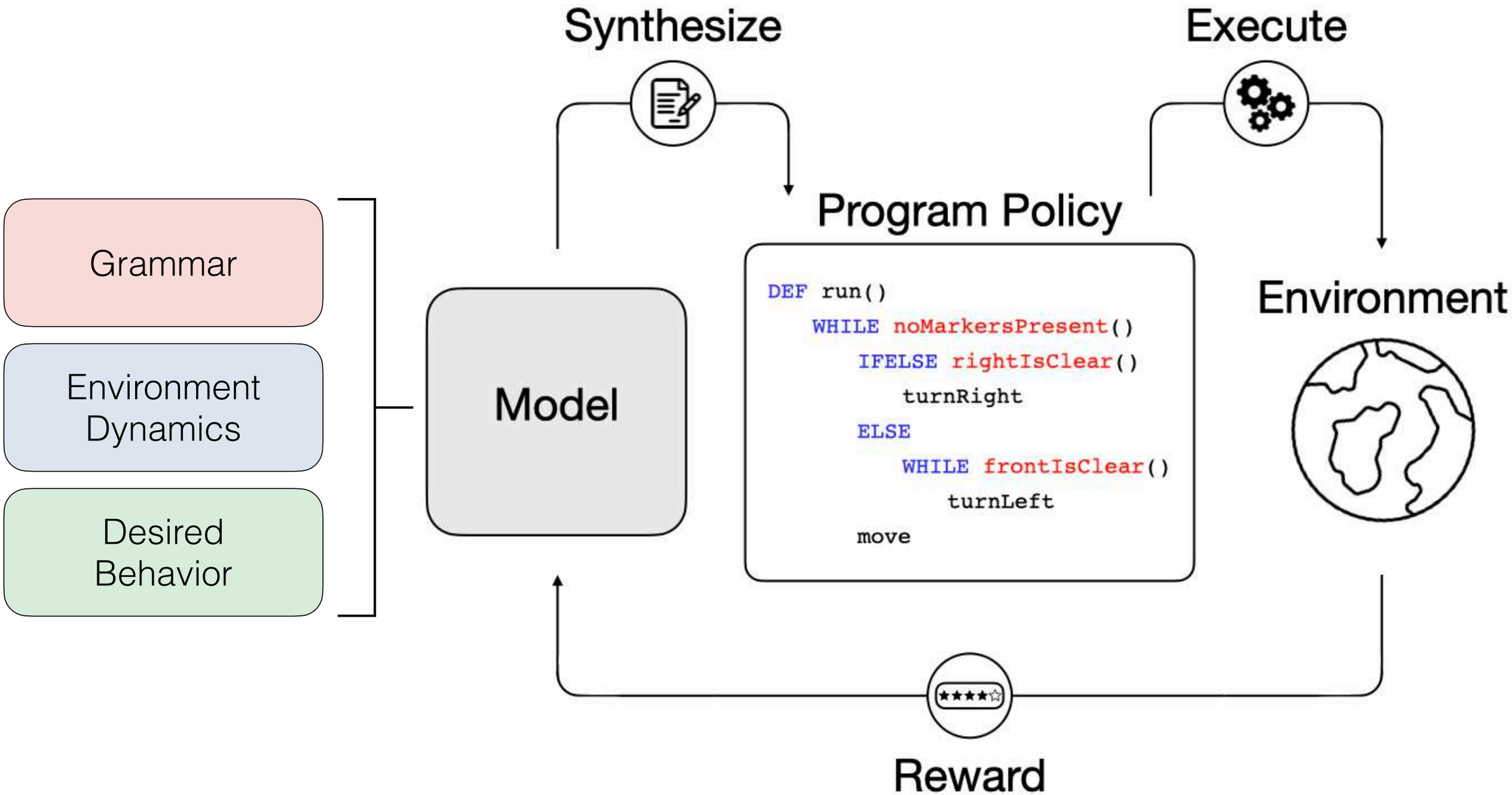
- **I/O pairs**: I/O strings, I/O images
- **Execution traces**: demonstration videos
- **States**: images, voxels
- **Language descriptions**
- **Reward function**

Learning to Synthesize Programs as Interpretable and Generalizable Policies

Dweep Trivedi, Jesse Zhang, Shao-Hua Sun, Joseph J. Lim

NeurIPS 2021

Motivation & Problem Formulation



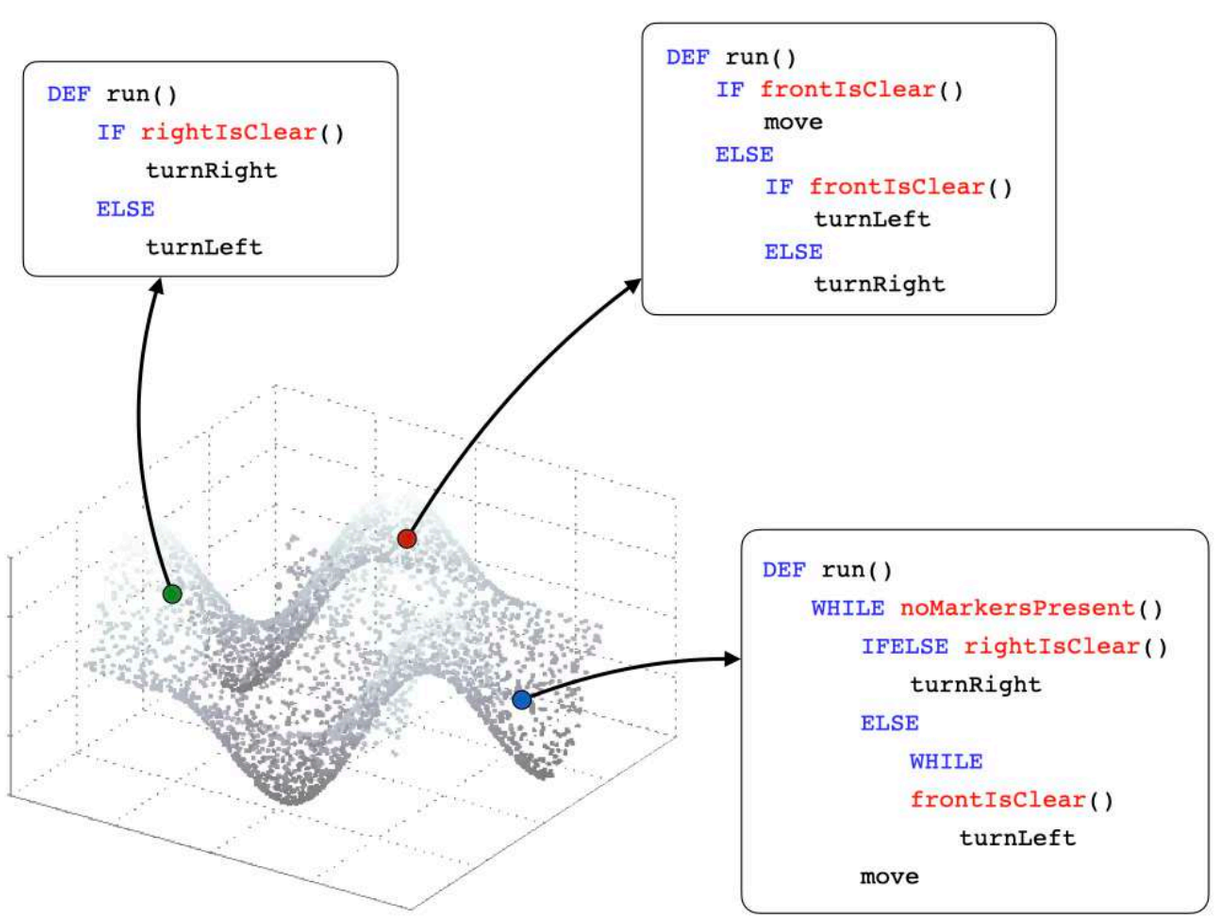
Method

Stage 1

Learn a program embedding space from randomly generated programs

Grammar

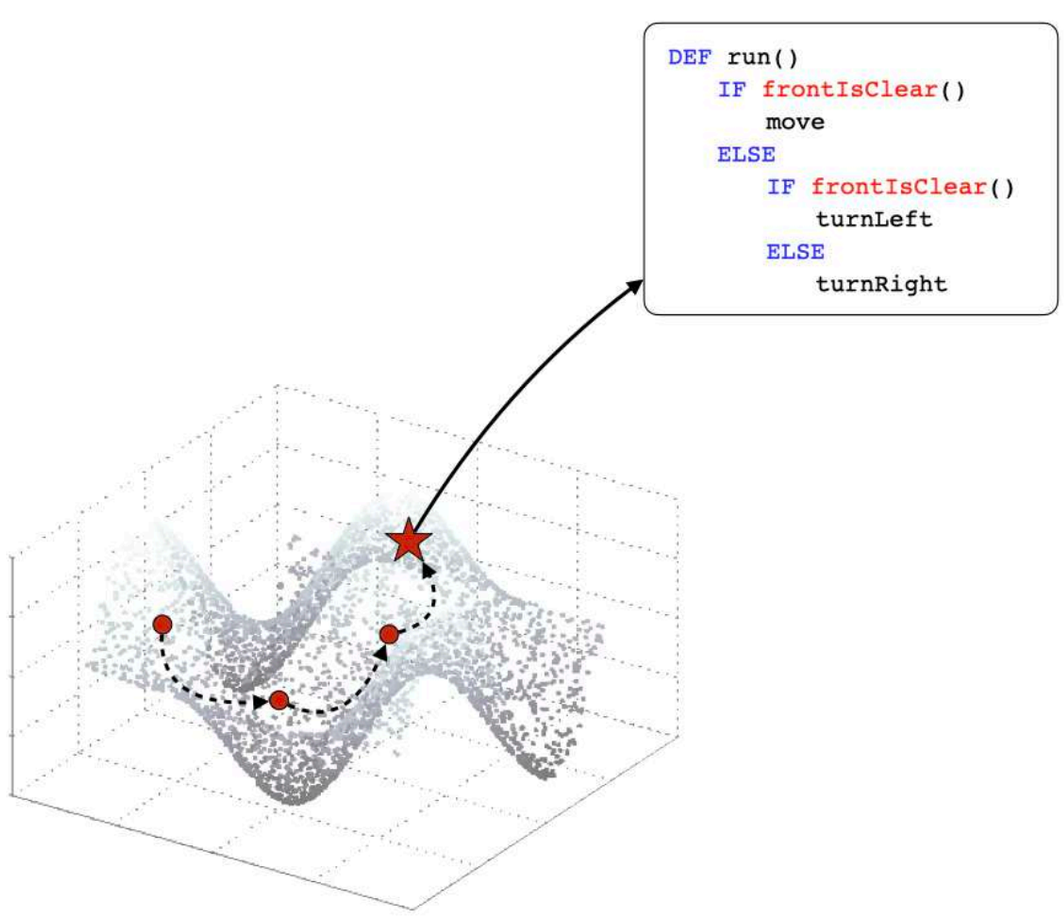
Environment Dynamics



Stage 2

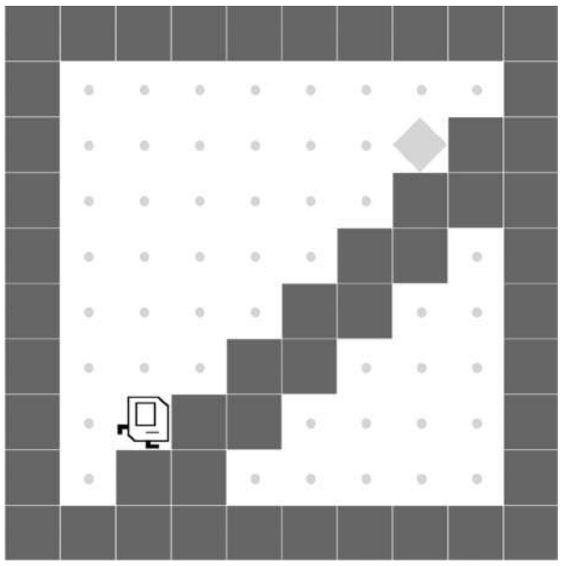
Search for a task-solving program

Desired Behavior

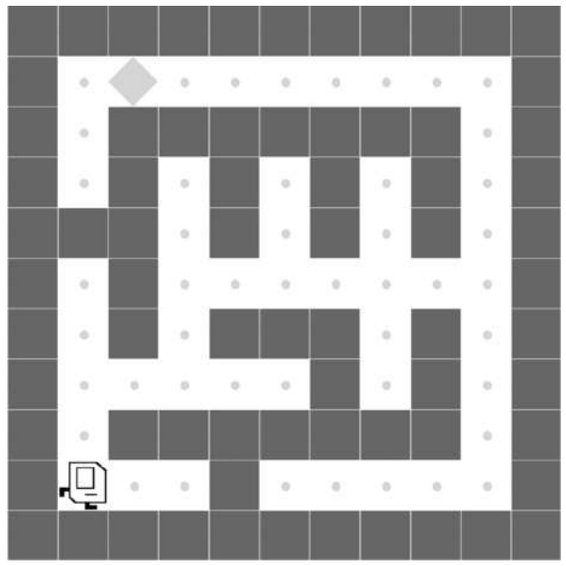


Karel Tasks

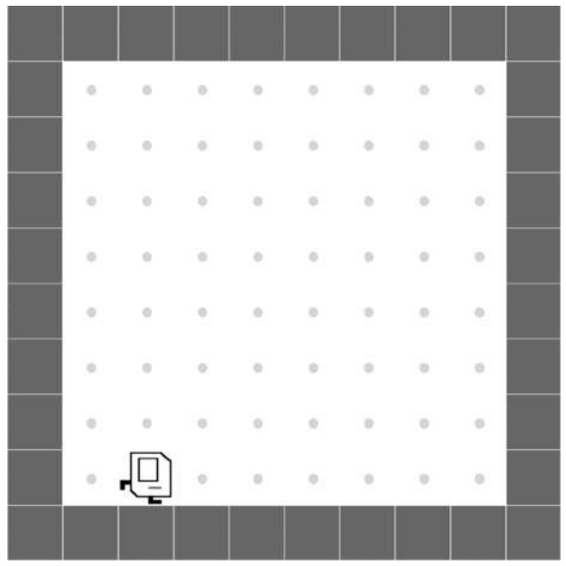
StairClimber



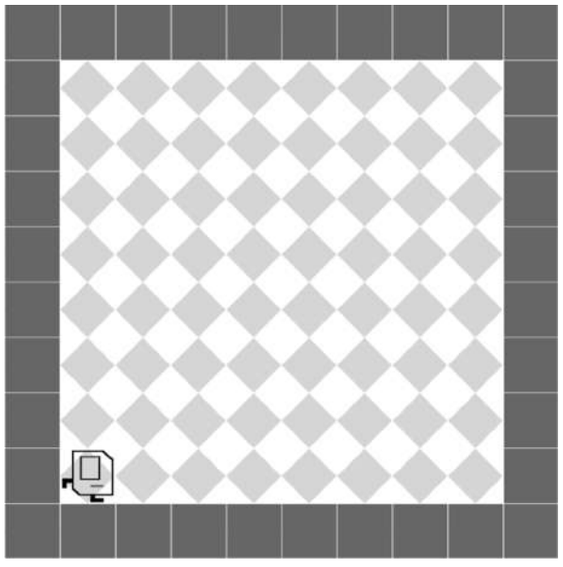
Maze



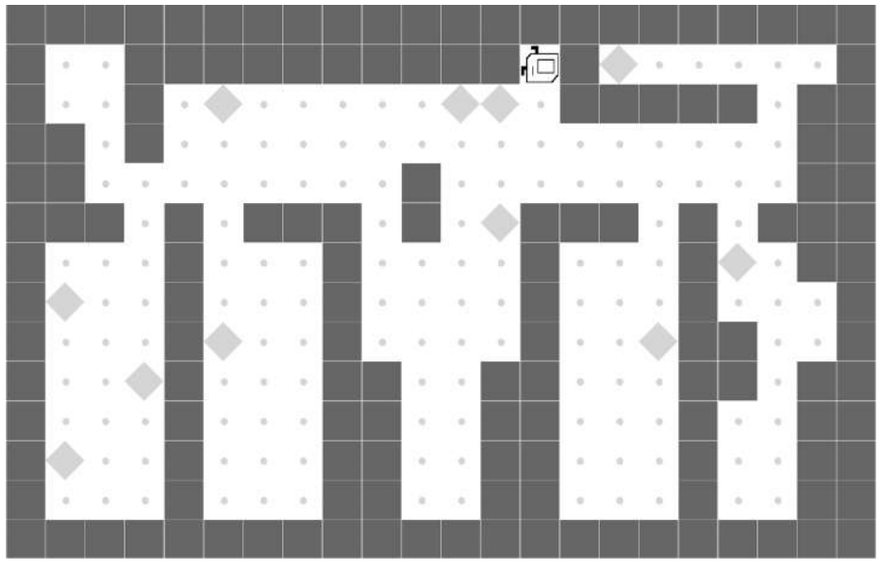
FourCorners



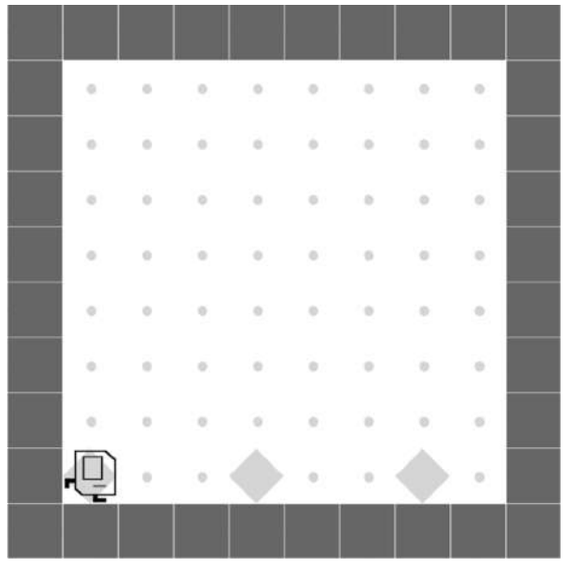
Harvester



CleanHouse

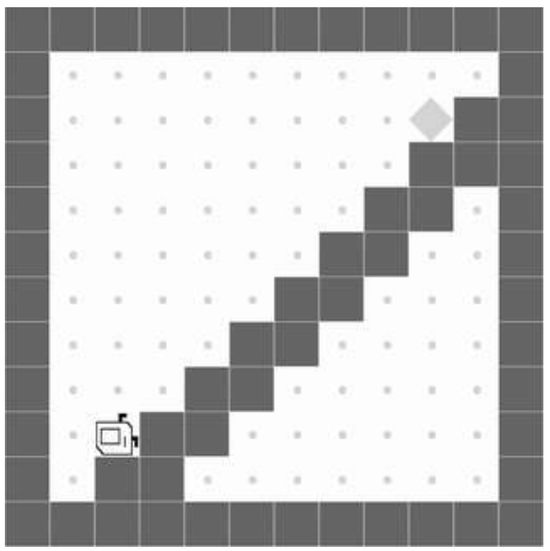


TopOff

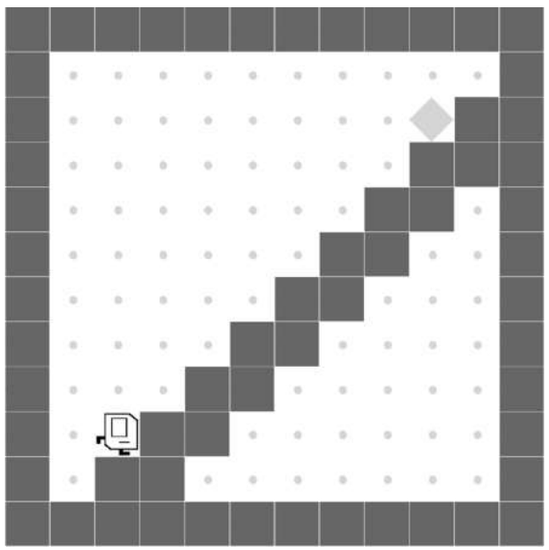


Qualitative Results

StairClimber

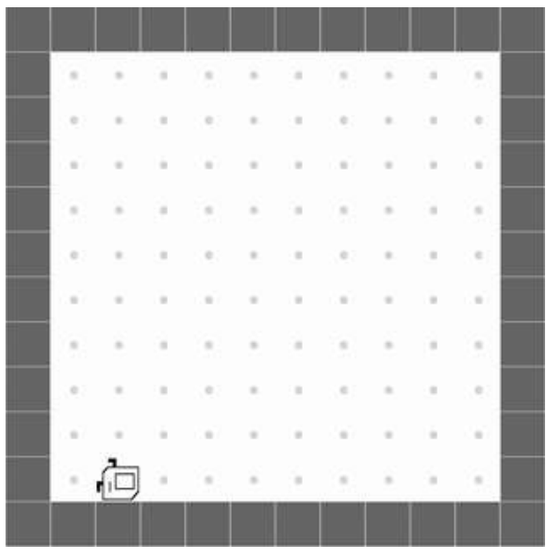


DRL

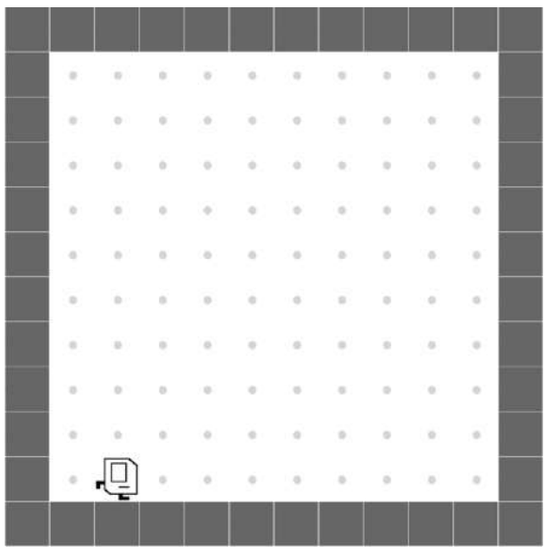


LEAPS

FourCorners

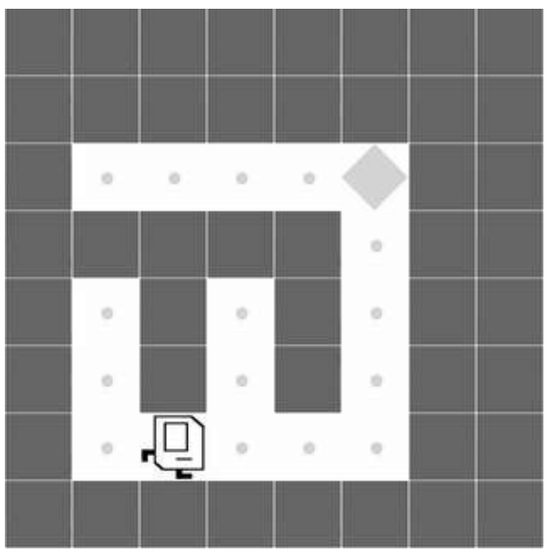


DRL

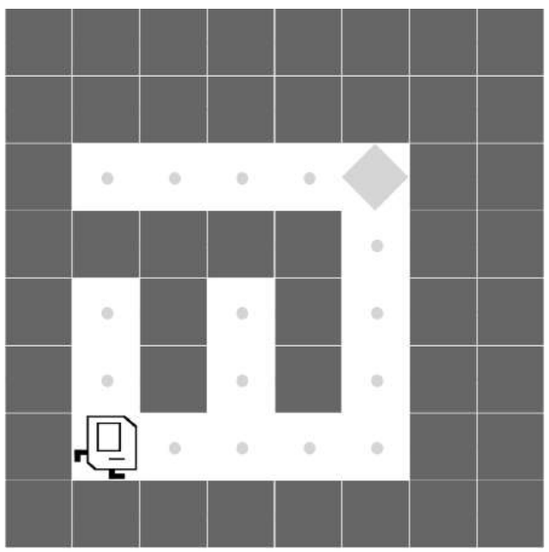


LEAPS

Maze

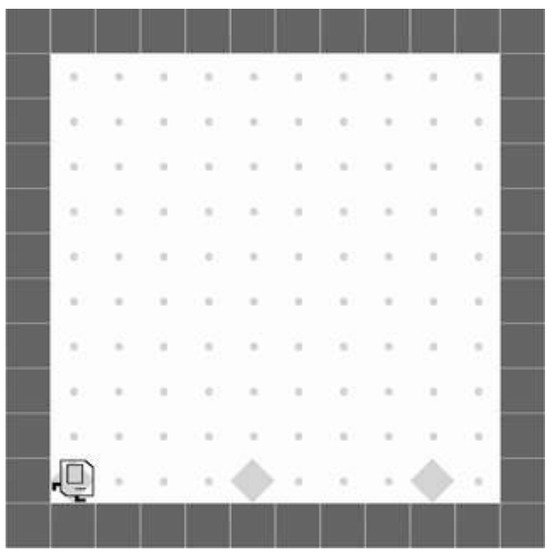


DRL

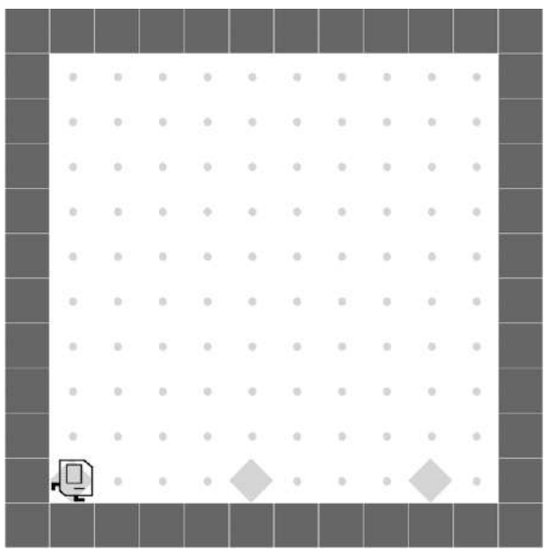


LEAPS

TopOff

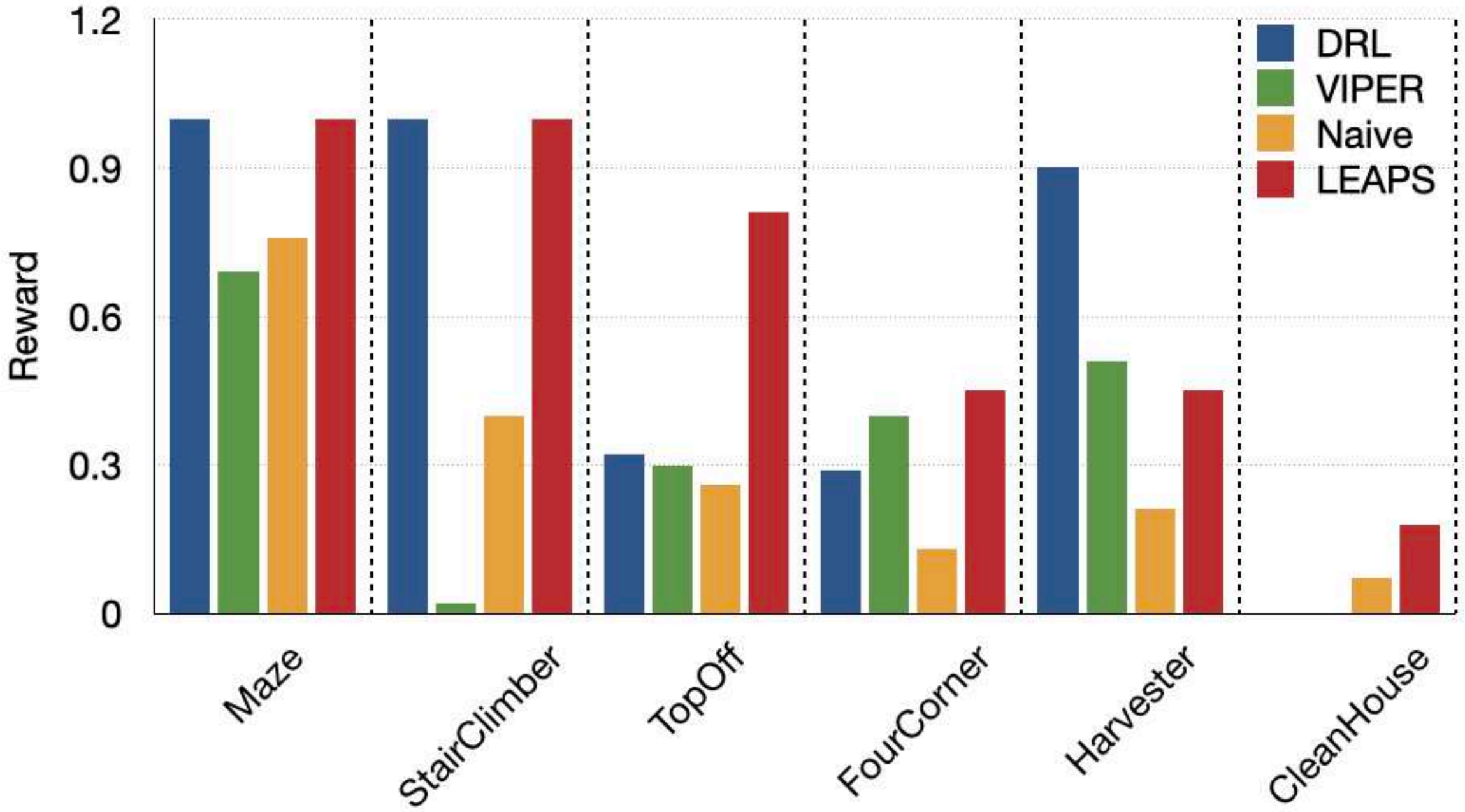


DRL



LEAPS

Quantitative Results

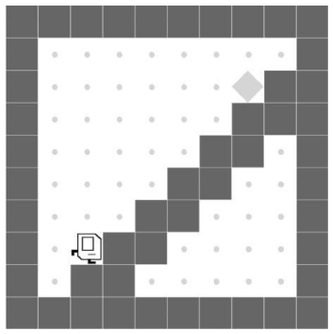


Zero-Shot Generalization

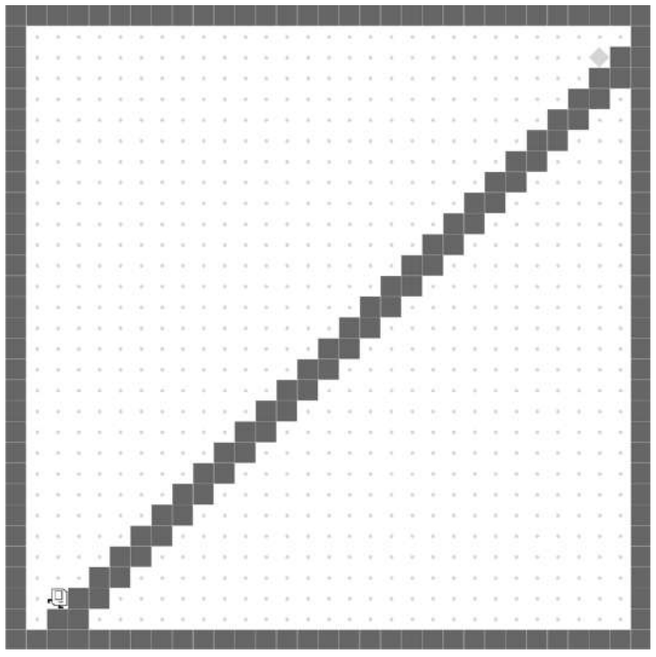
Learning on
8x8 grids

Evaluation on
100x100 grids

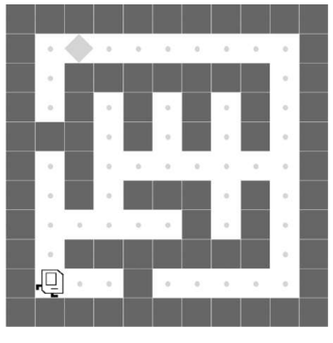
StairClimber



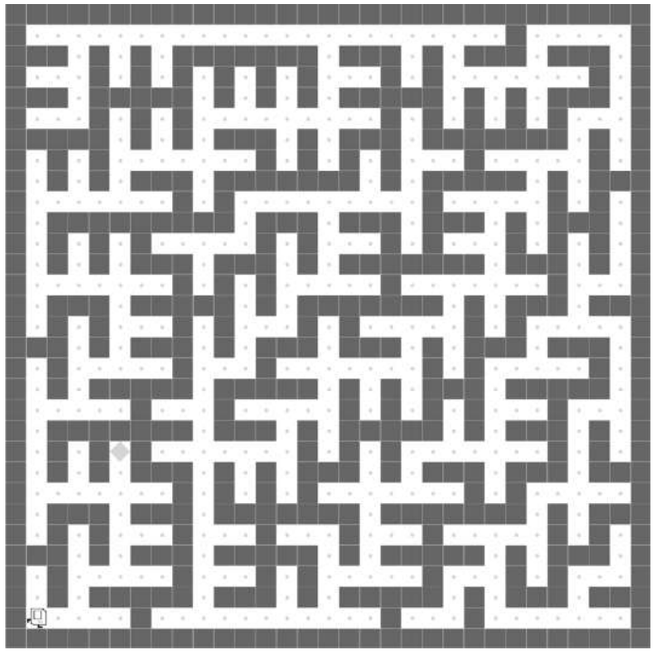
```
DEF run()
   WHILE noMarkersPresent()
      turnRight
      move
   WHILE rightIsClear()
      turnLeft
```



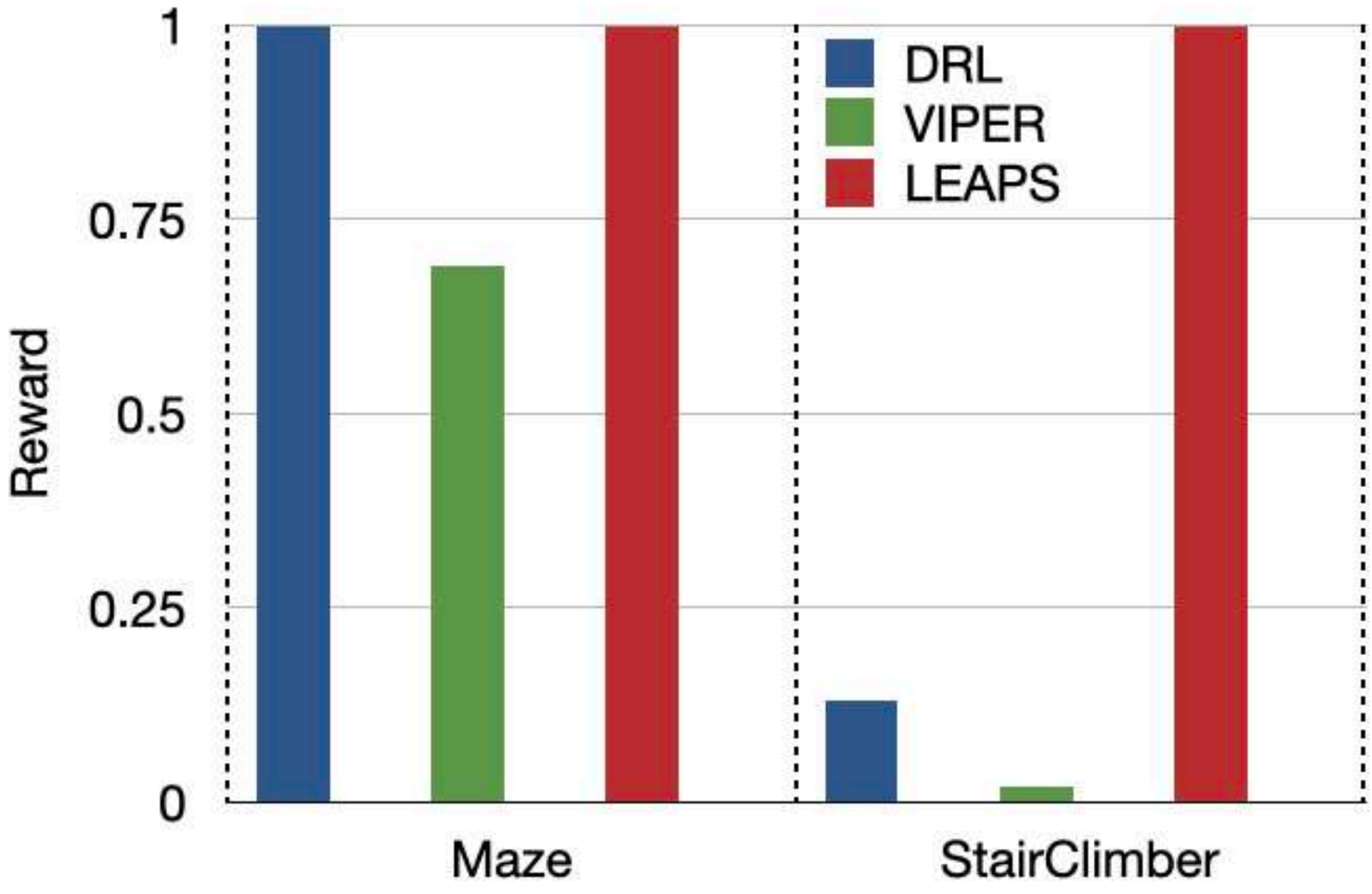
Maze



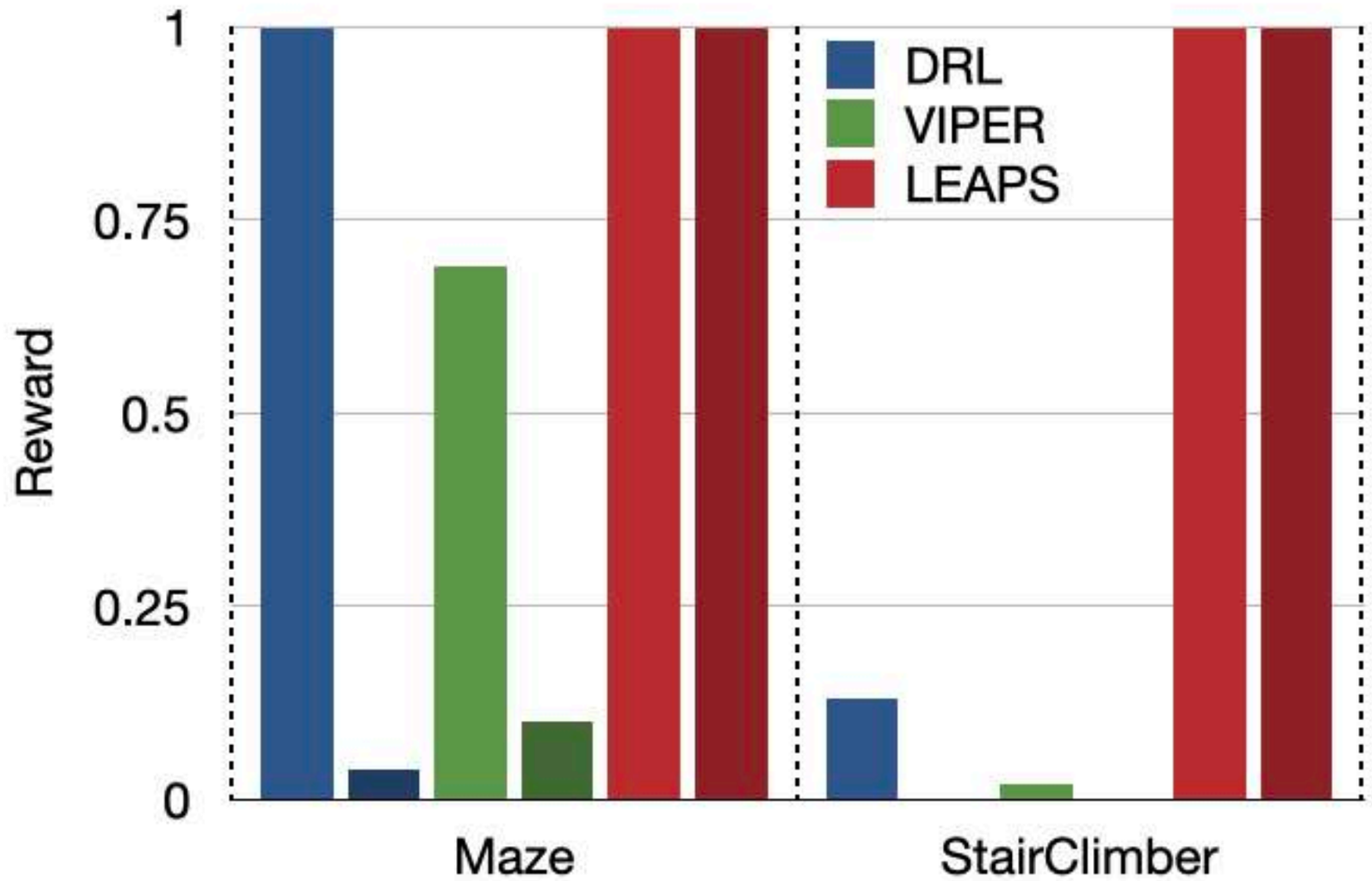
```
DEF run()
   IF frontIsClear()
      turnLeft
   WHILE noMarkersPresent()
      turnRight
      move
```



Zero-Shot Generalization

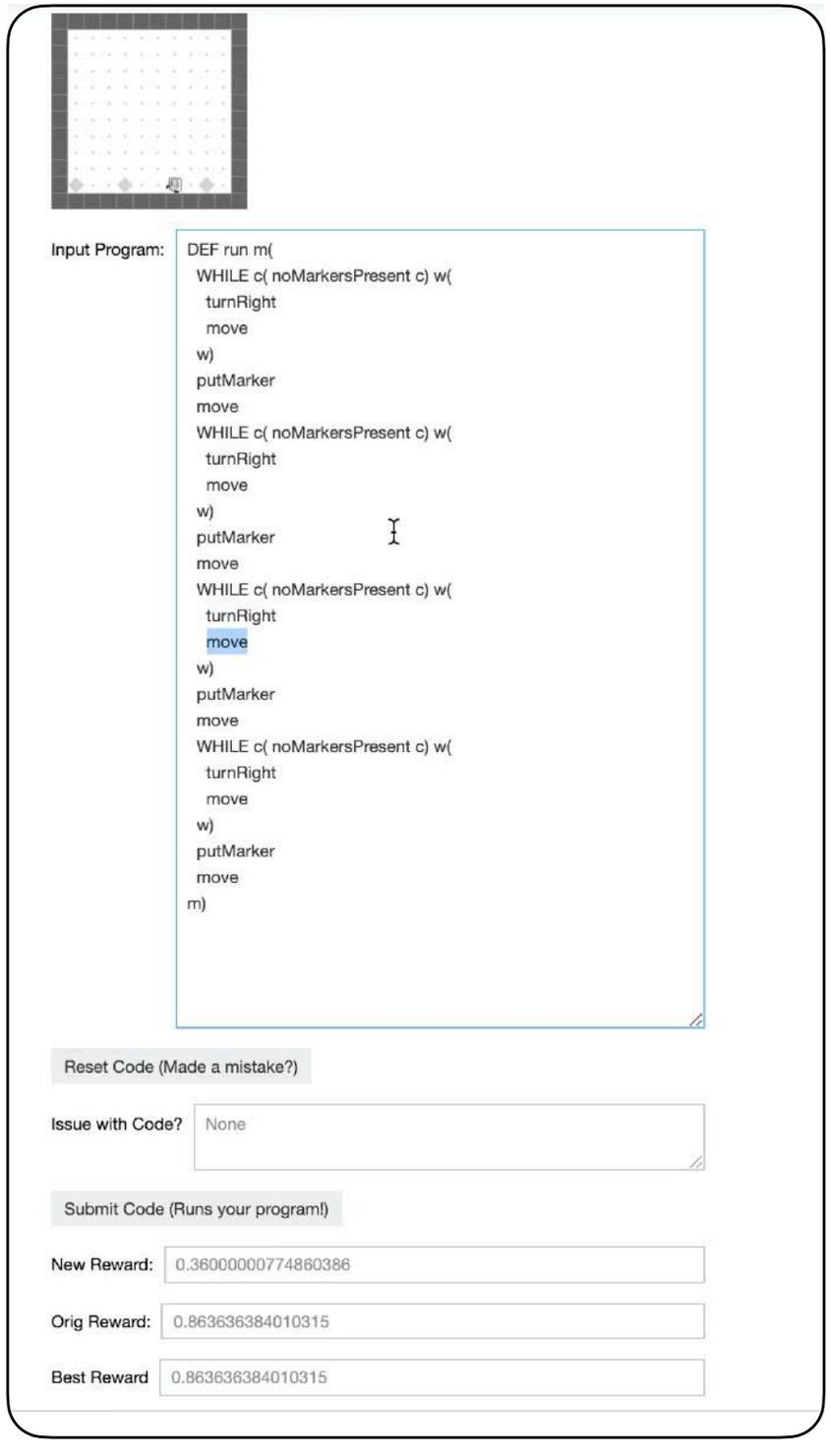


Zero-Shot Generalization

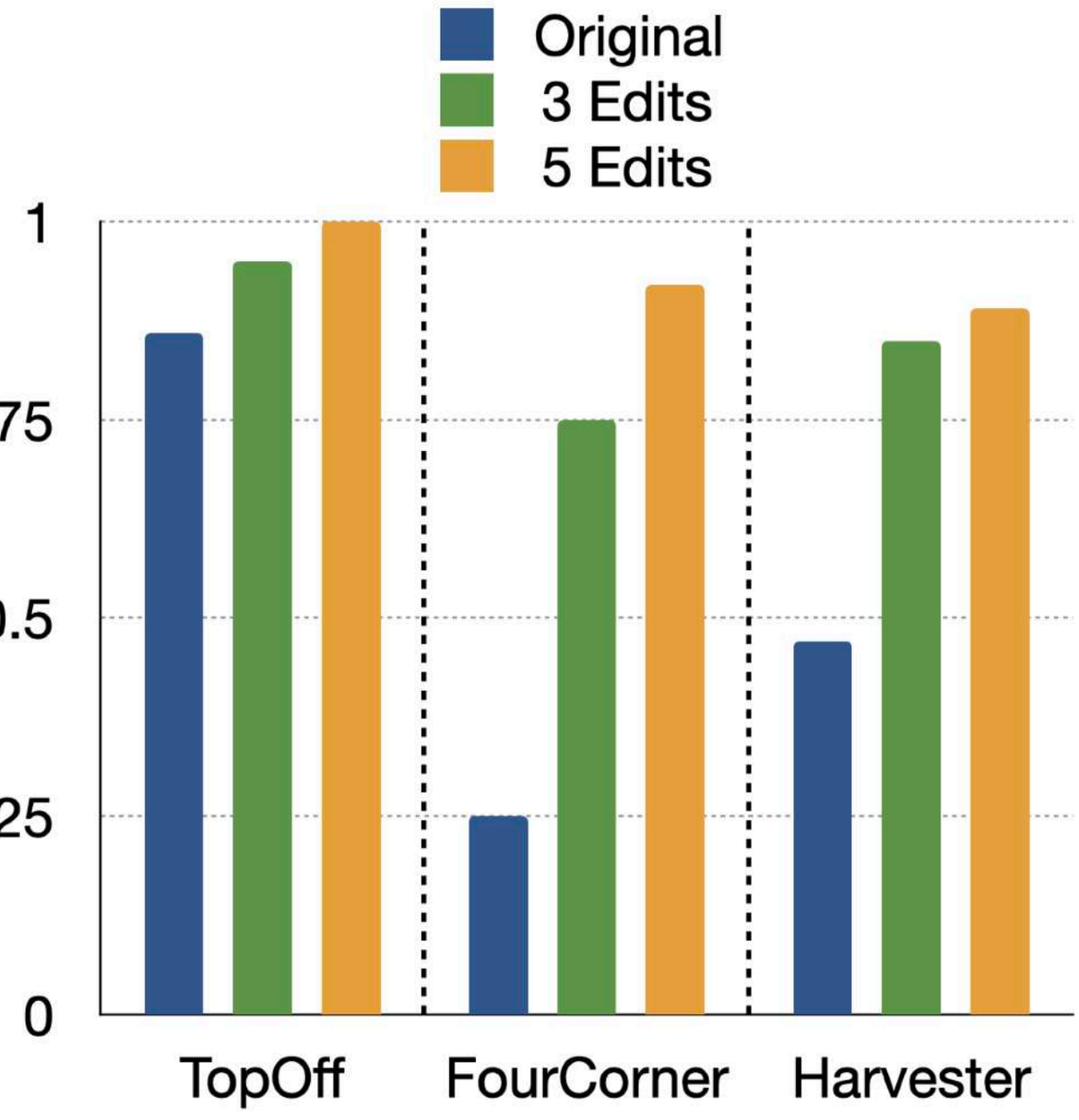


Interpretability

Human Debugging Interface



Improved Performance



Outline

- Recent Advance in Deep Learning
- Teach Neural Networks to Learn Rules
  - An Overview
  - Neural Program Induction
  - Neural Program Synthesis

Summary

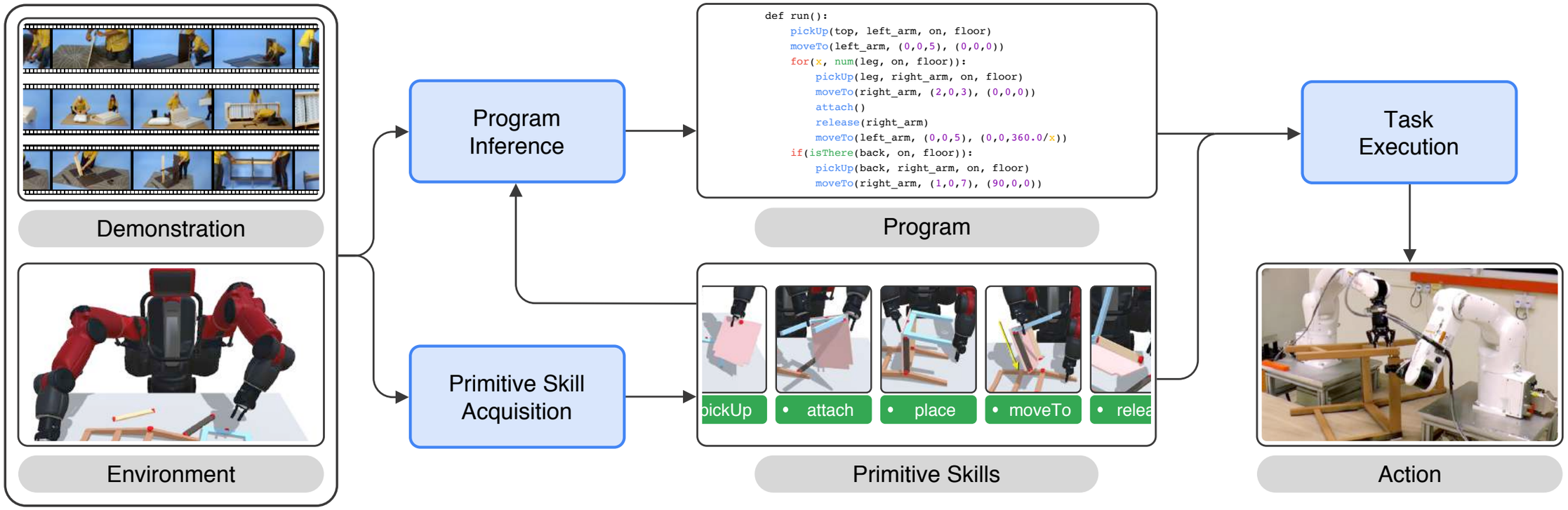
Recent advance in deep learning

Teach Neural Networks to Learn Rules

- Neural Program Synthesis
    - Infer the underlying program
    - Execute the program
- Neural Program Induction
    - Mimic the underlying program

My Research

Program-Guided Framework for Interpreting and Acquiring Complex Skills with Learning Robots

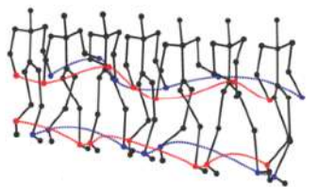


Program-Guided Framework for Interpreting and Acquiring Complex Skills with Learning Robots

Skill Specification → Program → High-level Plan → Low-level Execution

Video demonstration

Trajectory demonstration

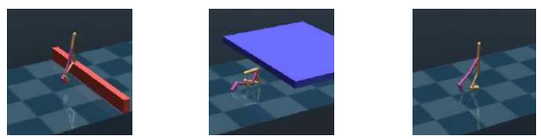


Language instruction

Stir-fry the onions until tender, and repeat this for garlic and carrots, if you have soy sauce, add some. Pour 2/3 cups the whisked eggs into the stir-fried and scramble.

Reward

Actions (sub-skills)



`Jump` `Crawl` `Walk`

Perceptions

`isCurb()` `isCeiling()`

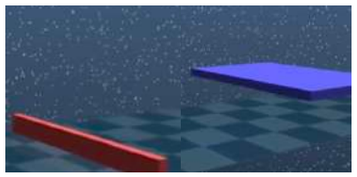
Control Flows

`while` `if` `else` `elseIf`

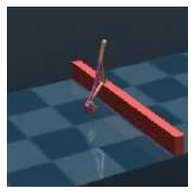
Program

```
DEF run()
    while True:
        if isCurb():
            jump
        elseIf isCeiling():
            crawl()
        else:
            walk()
```

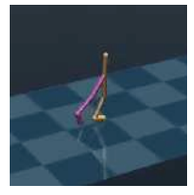
Observation



`Jump`

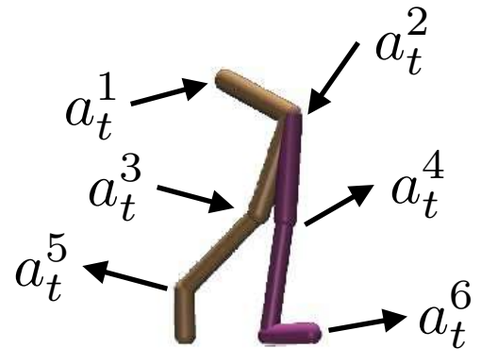


`Walk`



`Crawl`

Low-level execution

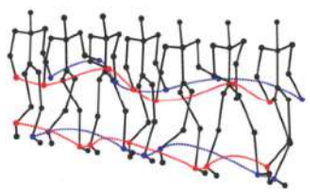


Program Inference

Skill Specification → Program → High-level Plan → Low-level Execution

Video demonstration

Trajectory demonstration

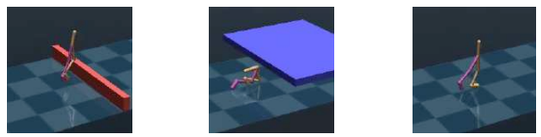


Language instruction

Stir-fry the onions until tender, and repeat this for garlic and carrots, if you have soy sauce, add some. Pour 2/3 cups the whisked eggs into the stir-fried and scramble.

Reward

Actions (sub-skills)



`Jump` `Crawl` `Walk`

Perceptions

`isCurb()` `isCeiling()`

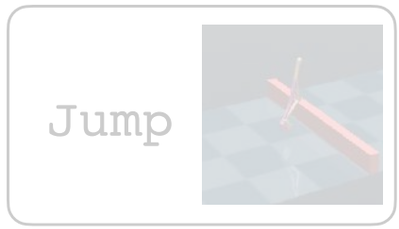
Control Flows

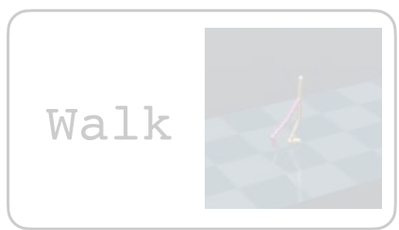
`while` `if` `else` `elseIf`

Program

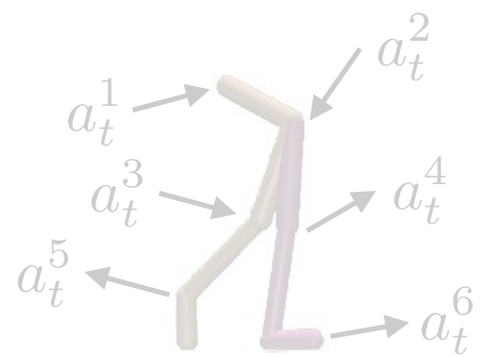
```
DEF run()
    while True:
        if isCurb():
            jump
        elseIf isCeiling():
            crawl()
        else:
            walk()
```

Observation



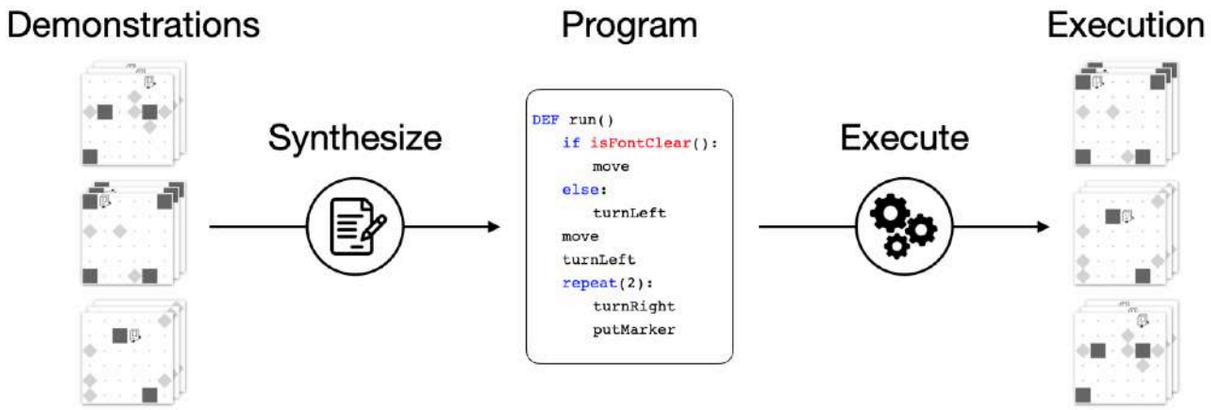


Low-level execution

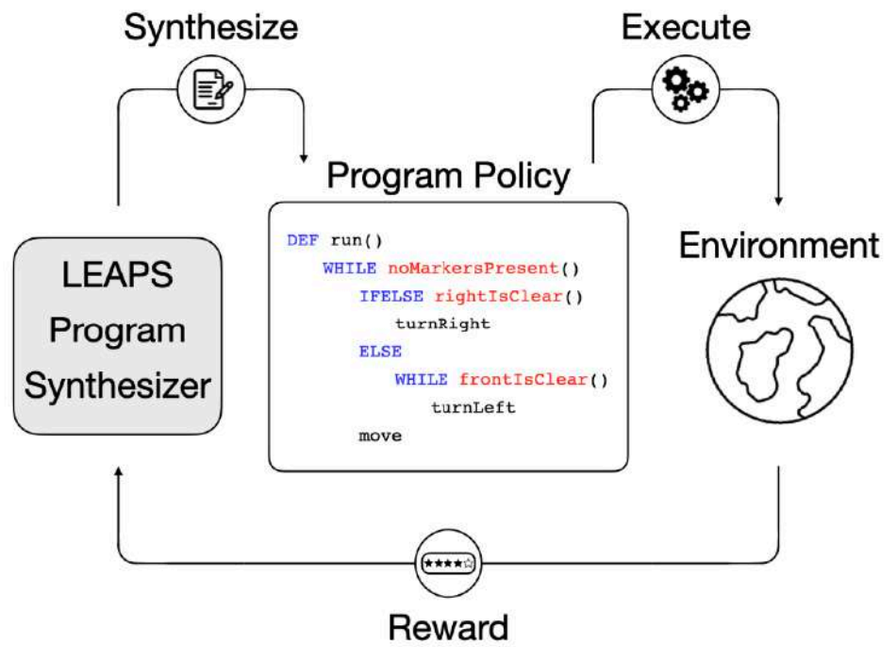


Program Inference

Imitation learning from demonstrations



Reinforcement learning from rewards



Covered Papers

Neural Program Induction

Neural Turing Machines

- Reinforcement Learning Neural Turing Machines - Revised
- Neural GPUs Learn Algorithms
- Neural Program Lattices
- Extensions and Limitations of the Neural GPU
- Neural Random-Access Machines

Learning Simple Algorithms from Examples

- Neural Program Meta-Induction

Neural Programmer-Interpreters

- Making Neural Programming Architectures Generalize via Recursion

Neural Task Programming: Learning to Generalize Across Hierarchical Tasks

- Neural Task Graphs: Generalizing to Unseen Tasks from a Single Video Demonstration

Covered Papers

Neural Program Synthesis

RobustFill: Neural Program Learning under Noisy I/O

DeepCoder: Learning to Write Programs

Leveraging Grammar and Reinforcement Learning for Neural Program Synthesis

- Neuro-Symbolic Program Synthesis
- Synthetic Datasets for Neural Program Synthesis

Neural Program Synthesis from Diverse Demonstration Videos

Improving Neural Program Synthesis with Inferred Execution Traces

Execution-Guided Neural Program Synthesis

Neural Scene De-rendering

Learning to Describe Scenes with Programs

pix2code: Generating Code from a Graphical User Interface Screenshot

Learning to Infer and Execute 3D Shape Programs

NL2Bash: A Corpus and Semantic Parser for Natural Language Interface to the Linux Operating System

Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning

Ain't Nobody Got Time For Coding: Structure-Aware Program Synthesis From Natural Language

Competition-Level Code Generation with AlphaCode

Learning to Synthesize Programs as Interpretable and Generalizable Policies

Thank You

Questions?