Neurosymbolic Learning

CIS 7000: Trustworthy Machine Learning, Apr 17

Ziyang Li, University of Pennsylvania

Ziyang Li

Jiani Huang

Prof. Mayur Naik

Joint Work With:
Jason Liu, Felix Zhu, Eric Zhao, William Dodds,
Neelay Velingker, Rajeev Alur

The need for Neurosymbolic AI

Two separate paradigms of programming

Classical Algorithms

Suited for exactly defined tasks on structured input domains

Deep Learning

Suited for tasks which cannot be hand-programmed or have unstructured input

Two separate paradigms of programming

Classical Algorithms

Suited for exactly defined tasks on structured input domains

e.g.

- Sort a list of numbers
- Find shortest path
- Solve boolean constraints

Deep Learning

Suited for tasks which cannot be hand-programmed or have unstructured input

Two separate paradigms of programming

Classical Algorithms

Suited for exactly defined tasks on structured input domains

e.g.

- Sort a list of numbers
- Find shortest path
- Solve boolean constraints

Deep Learning

Suited for tasks which cannot be hand-programmed or have unstructured input

e.g.

- Detecting objects in image
- Parse natural language text
- Control in physical environment

Neurosymbolic to combine both worlds…

Classical Algorithms

Suited for exactly defined tasks on structured input domains

Deep Learning

Suited for tasks which cannot be hand-programmed or have unstructured input

symbolic ⊕ neural = neurosymbolic

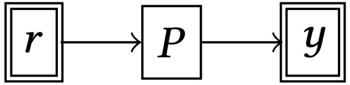
Neurosymbolic Learning…

=

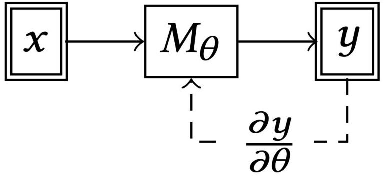
Machine learning with both neural and symbolic components

An Example of Neurosymbolic Learning

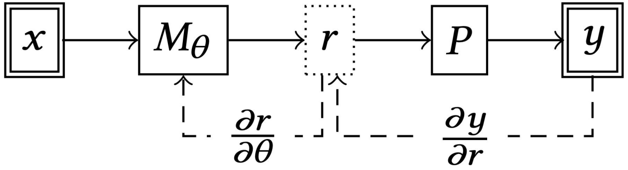
Neurosymbolic Learning of `addition(`, , `8)`



Logic Program

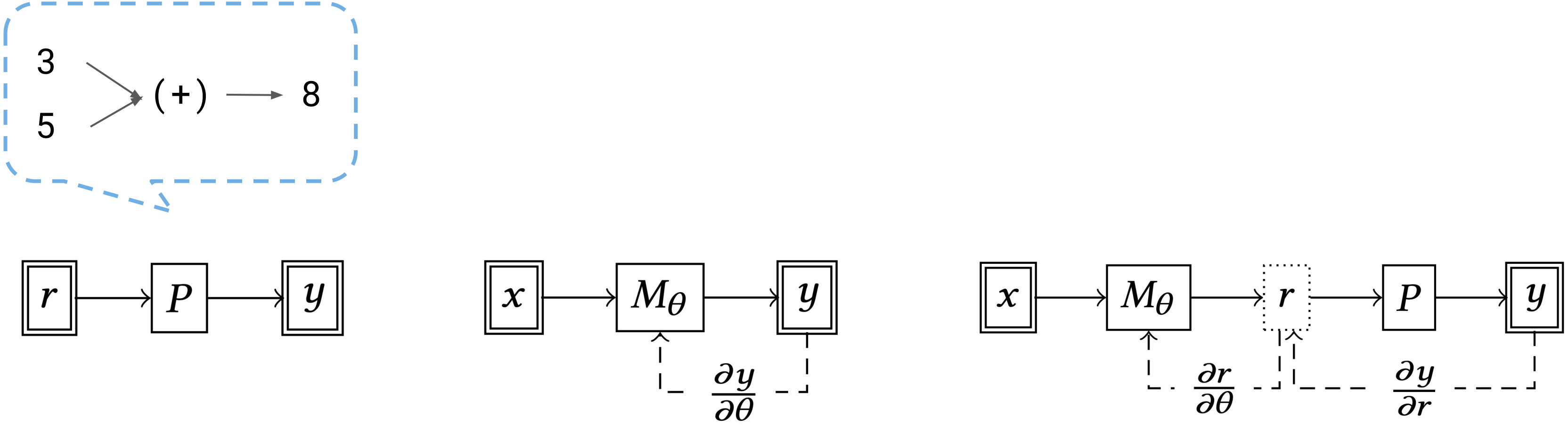


Neural Model



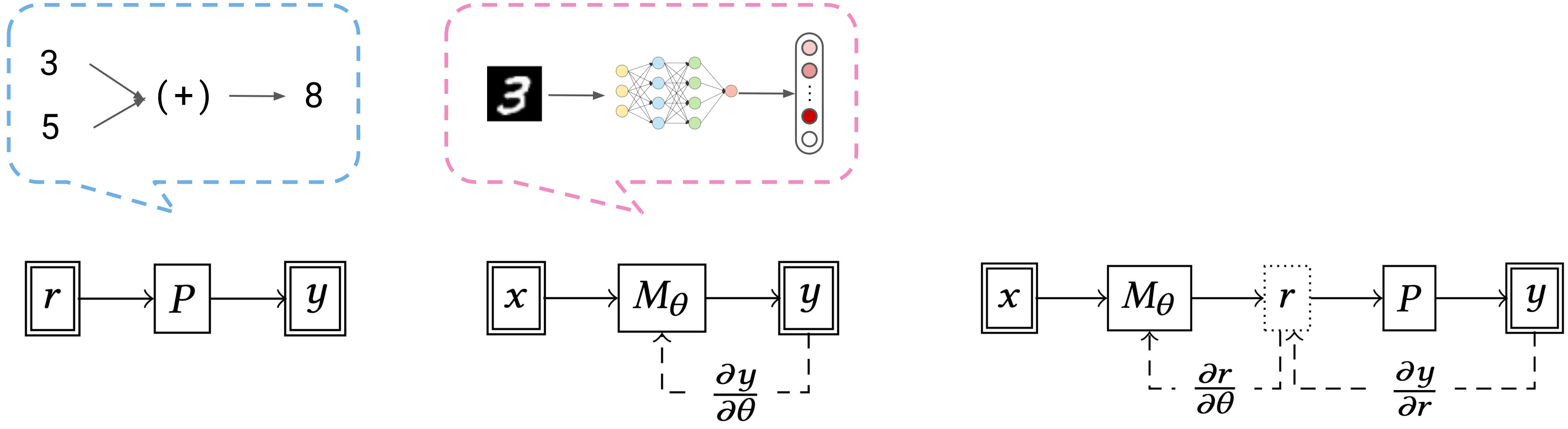
A Neurosymbolic Program

Neurosymbolic Learning of addition(, , 8)



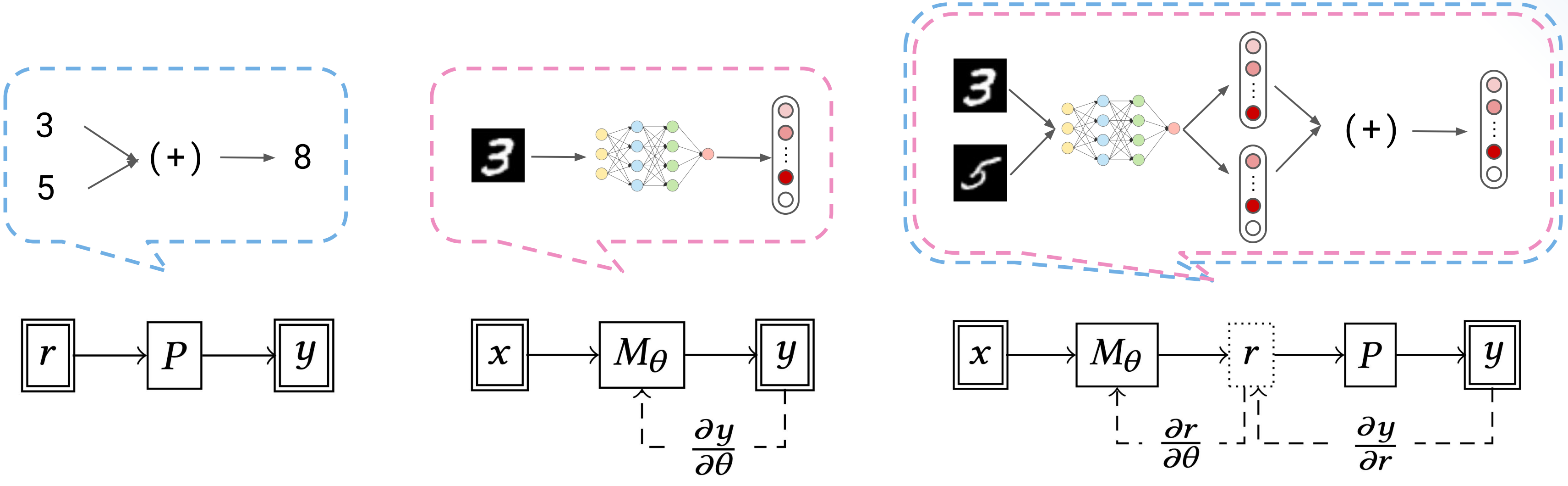
Logic Program   Neural Model   A Neurosymbolic Program

Neurosymbolic Learning of `addition(3, 5, 8)`



Logic Program | Neural Model | A Neurosymbolic Program

Neurosymbolic Learning of `addition`(3, 5, 8)

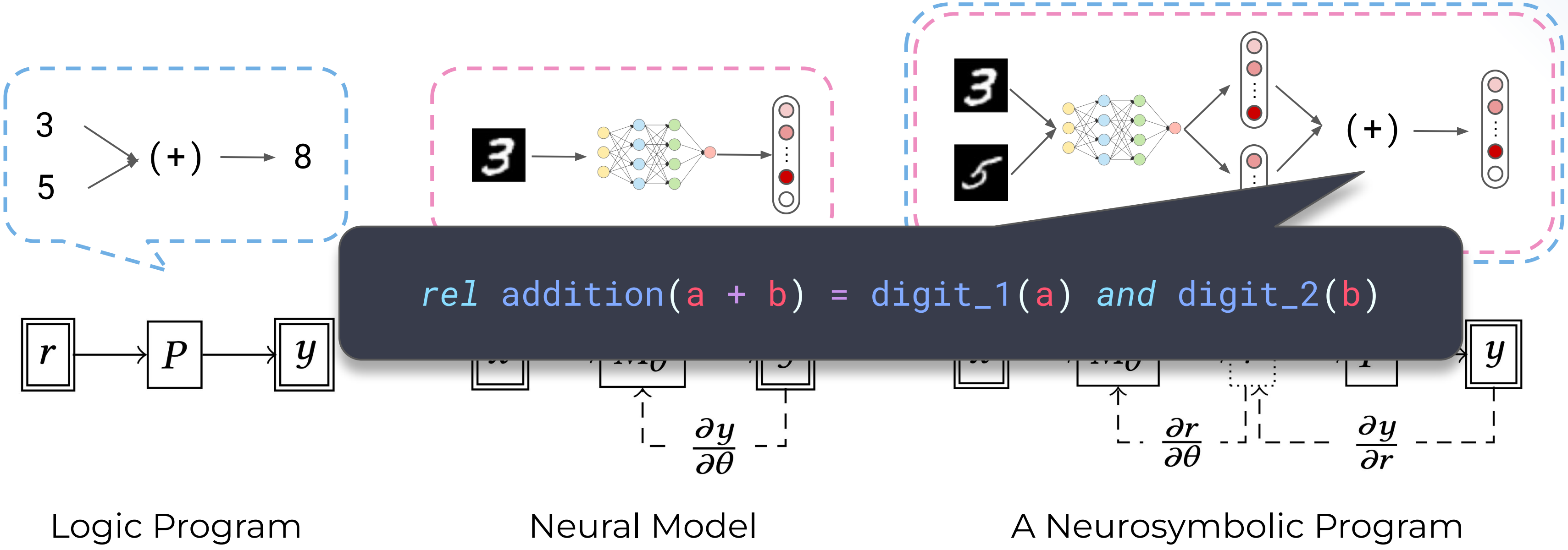


Logic Program

Neural Model

A Neurosymbolic Program

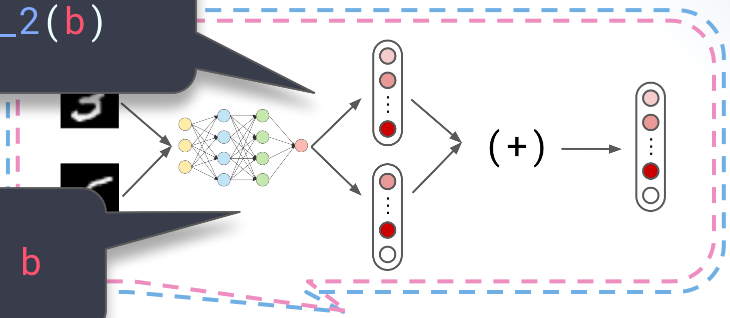
Neurosymbolic Learning of addition(3, 5, 8)



Neurosymbolic Learning of addition(3, 5, 8)

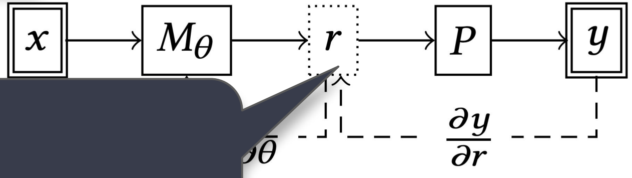
```
rel addition(a + b) = digit_1(a) and digit_2(b)
```

Scallop (Li et. al., Datalog like syntax)




```
addition(r) :- digit_1(a), digit_2(b), r is a + b
```

DeepProblog (Manheave et. al., Prolog like syntax)



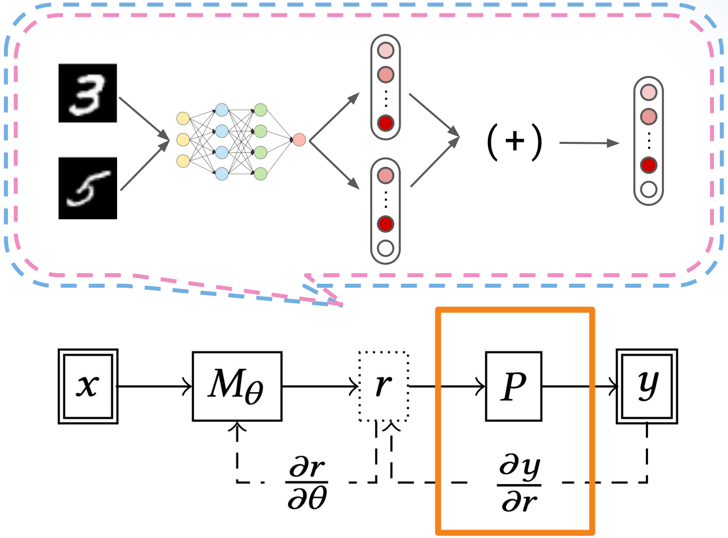

```
@ised.blackbox(a=IntRange(10), b=IntRange(10),
  output=IntRange(19))
def addition(a, b): return a + b
```

ymbolic Program

ISED (Breslin et. al., Python like syntax)

Neurosymbolic Learning of `addition(`, , `8)`

Neurosymbolic Frameworks aim to provide a programming interface for developers to write Neurosymbolic solutions.



A Neurosymbolic Program

A More Difficult Motivating Example: PacMan-Maze

A Motivating Example: PacMan-Maze

Step 0 Step 4 Step 7

State: 200x200 colored image
Action: Up, Down, Left, Right
(Environments are 5x5 grids randomized for each session)

How to combine neural and symbolic components?

State: 200x200 colored image
Action: Up, Down, Left, Right

(Environments are 5x5 grids randomized for each session)

1. What is the neural component?

   CNN that parses the image into entity positions

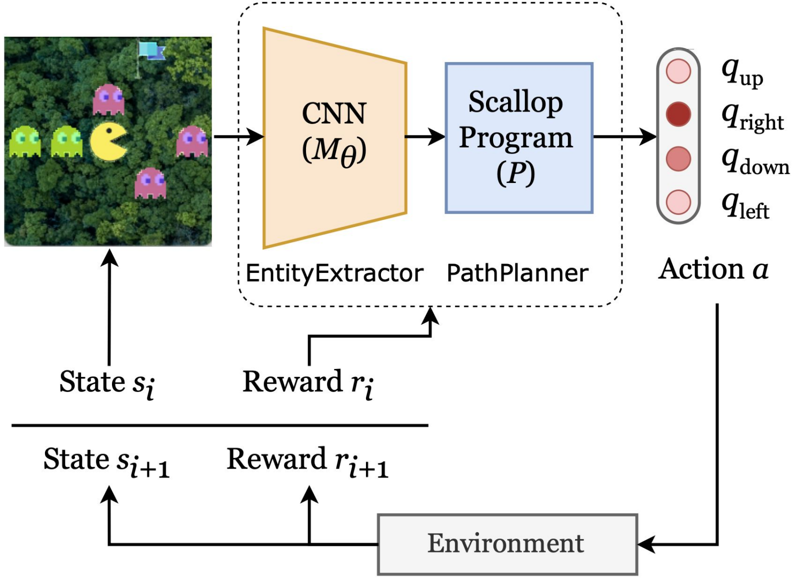
2. What does the symbolic program do?

   Plan the optimal action to take at each state, given the possibly noisy entity positions

How to combine neural and symbolic components?

State: 200x200 colored image
Action: Up, Down, Left, Right

(Environments are 5x5 grids randomized for each session)



Results after combining neural and symbolic

Step 0 Step 4 Step 7

State: 200x200 colored image
Action: Up, Down, Left, Right

(Environments are 5x5 grids randomized for each session)

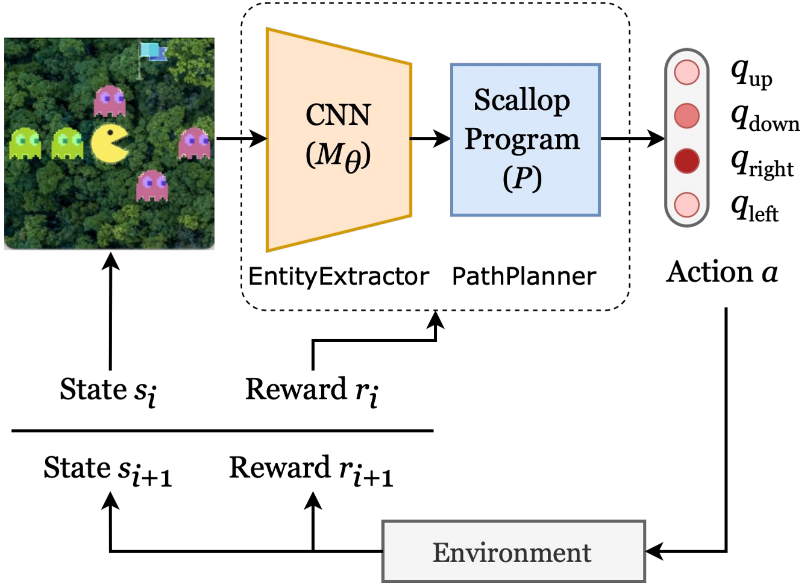
	Neurosymbolic (with Scallop)	**DQN**
Success rate (reaches the goal within 50 steps)	**99.4%**	84.9%
# of Training episodes (to achieve the success rate)	**50**	50K

(Note: this is not entirely a fair comparison since our Scallop program encodes system dynamics and human knowledge)

A Motivating Example: PacMan-Maze

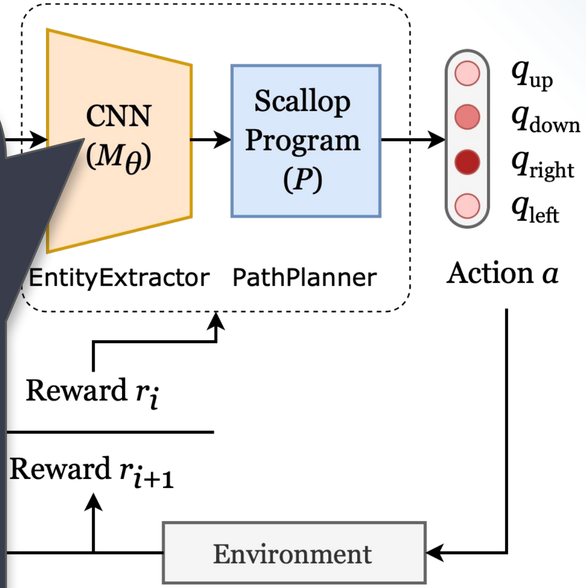
State: 200x200 colored image
Action: Up, Down, Left, Right

(Environments are 5x5 grids randomized for each session)

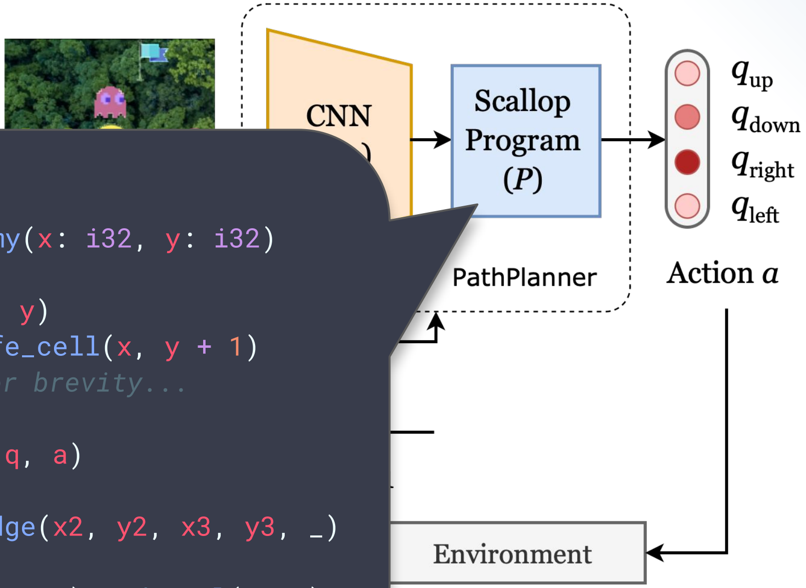


A Motivating Example: PacMan-Maze

```
class EntityExtractor(nn.Module):
  def __init__(self):
    super(EntityExtractor, self).__init__()
    self.conv1 = nn.Conv2d(...)
    self.conv2 = nn.Conv2d(...)
    self.fc1 = nn.Linear(in_features=288, out_features=256)
    self.fc2 = nn.Linear(in_features=256, out_features=4)
    self.relu = nn.ReLU()

  def forward(self, x):
    batch_size, _, _, _ = x.shape
    x = self.relu(self.conv1(x))
    x = self.relu(self.conv2(x))
    x = x.view(batch_size, -1)
    x = self.fc2(self.relu(self.fc1(x)))
    return torch.softmax(x, dim=1)
```

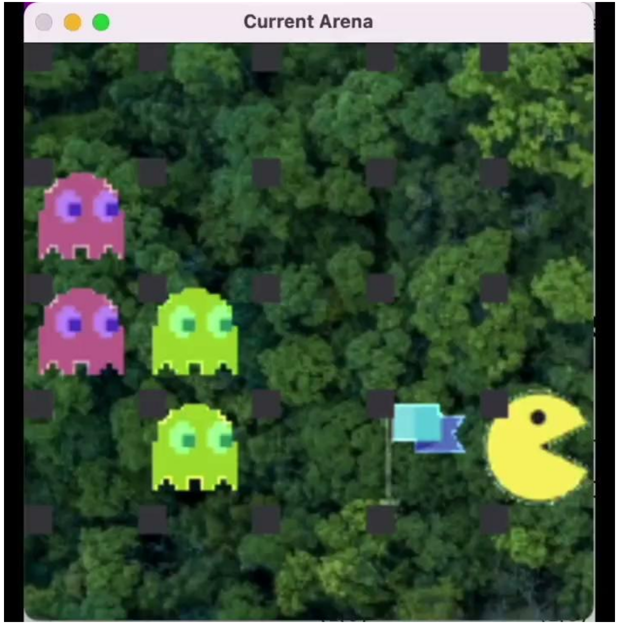


A Motivating Example: PacMan-Maze




```
type Action = UP | RIGHT | DOWN | LEFT
type actor(x: i32, y: i32), goal(x: i32, y: i32), enemy(x: i32, y: i32)

rel safe_cell(x, y) = grid_cell(x, y) and not enemy(x, y)
rel edge(x, y, x, y + 1, UP) = safe_cell(x, y) and safe_cell(x, y + 1)
// Rules for RIGHT, DOWN, and LEFT edges are omitted for brevity...

rel next_pos(p, q, a) = actor(x, y) and edge(x, y, p, q, a)
rel path(x, y, x, y) = next_pos(x, y, _)
rel path(x1, y1, x3, y3) = path(x1, y1, x2, y2) and edge(x2, y2, x3, y3, _)

rel next_action(a) = next_pos(p, q, a) and path(p, q, r, s) and goal(r, s)
```

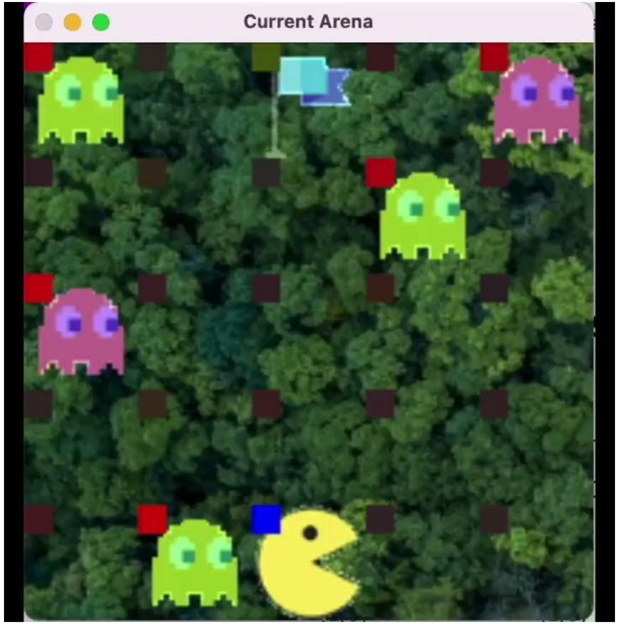
Demo – Training the Agent!



Marker on the top-left indicating NN prediction on what the cell represents (Saturation indicates confidence)

- Enemy (Red)
- PacMan (Blue)
- Goal (Green)

Demo – Testing the Agent!



Marker on the top-left indicating NN prediction on what the cell represents (Saturation indicates confidence)

- Enemy (Red)
- PacMan (Blue)
- Goal (Green)

Key Take Aways

Decomposing end-to-end neural solutions into separated perception + reasoning solutions...

1. **Improves accuracy**

   Reasoning module is generalizable to combinatorically diverse situations

2. **Learns faster**

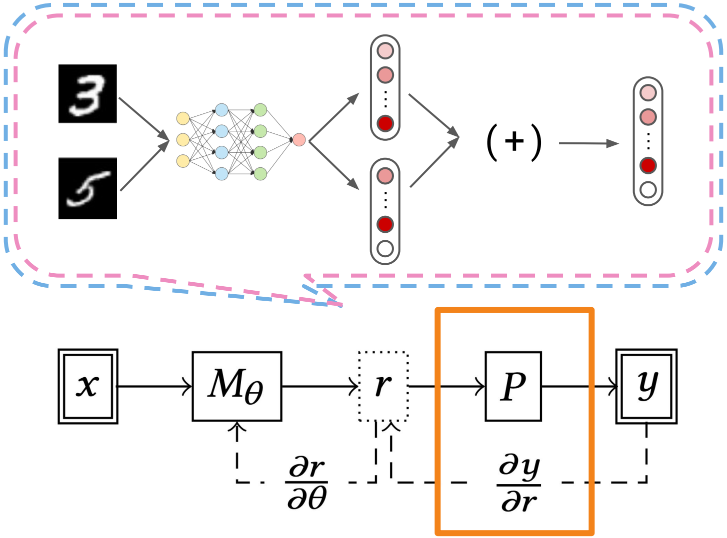
   Neural models are good at recognizing patterns; and they are used only for it

3. **Is explainable**

   Decomposition gives explicit meaning to intermediate information

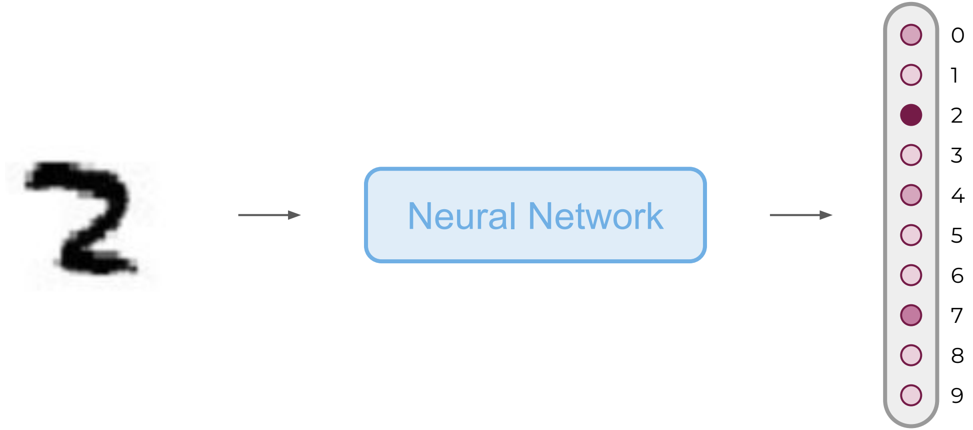
Differentiating Symbolic Programs

Back to adding two MNIST digits...

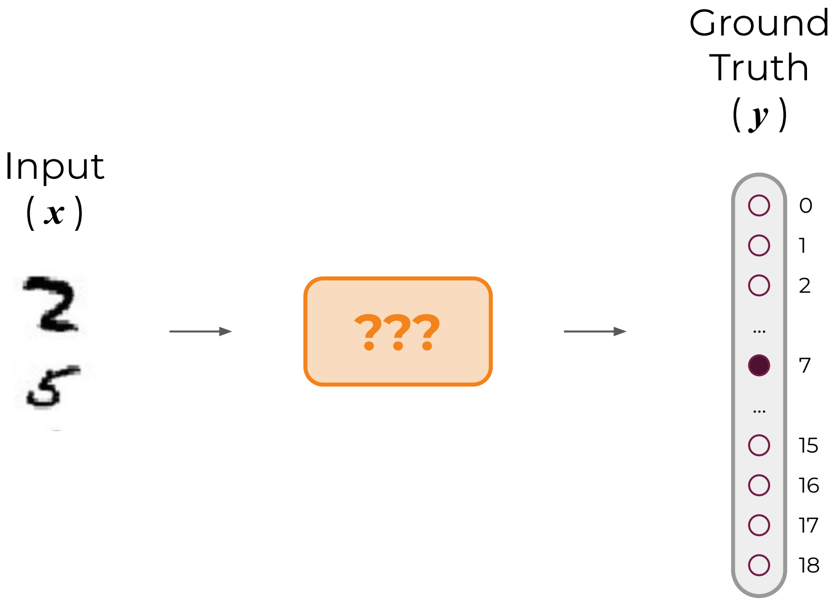


A Neurosymbolic Program

Using Neural Network to Classify One MNIST Digit

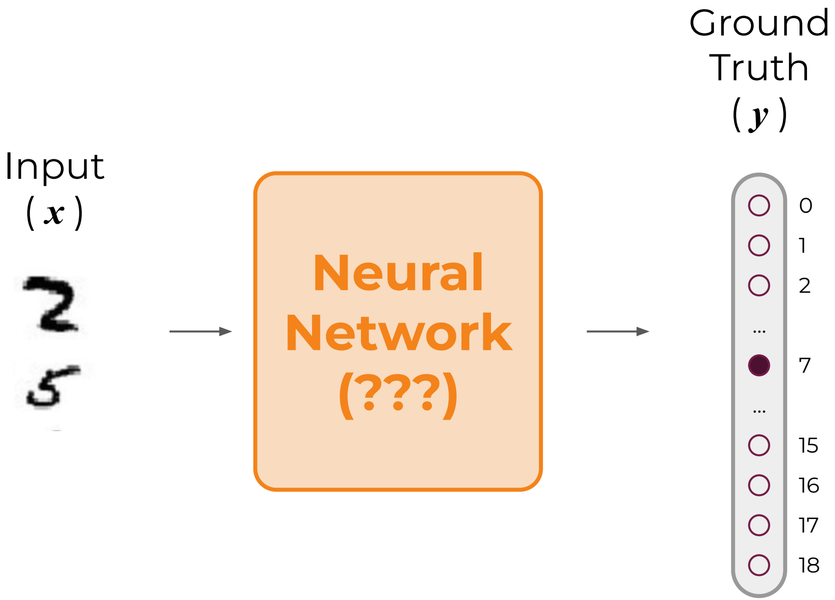


Training Loop for MNIST Addition Task

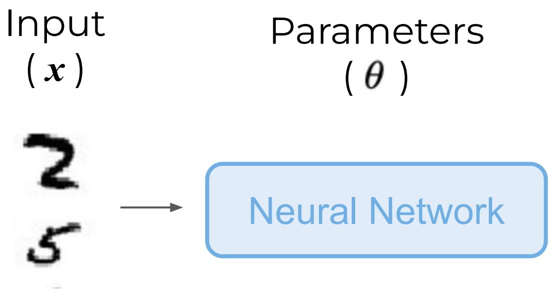


Task Initially Proposed by DeepProbLog (R Manhaeve et. al, 2018)

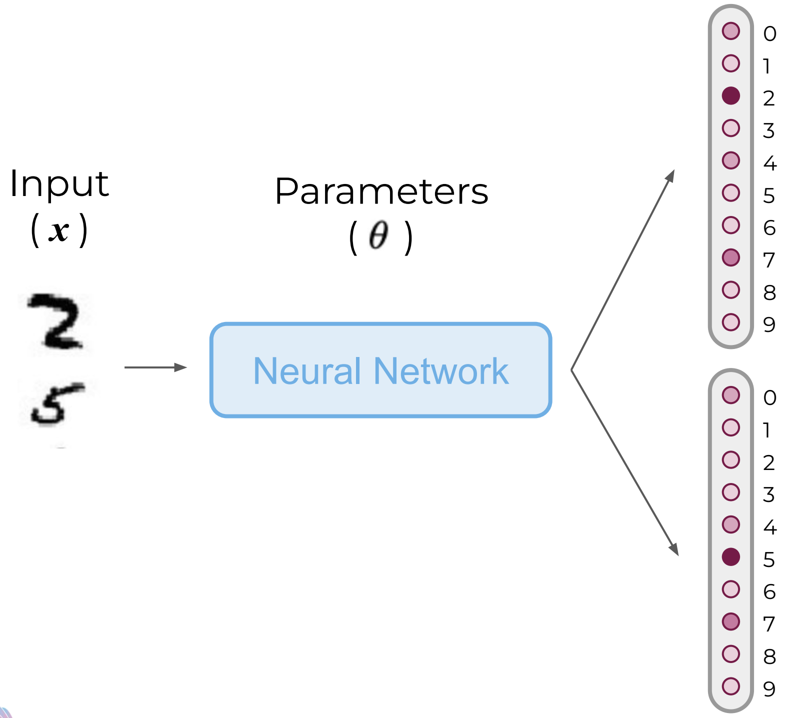
Training Loop for MNIST Addition Task



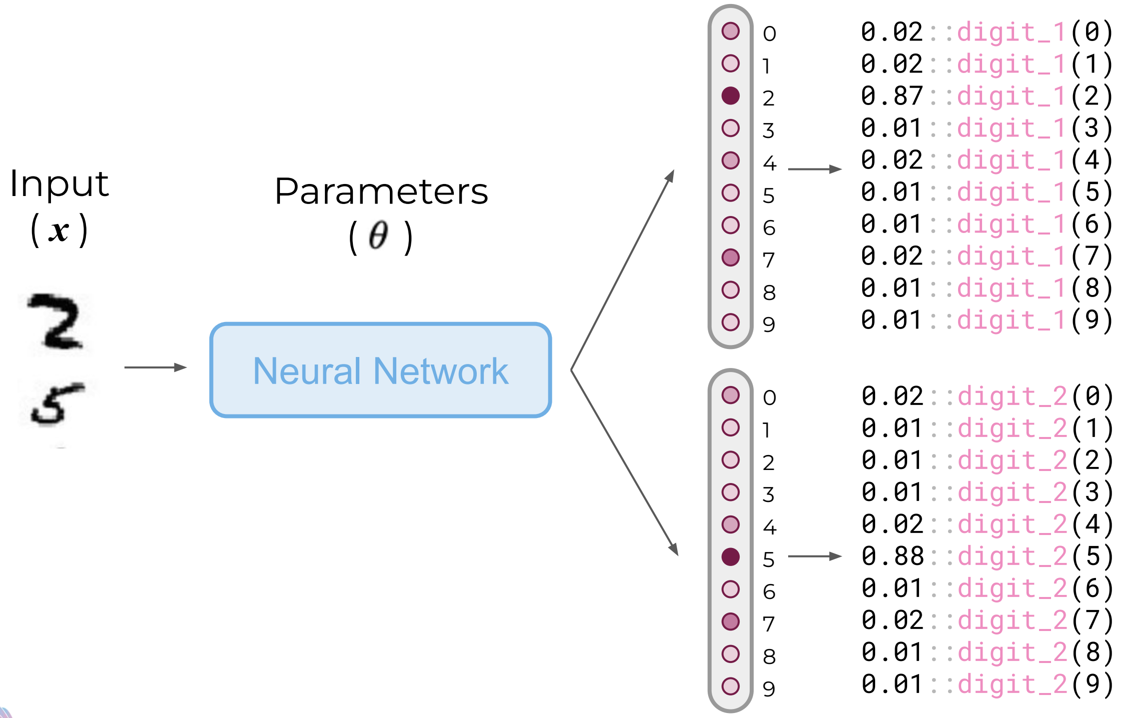
Training Loop for MNIST Addition Task



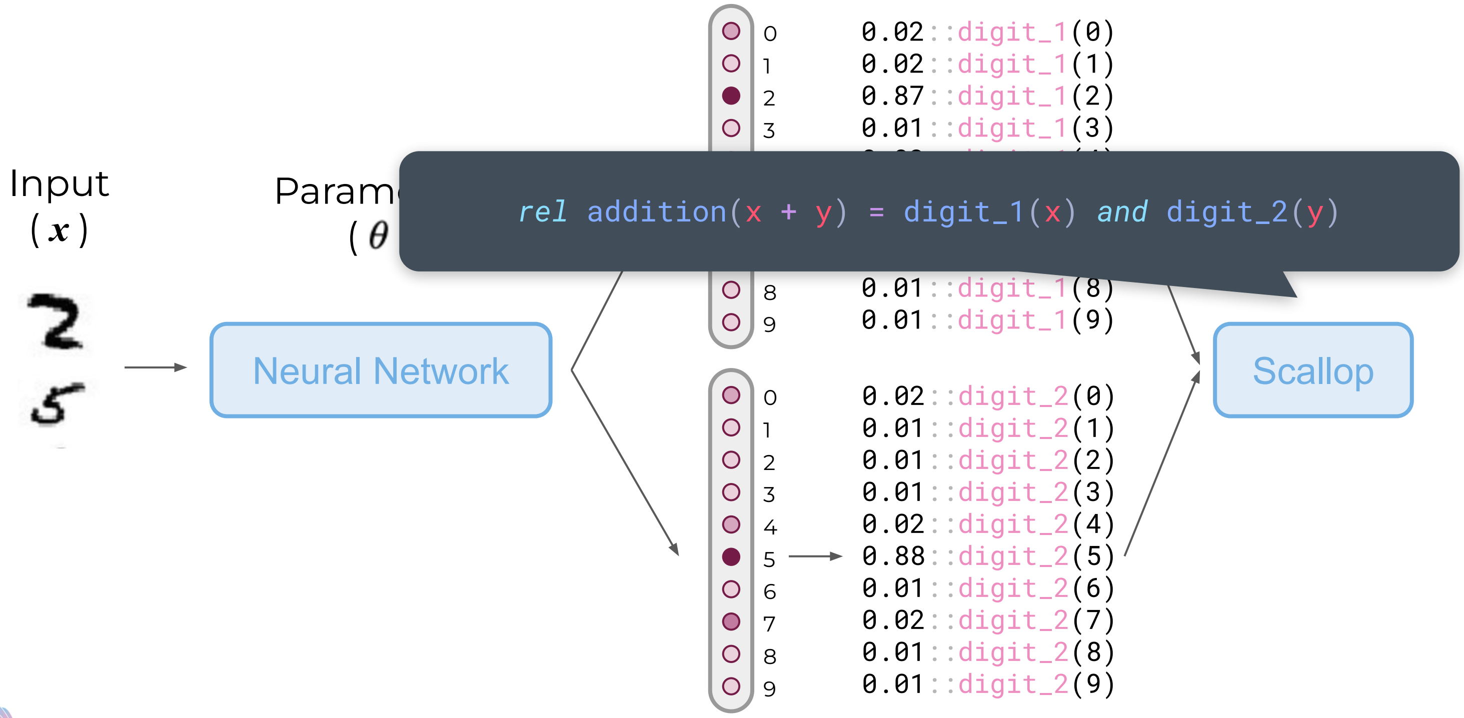
Training Loop for MNIST Addition Task



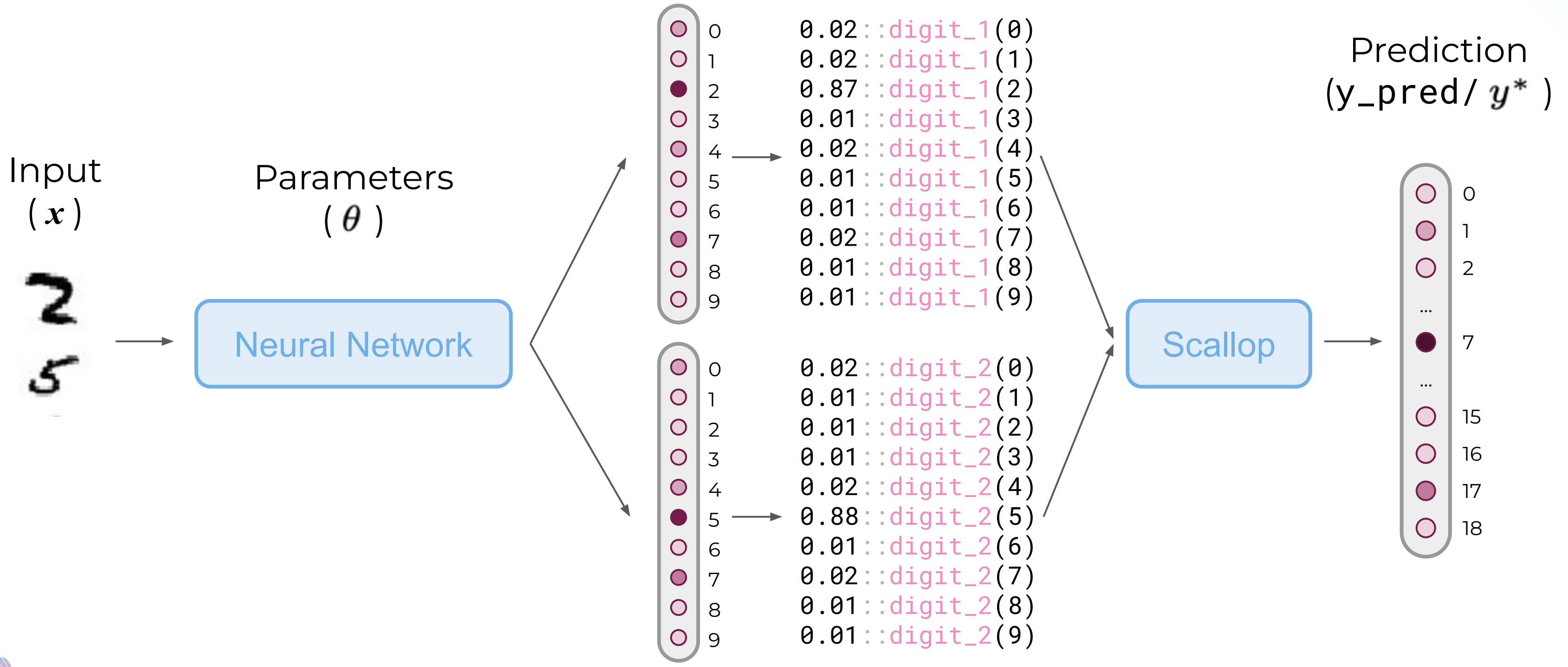
Training Loop for MNIST Addition Task

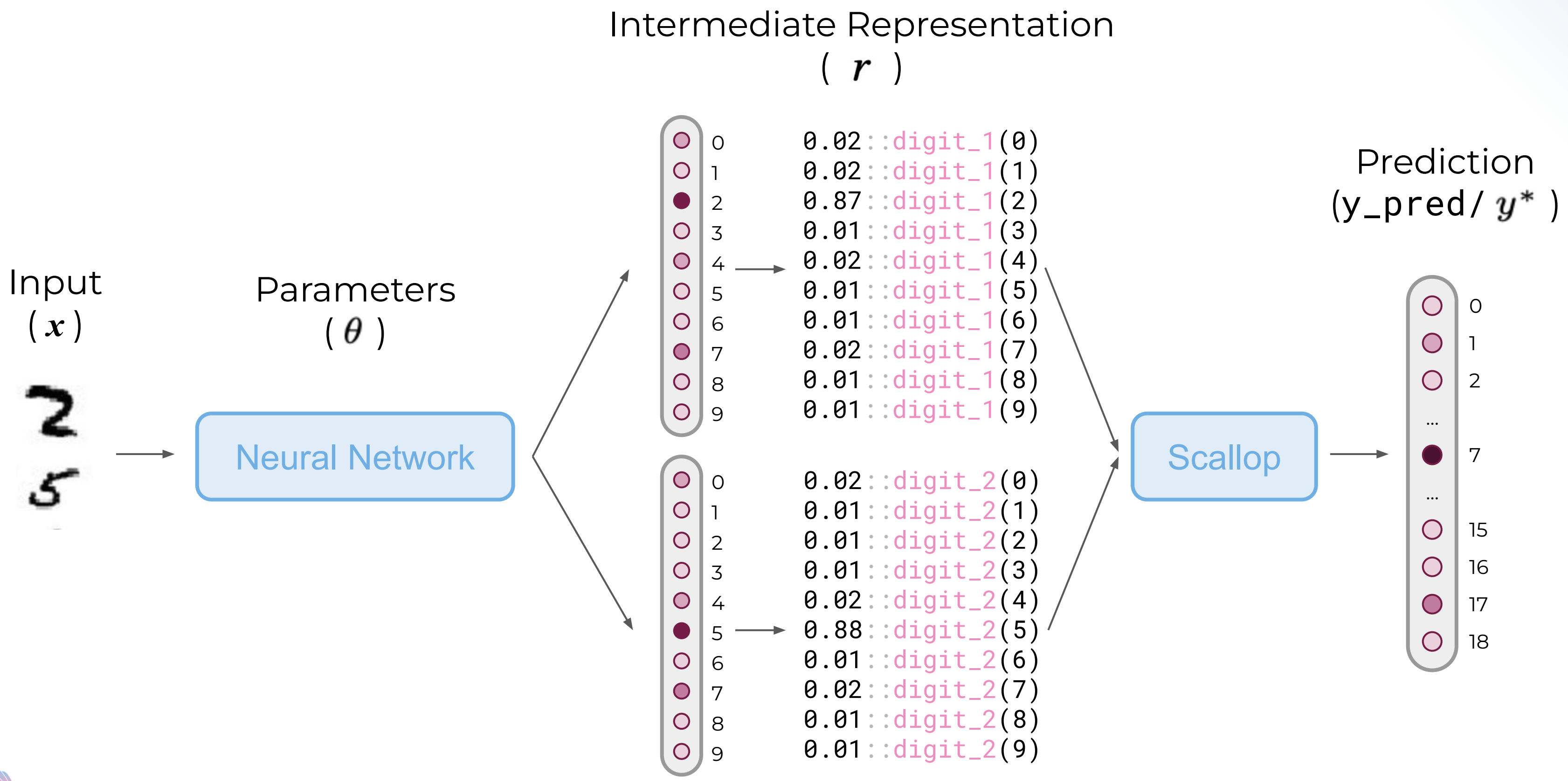


Training Loop for MNIST Addition Task

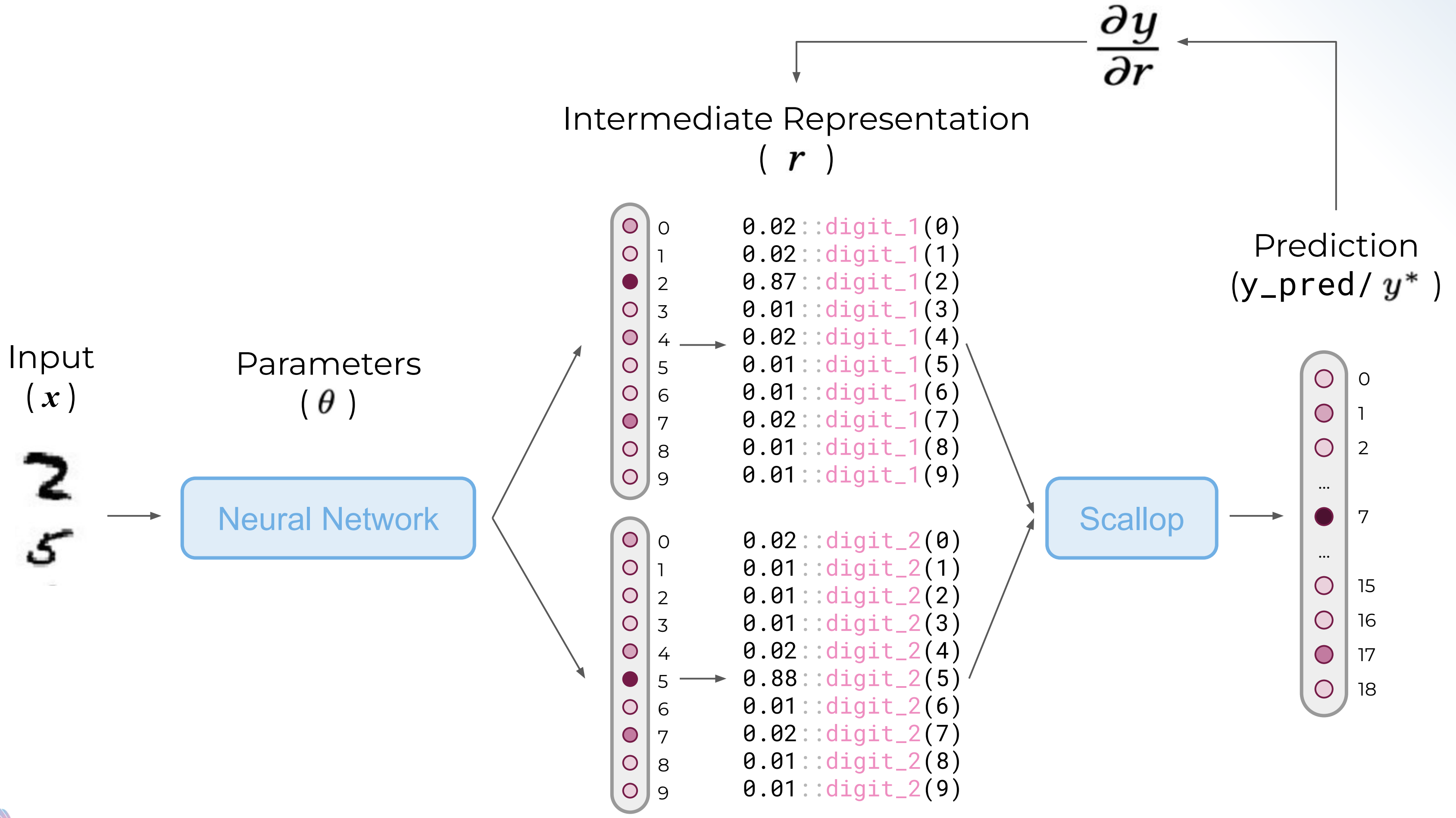


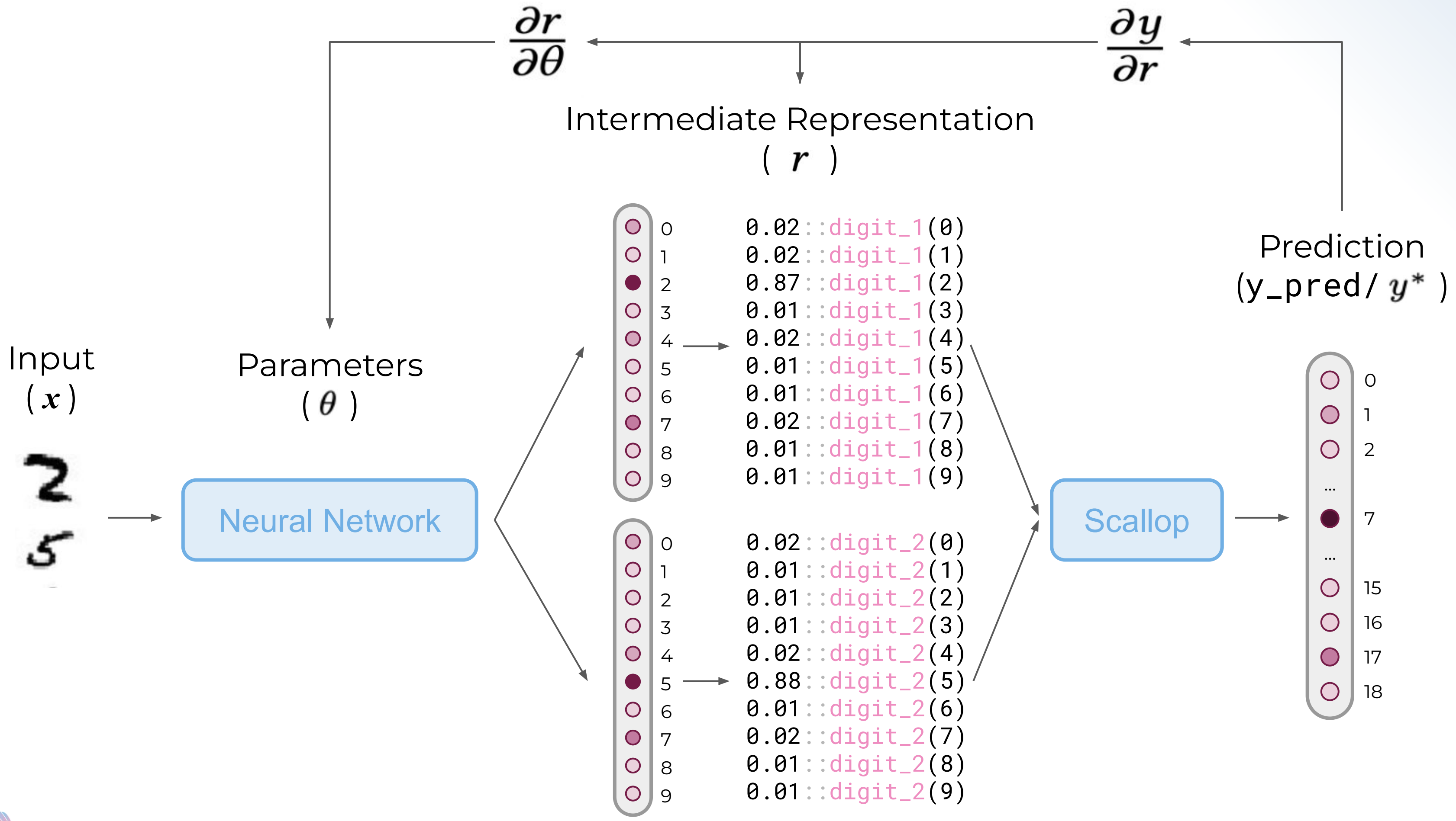
Training Loop for MNIST Addition Task



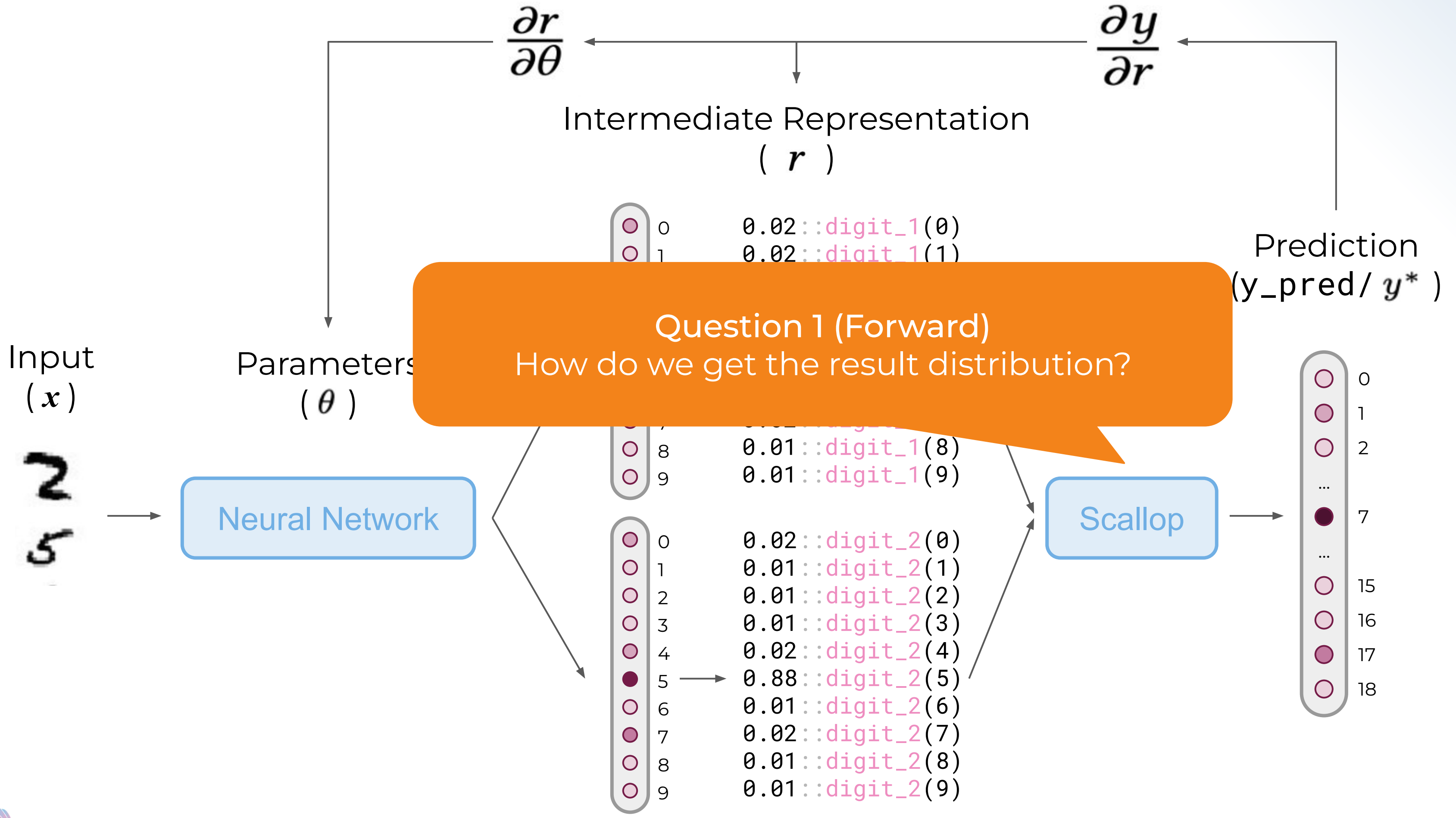


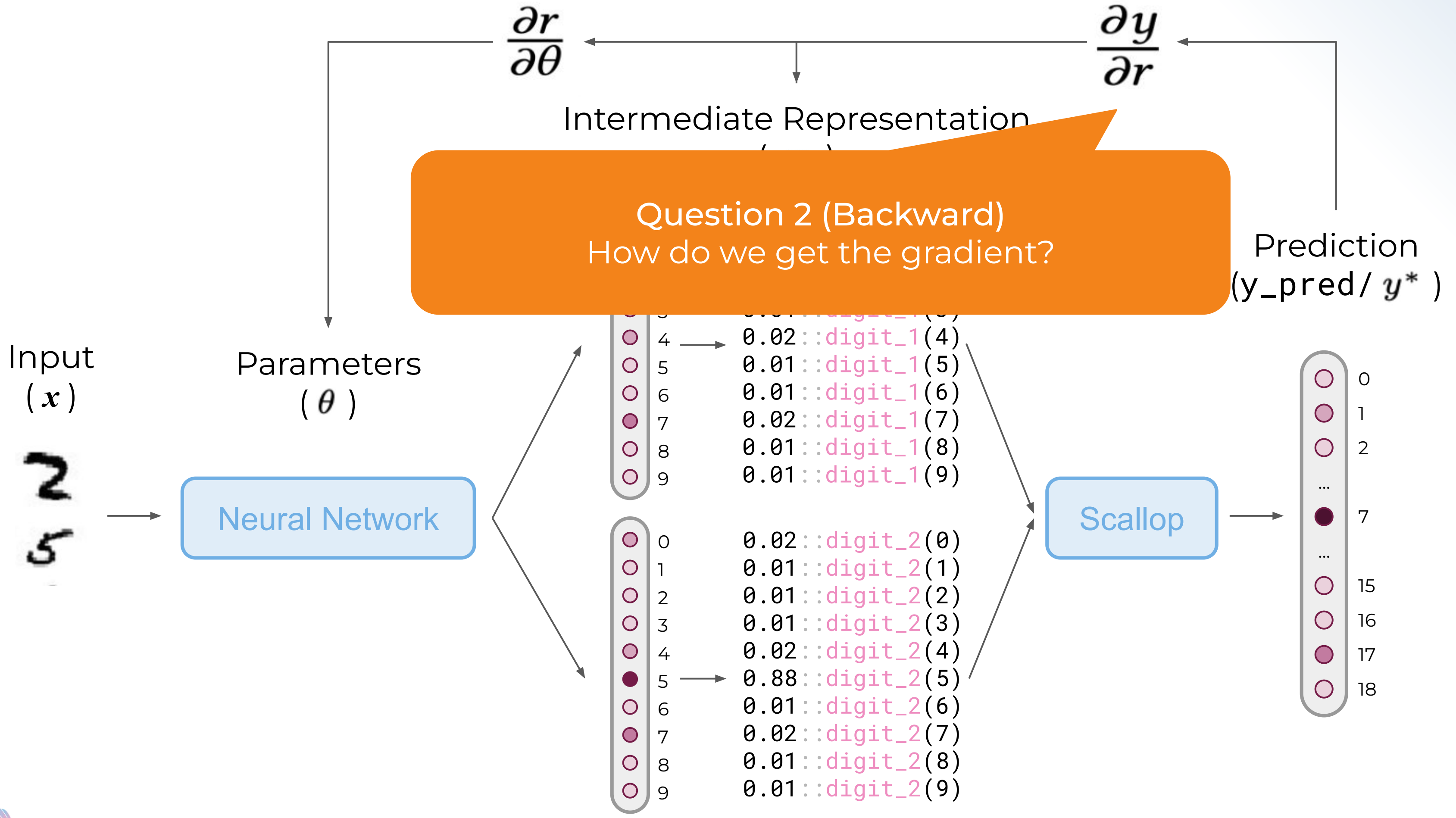
Intermediate Representation
(r)
Input
(x)
Parameters
(θ)
Neural Network
0.02::digit_1(0)
0.02::digit_1(1)
0.87::digit_1(2)
0.01::digit_1(3)
0.02::digit_1(4)
0.01::digit_1(5)
0.01::digit_1(6)
0.02::digit_1(7)
0.01::digit_1(8)
0.01::digit_1(9)
0.02::digit_2(0)
0.01::digit_2(1)
0.01::digit_2(2)
0.01::digit_2(3)
0.02::digit_2(4)
0.88::digit_2(5)
0.01::digit_2(6)
0.02::digit_2(7)
0.01::digit_2(8)
0.01::digit_2(9)
Scallop
Prediction
(y_pred/ y*)


$\frac{\partial y}{\partial r}$
Intermediate Representation
(r)
Input
(x)
Parameters
(θ)
Neural Network
0
1
2
3
4
5
6
7
8
9
0.02::digit_1(0)
0.02::digit_1(1)
0.87::digit_1(2)
0.01::digit_1(3)
0.02::digit_1(4)
0.01::digit_1(5)
0.01::digit_1(6)
0.02::digit_1(7)
0.01::digit_1(8)
0.01::digit_1(9)
0
1
2
3
4
5
6
7
8
9
0.02::digit_2(0)
0.01::digit_2(1)
0.01::digit_2(2)
0.01::digit_2(3)
0.02::digit_2(4)
0.88::digit_2(5)
0.01::digit_2(6)
0.02::digit_2(7)
0.01::digit_2(8)
0.01::digit_2(9)
Scallop
Prediction
(y_pred/ y^*)
0
1
2
...
7
...
15
16
17
18



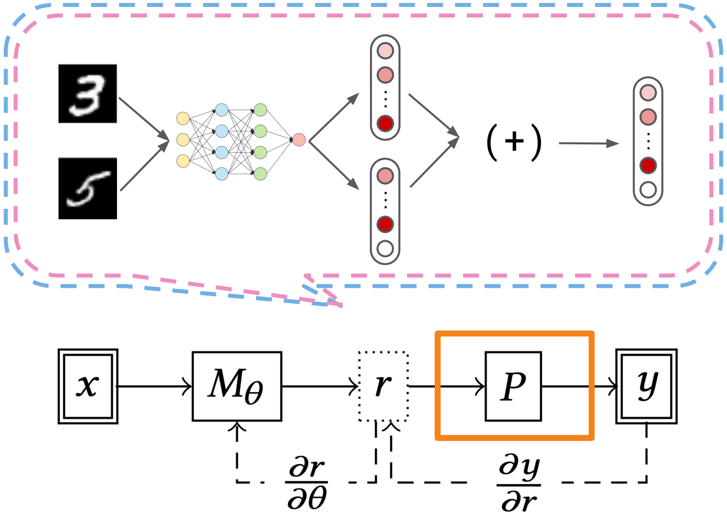
∂r/∂θ
∂y/∂r
Intermediate Representation
(r)
Prediction
(y_pred/ y*)
Input
(x)
Parameters
(θ)
Neural Network
Scallop
0.02::digit_1(0)
0.02::digit_1(1)
0.87::digit_1(2)
0.01::digit_1(3)
0.02::digit_1(4)
0.01::digit_1(5)
0.01::digit_1(6)
0.02::digit_1(7)
0.01::digit_1(8)
0.01::digit_1(9)
0.02::digit_2(0)
0.01::digit_2(1)
0.01::digit_2(2)
0.01::digit_2(3)
0.02::digit_2(4)
0.88::digit_2(5)
0.01::digit_2(6)
0.02::digit_2(7)
0.01::digit_2(8)
0.01::digit_2(9)


∂r/∂θ
∂y/∂r
Intermediate Representation
(r)
Prediction
(y_pred/ y*)
Input
(x)
Parameters
(θ)
Question 1 (Forward)
How do we get the result distribution?
Neural Network
Scallop
0.02::digit_1(0)
0.02::digit_1(1)
0.01::digit_1(8)
0.01::digit_1(9)
0.02::digit_2(0)
0.01::digit_2(1)
0.01::digit_2(2)
0.01::digit_2(3)
0.02::digit_2(4)
0.88::digit_2(5)
0.01::digit_2(6)
0.02::digit_2(7)
0.01::digit_2(8)
0.01::digit_2(9)
0
1
2
...
7
...
15
16
17
18



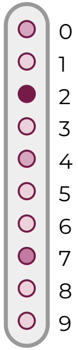
$\frac{\partial r}{\partial \theta}$
$\frac{\partial y}{\partial r}$
Intermediate Representation
Question 2 (Backward)
How do we get the gradient?
Prediction
(y_pred/ y^*)
Input
(x)
Parameters
(θ)
Neural Network
Scallop
4
5
6
7
8
9
0.02::digit_1(4)
0.01::digit_1(5)
0.01::digit_1(6)
0.02::digit_1(7)
0.01::digit_1(8)
0.01::digit_1(9)
0
1
2
3
4
5
6
7
8
9
0.02::digit_2(0)
0.01::digit_2(1)
0.01::digit_2(2)
0.01::digit_2(3)
0.02::digit_2(4)
0.88::digit_2(5)
0.01::digit_2(6)
0.02::digit_2(7)
0.01::digit_2(8)
0.01::digit_2(9)
0
1
2
...
7
...
15
16
17
18

Question 1: Forward probability estimation



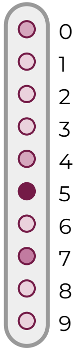
A Neurosymbolic Program

Question 1: Forward probability estimation




```
0.02::digit_1(0)
0.02::digit_1(1)
0.87::digit_1(2)
0.01::digit_1(3)
0.02::digit_1(4)
0.01::digit_1(5)
0.01::digit_1(6)
0.02::digit_1(7)
0.01::digit_1(8)
0.01::digit_1(9)
```

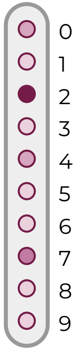
$\mathbf{Pr}(\text{digit1} = 0)$
$\mathbf{Pr}(\text{digit1} = 1)$
$\mathbf{Pr}(\text{digit1} = 2)$
$\mathbf{Pr}(\text{digit1} = 3)$
$\mathbf{Pr}(\text{digit1} = 4)$
$\mathbf{Pr}(\text{digit1} = 5)$
$\mathbf{Pr}(\text{digit1} = 6)$
$\mathbf{Pr}(\text{digit1} = 7)$
$\mathbf{Pr}(\text{digit1} = 8)$
$\mathbf{Pr}(\text{digit1} = 9)$




```
0.02::digit_2(0)
0.01::digit_2(1)
0.01::digit_2(2)
0.01::digit_2(3)
0.02::digit_2(4)
0.88::digit_2(5)
0.01::digit_2(6)
0.02::digit_2(7)
0.01::digit_2(8)
0.01::digit_2(9)
```

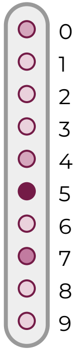
$\mathbf{Pr}(\text{digit2} = 0)$
$\mathbf{Pr}(\text{digit2} = 1)$
$\mathbf{Pr}(\text{digit2} = 2)$
$\mathbf{Pr}(\text{digit2} = 3)$
$\mathbf{Pr}(\text{digit2} = 4)$
$\mathbf{Pr}(\text{digit2} = 5)$
$\mathbf{Pr}(\text{digit2} = 6)$
$\mathbf{Pr}(\text{digit2} = 7)$
$\mathbf{Pr}(\text{digit2} = 8)$
$\mathbf{Pr}(\text{digit2} = 9)$

Question 1: Forward probability estimation




```
0.02::digit_1(0)
0.02::digit_1(1)
0.87::digit_1(2)
0.01::digit_1(3)
0.02::digit_1(4)
0.01::digit_1(5)
0.01::digit_1(6)
0.02::digit_1(7)
0.01::digit_1(8)
0.01::digit_1(9)
```

$\mathbf{Pr}(\text{digit1} = 0)$
$\mathbf{Pr}(\text{digit1} = 1)$
$\mathbf{Pr}(\text{digit1} = 2)$
$\mathbf{Pr}(\text{digit1} = 3)$
$\mathbf{Pr}(\text{digit1} = 4)$
$\mathbf{Pr}(\text{digit1} = 5)$
$\mathbf{Pr}(\text{digit1} = 6)$
$\mathbf{Pr}(\text{digit1} = 7)$
$\mathbf{Pr}(\text{digit1} = 8)$
$\mathbf{Pr}(\text{digit1} = 9)$

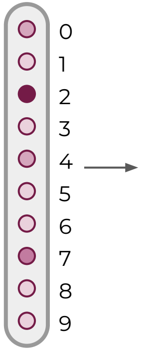



```
0.02::digit_2(0)
0.01::digit_2(1)
0.01::digit_2(2)
0.01::digit_2(3)
0.02::digit_2(4)
0.88::digit_2(5)
0.01::digit_2(6)
0.02::digit_2(7)
0.01::digit_2(8)
0.01::digit_2(9)
```

$\mathbf{Pr}(\text{digit2} = 0)$
$\mathbf{Pr}(\text{digit2} = 1)$
$\mathbf{Pr}(\text{digit2} = 2)$
$\mathbf{Pr}(\text{digit2} = 3)$
$\mathbf{Pr}(\text{digit2} = 4)$
$\mathbf{Pr}(\text{digit2} = 5)$
$\mathbf{Pr}(\text{digit2} = 6)$
$\mathbf{Pr}(\text{digit2} = 7)$
$\mathbf{Pr}(\text{digit2} = 8)$
$\mathbf{Pr}(\text{digit2} = 9)$

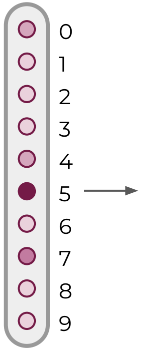
$$\mathbf{Pr}(\text{addition} = 0) = ?$$

Question 1: Forward probability estimation




```
0.02::digit_1(0)
0.02::digit_1(1)
0.87::digit_1(2)
0.01::digit_1(3)
0.02::digit_1(4)
0.01::digit_1(5)
0.01::digit_1(6)
0.02::digit_1(7)
0.01::digit_1(8)
0.01::digit_1(9)
```

$\mathbf{Pr}(\text{digit1} = 0)$
$\mathbf{Pr}(\text{digit1} = 1)$
$\mathbf{Pr}(\text{digit1} = 2)$
$\mathbf{Pr}(\text{digit1} = 3)$
$\mathbf{Pr}(\text{digit1} = 4)$
$\mathbf{Pr}(\text{digit1} = 5)$
$\mathbf{Pr}(\text{digit1} = 6)$
$\mathbf{Pr}(\text{digit1} = 7)$
$\mathbf{Pr}(\text{digit1} = 8)$
$\mathbf{Pr}(\text{digit1} = 9)$

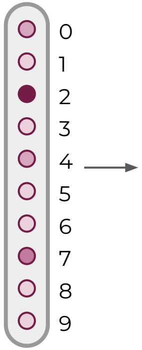



```
0.02::digit_2(0)
0.01::digit_2(1)
0.01::digit_2(2)
0.01::digit_2(3)
0.02::digit_2(4)
0.88::digit_2(5)
0.01::digit_2(6)
0.02::digit_2(7)
0.01::digit_2(8)
0.01::digit_2(9)
```

$\mathbf{Pr}(\text{digit2} = 0)$
$\mathbf{Pr}(\text{digit2} = 1)$
$\mathbf{Pr}(\text{digit2} = 2)$
$\mathbf{Pr}(\text{digit2} = 3)$
$\mathbf{Pr}(\text{digit2} = 4)$
$\mathbf{Pr}(\text{digit2} = 5)$
$\mathbf{Pr}(\text{digit2} = 6)$
$\mathbf{Pr}(\text{digit2} = 7)$
$\mathbf{Pr}(\text{digit2} = 8)$
$\mathbf{Pr}(\text{digit2} = 9)$

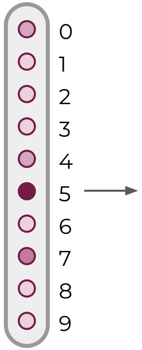
$$\mathbf{Pr}(\text{addition} = 0) = \mathbf{Pr}(\text{digit1} = 0) \times \mathbf{Pr}(\text{digit2} = 0)$$

Question 1: Forward probability estimation




```
0.02::digit_1(0)
0.02::digit_1(1)
0.87::digit_1(2)
0.01::digit_1(3)
0.02::digit_1(4)
0.01::digit_1(5)
0.01::digit_1(6)
0.02::digit_1(7)
0.01::digit_1(8)
0.01::digit_1(9)
```

$\mathbf{Pr}(\text{digit1} = 0)$
$\mathbf{Pr}(\text{digit1} = 1)$
$\mathbf{Pr}(\text{digit1} = 2)$
$\mathbf{Pr}(\text{digit1} = 3)$
$\mathbf{Pr}(\text{digit1} = 4)$
$\mathbf{Pr}(\text{digit1} = 5)$
$\mathbf{Pr}(\text{digit1} = 6)$
$\mathbf{Pr}(\text{digit1} = 7)$
$\mathbf{Pr}(\text{digit1} = 8)$
$\mathbf{Pr}(\text{digit1} = 9)$

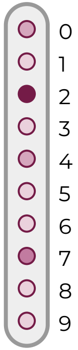



```
0.02::digit_2(0)
0.01::digit_2(1)
0.01::digit_2(2)
0.01::digit_2(3)
0.02::digit_2(4)
0.88::digit_2(5)
0.01::digit_2(6)
0.02::digit_2(7)
0.01::digit_2(8)
0.01::digit_2(9)
```

$\mathbf{Pr}(\text{digit2} = 0)$
$\mathbf{Pr}(\text{digit2} = 1)$
$\mathbf{Pr}(\text{digit2} = 2)$
$\mathbf{Pr}(\text{digit2} = 3)$
$\mathbf{Pr}(\text{digit2} = 4)$
$\mathbf{Pr}(\text{digit2} = 5)$
$\mathbf{Pr}(\text{digit2} = 6)$
$\mathbf{Pr}(\text{digit2} = 7)$
$\mathbf{Pr}(\text{digit2} = 8)$
$\mathbf{Pr}(\text{digit2} = 9)$

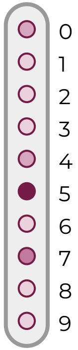
$$\mathbf{Pr}(\text{addition} = 1) = ???$$

Question 1: Forward probability estimation




```
0.02::digit_1(0)
0.02::digit_1(1)
0.87::digit_1(2)
0.01::digit_1(3)
0.02::digit_1(4)
0.01::digit_1(5)
0.01::digit_1(6)
0.02::digit_1(7)
0.01::digit_1(8)
0.01::digit_1(9)
```

$\mathbf{Pr}(\text{digit1} = 0)$
$\mathbf{Pr}(\text{digit1} = 1)$
$\mathbf{Pr}(\text{digit1} = 2)$
$\mathbf{Pr}(\text{digit1} = 3)$
$\mathbf{Pr}(\text{digit1} = 4)$
$\mathbf{Pr}(\text{digit1} = 5)$
$\mathbf{Pr}(\text{digit1} = 6)$
$\mathbf{Pr}(\text{digit1} = 7)$
$\mathbf{Pr}(\text{digit1} = 8)$
$\mathbf{Pr}(\text{digit1} = 9)$

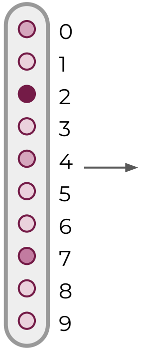



```
0.02::digit_2(0)
0.01::digit_2(1)
0.01::digit_2(2)
0.01::digit_2(3)
0.02::digit_2(4)
0.88::digit_2(5)
0.01::digit_2(6)
0.02::digit_2(7)
0.01::digit_2(8)
0.01::digit_2(9)
```

$\mathbf{Pr}(\text{digit2} = 0)$
$\mathbf{Pr}(\text{digit2} = 1)$
$\mathbf{Pr}(\text{digit2} = 2)$
$\mathbf{Pr}(\text{digit2} = 3)$
$\mathbf{Pr}(\text{digit2} = 4)$
$\mathbf{Pr}(\text{digit2} = 5)$
$\mathbf{Pr}(\text{digit2} = 6)$
$\mathbf{Pr}(\text{digit2} = 7)$
$\mathbf{Pr}(\text{digit2} = 8)$
$\mathbf{Pr}(\text{digit2} = 9)$

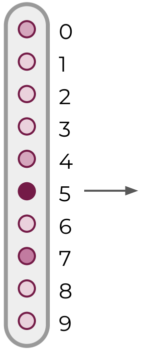
$$\mathbf{Pr}(\text{addition} = 1) \\ = \mathbf{Pr}(\text{digit1} = 0) \times \mathbf{Pr}(\text{digit2} = 1) + \\ \mathbf{Pr}(\text{digit1} = 1) \times \mathbf{Pr}(\text{digit2} = 0)$$

Question 1: Forward probability estimation




```
0.02::digit_1(0)
0.02::digit_1(1)
0.87::digit_1(2)
0.01::digit_1(3)
0.02::digit_1(4)
0.01::digit_1(5)
0.01::digit_1(6)
0.02::digit_1(7)
0.01::digit_1(8)
0.01::digit_1(9)
```

$\mathbf{Pr}(\text{digit1} = 0)$
$\mathbf{Pr}(\text{digit1} = 1)$
$\mathbf{Pr}(\text{digit1} = 2)$
$\mathbf{Pr}(\text{digit1} = 3)$
$\mathbf{Pr}(\text{digit1} = 4)$
$\mathbf{Pr}(\text{digit1} = 5)$
$\mathbf{Pr}(\text{digit1} = 6)$
$\mathbf{Pr}(\text{digit1} = 7)$
$\mathbf{Pr}(\text{digit1} = 8)$
$\mathbf{Pr}(\text{digit1} = 9)$

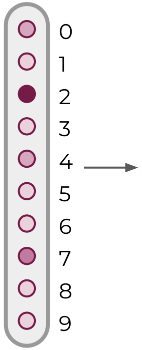



```
0.02::digit_2(0)
0.01::digit_2(1)
0.01::digit_2(2)
0.01::digit_2(3)
0.02::digit_2(4)
0.88::digit_2(5)
0.01::digit_2(6)
0.02::digit_2(7)
0.01::digit_2(8)
0.01::digit_2(9)
```

$\mathbf{Pr}(\text{digit2} = 0)$
$\mathbf{Pr}(\text{digit2} = 1)$
$\mathbf{Pr}(\text{digit2} = 2)$
$\mathbf{Pr}(\text{digit2} = 3)$
$\mathbf{Pr}(\text{digit2} = 4)$
$\mathbf{Pr}(\text{digit2} = 5)$
$\mathbf{Pr}(\text{digit2} = 6)$
$\mathbf{Pr}(\text{digit2} = 7)$
$\mathbf{Pr}(\text{digit2} = 8)$
$\mathbf{Pr}(\text{digit2} = 9)$

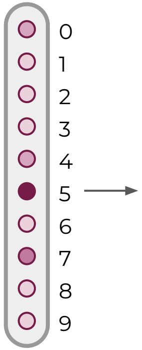
$$\mathbf{Pr}(\text{addition} = 2)$$
$$= \mathbf{???}$$

Question 1: Forward probability estimation




```
0.02::digit_1(0)
0.02::digit_1(1)
0.87::digit_1(2)
0.01::digit_1(3)
0.02::digit_1(4)
0.01::digit_1(5)
0.01::digit_1(6)
0.02::digit_1(7)
0.01::digit_1(8)
0.01::digit_1(9)
```

$\mathbf{Pr}(\text{digit1} = 0)$
$\mathbf{Pr}(\text{digit1} = 1)$
$\mathbf{Pr}(\text{digit1} = 2)$
$\mathbf{Pr}(\text{digit1} = 3)$
$\mathbf{Pr}(\text{digit1} = 4)$
$\mathbf{Pr}(\text{digit1} = 5)$
$\mathbf{Pr}(\text{digit1} = 6)$
$\mathbf{Pr}(\text{digit1} = 7)$
$\mathbf{Pr}(\text{digit1} = 8)$
$\mathbf{Pr}(\text{digit1} = 9)$

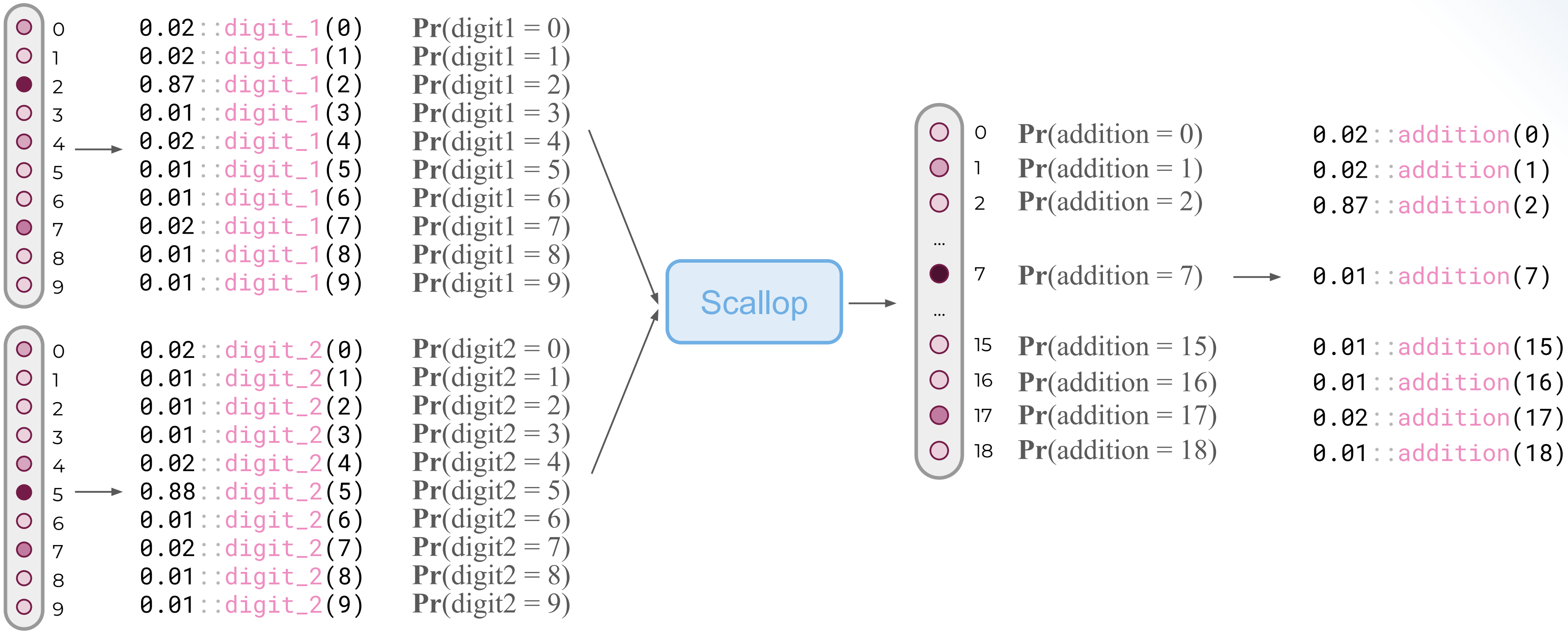



```
0.02::digit_2(0)
0.01::digit_2(1)
0.01::digit_2(2)
0.01::digit_2(3)
0.02::digit_2(4)
0.88::digit_2(5)
0.01::digit_2(6)
0.02::digit_2(7)
0.01::digit_2(8)
0.01::digit_2(9)
```

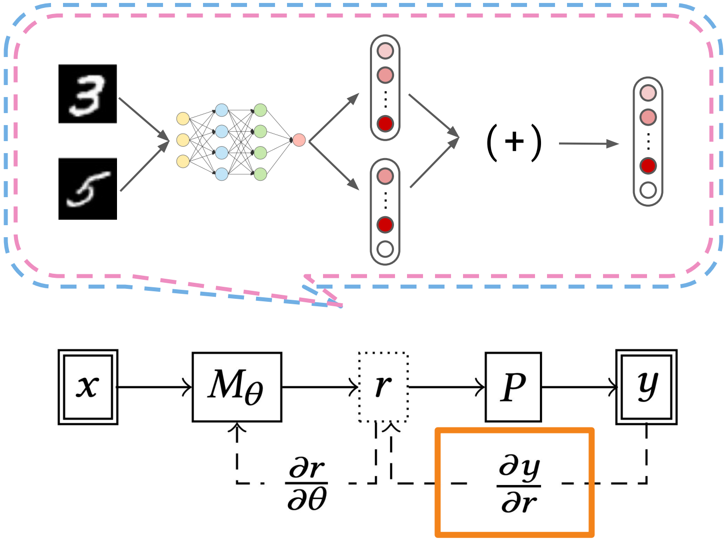
$\mathbf{Pr}(\text{digit2} = 0)$
$\mathbf{Pr}(\text{digit2} = 1)$
$\mathbf{Pr}(\text{digit2} = 2)$
$\mathbf{Pr}(\text{digit2} = 3)$
$\mathbf{Pr}(\text{digit2} = 4)$
$\mathbf{Pr}(\text{digit2} = 5)$
$\mathbf{Pr}(\text{digit2} = 6)$
$\mathbf{Pr}(\text{digit2} = 7)$
$\mathbf{Pr}(\text{digit2} = 8)$
$\mathbf{Pr}(\text{digit2} = 9)$

$$
\begin{aligned}
&\mathbf{Pr}(\text{addition} = 2) \\
&= \mathbf{Pr}(\text{digit1} = 0) \times \mathbf{Pr}(\text{digit2} = 2) + \\
&\mathbf{Pr}(\text{digit1} = 1) \times \mathbf{Pr}(\text{digit2} = 1) + \\
&\mathbf{Pr}(\text{digit1} = 2) \times \mathbf{Pr}(\text{digit2} = 0) +
\end{aligned}
$$

Question 1: Forward probability estimation

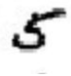


Question 2: Backward gradient estimation

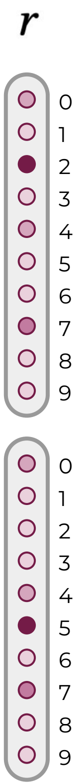


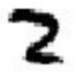
A Neurosymbolic Program

```
0.02::digit_1(0)
0.02::digit_1(1)
0.87::digit_1(2)
0.01::digit_1(3)
0.02::digit_1(4)
0.01::digit_1(5)
0.01::digit_1(6)
0.02::digit_1(7)
0.01::digit_1(8)
0.01::digit_1(9)
```

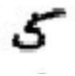


```
0.02::digit_2(0)
0.01::digit_2(1)
0.01::digit_2(2)
0.01::digit_2(3)
0.02::digit_2(4)
0.88::digit_2(5)
0.01::digit_2(6)
0.02::digit_2(7)
0.01::digit_2(8)
0.01::digit_2(9)
```

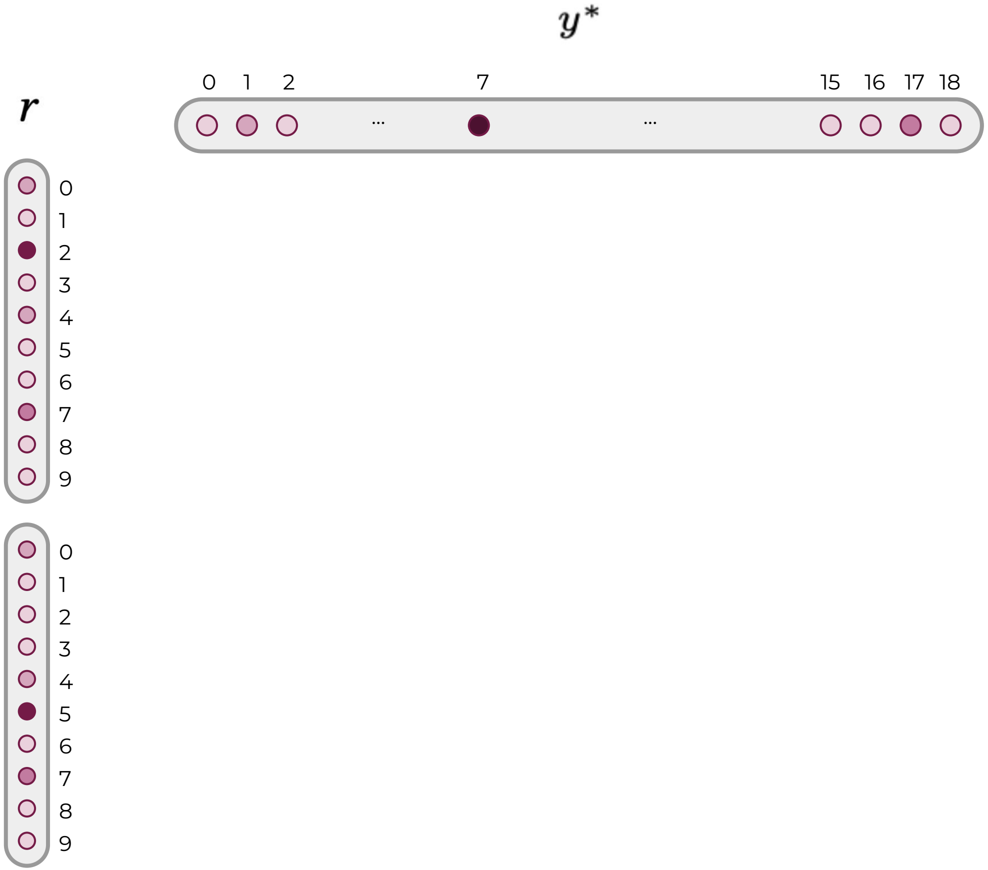


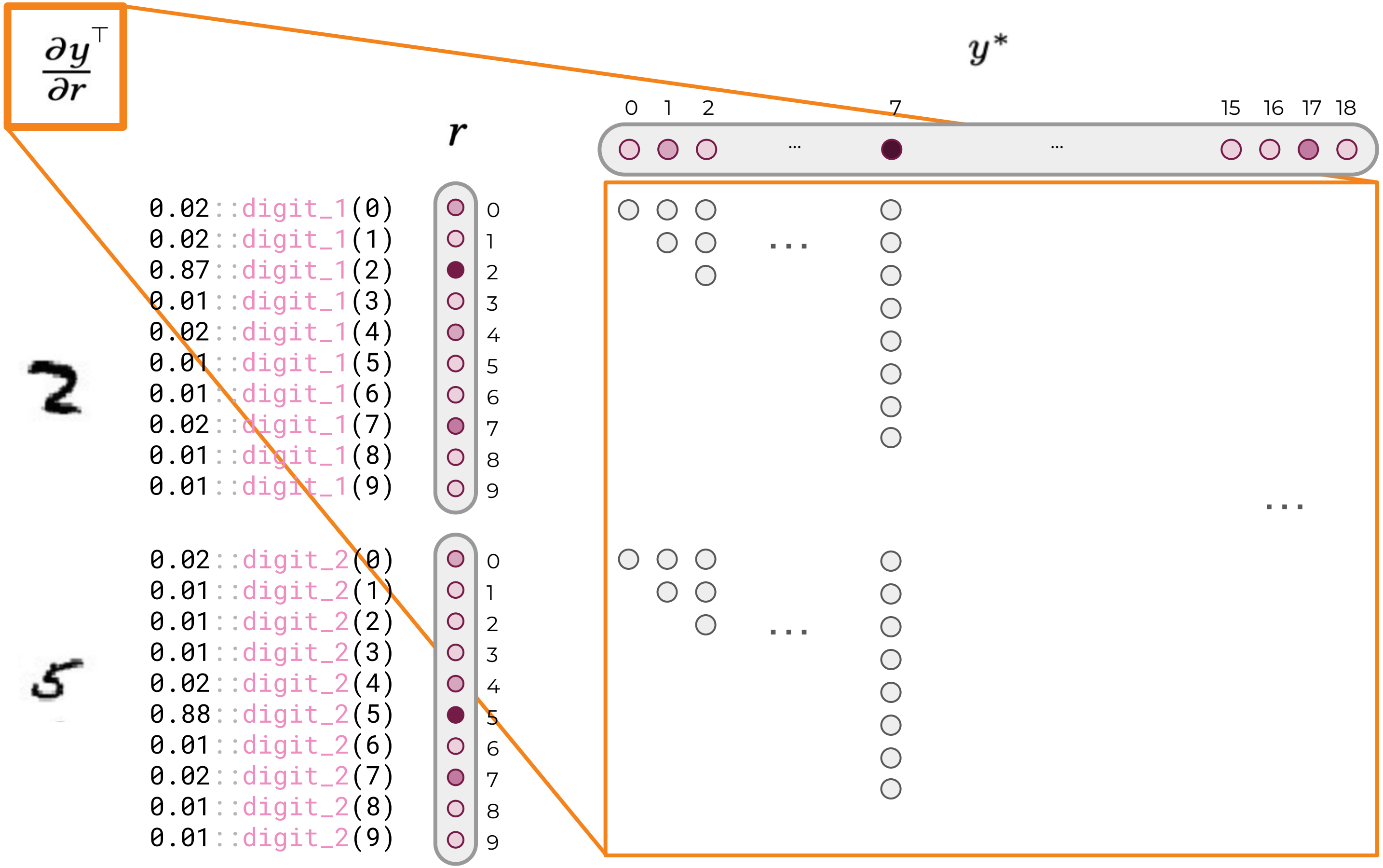


```
0.02::digit_1(0)
0.02::digit_1(1)
0.87::digit_1(2)
0.01::digit_1(3)
0.02::digit_1(4)
0.01::digit_1(5)
0.01::digit_1(6)
0.02::digit_1(7)
0.01::digit_1(8)
0.01::digit_1(9)
```



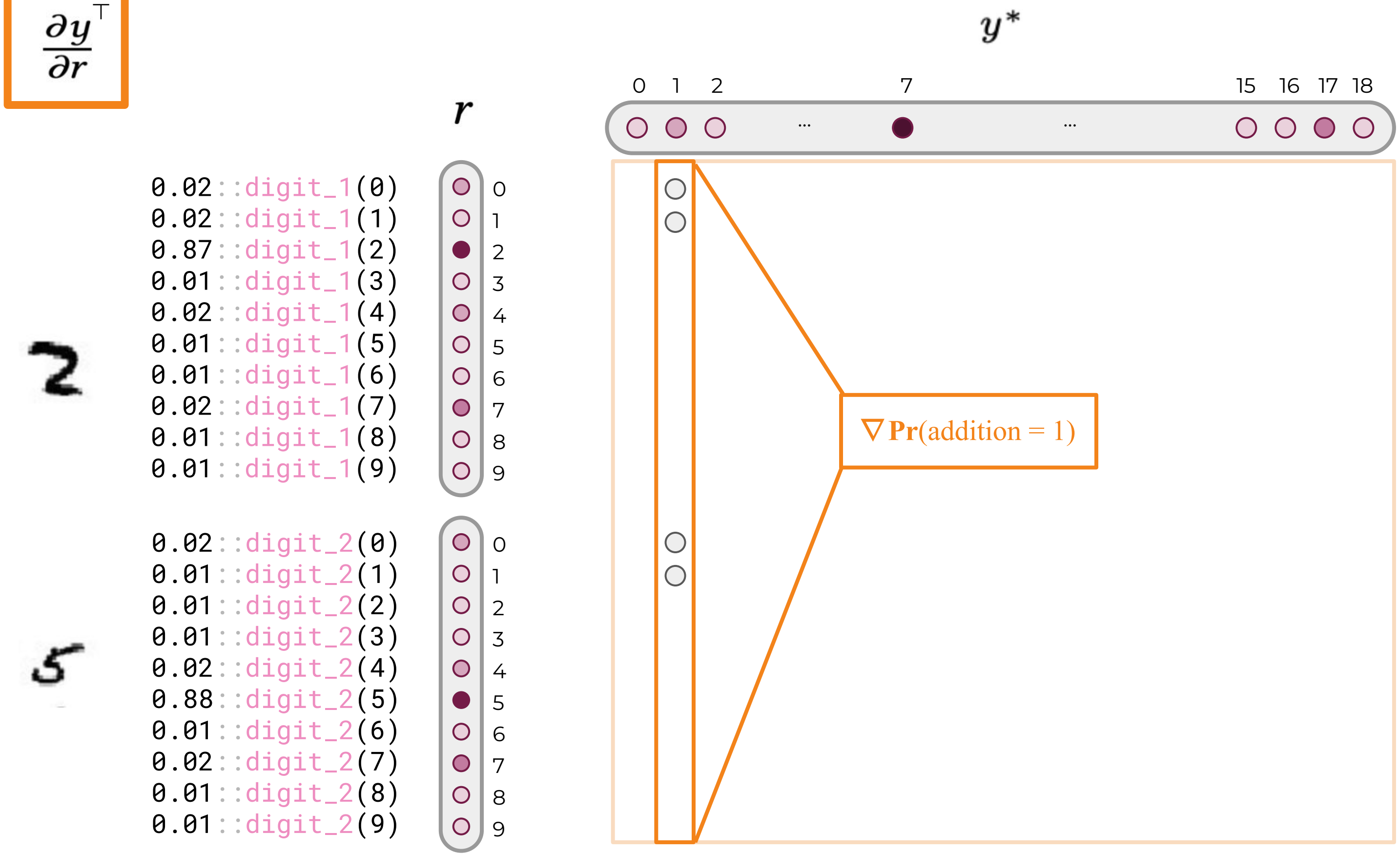
```
0.02::digit_2(0)
0.01::digit_2(1)
0.01::digit_2(2)
0.01::digit_2(3)
0.02::digit_2(4)
0.88::digit_2(5)
0.01::digit_2(6)
0.02::digit_2(7)
0.01::digit_2(8)
0.01::digit_2(9)
```

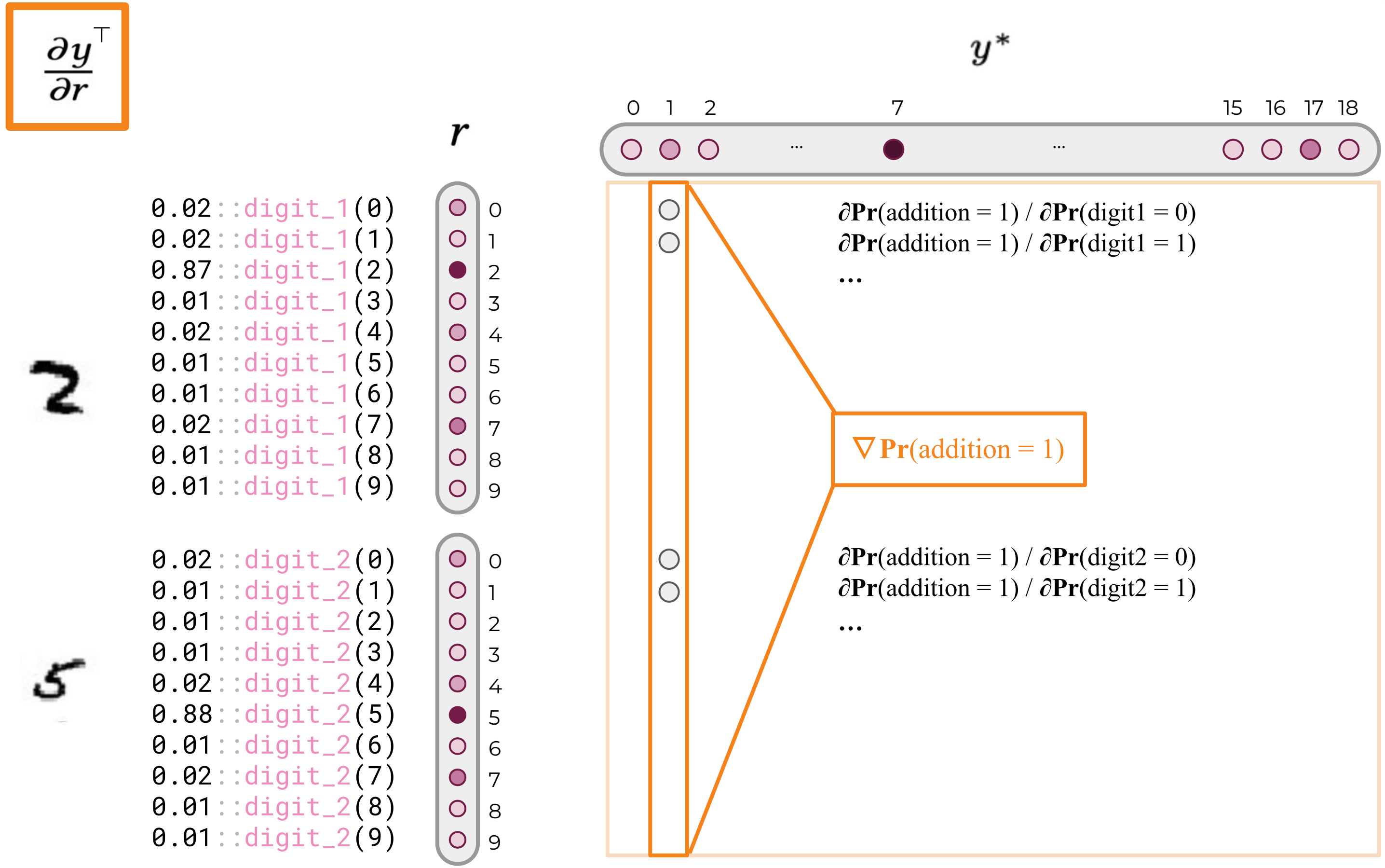


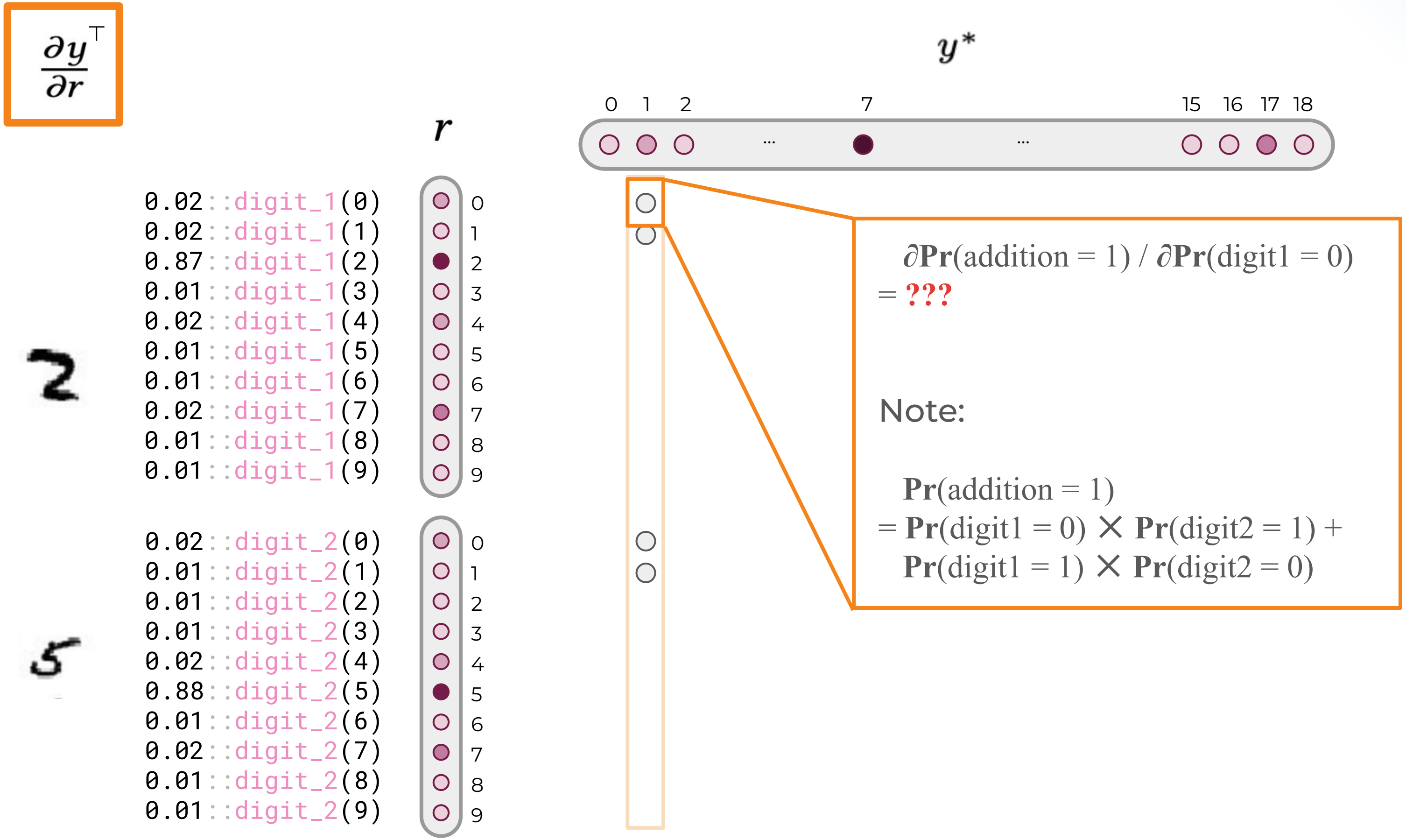


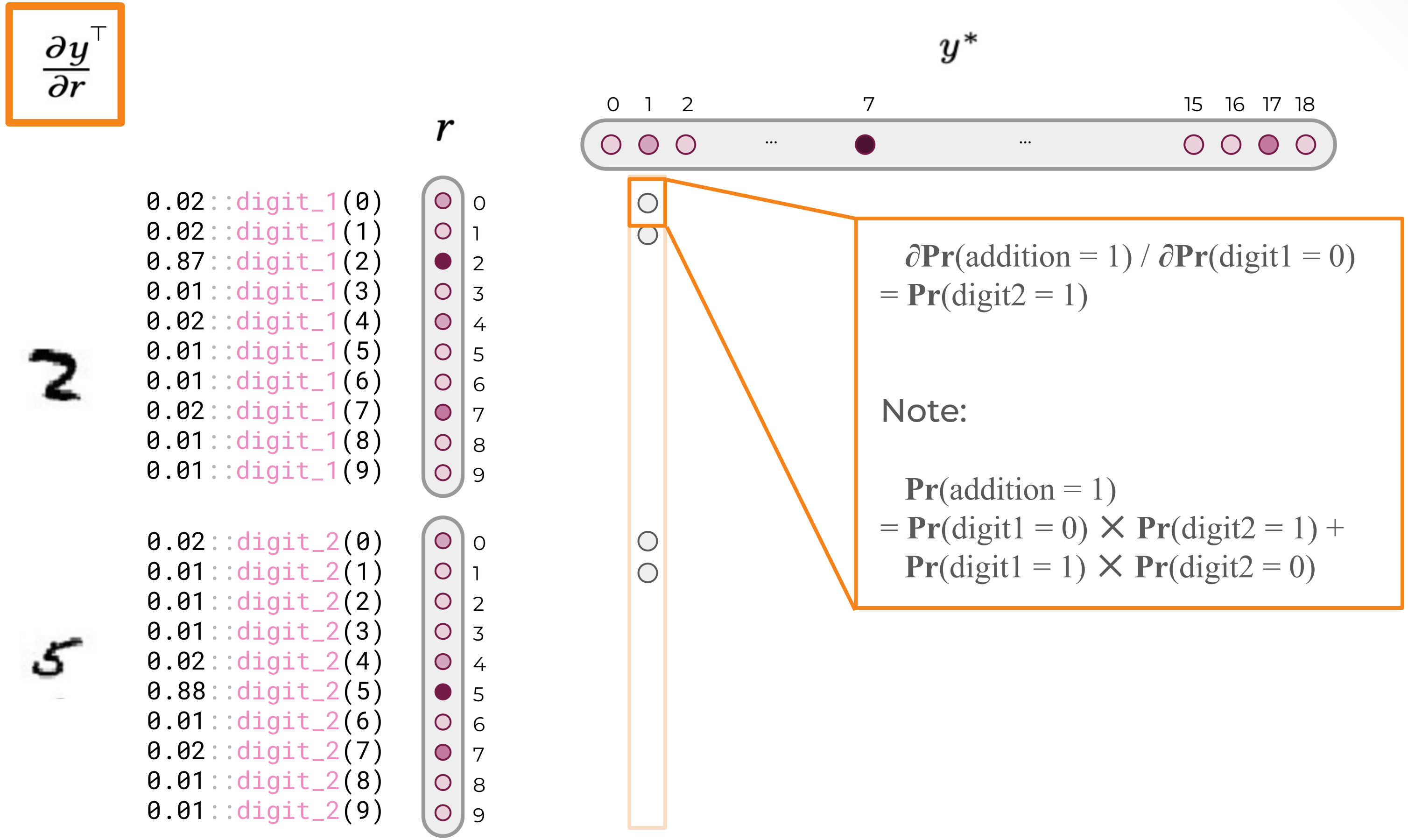
$\frac{\partial y}{\partial r}^\top$
y^*
r
0.02::digit_1(0)
0.02::digit_1(1)
0.87::digit_1(2)
0.01::digit_1(3)
0.02::digit_1(4)
0.01::digit_1(5)
0.01::digit_1(6)
0.02::digit_1(7)
0.01::digit_1(8)
0.01::digit_1(9)
0.02::digit_2(0)
0.01::digit_2(1)
0.01::digit_2(2)
0.01::digit_2(3)
0.02::digit_2(4)
0.88::digit_2(5)
0.01::digit_2(6)
0.02::digit_2(7)
0.01::digit_2(8)
0.01::digit_2(9)

$$\frac{\partial y}{\partial r}^{\top}$$



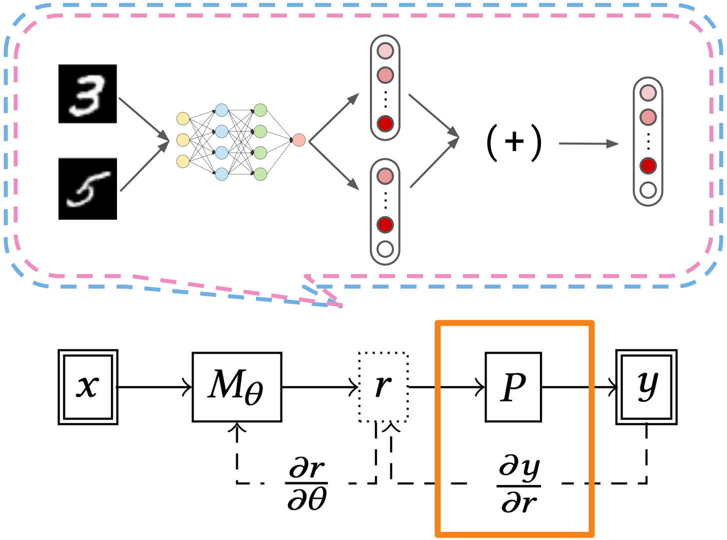

$\frac{\partial y}{\partial r}^{\top}$
r
y^*
0 1 2 ... 7 ... 15 16 17 18
0.02::digit_1(0)
0.02::digit_1(1)
0.87::digit_1(2)
0.01::digit_1(3)
0.02::digit_1(4)
0.01::digit_1(5)
0.01::digit_1(6)
0.02::digit_1(7)
0.01::digit_1(8)
0.01::digit_1(9)
0 1 2 3 4 5 6 7 8 9
0.02::digit_2(0)
0.01::digit_2(1)
0.01::digit_2(2)
0.01::digit_2(3)
0.02::digit_2(4)
0.88::digit_2(5)
0.01::digit_2(6)
0.02::digit_2(7)
0.01::digit_2(8)
0.01::digit_2(9)
0 1 2 3 4 5 6 7 8 9
∂Pr(addition = 1) / ∂Pr(digit1 = 0)
∂Pr(addition = 1) / ∂Pr(digit1 = 1)
...
∇ Pr(addition = 1)
∂Pr(addition = 1) / ∂Pr(digit2 = 0)
∂Pr(addition = 1) / ∂Pr(digit2 = 1)
...


$\frac{\partial y}{\partial r}^{\top}$
r
y^*
0 1 2 ... 7 ... 15 16 17 18
0.02::digit_1(0)
0.02::digit_1(1)
0.87::digit_1(2)
0.01::digit_1(3)
0.02::digit_1(4)
0.01::digit_1(5)
0.01::digit_1(6)
0.02::digit_1(7)
0.01::digit_1(8)
0.01::digit_1(9)
0.02::digit_2(0)
0.01::digit_2(1)
0.01::digit_2(2)
0.01::digit_2(3)
0.02::digit_2(4)
0.88::digit_2(5)
0.01::digit_2(6)
0.02::digit_2(7)
0.01::digit_2(8)
0.01::digit_2(9)
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
$\partial \mathbf{Pr}(\text{addition} = 1) \,/\, \partial \mathbf{Pr}(\text{digit1} = 0)$
$= ???$
Note:
$\mathbf{Pr}(\text{addition} = 1)$
$= \mathbf{Pr}(\text{digit1} = 0) \times \mathbf{Pr}(\text{digit2} = 1) +$
$\mathbf{Pr}(\text{digit1} = 1) \times \mathbf{Pr}(\text{digit2} = 0)$



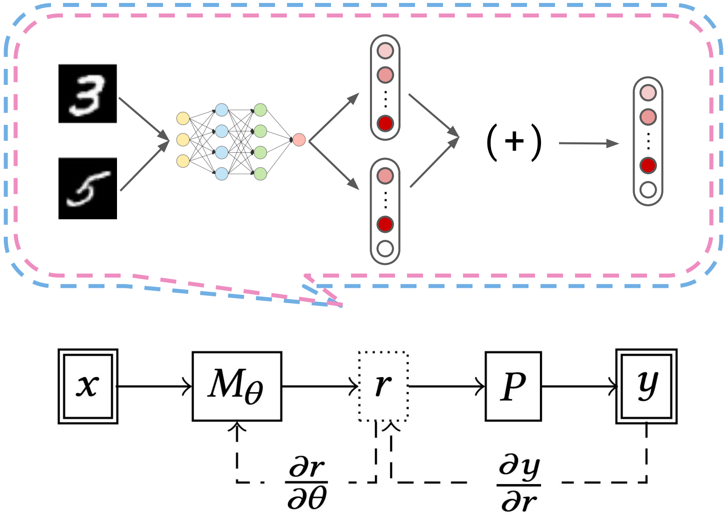
$\frac{\partial y}{\partial r}^{\top}$
r
y^*
0 1 2 ... 7 ... 15 16 17 18
0.02::digit_1(0)
0.02::digit_1(1)
0.87::digit_1(2)
0.01::digit_1(3)
0.02::digit_1(4)
0.01::digit_1(5)
0.01::digit_1(6)
0.02::digit_1(7)
0.01::digit_1(8)
0.01::digit_1(9)
0.02::digit_2(0)
0.01::digit_2(1)
0.01::digit_2(2)
0.01::digit_2(3)
0.02::digit_2(4)
0.88::digit_2(5)
0.01::digit_2(6)
0.02::digit_2(7)
0.01::digit_2(8)
0.01::digit_2(9)
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
$\partial \mathbf{Pr}(\text{addition} = 1) \,/\, \partial \mathbf{Pr}(\text{digit1} = 0)$
$= \mathbf{Pr}(\text{digit2} = 1)$
Note:
$\mathbf{Pr}(\text{addition} = 1)$
$= \mathbf{Pr}(\text{digit1} = 0) \times \mathbf{Pr}(\text{digit2} = 1) +$
$\mathbf{Pr}(\text{digit1} = 1) \times \mathbf{Pr}(\text{digit2} = 0)$

The Questions Are Solved!



A Neurosymbolic Program

The Questions Are Solved! Or are we...?



A Neurosymbolic Program

The Questions Are Solved! Or are we...?

1. **Scalability Issue**

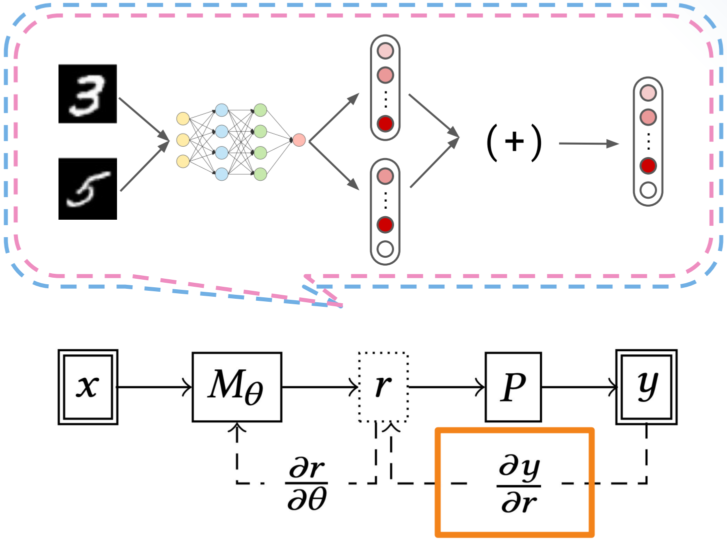
   Doing exact probabilistic estimation is too time consuming! We need to speed it up!

   Solution: approximation, heuristics, etc.

2. **Expressiveness Issue**

   This framework is too primitive, need to support more operations than simply summations.

   Solution: a programming language that supports negation, recursion, aggregations, etc.



A Neurosymbolic Program

Scallop, A Neurosymbolic Programming Language

Relational Programming and Scallop

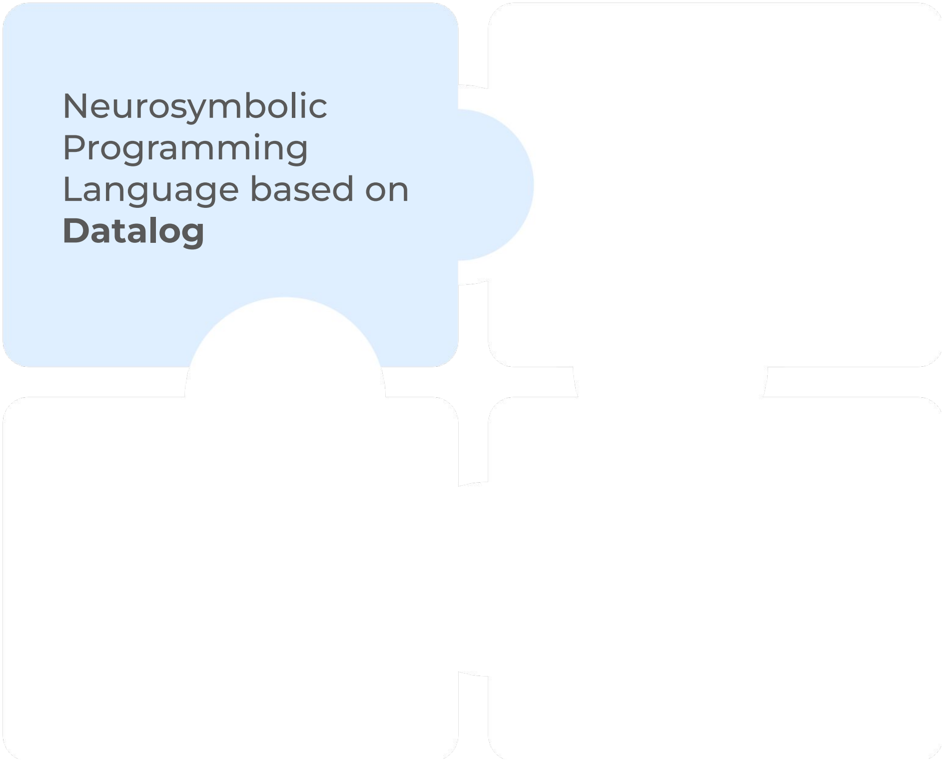
Relational Programming

Datalog, Prolog, **Scallop**

```
rel grandmother(a, b) =
  father(a, x) and mother(x, b)
```

Scallop, a Neurosymbolic Programming Language

Scallop: A Neurosymbolic Programming Language



Scallop: A Neurosymbolic Programming Language

Neurosymbolic Programming Language based on **Datalog**

Expressive Logic:

- Recursion
- Negation
- Aggregation

Scallop: A Neurosymbolic Programming Language

Neurosymbolic Programming Language based on **Datalog**

Expressive Logic:

- Recursion
- Negation
- Aggregation

Rich Reasoning Framework:

- Discrete
- Probabilistic
- Differentiable

Scallop: A Neurosymbolic Programming Language

Neurosymbolic Programming Language based on **Datalog**

Expressive Logic:
- Recursion
- Negation
- Aggregation

Foreign Interface:
- Functions
- Predicates
- Aggregators
- Attributes

Rich Reasoning Framework:
- Discrete
- Probabilistic
- Differentiable

Scallop: A Neurosymbolic Programming Language

Neurosymbolic Programming Language based on **Datalog**

Expressive Logic:
- Recursion
- Negation
- Aggregation

Goal: Provide a unified language for AI developers

Foreign Interface:
- Predicates
- Attributes
- Aggregators

Rich Reasoning Framework:
- Discrete
- Probabilistic
- Differentiable

Scallop has been applied to many ML tasks...

CLUTRR: Kinship Reasoning

CLUTRR: Kinship Reasoning

Context:

[Cristina] was afraid of heights just like her daughters, [**Sheila**] and [Diana]. However, [Diana]'s father, [Jonathan], loved heights and even went skydiving a few times. [**Ruth**] and her son, [Jeremy], went to the park, and had a wonderful time. [Jeremy] went to the bakery with his uncle [Jonathan] to pick up some bread for lunch.

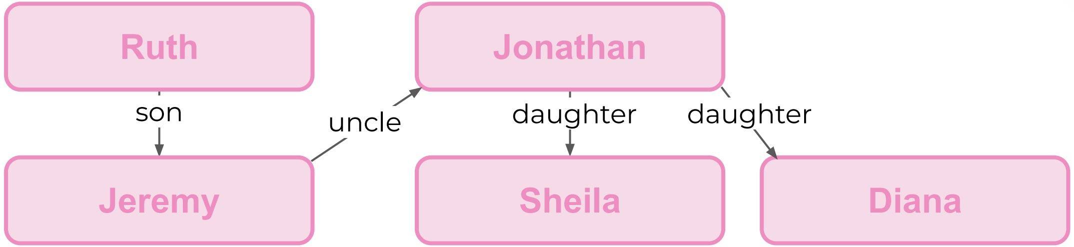
Question:

What is the relationship between **Ruth** and **Sheila**?

CLUTRR: Kinship Reasoning

Context:

[Cristina] was afraid of heights just like her daughters, [**Sheila**] and [Diana]. However, [Diana]'s father, [Jonathan], loved heights and even went skydiving a few times. [**Ruth**] and her son, [Jeremy], went to the park, and had a wonderful time. [Jeremy] went to the bakery with his uncle [Jonathan] to pick up some bread for lunch.



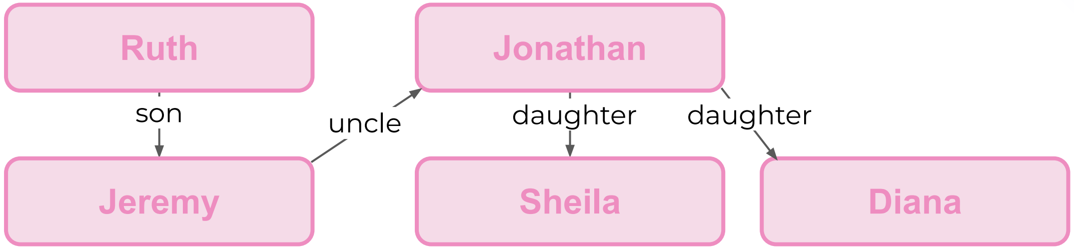
Question:

What is the relationship between **Ruth** and **Sheila**?

CLUTRR: Kinship Reasoning

Context:

[Cristina] was afraid of heights just like her daughters, [**Sheila**] and [Diana]. However, [Diana]'s father, [Jonathan], loved heights and even went skydiving a few times. [**Ruth**] and her son, [Jeremy], went to the park, and had a wonderful time. [Jeremy] went to the bakery with his uncle [Jonathan] to pick up some bread for lunch.



Question:

What is the relationship between **Ruth** and **Sheila**?

Answer:

Sheila is **Ruth**'s niece.

CLUTRR: Kinship Reasoning

Context:

[Cristina] was afraid of heights just like her daughters, [**Sheila**] and [Diana].

However, [Diana]'s father, [Jonathan], loved heights and even went skydiving a few times. [**Ruth**] and her son, [Jeremy], went to the park, and had a wonderful time. [Jeremy] went to the bakery with his uncle [Jonathan] to pick up some bread for lunch.

```
0.951::context(DAUGHTER, "Cristina", "Sheila")
0.002::context(MOTHER, "Cristina", "Sheila")
0.004::context(FATHER, "Cristina", "Sheila")
…
0.001::context(NA, "Cristina", "Sheila")
0.942::context(DAUGHTER, "Christina", "Diana")
0.015::context(MOTHER, "Christina", "Diana")
…
0.002::context(NA, "Sheila", "Diana")
```

Symbolic Context

For each pair of names, classify them into 21 types of kinship

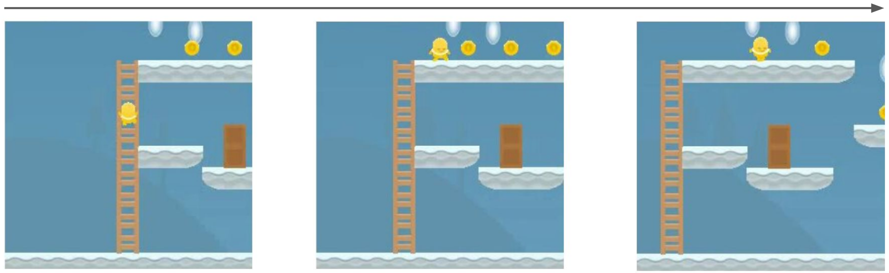
Mugen: Text/Video Retrieval

Mugen using Scallop...

Mugen is a dataset containing multi-modality data (video, text, audio, etc)

In this task, we consider the video-text-alignment. The model takes in video and text, and returns whether they are aligned or not.

Video timeline



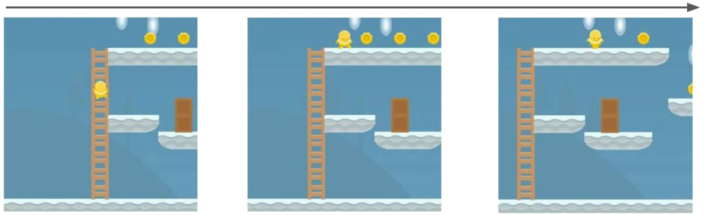
Text description

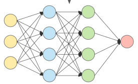
Mugen climbs up on a ladder, and walks to the right and collects a few coins

Aligned? Yes

Mugen using Scallop…

Video timeline

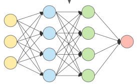




```
0.99::action(FRAME0, “climb”); …
0.85::action(FRAME5, “walk”); …
0.85::mod(FRAME5, “collect-coin”); …
0.01::action(FRAME10, “jump”); …
```

Text description

Mugen climbs up on a ladder, and walks to the right and collects a few coins



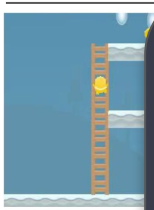
```
expr(0, “climb”)
expr(1, “walk”)
expr(2, “collect-coin”)
```

Scallop Program

```
0.99::match()
```

Mugen using Scallop...

Video timeline



```
rel match_single(tid, vid, vid + 1) = expr(tid, a), action(vid, a)
rel match_sub(tid, tid, vid_start, vid_end) =
   match_single(tid, vid_start, vid_end)
rel match_sub(tid, tid, vid_start, vid_end) =
   match_sub(tid, tid, vid_start, vid_mid), match_single(tid, vid_mid, vid_end)
rel match_sub(tid_start, tid_end, vid_start, vid_end) =
   match_sub(tid_start, tid_end - 1, vid_start, vid_mid),
   match_single(tid_end, vid_mid, vid_end)
rel match() = expr_start(tid_start), expr_end(tid_end), action_start(vid_start),
   action_end(vid_end), match_sub(tid_start, tid_end, vid_start, vid_end)
```

walks
s

```
0.99::action(FRAME0, “climb”); …
0.85::action(FRAME5, “walk”); …
0.85::mod(FRAME5, “collect-coin”); …
0.01::action(FRAME10, “jump”); …
```

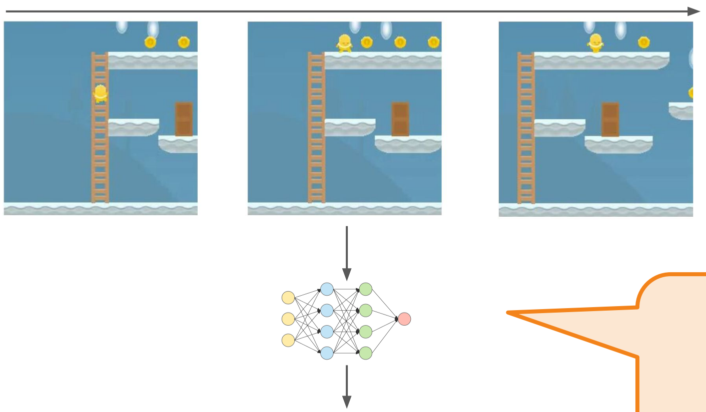
Scallop Program

```
expr(0, “climb”)
expr(1, “walk”)
expr(2, “collect-coin”)
```

```
0.99::match()
```

Mugen using Scallop…

Video timeline



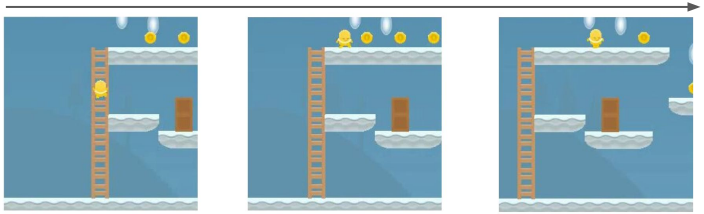
```
0.99::action(FRAME0, “climb”); …
0.85::action(FRAME5, “walk”); …
0.85::mod(FRAME5, “collect-coin”); …
0.01::action(FRAME10, “jump”); …
```

In this process, Scallop helps to extract detailed actions from the video, providing better interpretability and explainability

Mugen using Scallop...

Training video- and text-retrieval models under constrastive learning

Video timeline



aligned

not-aligned

Potential Text Descriptions...

Choice A: Mugen climbs up on a ladder, and walks to the right and collects a few coins

Choice B: Mugen collects 5 coins before jumping to kill an enemy

Choice C: Mugen jumps twice to the right and uses the key to open the door

20bn: Video Reasoning via Linear-Temporal Specifications

Grounding predicates through actions with Scallop…

Given a real life video of people doing a pre-specified task, ground the details of the shown objects indicated in the video

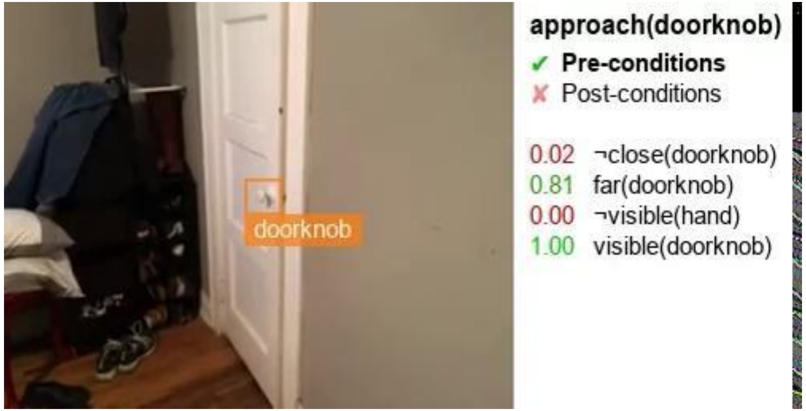
Example:

- Task: `attach(“sticky note”, “cabinet”)`
- Predicates:
    - `far(“sticky note”)`
    - `touching(“sticky note”, “hand”)`
    - `visible(“sticky note”)`
    - …


```
rel precond("attach", a, b) =
 touching(a, "hand") and visible("hand") and visible(a)
rel postcond("attach", a, b) =
 not touching(a, "hand") and touching(a, b) and far(a)
```

Grounding predicates through actions (Migimatsu et. al. 2022)

Grounding predicates through actions with Scallop…



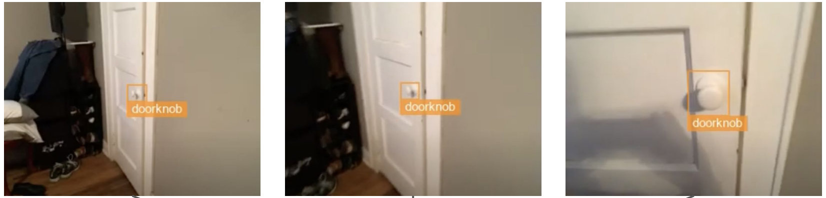

```
rel precond("approach", x) =
  far(x) and visible(x)
rel postcond("approach", x) =
  close(x) and not far(x) and visible(x)
```

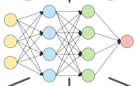
```
rel precond("attach", a, b) =
 touching(a, "hand") and visible("hand") and visible(a)
rel postcond("attach", a, b) =
 not touching(a, "hand") and touching(a, b) and far(a)
```


[Grounding predicates through actions (Migimatsu et. al. 2022)](#)

Grounding predicates through actions with Scallop...

Video timeline





Neural Model

```
0.85::far("doorknob")
0.99::visible("doorknob")
```

...

```
0.01::far("doorknob")
0.95::visible("doorknob")
```

Event

`approach("doorknob")`

Scallop Program

```
rel precond("approach", x) =
  far(x) and visible(x)
rel postcond("approach", x) =
  close(x) and not far(x) and visible(x)
```


[Grounding predicates through actions (Migimatsu et. al. 2022)](#)

Scallop 🤝 Foundation Models

Foundation Models

Claude 2
ANTHROP\C

OpenAI
CLIP

Stable Diffusion

ViLT

T5

LLaMA-2

ResNet

DSFD

OpenAI
ChatGPT 4.0

VICUNA

OWL-ViT

Midjourney

GitHub
Copilot

Segment Anything
Research by Meta AI

S3D

Foundation Models: Recap

Claude 2

OpenAI CLIP

Stable Diffusion

ViLT

What is the right **abstraction layer** to program with **foundation models**?

DSFD

ResNet

Midjourney

GitHub Copilot

Segment Anything
Research by Meta AI

S3D

Relational Knowledge Extraction with GPT

Context:

[Cristina] was afraid of heights just like her daughters, [**Sheila**] and [Diana]. However, [Diana]'s father, [Jonathan], loved heights and even went skydiving a few times. [**Ruth**] and her son, [Jeremy], went to the park, and had a wonderful time. [Jeremy] went to the bakery with his uncle [Jonathan] to pick up some bread for lunch.

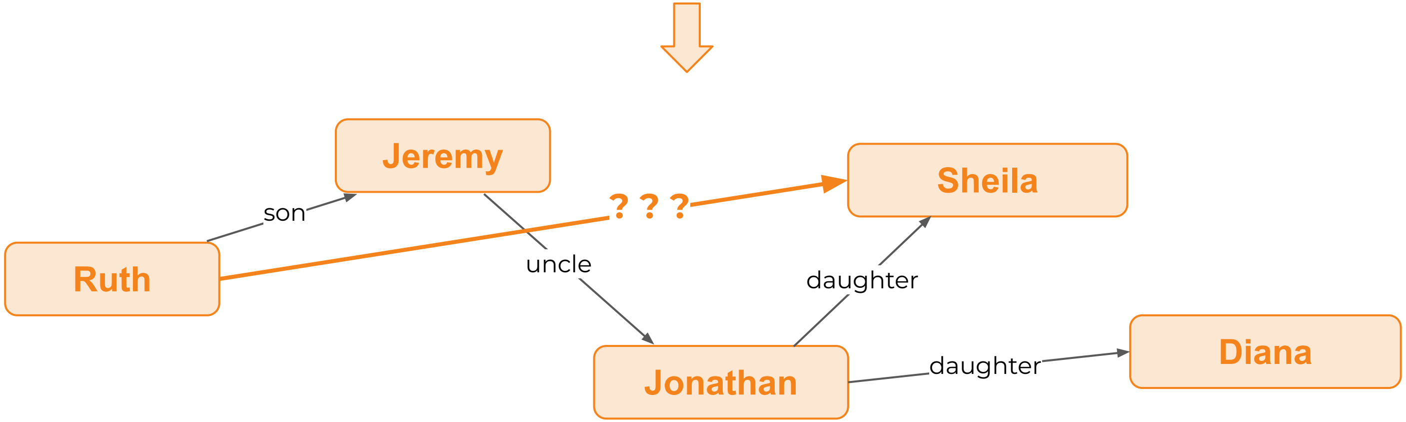
Question:

What is the relationship between **Ruth** and **Sheila**?

Relational Knowledge Extraction with GPT

Context:

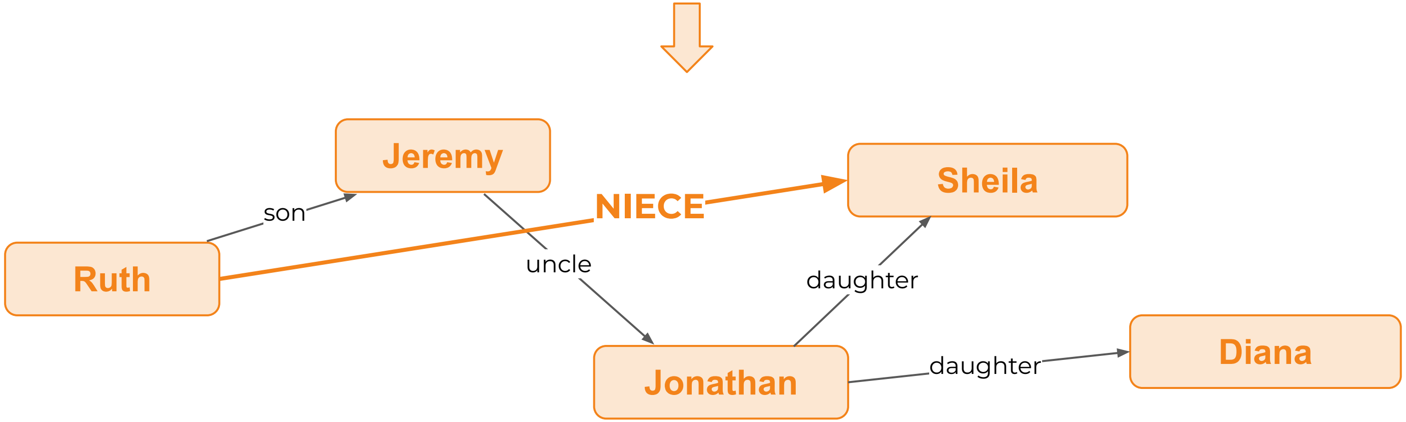
[Cristina] was afraid of heights just like her daughters, [**Sheila**] and [Diana]. However, [Diana]'s father, [Jonathan], loved heights and even went skydiving a few times. [**Ruth**] and her son, [Jeremy], went to the park, and had a wonderful time. [Jeremy] went to the bakery with his uncle [Jonathan] to pick up some bread for lunch.



Relational Knowledge Extraction with GPT

Context:

[Cristina] was afraid of heights just like her daughters, [**Sheila**] and [Diana]. However, [Diana]'s father, [Jonathan], loved heights and even went skydiving a few times. [**Ruth**] and her son, [Jeremy], went to the park, and had a wonderful time. [Jeremy] went to the bakery with his uncle [Jonathan] to pick up some bread for lunch.



Relational Knowledge Extraction with GPT

OpenAI
ChatGPT 4.0

```
@gpt_extract_relation(
  prompt="Please extract the kinship relationships from the context:",
  examples=[(“Alice is Bob’s mother”, [(“alice”, “bob”, “son”), ...]), ...])
type parse_relations(bound context: String, sub: String, obj: String, rela: String), …
```

Relational Knowledge Extraction with GPT

Context: [Cristina] was afraid of heights just like her daughters, [**Sheila**] and [Diana]. However, [Diana]'s father, [Jonathan], loved heights and even went skydiving a few times. [**Ruth**] and her son, [Jeremy], went to the park, and had a wonderful time. [Jeremy] went to the bakery with his uncle [Jonathan] to pick up some bread for lunch. What is the relationship between **Sheila** and **Ruth**?

⇩

```
@gpt_extract_relation(
  prompt="Please extract the kinship relationships from the context:",
  examples=[(“Alice is Bob’s mother”, [(“alice”, “bob”, “son”), ...]), ...])
type parse_relations(bound context: String, sub: String, obj: String, rela: String), …
```

Relational Knowledge Extraction with GPT

Context: [Cristina] was afraid of heights just like her daughters, [**Sheila**] and [Diana]. However, [Diana]'s father, [Jonathan], loved heights and even went skydiving a few times. [**Ruth**] and her son, [Jeremy], went to the park, and had a wonderful time. [Jeremy] went to the bakery with his uncle [Jonathan] to pick up some bread for lunch. What is the relationship between **Sheila** and **Ruth**?

⇩

```
@gpt_extract_relation(
  prompt="Please extract the kinship relationships from the context:",
  examples=[("Alice is Bob's mother", [("alice", "bob", "son"), ...]), ...])
type parse_relations(bound context: String, sub: String, obj: String, rela: String), …
```

⇩

sub	obj	rela
cristina	diana	daughter
jeremy	jonathan	uncle
...	...	...

Relational Knowledge Extraction with GPT

Context: [Cristina] was afraid of heights just like her daughters, [**Sheila**] and [Diana]. However, [Diana]'s father, [Jonathan], loved heights and even went skydiving a few times. [**Ruth**] and her son, [Jeremy], went to the park, and had a wonderful time. [Jeremy] went to the bakery with his uncle [Jonathan] to pick up some bread for lunch. What is the relationship between **Sheila** and **Ruth**?

⇩

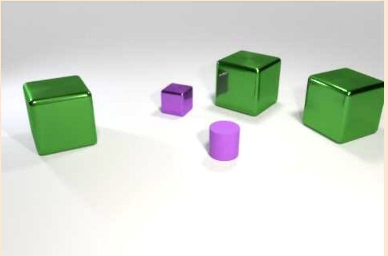
```
@gpt_extract_relation(
  prompt="Please extract the kinship relationships from the context:",
  examples=[(“Alice is Bob’s mother”, [(“alice”, “bob”, “son”), ...]), ...])
type parse_relations(bound context: String, sub: String, obj: String, rela: String), …

rel kinship(p1,p2,rela) = context(ctx) and parse_relations(ctx,p1,p2,rela)
rel kinship(p1,p3,r3) = kinship(p1,p2,r1) and kinship(p2,p3,r2) and composition(r1,r2,r3)
rel answer(r) = question(p1,p2) and kinship(p1,p2,r)
```

⇩

answer
niece

Domain Specific Language (DSL) in Scallop

Question: How many green objects are there in the image?



```
Count(FilterColor(Scene(), "green"))
```

Domain Specific Language (DSL) in Scallop

Question: How many green objects are there in the image?

```
Count(FilterColor(Scene(), "green"))
```

```
type Expr = Scene()
          | FilterColor(Expr, String)
          | Count(Expr)
          | Exists(Expr)
          | ...
```

Domain Specific Language (DSL) in Scallop

Syntax of a Query DSL:

```
type Expr = Scene() | FilterColor(Expr, String)
          | Count(Expr) | Exists(Expr) | ...
```

Semantics:

```
type eval<T>(bound expr: Expr, free output: T)
rel eval<Object>(e, o) = case e is Scene() and object(o)
rel eval<Object>(e, o) = case e is FilterShape(e1, s) and eval<Object>(e1, o) and shape(o, s)
rel eval<Object>(e, o) = case e is FilterColor(e1, c) and eval<Object>(e1, o) and color(o, c)
rel eval<usize>(e, n) = n := count(o: eval<Object>(e1, o) where e: case e is Count(e1))
rel eval<bool>(e, b) = b := exists(o: eval<Object>(e1, o) where e: case e is Exists(e1))
```

Semantic Parsing with GPT

Question: How many green objects are there in the image?

Syntax of a query DSL:

```
type Expr = Scene() | FilterColor(Expr, String)
          | Count(Expr) | Exists(Expr) | ...
```

Semantic Parsing with GPT

Question: How many green objects are there in the image?

Syntax of a query DSL:

```
type Expr = Scene() | FilterColor(Expr, String)
          | Count(Expr) | Exists(Expr) | ...
```

⇩

```
@gpt_complete(prompt="The programmatic representation of \"{{question}}\" is {{answer}}",
  examples=[("Is there a sphere?", "Exists(FilterShape(Scene(), \"sphere\"))")])
type semantic_parse(bound question: String, answer: Expr)
```

Semantic Parsing with GPT

Question: How many green objects are there in the image?

Syntax of a query DSL:

```
type Expr = Scene() | FilterColor(Expr, String)
          | Count(Expr) | Exists(Expr) | ...
```

⇩

```
@gpt_complete(prompt="The programmatic representation of \"{{question}}\" is {{answer}}",
  examples=[("Is there a sphere?", "Exists(FilterShape(Scene(), \"sphere\"))")])
type semantic_parse(bound question: String, answer: Expr)
```

⇩

question	answer
How many green objects are there in the image?	Count(FilterColor(Scene(), "green"))

Image Classification as Probabilistic Relation

OpenAI CLIP

```
@clip_classifier(["cat","dog"])
type cat_or_dog(
  bound img: Tensor,
  free label: String,
)
```

Image Classification as Probabilistic Relation

id	image
0	
1	
...	...

```
@clip_classifier(["cat","dog"])
type cat_or_dog(
  bound img: Tensor,
  free label: String,
)
```

Image Classification as Probabilistic Relation

id	image
0	
1	
...	...

```
@clip_classifier(["cat","dog"])
type cat_or_dog(
  bound img: Tensor,
  free label: String,
)
```

prob	id	label
0.00	0	cat
0.99	**0**	**dog**
0.98	**1**	**cat**
0.02	1	dog
...	...	...

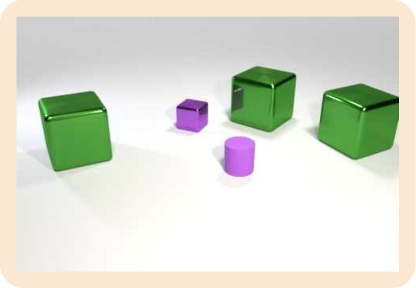
Image Segmentation as Probabilistic Relation

Segment Anything

Research by Meta AI

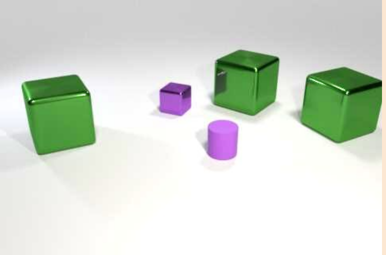
```
@segment_anything
type image_segment(
  bound img: Tensor,
  free id: u32,
  free segment: Tensor,
)
```

Image Segmentation as Probabilistic Relation



```
@segment_anything
type image_segment(
  bound img: Tensor,
  free id: u32,
  free segment: Tensor,
)
```

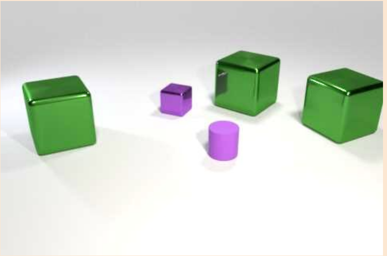
Image Segmentation as Probabilistic Relation



```
@segment_anything
type image_segment(
  bound img: Tensor,
  free id: u32,
  free segment: Tensor,
)
```

prob	id	segment
0.99	0	
0.98	1	
...	...	...

Combining Foundation Models



Question: How many green objects are there in the image?

```
@segment_anything
type image_segment(
  bound img: Tensor,
  free id: u32,
  free segment: Tensor)

@clip_classifier(["green","red",…])
type obj_color(
  bound object_segment: Tensor,
  free label: String)

@gpt_complete(prompt=
  "Please semantically parse the
   following question…")
type semantic_parse(
  bound question: String,
  free answer: Expr)
```

prob	count
0.00	0
0.03	1
0.02	2
0.91	**3**
...	...

Domain Specific Language (DSL) in Scallop

Syntax of a Query DSL:

```
type Expr = Scene() | FilterColor(Expr, String)
          | Count(Expr) | Exists(Expr) | ...
```

Semantics:

```
type eval<T>(bound expr: Expr, free output: T)
rel eval<Object>(e, o) = case e is Scene() and object(o)
rel eval<Object>(e, o) = case e is FilterShape(e1, s) and eval<Object>(e1, o) and shape(o, s)
rel eval<Object>(e, o) = case e is FilterColor(e1, c) and eval<Object>(e1, o) and color(o, c)
rel eval<usize>(e, n) = n := count(o: eval<Object>(e1, o) where e: case e is Count(e1))
rel eval<bool>(e, b) = b := exists(o: eval<Object>(e1, o) where e: case e is Exists(e1))
```

Scallop + LLM for Program Analysis

```
type input_program(program: String)

@gpt_extract_info(
  header="""Please point out the dataflow graph in the given Java program""",
  prompts=["What are the dataflow edges?",
           "What are the sources of user inputs?",
           "What are the sinks that may result in vulnerabilities?"],
  examples=[(
    ["public int f(int c) { int i = 0; int out = 0; while (i < c) { out += i; } int j = 42 / out; return out; }"],
    [[("i", "out")], [("c",)], [("out", "int j = 42 / out;")]]
  )])
type gen_dataflow_edge(bound program: String, from: String, to: String),
     gen_source(bound program: String, source: String),
     gen_sink(bound program: String, sink: String, loc: String)

rel source(s) = input_program(pgm) and gen_source(pgm, s)
rel sink(s, l) = input_program(pgm) and gen_sink(pgm, s, l)
rel edge(a, b) = input_program(pgm) and gen_dataflow_edge(pgm, a, b)
rel path(a, b) = edge(a, b) or (path(a, c) and edge(c, b))

rel vul(loc) = source(src) and sink(snk, loc) and path(src, snk)
```

Scallop + LLM for Program Analysis

Sample Java File:

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
  // some code
  response.setContentType("text/html;charset=UTF-8");
  javax.servlet.http.Cookie[] theCookies = request.getCookies();
  String param = "noCookieValueSupplied";
  if (theCookies != null) {
    for (javax.servlet.http.Cookie theCookie : theCookies) {
      if (theCookie.getName().equals("Cdsr92")) {
        param = java.net.URLDecoder.decode(theCookie.getValue(), "UTF-8");
        break;
      }
    }
  }
  String fileName = null;
  java.io.FileOutputStream fos = null;
  try {
    fileName = org.pck.bcks.helpers.Utils.TESTFILES_DIR + param;
    fos = new java.io.FileOutputStream(fileName, false);
    response.getWriter().println(ESAPI.encoder().encodeForHTML(fileName));
  } catch (Exception e) {
    // System.out.println("File exception caught and swallowed");
  } finally {
    // we tried...
  }
}
```

Extracted dataflow and source/sink information

edge:

fileName	fos
param	fileName
request	rd
request	theCookies
request	userCookie
theCookies	param

sink:

fos	new java.io.FileOutputStream(fileName, false)

source:

request
theCookies

vul:

new java.io.FileOutputStream(fileName, false)

scallop-lang.github.io

Documentation | Downloads | Resources | Tutorials

Questions