Neural Program Synthesis from Diverse Demonstration Videos

Shao-Hua Sun*¹  Hyeonwoo Noh*²  Sriram Somasundaram¹  Joseph J. Lim¹

¹University of Southern California
²Pohang University of Science and Technology
*Equal contribution
A **program** is an **interpretable** and **executable** way to describe behaviors.

```python
def run()
    if frontIsClear():
        move()
    else:
        turnLeft()
    move()
    turnLeft()
    repeat(2):
        turnRight()
    putMarker()
```
A **program** is an interpretable and executable way to describe behaviors.
A program is an interpretable and executable way to describe behaviors.
Hey, can you...
Program Synthesis

Task specification

Input/output pairs

Devlin et al. "Robustfill: Neural program learning under noisy i/o." ICML 2017
Rudy R et al. "Leveraging grammar and reinforcement learning for neural program synthesis." ICLR 2018
Program Synthesis

Task specification

Input/output pairs

Program

```
def run()
    if frontIsClear():
        move()
    else:
        turnLeft()
        move()
        turnLeft
        repeat(2):
            turnRight()
            putMarker()
```
Program Synthesis

Task specification

Input/output pairs

Program

Devlin et al. "Robustfill: Neural program learning under noisy i/o." ICML 2017
Rudy R et al. "Leveraging grammar and reinforcement learning for neural program synthesis." ICLR 2018
def run():
    if frontIsClear():
        move()
    else:
        turnLeft()
        move()
        turnLeft
    repeat(2):
        turnRight()
        putMarker()
Problem Formulation

**Input**
a set of demo videos

Program
Problem Formulation

Input
a set of demo videos

Output
a program describing the demonstrated behavior

Program
Problem Formulation

Input
a set of demo videos

Output
a program describing the
demonstrated behavior

Challenges

• Extracting unique behaviors in each demo

• Summarizing diverse behaviors as a program
Problem Formulation

Input
a set of demo videos

Output
a program describing the demonstrated behavior

**Challenges**

- **Extracting unique behaviors** in each demo  
  **Reviewer module**
- **Summarizing** diverse behaviors as a program  
  **Relation module**
Model Overview

Extract unique behaviors

Demos

Encoder

Reviewer Module

Demo features

Encoder

Relation Module

Program vector

Decoder

Program

Demo features
Model Overview

- Extract unique behaviors
- Demo features
- Relation Module
- Program vector
- Decoder
- Program

Demos

Encoder

Reviewer Module

Encoder

Encoder
Model Overview

Demos

Encoder

Demos

Encoder

Reviewer Module

Demo features

Relation Module

Program vector

Decoder

Program

Extract unique behaviors

Summarize

Decode

Model Overview

Extract unique behaviors

Summarize

Decode
Model Overview

Demos

Encoder

Reviewer Module

Demo features

Program vector

Relation Module

Decoder

Program

Extract unique behaviors

Summarize

Decode

Program vector

Demo features

Reviewer Module

Encoder

Encoder

Encoder

Extract unique behaviors

Demos
Reviewer Module

Demo<sub>1</sub> → Encoder

<table>
<thead>
<tr>
<th>t=1</th>
<th>t=2</th>
<th>...</th>
<th>t=T</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN</td>
<td>CNN</td>
<td>...</td>
<td>CNN</td>
</tr>
</tbody>
</table>

LSTM

Demo<sub>1</sub>
Reviewer Module

Demo

Encoder

CNN

CNN

CNN

LSTM

Demo1

Demo2
Reviewer Module

Encoder

Demo\textsubscript{k}

\begin{align*}
  t=1 & & \text{CNN} \\
  t=2 & & \text{CNN} \\
  \vdots & & \vdots \\
  t=T & & \text{CNN} \\
\end{align*}

LSTM

Demo\textsubscript{1}

Demo\textsubscript{2}

Demo\textsubscript{k}
Reviewer Module

Encoder

Demo_{k}

CNN

LSTM

Demon_{1}

Demon_{2}

Overall Tendency
Reviewer Module

Encoder

Demo₁

\[ \text{CNN} \] \[ \text{CNN} \] \[ \ldots \] \[ \text{CNN} \]

LSTM

\[ \text{t}=1 \] \[ \text{t}=2 \] \[ \ldots \] \[ \text{t}=T \]

Demo₁

Demo₂

\ldots

Demoₖ

Overall Tendency

Review each demo
Reviewer Module

Encoder

Demo$_{1}$

\[ t=1 \quad t=2 \quad \ldots \quad t=T \]

CNN

CNN

CNN

LSTM

Demo$_{1}$

Demo$_{2}$

\[ \ldots \]

Demo$_{k}$

\[ \text{Overall Tendency} \]

Overall Tendency

Reviewer LSTM
Reviewer Module

Encoder

Demo$_1$ →

Overall
Tendency

Reviewer LSTM
Reviewer Module

Encoder

Demo\textsubscript{1} \rightarrow \text{CNN} \rightarrow \text{CNN} \rightarrow \text{CNN} \ldots \rightarrow \text{LSTM} \rightarrow \text{Reviewer LSTM}

Overall Tendency \rightarrow \text{Demo feature\textsubscript{1}}
Model Overview

Demos

Encoder

Encoder

Encoder

Reviewer Module

Demo features
Model Overview

Demos

- Encoder

Reviewer Module

Demo features

Relation Module

Program vector

Decoder

Program

Extract unique behaviors

Summarize

Decode
if frontIsClear():
    move()
else:
    turnLeft()
Relation Module

if frontIsClear():
    move()
else:
    turnLeft()

Compare demo pairs to infer branching conditions
Relation Module

Demo pairs

Reviewer Module

Relation Module

Reviewer Module

Demo pairs

\[ g_\theta \]

\[ g_\theta \]

\[ g_\theta \]

Program vector

Model Overview

- **Demos**
  - Encoder
  - Encoder
  - Encoder

- **Demo features**
  - Reviewer Module
  - Relation Module

- **Program vector**
  - Decoder
  - Program

- **Extract unique behaviors**
  - Summarize
  - Decode
Decode a Program

**Program vector**

```
def run()  move()  if  <end>
```

LSTM
Experiments
Environments

Karel

```python
def run()
    if frontIsClear():
        move()
    else:
        turnLeft()
        move()
        turnLeft
    repeat(2):
        turnRight()
        putMarker()
```
Baselines

- Program *synthesis* baseline
- Program *induction* baseline

![Diagram showing baselines with encoder, decoder, and average vector without reviewer and relation modules.](image-url)
Baselines

- Program **synthesis** baseline
- Program **induction** baseline
Example Result: Karel

Ground truth

```python
def run():
    if frontIsClear():
        move()
    else:
        turnLeft()
    move()
    repeat(2):
        turnRight()
        putMarker()
```

Synthesis baseline

```python
def run():
    move()
    move()
    turnRight()
    putMarker()
    turnRight()
    putMarker()
```

Miss the if-else statement
Example Result: Karel

Ground truth

```python
def run():
    if frontIsClear():
        move()
    else:
        turnLeft()
    move()
    repeat(2):
        turnRight()
        putMarker()
```

Synthesis baseline

```python
def run():
    move()
    move()
    turnRight()
    putMarker()
    turnRight()
    putMarker()
```

Ours

```python
def run():
    if frontIsClear():
        move()
    else:
        turnLeft()
    turnRight()
    putMarker()
    turnRight()
    putMarker()
```
Example Result: ViZDoom

Ground truth

```python
def run():
    if inTarget(Demon):
        attack()
        moveLeft()
    else:
        moveRight()
    if isThere(Demon):
        attack()
    moveLeft()
```

Synthesis baseline

```python
def run():
    while(inTarget(HellKnight)):
        attack()
    if isThere(Demon):
        moveRight()
        attack()
    else:
        moveLeft()
```
Example Result: ViZDoom

Ground truth

def run():
    if inTarget(Demon):
        attack()
        moveLeft()
    else:
        moveRight()
    if isThere(Demon):
        attack()
        moveLeft()

Synthesis baseline

def run():
    while(inTarget(HellKnight)):
        attack()
        if isThere(Demon):
            moveRight()
            attack()
        else:
            moveLeft()

Ours

def run():
    if inTarget(Demon):
        attack()
        moveLeft()
    else:
        moveRight()
    if isThere(Demon):
        attack()
        moveLeft()
Quantitative Result: Infer Programs
Sequence Accuracy

- Measure the accuracy based on code sequences

<table>
<thead>
<tr>
<th>Synthesized program</th>
<th>Ground truth program</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>def run():</code></td>
<td><code>def run():</code></td>
</tr>
<tr>
<td><code>if A():</code></td>
<td><code>if A():</code></td>
</tr>
<tr>
<td><code>x()</code></td>
<td><code>x()</code></td>
</tr>
<tr>
<td><code>else:</code></td>
<td><code>else:</code></td>
</tr>
<tr>
<td><code>while(B()):</code></td>
<td><code>repeat(5):</code></td>
</tr>
<tr>
<td><code>y()</code></td>
<td><code>y()</code></td>
</tr>
<tr>
<td><code>z()</code></td>
<td><code>z()</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Synthesized program</th>
<th>Ground truth program</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>def run():</code></td>
<td><code>def run():</code></td>
</tr>
<tr>
<td><code>if A():</code></td>
<td><code>if A():</code></td>
</tr>
<tr>
<td><code>x()</code></td>
<td><code>x()</code></td>
</tr>
<tr>
<td><code>else:</code></td>
<td><code>else:</code></td>
</tr>
<tr>
<td><code>while(B()):</code></td>
<td><code>repeat(5):</code></td>
</tr>
<tr>
<td><code>y()</code></td>
<td><code>y()</code></td>
</tr>
<tr>
<td><code>z()</code></td>
<td><code>z()</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Synthesized program</th>
<th>Ground truth program</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>def run():</code></td>
<td><code>def run():</code></td>
</tr>
<tr>
<td><code>if A():</code></td>
<td><code>if not A():</code></td>
</tr>
<tr>
<td><code>x()</code></td>
<td><code>y()</code></td>
</tr>
<tr>
<td><code>else:</code></td>
<td><code>else:</code></td>
</tr>
<tr>
<td><code>else:</code></td>
<td><code>else:</code></td>
</tr>
<tr>
<td><code>y()</code></td>
<td><code>x()</code></td>
</tr>
<tr>
<td><code>z()</code></td>
<td><code>y()</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Synthesized program</th>
<th>Ground truth program</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>def run():</code></td>
<td><code>def run():</code></td>
</tr>
<tr>
<td><code>if A():</code></td>
<td><code>if A():</code></td>
</tr>
<tr>
<td><code>x()</code></td>
<td><code>x()</code></td>
</tr>
<tr>
<td><code>else:</code></td>
<td><code>else:</code></td>
</tr>
<tr>
<td><code>else:</code></td>
<td><code>else:</code></td>
</tr>
<tr>
<td><code>y()</code></td>
<td><code>y()</code></td>
</tr>
<tr>
<td><code>z()</code></td>
<td><code>y()</code></td>
</tr>
</tbody>
</table>
Sequence Accuracy

- Measure the accuracy based on code sequences
Sequence Accuracy

- Measure the accuracy based on code sequences

Evaluation Metrics

Sequence Accuracy

- Measure the accuracy based on code sequences

Program aliasing

Different codes with identical program semantics
Program Accuracy

- Compare programs in the program semantic space
- With some assumptions
  - ex. termination of loops

```python
def run():
    if A():
        x()
    else:
        while(B()):
            y()
            z()
```

```python
def run():
    if A():
        x()
    else:
        repeat(5):
            y()
            z()
```

```python
def run():
    if A():
        x()
    else:
        if not A():
            y()
        else:
            y()
```

```python
def run():
    if A():
        x()
    else:
        y()
```

```python
def run():
    if A():
        x()
    else:
        y()
```

```python
def run():
    if A():
        x()
    else:
        y()
```

```python
def run():
    if A():
        x()
    else:
        y()
```

```python
def run():
    if A():
        x()
    else:
        y()
```

```python
def run():
    if A():
        x()
    else:
        y()
```

```python
def run():
    if A():
        x()
    else:
        y()
```

```python
def run():
    if A():
        x()
    else:
        y()
```

```python
def run():
    if A():
        x()
    else:
        y()
```
Quantitative Result: Infer Programs

Synthesis baseline vs Ours-full

- Sequence Accuracy: 35.7 vs 41.0
- Program Accuracy: 42.4 vs 48.9
Quantitative Result: Execution
Quantitative Result: Execution

- Induction baseline
- Synthesis baseline
- Ours (with the reviewer and relation modules)

<table>
<thead>
<tr>
<th></th>
<th>Karel</th>
<th>ViZDoom</th>
</tr>
</thead>
<tbody>
<tr>
<td>Induction</td>
<td>62.8</td>
<td>35.1</td>
</tr>
<tr>
<td>Synthesis</td>
<td>64.1</td>
<td>48.2</td>
</tr>
<tr>
<td>Ours</td>
<td>72.1</td>
<td>78.4</td>
</tr>
</tbody>
</table>

ViZDoom has the highest execution accuracy among the tested algorithms.
Quantitative Result: Execution

- **Induction baseline**
- **Synthesis baseline**
- **Ours (with the reviewer and relation modules)**

<table>
<thead>
<tr>
<th></th>
<th>Karel</th>
<th>ViZDoom</th>
</tr>
</thead>
<tbody>
<tr>
<td>Induction baseline</td>
<td>62.8</td>
<td>35.1</td>
</tr>
<tr>
<td>Synthesis baseline</td>
<td>64.1</td>
<td>48.2</td>
</tr>
<tr>
<td>Ours</td>
<td>72.1</td>
<td>78.4</td>
</tr>
</tbody>
</table>
Quantitative Result: Execution

- Induction baseline
- Synthesis baseline
- Ours (with the reviewer and relation modules)

Execution Accuracy

<table>
<thead>
<tr>
<th>Task</th>
<th>Induction</th>
<th>Synthesis</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Karel</td>
<td>62.8</td>
<td>64.1</td>
<td>72.1</td>
</tr>
<tr>
<td>ViZDoom</td>
<td>35.1</td>
<td>48.2</td>
<td>78.4</td>
</tr>
</tbody>
</table>

Synthesize programs to describe behaviors
Quantitative Result: Execution

- Induction baseline
- Synthesis baseline
- Ours (with the reviewer and relation modules)

<table>
<thead>
<tr>
<th></th>
<th>Karel</th>
<th>ViZDoom</th>
</tr>
</thead>
<tbody>
<tr>
<td>Induction</td>
<td>62.8</td>
<td>35.1</td>
</tr>
<tr>
<td>Synthesis</td>
<td>64.1</td>
<td>48.2</td>
</tr>
<tr>
<td>Ours (with</td>
<td>72.1</td>
<td>78.4</td>
</tr>
<tr>
<td>reviewer and</td>
<td></td>
<td></td>
</tr>
<tr>
<td>relation modules</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Synthesize programs to describe behaviors
Reviewer module
Relation module
More Results
Effect of Additional Demos

- Induction baseline
- Synthesis baseline
- Ours (with the reviewer and relation module)

Execution Accuracy vs. Number of input demos
Incomplete Seen Denominations

The seen demos do not completely capture the underlying behavior.
Each program: **single if-else statement** with two branching consequences

### If-else experiment

<table>
<thead>
<tr>
<th></th>
<th>Induction baseline</th>
<th>Synthesis baseline</th>
<th>Ours-full</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence Accuracy</td>
<td>36.1</td>
<td>58.8</td>
<td>89.4</td>
</tr>
<tr>
<td>Program Accuracy</td>
<td>44.4</td>
<td>69.1</td>
<td></td>
</tr>
<tr>
<td>Execution Accuracy</td>
<td>26.5</td>
<td>59.9</td>
<td></td>
</tr>
</tbody>
</table>
Conclusion
def run()
    if frontIsClear():
        move()
    else:
        turnLeft()
        move()
        turnLeft
    repeat(2):
        turnRight()
        putMarker()
Neural Program Synthesis from Diverse Demonstration Videos

Extract unique behaviors  Summarize  Decode
Codes, datasets, and checkpoints are available at demo2program
Questions?

```python
def run():
    if frontIsClear():
        move()
    else:
        turnLeft()
        move()
        turnLeft()
        repeat(2):
            turnRight()
            putMarker()
```