Learning to Synthesize Programs as 
**Interpretable and Generalizable** Policies

Shao-Hua Sun

Ph.D. candidate in Computer Science
at the University of Southern California (USC)
Reinforcement Learning

Goal: maximize \( \sum_{t=0}^{t=H} \gamma^t R_t(s_t, a_t) \)
Reinforcement Learning

\[ t = H \]
\[ X_t = 0 \]
\[ R_t(s_t, a_t) \]

Goal: maximize
\[
\sum_{t=0}^{t=H} \gamma^t R_t(s_t, a_t)
\]
Reinforcement Learning

\[ t = H \]

\[ s_t, a_t \]

Goal: maximize

\[ \sum_{t=0}^{t=H} \gamma^t R_t(s_t, a_t) \]
Reinforcement Learning

\[ t = H \]

\[ X_t \]

\[ R_t(s_t, a_t) \]

Goal: maximize

\[ \sum_{t=0}^{t=H} \gamma^t R_t(s_t, a_t) \]

Environment

Agent / Policy

Observation

Reward

Action
Reinforcement Learning

Agent / Policy

Environment

Goal: maximize \( \sum_{t=0}^{t=H} \gamma^t R_t(s_t, a_t) \)
Reinforcement Learning

$$t = H$$

$$X_t = 0$$

$$t R_t(s_t, a_t)$$

Goal: maximize

$$\sum_{t=0}^{t=H} \gamma^t R_t(s_t, a_t)$$
Reinforcement Learning

Goal: maximize $\sum_{t=0}^{t=H} \gamma^t R_t(s_t, a_t)$
Reinforcement Learning

Goal: maximize
\[
\sum_{t=0}^{t=H} \gamma^t R_t(s_t, a_t)
\]
Reinforcement Learning

Goal: maximize \( \sum_{t=0}^{t=H} \gamma^t R_t(s_t, a_t) \)
Advances in Deep Reinforcement Learning

- Robot Manipulation
- Robot Locomotion
- Autonomous Driving
- Game AI
Issues with Deep Reinforcement Learning (DRL)

Generalization

Simple task

->

Complex task

Interpretability

Trust

Safety

Contestability
DEF run()
    IF frontIsClear()
        move
    ELSE
        IF frontIsClear()
            turnLeft
        ELSE
            turnRight
DEF run() 
WHILE noMarkersPresent() 
IF ELSE rightIsClear() 
  turnRight 
ELSE 
  WHILE frontIsClear() 
    move 
  turnLeft 
ELSE
DEF run()
WHILE noMarkersPresent()
IFELSE rightIsClear()
    turnRight
ELSE
    WHILE frontIsClear()
        turnLeft
    move
ENDIF

Program Policy

Synthesize

Model

Environment

Execute

Grammar

Environment Dynamics

Desired Behavior

Reward
LEAPS: Learning Embeddings for Latent Program Synthesis

Stage 1
Learning a program embedding space from randomly generated programs

Grammar

Environment Dynamics

Stage 2
Searching for a task-solving program

Desired Behavior

---

```
DEF run()
  IF rightIsClear()
    turnRight
  ELSE
    turnLeft
```

```
DEF run()
  IF frontIsClear()
    move
  ELSE
    IF frontIsClear()
      turnLeft
    ELSE
      turnRight
```

```
DEF run()
  IF frontIsClear()
    move
  ELSE
    IF frontIsClear()
      turnLeft
    ELSE
      turnRight
```

---
Learning a Program Embedding Space

\[ \mathcal{L}^P \]

Program \( \rho \)

Latent Program \( z \)

Reconstructed Program \( \hat{\rho} \)

Execute

Environment

\( \pi(a|s, z) \)

\( a_1, a_2, \ldots, a_t \)

Learnable mapping

Training Objective

Latent Program

\( q_\phi \)

\( p_\theta \)

\( \mathcal{L}^L \)

\( \mathcal{L}^R \)
Latent Program Search with Cross-Entropy Method
Baselines

DRL

VIPER (Decision Tree)

Naive Program Synthesis

Raw State

Distillation

Program Token Generated at $t$

Program Synthesized So Far

```python
def run():
    if frontIsClear():
        move()
    else:
        turnLeft()
```
Qualitative Results

StairClimber

DRL

LEAPS

FourCorners

DRL

LEAPS

Maze

DRL

LEAPS

TopOff

DRL

LEAPS
CEM trajectory Visualization

Iteration: 1
Avg. Reward: 0.1000
Next Candidate Program Reward: 0.1000
Zero-shot Generalization

Learning on 8 x 8

StairClimber

Maze

Evaluation on 100 x 100
DEF run()
WHILE noMarkersPresent()
  turnRight
  move
WHILE rightIsClear()
  turnLeft

DEF run()
IF frontIsClear()
  turnLeft
  move
WHILE noMarkersPresent()
  turnRight
  move
Results - Zero-shot Generalization

- Maze
- StairClimber
- TopOff

Reward

- DRL
- VIPER
- LEAPS
Results - Zero-shot Generalization

- **Maze**
  - DRL: 1.0
  - VIPER: 0.75
  - LEAPS: 0.75

- **StairClimber**
  - DRL: 0.75
  - VIPER: 0.5
  - LEAPS: 0.5

- **TopOff**
  - DRL: 0.25
  - VIPER: 0.25
  - LEAPS: 0.25
Interpretability

Human Debugging Interface

Performance Improvement

Reward

0
0.25
0.5
0.75
1

TopOff
FourCorner
Harvester

Original
3 Edits
5 Edits
Takeaways

- We learn to synthesize a program as a policy
- LEAPS
  - Learn a program embedding space
  - Search for a task-solving program
- Our synthesized programs achieve better
  - Task performance
  - Zero-shot *generalization*
  - *Interpretability*
Thank You

Questions?

Paper and code
clvrai.com/leaps