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

Shao-Hua Sun
Ph.D. candidate in Computer Science at the University of Southern California (USC)
Machine Learning

Robot Learning

Robotics
Applications of AI

- Natural Language Processing
- Speech Processing
- Computer Vision
- Gaming
- Healthcare
- Finance
- Social Media
- Data Security
- Education
- Entertainment
- Robotics
Robot Learning

Environment
- Structured
- Unstructured

Object
- Known
- Unseen

Task
- Pre-defined / Pre-programmed
- Diverse and Novel
Supervised Learning

Image Classification

<table>
<thead>
<tr>
<th>Model</th>
<th>Image size</th>
<th>F parameters</th>
<th>Map of the image</th>
<th>Top1 Acc. (%)</th>
<th>Top5 Acc. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAIMA</td>
<td>100x100</td>
<td>100</td>
<td>100 Classic</td>
<td>92.1</td>
<td>94.2</td>
</tr>
<tr>
<td>NAIMA</td>
<td>100x100</td>
<td>100</td>
<td>100 Classic</td>
<td>92.1</td>
<td>94.2</td>
</tr>
<tr>
<td>NAIMA</td>
<td>100x100</td>
<td>100</td>
<td>100 Classic</td>
<td>92.1</td>
<td>94.2</td>
</tr>
</tbody>
</table>

Instance Segmentation

Visual Question Answering

Table 3. Performance of different models and their published rates of the same models in 2014. Note that the published rates of the same models in 2014 are calculated to the best of our knowledge. Models for [23] are trained from scratch.

Machine Translation

Word Embeddings

Named Entity Recognition

Table 2. VQA results: The RERF ensemble is a system which uses different pre-training checkpoints and fine-tuning steps.

Question Answering

Speech Recognition

Speech Synthesis (text-to-speech)
Supervised Learning

Image Classification

Instance Segmentation

Visual Question Answering

Machine Translation

Word Embeddings

Named Entity Recognition

Question Answering

Speech Recognition

Speech Synthesis (text-to-speech)
Image Classification

Table 2. Performance of and ablation study on ImageNet, using AlexNet 100% pre-trained model. The number of compact labels in each set was calculated based on the AlexNet pre-trained model.

<table>
<thead>
<tr>
<th>Model</th>
<th>Image size</th>
<th>F parameters</th>
<th>Math-AID</th>
<th>Top1 Acc. (%)</th>
<th>Top5 Acc. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-V1</td>
<td>224 x 224</td>
<td>11.8M</td>
<td>1.94</td>
<td>70.0</td>
<td>92.2</td>
</tr>
<tr>
<td>ResNet-V2</td>
<td>256 x 256</td>
<td>52.2M</td>
<td>3.08</td>
<td>73.0</td>
<td>94.2</td>
</tr>
<tr>
<td>Inception-V1</td>
<td>299 x 299</td>
<td>17.0M</td>
<td>2.41</td>
<td>70.8</td>
<td>92.4</td>
</tr>
<tr>
<td>Inception-V2</td>
<td>300 x 300</td>
<td>21.0M</td>
<td>1.93</td>
<td>74.0</td>
<td>93.2</td>
</tr>
<tr>
<td>Inception-V3</td>
<td>352 x 352</td>
<td>31.0M</td>
<td>2.12</td>
<td>74.7</td>
<td>94.2</td>
</tr>
<tr>
<td>Inception-V4</td>
<td>377 x 377</td>
<td>33.0M</td>
<td>2.39</td>
<td>74.8</td>
<td>94.5</td>
</tr>
<tr>
<td>ResNet-152</td>
<td>331 x 331</td>
<td>68.0M</td>
<td>3.15</td>
<td>78.0</td>
<td>95.3</td>
</tr>
<tr>
<td>ResNet-101</td>
<td>224 x 224</td>
<td>60.0M</td>
<td>2.41</td>
<td>71.1</td>
<td>92.6</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>224 x 224</td>
<td>44.0M</td>
<td>2.41</td>
<td>67.0</td>
<td>90.7</td>
</tr>
</tbody>
</table>

Visual Question Answering

Speech Recognition

Speech Synthesis (text-to-speech)
Machine Translation

English sentence:
France is never cold in September

French sentence:
la france est jamais froid en septembre
Automatic Speech Recognition

This is a supervised learning method
Supervised Learning

Image Classification

Instance Segmentation

Visual Question Answering

Machine Translation

Word Embedding

Named Entity Recognition

Question Answering

Speech Recognition

Speech Synthesis (text-to-speech)
Supervised Learning

Image Classification

Instance Segmentation

Visual Question Answering

Machine Translation

Word Embeddings

Speech Recognition

Named Entity Recognition

Speech Synthesis (text-to-speech)
Supervised Learning
Supervised Learning
Supervised Learning
Robot Learning via Reinforcement Learning

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

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

Goal: maximize

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

Environment

Action

Agent / Policy

Observation

Reward

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

Observation $o_t$

Reward $R_t$

Agent / Policy

Environment

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

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

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

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

Environment

Agent / Policy

Observation

Reward $R_{t+1}$

Uninterpretable

Not generalizable

Limited to short-horizon tasks

No skill-reuse
Program-Guided Framework for Interpreting and Acquiring Complex Skills with Learning Robots

Interpretable
Programmatic / Generalizable
Hierarchical
Modular
Jump
Walk
Crawl

Video demonstration
Trajectory demonstration
Language instruction
Reward

Video demonstration
Trajectory demonstration
Language instruction
Reward

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.

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

Program

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()
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.
Neural Program Synthesis from Diverse Demonstration Videos

ICML 2018

Program

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

 Skill Specification

Program

High-level Plan

Low-level Execution

Video demonstration

Trajectory demonstration

Hyeonwoo Noh

Sriram Somasundaram

Joseph J. Lim
Imitation Learning

Demonstrations

Imitate

Neural Network Policy

Execution
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

**Demos**

- Encoder
- Encoder
- Encoder

**Demo features**

- Reviewer Module

**Program vector**

- Relation Module

**Program**

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

**Extract unique behaviors**

**Summarize**

**Decode**
Model Overview

Extract unique behaviors  Summarize  Decode

Program

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

Program vector

Relation Module

Demo features

Reviewer Module

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

Encoder

Encoder

Encoder

Demos
Model Overview

Extract unique behaviors  Summarize  Decode

Demo features

Decoder

Program vector

Program

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

Karel

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

ViZDoom

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

Neural Network Policy

Demos → Execution

Program Policy

Demos → Program → Execution

Execution Accuracy

Karel

ViZDoom

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

9/10 demos
Minor branch
1/10 demos
Major branch
Takeaway

- Synthesize programs to imitate demonstrations

Demonstrations → Synthesize → Program Policy → Execute

DEF run()
if isFontClear():
    move
ever:
    turnLeft
    move
    turnLeft
    repeat(2):
    turnRight
    putMarker
Learning to Synthesize Programs as Interpretable and Generalizable Policies

NeurIPS 2021

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

Reward

Skill Specification

Program

High-level Plan

Low-level Execution

Dweep Trivedi

Jesse Zhang

Joseph J. Lim
Reinforcement Learning

Environmental System

Neural Network Policy

Execute

Reward
Reinforcement Learning by Synthesizing Programs

```
DEF run()
  WHILE noMarkersPresent()
    IFELSE rightIsClear()
      turnRight
    ELSE
      WHILE frontIsClear()
        turnLeft
        move
  END
END
```
Reinforcement Learning by Synthesizing Programs

Grammar

Environment Dynamics

Desired Behavior

Model

Synthesize

Program Policy

Execute

Environment

Reward

\[
\text{DEF run()}
\begin{align*}
&\text{WHILE noMarkersPresent() } \\
&\text{IFELSE rightIsClear() } \\
&\text{turnRight } \\
&\text{ELSE } \\
&\text{WHILE frontIsClear() } \\
&\text{turnLeft } \\
&\text{move }
\end{align*}
\]
Method Overview

Stage 1
Learn a program embedding space from randomly generated programs

Grammar
Environment Dynamics

Stage 2
Search for a task-solving program

Desired Behavior
Stage 1: Learning a Program Embedding Space
Stage 2: Latent Program Search with Cross Entropy Method

Cross Entropy Method

Candidate Latent Program + Noise → Sample → Next Candidate Latent Programs → \( \rho \theta \)

\( r \) → \( s \) → Predicted Program

Environment

\( a \)
Cross Entropy Method Trajectory Visualization

- Dataset Program
- GT Program
- CEM Population
- CEM Next Center

Iteration: 1
Avg. Reward: 0.1000
Next Candidate Program Reward: 0.1000
Karel Tasks

StairClimber

Maze

FourCorners

Harvester

CleanHouse

TopOff
Quantitative Results

- Maze
- StairClimber
- TopOff
- FourCorner
- Harvester
- CleanHouse

- DRL
- VIPER
- Naive
- LEAPS
Zero-shot Generalization

Learning on 8x8 grids

StairClimber

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

Evaluation on 100x100 grids

Maze

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

![Bar chart showing reward for different tasks and algorithms: DRL, VIPER, LEAPS. The tasks include Maze and StairClimber. The reward values are 0.5 for Maze and 1 for StairClimber for DRL, 0.25 for Maze and 1 for StairClimber for VIPER, and 0 for Maze and 1 for StairClimber for LEAPS.]
Zero-shot Generalization

![Graph showing reward for Maze and StairClimber environments for DRL, VIPER, and LEAPS.]

- **Reward**
- **Maze**
  - DRL: ~0.8
  - VIPER: ~0.5
  - LEAPS: ~1.0
- **StairClimber**
  - DRL: ~0.2
  - VIPER: ~0.3
  - LEAPS: ~1.0
Interpretability

Human Debugging Interface

Improved Performance

- TopOff
- FourCorner
- Harvester

Reward

- Original
- 3 Edits
- 5 Edits

Results:

- New Reward: 0.3500000000000000000
- Orig Reward: 0.88363636364010315
- Best Reward: 0.88363636364010315
Takeaway

- Synthesize generalizable and interpretable programs from rewards
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.
Program Inference

Imitation learning from demonstrations

Reinforcement learning from rewards
Program Inference

Imitation learning from demonstrations

Reinforcement learning from rewards
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.
Program Guided Agent

ICLR 2020 (Spotlight)

def run():
    if is_there[River]:
        mine(Wood)
        build_bridge()
        if agent[Iron]<3:
            mine(Iron)
            place(Iron, 1, 1)
        else:
            goto(4, 2)
        while env[Gold]>0:
            mine(Gold)

Te-Lin Wu
Joseph J. Lim
Problem Formulation

Program (task)

Observation

Plan (subtasks)

Execution

```python
def run():
    if is_there[River]:
        mine(Wood)
    build_bridge()
    if agent[Iron]<3:
        mine(Iron)
        place(Iron, 1, 1)
    else:
        goto(4, 2)
    while env[Gold]>0:
        mine(Gold)
```

```
<table>
<thead>
<tr>
<th></th>
<th>x3</th>
<th>x1</th>
<th>x0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
def run():
    if is_there[River]:
        mine(Wood)
        build_bridge()
        if agent[Iron] < 3:
            mine(Iron)
            place(Iron, 2, 3)
        else:
            goto(4, 2)
    while env[Gold] > 0:
        mine(Gold)

Instructions
Programs

def run():
    while agent[Wood] <= 11:
        place(Wood, 2, 4)
        place(Iron, 1, 1)
        place(Iron, 8, 5)
        mine(Gold)
        mine(Gold)
        mine(Gold)
        repeat(4):
            sell(Gold)
            sell(Iron)

Natural Language
Descriptions

If a river is in the environment, mine a wood and then use it to build a bridge. And then if agent has less than there iron, place an iron at (2,3). Otherwise if no river, goto location (4,2). Finally, whenever there's still gold in the environment, mine a gold.

While agent has no more than 11 wood, place wood at (2,4) and iron at (1,1), then place iron at (8,5) and mine gold twice, then mine gold. After the preceding procedure, sell gold and sell iron 4 times.
End-to-end Learning Baseline

def run():
    while env[Gold] > 0:
        mine(Gold)
        if is_there[River]:
            build_bridge()
            place(Wood, 2, 3)

Program
If an agent has more than 1 iron, place an iron on (2,3), and then if there are less than 3 gold in the environment, mine gold; otherwise, goto (4,2). While gold in the environment is larger than 2, keep mining gold.
def run():
    while env[Gold] > 0:
        mine(Gold)
    if is_there[River]:
        build_bridge()
        place(Wood, 2, 3)
## Quantitative Results

### Generalization

<table>
<thead>
<tr>
<th>Instruction Method</th>
<th>Natural language descriptions</th>
<th>Programs</th>
<th>Ours (concat)</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Seq-LSTM</td>
<td>Tree-RNN</td>
<td>Transformer</td>
<td>Ours</td>
</tr>
<tr>
<td>Dataset</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>test</td>
<td>54.9±1.8%</td>
<td>56.7±1.9%</td>
<td>49.4±1.6%</td>
<td>94.0±0.5%</td>
</tr>
<tr>
<td>test-complex</td>
<td>32.4±4.9%</td>
<td>50.1±1.2%</td>
<td>49.4±1.6%</td>
<td>94.0±0.5%</td>
</tr>
<tr>
<td>Generalization gap</td>
<td>40.9%</td>
<td>31.6%</td>
<td>17.2%</td>
<td>3.8%</td>
</tr>
<tr>
<td></td>
<td>27.2%</td>
<td>15.8%</td>
<td>3.8%</td>
<td>2.3%</td>
</tr>
</tbody>
</table>

(a) Instruction Length  
(b) Instruction Complexity
# Quantitative Results

## Natural Languages < Programs

<table>
<thead>
<tr>
<th>Instruction Method</th>
<th>Natural language descriptions</th>
<th>Programs</th>
<th>Generalization gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seq-LSTM</td>
<td>54.9±1.8%</td>
<td>56.7±1.9%</td>
<td>31.6%</td>
</tr>
<tr>
<td>Transformer</td>
<td>52.5±2.6%</td>
<td>50.1±1.2%</td>
<td>15.8%</td>
</tr>
<tr>
<td>Seq-LSTM</td>
<td>32.4±4.9%</td>
<td>49.4±1.6%</td>
<td>17.2%</td>
</tr>
<tr>
<td>Transformer</td>
<td>38.2±2.6%</td>
<td>42.2±2.4%</td>
<td>40.9±1.5%</td>
</tr>
<tr>
<td>Ours (concat)</td>
<td>88.6±0.8%</td>
<td>85.2±0.8%</td>
<td>3.8%</td>
</tr>
<tr>
<td>Ours</td>
<td>94.0±0.5%</td>
<td>91.8±0.2%</td>
<td>2.3%</td>
</tr>
</tbody>
</table>

### Diagrams

(a) Instruction Length

(b) Instruction Complexity
Quantitative Results

### Table

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Natural language descriptions</th>
<th>Programs</th>
<th>Generalization gap</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Natural language descriptions</td>
<td>Programs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Seq-LSTM</td>
<td>Transformer</td>
<td>Seq-LSTM</td>
</tr>
<tr>
<td>test</td>
<td>54.9±1.8%</td>
<td>52.5±2.6%</td>
<td>56.7±1.9%</td>
</tr>
<tr>
<td>test-complex</td>
<td>32.4±4.9%</td>
<td>38.2±2.6%</td>
<td>38.8±1.2%</td>
</tr>
<tr>
<td>Generalization gap</td>
<td>40.9%</td>
<td>27.2%</td>
<td>31.6%</td>
</tr>
</tbody>
</table>

### Graphs

- **(a) Instruction Length**
  - Tree-RNN
  - Transformer
  - Seq-LSTM

- **(b) Instruction Complexity**
  - Tree-RNN
  - Transformer
  - Program
  - Natural Language
Takeaway

• Specific tasks using programs

```python
def run():
    if is_there[River]:
        mine(Wood)
        build_bridge()
    if agent[Iron]<3:
        mine(Iron)
        place(Iron, 1, 1)
    else:
        goto(4, 2)
    while env[Gold]>0:
        mine(Gold)
```

• Leverage the structure of programs with a modular framework
Composing Complex Skills by Learning Transition Policies

ICLR 2019

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

Jump → Walk → Crawl

Youngwoon Lee
Sriram Somasundaram
Edward Hu
Joseph J. Lim
Learned Skills

\[ S \rightarrow \pi \rightarrow a \]

- Walk
- Jump
- Crawl
Compose Complex Skills

High-level plan

Jump

Walk

Crawl
Compose Complex Skills

High-level plan

Jump
Walk
Crawl

Sequentially execute corresponding policies

Jump
Jump
Jump
Walk
Walk
Crawl
Compose Complex Skills

High-level plan

Jump
Walk
Crawl

Sequentially execute corresponding policies

An end state of a previous policy might not be a good initial state of the following policy
Compose Complex Skills

High-level plan

Sequentially execute corresponding policies

Transition policies
Learning Transition Policies

- **Predictor** learns to judge if a state is good for executing the next policy.
- **Transition policy** learns from the predicted rewards.
Qualitative Results

Locomotion

- Crawl
- Transition
- Walk
- Transition
- Walk Forward

Manipulation

- Pick
- Transition
- Toss
- Hit
Quantitative Results - Sample Efficiency

Manipulation

(a) Repetitive picking up  
(b) Repetitive catching  
(c) Serve

Locomotion

(d) Patrol  
(e) Hurdle  
(f) Obstacle course
• Learning *transition policies* to smoothly compose learned skills
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.
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.

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

Video demonstration
Trajectory demonstration
Language instruction
Reward

Def run():
    while True:
        if isCurb():
            jump
        elseif isCeiling():
            crawl()
        else:
            walk()
Goal: acquire a diverse set of primitive skills efficiently

Key directions

Meta-learning Meta-RL

Learning from experts
Multimodal Model-Agnostic Meta-Learning via Task-Aware Modulation

NeurIPS 2019 (Spotlight)

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

Risto Vuorio  Hexiang Hu  Joseph J. Lim
Model-Agnostic Meta-Learning (MAML)

velocity = 1 m/s

velocity = 2 m/s

velocity = 3 m/s

velocity = 2.5 m/s

Fast adapt
Multimodal Task Distribution

Walk

Jump

Crawl
Multimodal Model-Agnostic Meta-Learning (MMAML)

Modulation Network

\[
\begin{align*}
\{ \{ x \} \} & \quad \text{Samples} \\
\{ y \} & \quad \times \quad K \\
\text{Task Encoder} & \\
\text{Task Embedding} & \\
\mathcal{U} & \\
\text{MLPs} & \\
\end{align*}
\]

Task Network

\[
\begin{align*}
x & \quad \theta_1 \\
\tau_1 & \\
\theta_2 & \\
\tau_2 & \\
\vdots & \\
\theta_n & \\
\tau_n & \\
\hat{y} & \\
\end{align*}
\]
Training Algorithm

Outer loop
- Task Encoder: produce the task embedding $\omega_g$
- MLPs: modulate the task network blocks $\omega_h$

Inner loop
- Task network: fast adapt through gradient updates $\theta$

Algorithm 1 MMAML Meta-Training Procedure.

1: **Input**: Task distribution $P(T)$, Hyper-parameters $\alpha$ and $\beta$
2: Randomly initialize $\theta$ and $\omega$.
3: while not DONE do
4: Sample batches of tasks $T_j \sim P(T)$
5: for all $j$ do
6: Infer $v = h(\{x,y\}_K; \omega_h)$ with $K$ samples from $D_{T_j}^{train}$.
7: Generate parameters $\tau = \{g_i(v; \omega_g) \mid i = 1, \cdots, N\}$ to modulate each block of the task network $f$.
8: Evaluate $\nabla_\theta \mathcal{L}_{T_j}(f(x; \theta, \tau); D_{T_j}^{train})$ w.r.t the $K$ samples
9: Compute adapted parameter with gradient descent:
   $\theta'_{T_j} = \theta - \alpha \nabla_\theta \mathcal{L}_{T_j}(f(x; \theta, \tau); D_{T_j}^{train})$
10: end for
11: Update $\theta$ with $\beta \nabla_\theta \sum_{T_j \sim P(T)} \mathcal{L}_{T_j}(f(x; \theta', \tau); D_{T_j}^{val})$
12: Update $\omega_g$ with $\beta \nabla_{\omega_g} \sum_{T_j \sim P(T)} \mathcal{L}_{T_j}(f(x; \theta', \tau); D_{T_j}^{val})$
13: Update $\omega_h$ with $\beta \nabla_{\omega_h} \sum_{T_j \sim P(T)} \mathcal{L}_{T_j}(f(x; \theta', \tau); D_{T_j}^{val})$
14: end while
Regression

(a) MMAML post modulation vs. other prior models

(b) MMAML post adaptation vs. other posterior models

<table>
<thead>
<tr>
<th>Method</th>
<th>2 Modes Post Modulation</th>
<th>2 Modes Post Adaptation</th>
<th>3 Modes Post Modulation</th>
<th>3 Modes Post Adaptation</th>
<th>5 Modes Post Modulation</th>
<th>5 Modes Post Adaptation</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAML [1]</td>
<td>-</td>
<td>1.085</td>
<td>-</td>
<td>1.231</td>
<td>-</td>
<td>1.668</td>
</tr>
<tr>
<td>Multi-MAML</td>
<td>-</td>
<td>0.433</td>
<td>-</td>
<td>0.713</td>
<td>-</td>
<td>1.082</td>
</tr>
<tr>
<td>LSTM Learner</td>
<td>0.362</td>
<td>-</td>
<td>0.548</td>
<td>-</td>
<td>0.898</td>
<td>-</td>
</tr>
<tr>
<td>Ours: MMAML (Softmax)</td>
<td>1.548</td>
<td>0.361</td>
<td>2.213</td>
<td>0.444</td>
<td>2.421</td>
<td>0.939</td>
</tr>
<tr>
<td>Ours: MMAML (FiLM)</td>
<td>2.421</td>
<td>0.336</td>
<td>1.923</td>
<td>0.444</td>
<td>2.166</td>
<td>0.868</td>
</tr>
</tbody>
</table>
Image Classification

(a) Omniglot  (b) Mini-ImageNet  (c) FC100  (d) CUB  (e) Aircraft

<table>
<thead>
<tr>
<th>Method &amp; Setup</th>
<th>2 Modes</th>
<th>3 Modes</th>
<th>5 Modes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5-way</td>
<td>20-way</td>
<td>5-way</td>
</tr>
<tr>
<td></td>
<td>1-shot</td>
<td>1-shot</td>
<td>1-shot</td>
</tr>
<tr>
<td></td>
<td>9-shot</td>
<td>5-shot</td>
<td>5-shot</td>
</tr>
<tr>
<td>MAML [1]</td>
<td>66.80%</td>
<td>44.69%</td>
<td>54.55%</td>
</tr>
<tr>
<td>Multi-MAML</td>
<td>66.85%</td>
<td>53.15%</td>
<td>55.90%</td>
</tr>
<tr>
<td>MMAML (ours)</td>
<td>69.93%</td>
<td>47.80%</td>
<td>57.47%</td>
</tr>
</tbody>
</table>
Reinforcement Learning

Goal modes

ProMP

Ours
Learned Task Embedding (tSNE plot)
Learned Task Embedding (tSNE plot)

5-mode Classification

4-mode Reacher

3-mode Regression
Takeaway

- MAML struggles at learning from multimodal task distributions
- We propose multimodal MAML to alleviate the issue
Skill-based Meta-Reinforcement Learning

ICLR 2022

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

Taewook Nam  Karl Pertsch  Sung Ju Hwang  Joseph J. Lim
Meta-RL with Skills

- Meta-RL
  - Fast adapt to novel tasks
  - Limited to short-horizon & dense-reward tasks

- Skill-based RL
  - Task-agnostic offline data
  - Learning a novel task requires many samples

Ours
- Learn skills from task-agnostic offline data
- Meta-learn on long-horizon, sparse-reward tasks
SiMPL: **Skill-based Meta Policy Learning**
Environments

(a) Maze Navigation

(b) Kitchen Manipulation
Results

Maze Navigation

Kitchen Manipulation

Episode 0
Episode 20
Episode 100

Episode 0
Episode 20
Episode 100

SiMPL (Ours)
SPIRL
MTRL
PEARL-ft
PEARL
SAC

SiMPL (Ours)
SPIRL
MTRL
PEARL-ft
PEARL
SAC

Meta-training Tasks
Target Task
Agent Trajectory

Meta-training Tasks
Target Task
Agent Trajectory
Generalizable Imitation Learning from Observation via Inferring Goal Proximity

NeurIPS 2021

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

Youngwoon Lee  Andrew Szot  Joseph J. Lim
Learning from **Demonstration**

with expert's actions

Demo: \{s_1, a_1, s_2, a_2, s_3, a_3, \ldots\}

vs.

Learning from **Observation**

without expert's actions

Demo: \{s_1, s_2, s_3, \ldots\}
Learning from Observation via Inferring Goal Proximity
Experiments
Program-Guided Framework for Interpreting and Acquiring Complex Skills with Learning Robots

Interpretable
Programmatic / Generalizable
Hierarchical
Modular
Program-Guided Framework for Interpreting and Acquiring Complex Skills with Learning Robots

Skill Specification → Program → High-level Plan → Low-level Execution
Neural Program Synthesis from Diverse Demonstration Videos

Def run():
    while frontIsClear():
        move()
        turnRight()
        if thereIsPig():
            attack()
        else:
            if not thereIsWolf():
                spawnPig()
            else:
                giveBone()
Learning to Synthesize Programs as Interpretable and Generalizable Policies

def run():
    while frontIsClear():
        move()
        turnRight()
        if thereIsPig():
            attack()
        else:
            if not thereIsWolf():
                spawnPig()
            else:
                giveBone()

Neural Program Synthesis from Diverse Demonstration Videos

NeurIPS 2021

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

Learning to Synthesize Programs as Interpretable and Generalizable Policies

NEAPS Program Synthesizer

Synthesize Program Policy

Program Policy

LEAPS Program Synthesizer

Synthesize

Execute

Reward

NeurIPS 2021
Neural Program Synthesis from Diverse Demonstration Videos

ICML 2018

def run():
    while frontIsClear():
        move()
        turnRight()
        if thereIsPig():
            attack()
        else:
            if not thereIsWolf():
                spawnPig()
            else:
                giveBone()

Synthesized Program

def run():
    while frontIsClear():
        move()
        turnRight()
        if thereIsPig():
            attack()
        else:
            if not thereIsWolf():
                spawnPig()
            else:
                giveBone()

Program

High-level Plan

Low-level Execution

Program Guided Agent

ICLR 2020 (Spotlight)
Neural Program Synthesis from Diverse Demonstration Videos

Learning to Synthesize Programs as Interpretable and Generalizable Policies

Composing Complex Skills by Learning Transition Policies

High-level Plan

Low-level Execution

NeurIPS 2021

ICLR 2020 (Spotlight)

NeurIPS 2021

ICLR 2019

ICML 2018
Learning to Synthesize Programs as Interpretable and Generalizable Policies

NeurIPS 2021

Program

High-level Plan

Low-level Execution

Skill Specification

Program

ICML 2018

Neural Program Synthesis from Diverse Demonstration Videos

Synthesized Program

def run():
    while frontIsClear():
        move()
        turnRight()
    if thereIsPig():
        attack()
    else:
        if not thereIsWolf():
            spawnPig()
        else:
            giveBone()

Synthesized Program

def run():
    while frontIsClear():
        move()
        turnRight()
    if thereIsPig():
        attack()
    else:
        if not thereIsWolf():
            spawnPig()
        else:
            giveBone()

Neural Program Synthesis from Diverse Demonstration Videos

Synthesized Program

def run():
    while frontIsClear():
        move()
        turnRight()
    if thereIsPig():
        attack()
    else:
        if not thereIsWolf():
            spawnPig()
        else:
            giveBone()

Neural Program Synthesis from Diverse Demonstration Videos

Synthesized Program

def run():
    while frontIsClear():
        move()
        turnRight()
    if thereIsPig():
        attack()
    else:
        if not thereIsWolf():
            spawnPig()
        else:
            giveBone()

Neural Program Synthesis from Diverse Demonstration Videos

Synthesized Program

def run():
    while frontIsClear():
        move()
        turnRight()
    if thereIsPig():
        attack()
    else:
        if not thereIsWolf():
            spawnPig()
        else:
            giveBone()

Neural Program Synthesis from Diverse Demonstration Videos

Synthesized Program

def run():
    while frontIsClear():
        move()
        turnRight()
    if thereIsPig():
        attack()
    else:
        if not thereIsWolf():
            spawnPig()
        else:
            giveBone()

Neural Program Synthesis from Diverse Demonstration Videos

Synthesized Program

def run():
    while frontIsClear():
        move()
        turnRight()
    if thereIsPig():
        attack()
    else:
        if not thereIsWolf():
            spawnPig()
        else:
            giveBone()

Neural Program Synthesis from Diverse Demonstration Videos

Synthesized Program

def run():
    while frontIsClear():
        move()
        turnRight()
    if thereIsPig():
        attack()
    else:
        if not thereIsWolf():
            spawnPig()
        else:
            giveBone()

Neural Program Synthesis from Diverse Demonstration Videos

Synthesized Program

def run():
    while frontIsClear():
        move()
        turnRight()
    if thereIsPig():
        attack()
    else:
        if not thereIsWolf():
            spawnPig()
        else:
            giveBone()

Neural Program Synthesis from Diverse Demonstration Videos

Synthesized Program

def run():
    while frontIsClear():
        move()
        turnRight()
    if thereIsPig():
        attack()
    else:
        if not thereIsWolf():
            spawnPig()
        else:
            giveBone()

Neural Program Synthesis from Diverse Demonstration Videos

Synthesized Program

def run():
    while frontIsClear():
        move()
        turnRight()
    if thereIsPig():
        attack()
    else:
        if not thereIsWolf():
            spawnPig()
        else:
            giveBone()

Neural Program Synthesis from Diverse Demonstration Videos

Synthesized Program

def run():
    while frontIsClear():
        move()
        turnRight()
    if thereIsPig():
        attack()
    else:
        if not thereIsWolf():
            spawnPig()
        else:
            giveBone()

Neural Program Synthesis from Diverse Demonstration Videos

Synthesized Program

def run():
    while frontIsClear():
        move()
        turnRight()
    if thereIsPig():
        attack()
    else:
        if not thereIsWolf():
            spawnPig()
        else:
            giveBone()

Neural Program Synthesis from Diverse Demonstration Videos

Synthesized Program

def run():
    while frontIsClear():
        move()
        turnRight()
    if thereIsPig():
        attack()
    else:
        if not thereIsWolf():
            spawnPig()
        else:
            giveBone()

Neural Program Synthesis from Diverse Demonstration Videos

Synthesized Program

def run():
    while frontIsClear():
        move()
        turnRight()
    if thereIsPig():
        attack()
    else:
        if not thereIsWolf():
            spawnPig()
        else:
            giveBone()
Toward Multimodal Model-Agnostic Meta-Learning

Multimodal Model-Agnostic Meta-Learning via Task-Aware Modulation

Skill-based Meta-Reinforcement Learning

Generalizable Imitation Learning from Observation via Inferring Goal Proximity

NeurIPS 2019 (Spotlight)

NeurIPS 2021

Meta-learning workshop @ NeurIPS 2018

ICLR 2022

NeurIPS 2021 (Spotlight)
Feedback Adversarial Learning: Spatial Feedback for Improving Generative Adversarial Networks

Multi-view to Novel view: Synthesizing Views with Self-Learned Confidence

CVPR 2019

ECCV 2018
Joseph J. Lim
PI @ USC / CLVR

Youngwoon Lee
PhD @ USC / CLVR

Karl Pertsch
PhD @ USC / CLVR

Jesse Zhang
PhD @ USC / CLVR

Minyoung Huh
PhD @ MIT with Prof. Pulkit Agrawal & Prof. Phillip Isola

Edward Hu
PhD @ UPenn with Prof. Jayaraman

Hyeonwoo Noh
Research Scientist @ OpenAI

Ning Zhang
Research Scientist @ Facebook

Risto Vuorio
PhD @ Oxford with Prof. Whiteson

Te-Lin Wu
PhD @ UCLA with Prof. Nanyun Peng

Andrew Szot
PhD @ Georgia Tech with Prof. Zsolt Kira & Prof. Dhruv Batra

Sung Ju Hwang
Associate Professor @ KAIST

Dweep Trivedi
Visitor @ USC / CLVR

Sriram Somasundaram
MS @ Stanford

Hexiang Hu
Research Scientist @ Google

Yuan-Hong Liao
PhD @ U of T with Prof. Sanja Fidler

Jesse Zhang
PhD @ USC / CLVR

Karl Pertsch
PhD @ USC / CLVR

Minyoung Huh
PhD @ MIT with Prof. Pulkit Agrawal & Prof. Phillip Isola

Edward Hu
PhD @ UPenn with Prof. Jayaraman

Hyeonwoo Noh
Research Scientist @ OpenAI

Ning Zhang
Research Scientist @ Facebook

Risto Vuorio
PhD @ Oxford with Prof. Whiteson

Te-Lin Wu
PhD @ UCLA with Prof. Nanyun Peng

Andrew Szot
PhD @ Georgia Tech with Prof. Zsolt Kira & Prof. Dhruv Batra

Sung Ju Hwang
Associate Professor @ KAIST

Dweep Trivedi
Visitor @ USC / CLVR

Sriram Somasundaram
MS @ Stanford
Thank You

Questions?