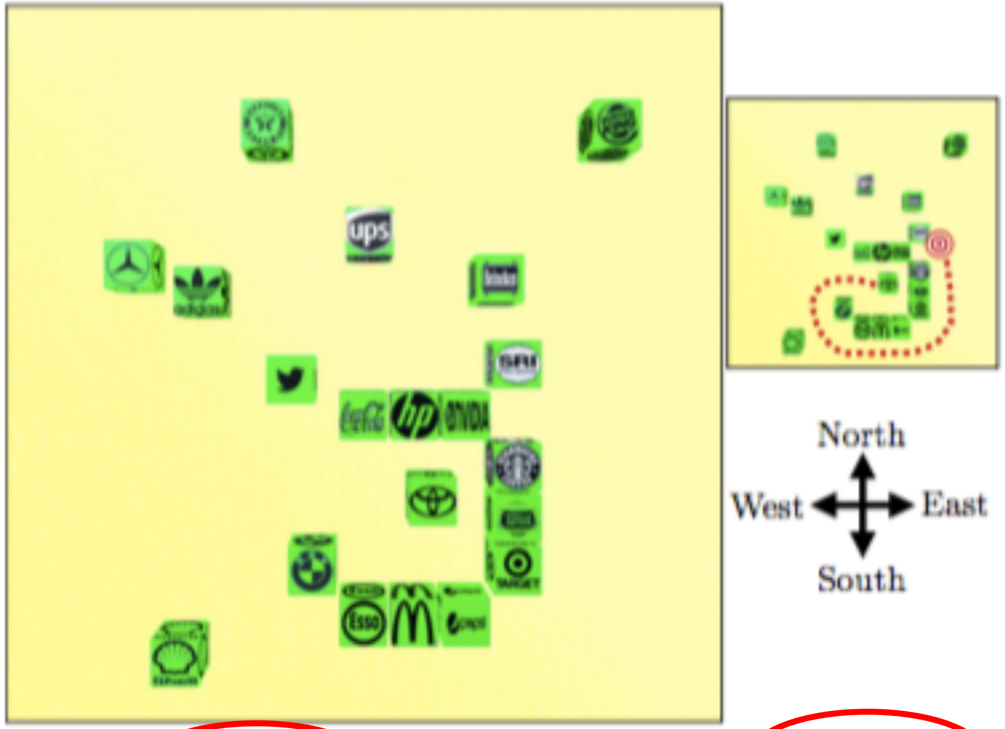
Mapping instructions and visual observations to actions with reinforcement learning

Dipendra Misra, John Langford, and Yoav Artzi

Task



Put the Toyota block in the same row as the SRI block, in the first open space to the right of the SRI block

Move Toyota to the immediate right of SRI, evenly aligned and slightly separated

Move the Toyota block around the pile and place it just to the right of the SRI block

Place Toyota block just to the right of The SRI Block

Toyota, right side of SRI

Features:

- One single model
- Limited data

Task

- Training inputs:

$$\{(x^i, s_1^i, e^i\}_{i=1}^N$$

  - x^i is an instruction is a sequence $< x_1, x_2, \ldots x_n >$ where x_i is a token
  - s_1^i is a start state
  - e^i is an execution demonstration of *x* starting at s1, which is an m-length sequence <(s_1,a_1), …(s_m, a_m)>, where $s_j \ \in S, a_j \ \in A \ and \ a_m = \mathrm{STOP}$

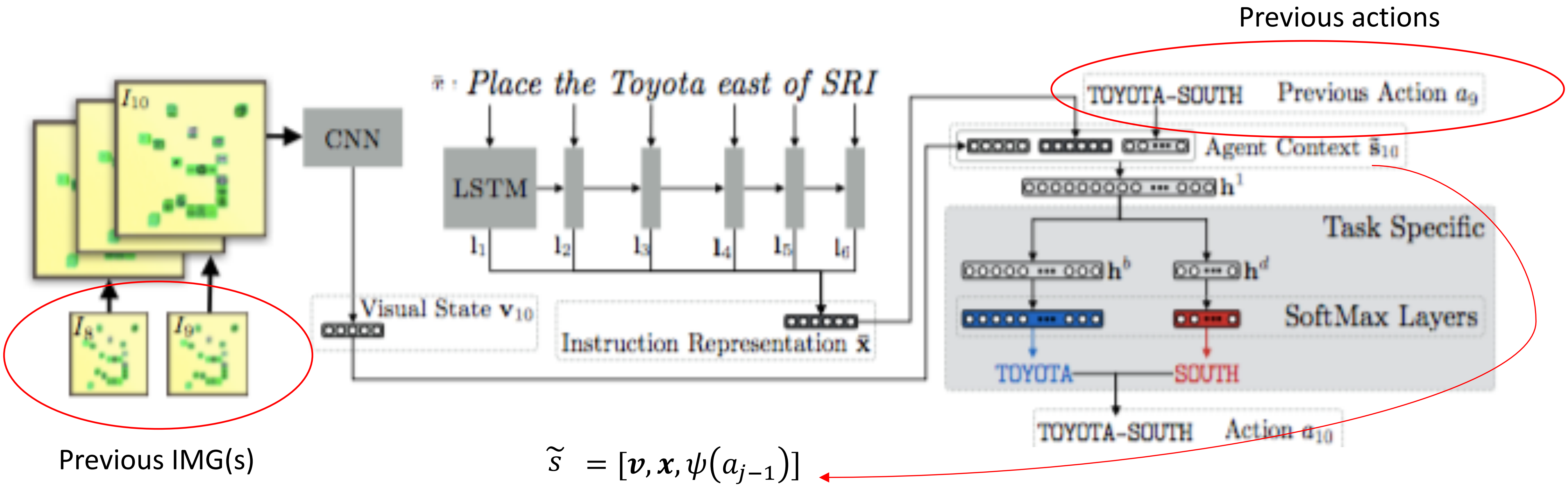
Task

- Testing inputs:

$$\{(x^i, s_1^i, s_g^i\}_{i=1}^{M}$$

  - x^i is an instruction is a sequence $< x_1, x_2, \dots x_n >$ where $x_i \ is$ a token
  - s_1^i is a start state
  - s_g^i is a goal state

Architecture



$$\tilde{s} = [\boldsymbol{v}, \boldsymbol{x}, \psi(a_{j-1})]$$

State context

Actions

Action is decomposed into direction a^D and block a^B. We compute the feed forward network:

$$
\begin{aligned}
\mathbf{h}^1 &= \max(\mathbf{W}^{(1)}\tilde{\mathbf{s}}_j + \mathbf{b}^{(1)}, 0) \\
\mathbf{h}^D &= \mathbf{W}^{(D)}\mathbf{h}^1 + \mathbf{b}^{(D)} \\
\mathbf{h}^B &= \mathbf{W}^{(B)}\mathbf{h}^1 + \mathbf{b}^{(B)} ,
\end{aligned}
$$

$$
\begin{aligned}
P(a_j^D = d \mid \bar{x}, s_j, a_{j-1}) &\propto \exp(\mathbf{h}_d^D) \\
P(a_j^B = b \mid \bar{x}, s_j, a_{j-1}) &\propto \exp(\mathbf{h}_b^B) .
\end{aligned}
$$

$$
P(a \mid x, s, a_{j-1}) = P(a_j^D = d \mid x, s, a_{j-1}) * P(a_j^B = d \mid x, s, a_{j-1})
$$

Reward function

The final shaped reward is the sum of reward shaping and the problem reward

$$R^{(i)}(s,a) = \begin{cases} 1.0 & \text{if } s = s_{m^{(i)}} \text{ and } a = \text{STOP} \\ -1.0 & s \neq s_{m^{(i)}} \text{ and } a = \text{STOP} \\ -1.0 & a \text{ fails to execute} \\ -\delta & \text{else} \end{cases},$$

where $m^{(i)}$ is the length of $\bar{e}^{(i)}$.

Reward shaping:

- Distance-based shaping (F_1)
- Trajectory-based shaping(F_2)

Reward shaping

- Distance-based shaping (if the agent moved closer to the goal state)

$$F_1^{(i)}(s_j, a_j, s_{j+1}) = \phi_1^{(i)}(s_{j+1}) - \phi_1^{(i)}(s_j) \quad .$$

The **potential $\boldsymbol{\phi_1^i}$** is proportional to the negative distance from the goal state:

$$\phi_1^{(i)}(s) = -\eta \| s - s_g^{(i)} \|$$

- Trajectory-based shaping (considering the previous state and action)

$$F_2^{(i)}(s_{j-1}, a_{j-1}, s_j, a_j) = \phi_2^{(i)}(s_j, a_j) - \phi_2^{(i)}(s_{j-1}, a_{j-1})$$

Encourage the agent to take action close to the execution demonstration state

Policy gradient objective

- Contextual Bandit

Suitable for the few-sample regime common in natural language problem

Policy is learned from agent context rather than the world state

Immediate reward

$$\nabla_\theta \mathcal{J} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}[\nabla_\theta \log \pi(\tilde{s}, a) R^{(i)}(s, a)]$$

Policy gradient objective

- Contextual Bandit

Immediate reward

$$\nabla_\theta \mathcal{J} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}[\nabla_\theta \log \pi(\tilde{s}, a) R^{(i)}(s, a)]$$

$$\theta \leftarrow \theta + \mu \text{ADAM}(\tfrac{1}{j} \textstyle\sum_{j'=1}^{j} \Delta_{j'})$$

Average for this episode

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta}), \forall a \in \mathcal{A}, s \in \mathcal{S}, \boldsymbol{\theta} \in \mathbb{R}^n$
Initialize policy weights $\boldsymbol{\theta}$
Repeat forever:
    Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
    For each step of the episode $t = 0, \ldots, T-1$:
        $G_t \leftarrow$ return from step t
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G_t \nabla_{\boldsymbol{\theta}} \log \pi(A_t|S_t, \boldsymbol{\theta})$

Total reward for an episode

Policy gradient objective

- Entropy Penalty

To avoid falling into negative reward and rarely completing the task in the early training

$$\nabla_\theta \mathcal{J} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}[\nabla_\theta \log \pi(\tilde{s}, a) R^{(i)}(s, a) + \lambda \nabla_\theta H(\pi(\tilde{s}, \cdot))]$$

entropy

Algorithm

Algorithm 1 Policy gradient learning

Input: Training set $\{(\bar{x}^{(i)}, s_1^{(i)}, \bar{e}^{(i)})\}_{i=1}^{N}$, learning rate μ, epochs T, horizon J, and entropy regularization term λ.

Definitions: $\textsc{Img}(s)$ is a camera sensor that reports an RGB image of state s. π is a probabilistic neural network policy parameterized by θ, as described in Section 4. $\textsc{Execute}(s, a)$ executes the action a at the state s, and returns the new state. $R^{(i)}$ is the reward function for example i. $\textsc{Adam}(\Delta)$ applies a per-feature learning rate to the gradient Δ (Kingma and Ba, 2014).

Output: Policy parameters θ.

```
» Iterate over the training data.
for t = 1 to T, i = 1 to N do
    I_{1-K}, ..., I_0 = 0⃗
    a_0 = NONE, s_1 = s_1^(i)
    j = 1
    » Rollout up to episode limit.
    while j ≤ J and a_j ≠ STOP do
        » Observe world and construct agent context.
        I_j = IMG(s_j)
        s̃_j = (x̄^(i), I_j, I_{j-1}, ..., I_{j-K}, a^d_{j-1})
        » Sample an action from the policy.
        a_j ~ π(s̃_j, a)
        s_{j+1} = EXECUTE(s_j, a_j)
        » Compute the approximate gradient.
        Δ_j ← ∇_θ log π(s̃_j, a_j) R^(i)(s_j, a_j)
                  + λ ∇_θ H(π(s̃_j, ·))
        j += 1
    θ ← θ + μ ADAM( (1/j) Σ_{j'=1}^{j} Δ_{j'} )
return θ
```

Initialization

Construct state context

Sample an action

Compute policy gradient

Results

Distance error:
The sum of Euclidean distances for each block between its position at the end of the execution and in the gold goal state

F1 is better than F2

Algorithm	Distance Error		Min. Distance	
	Mean	Med.	Mean	Med.
Demonstrations	0.35	0.30	0.35	0.30
Baselines				
STOP	5.95	5.71	5.95	5.71
RANDOM	15.3	15.70	5.92	5.70
SUPERVISED	4.65	4.45	3.72	3.26
REINFORCE	5.57	5.29	4.50	4.25
DQN	6.04	5.78	5.63	5.49
Our Approach	3.60	3.09	2.72	2.21
w/o Sup. Init	3.78	3.13	2.79	2.21
w/o Prev. Action	3.95	3.44	3.20	2.56
w/o F_1	4.33	3.74	3.29	2.64
w/o F_2	3.74	3.11	3.13	2.49
w/ Distance Reward	8.36	7.82	5.91	5.70
Ensembles				
SUPERVISED	4.64	4.27	3.69	3.22
REINFORCE	5.28	5.23	4.75	4.67
DQN	5.85	5.59	5.60	5.46
Our Approach	**3.59**	**3.03**	**2.63**	**2.15**

Table 2: Mean and median (Med.) development results.

Algorithm	Distance Error		Min. Distance	
	Mean	Med.	Mean	Med.
Demonstrations	0.37	0.31	0.37	0.31
STOP	6.23	6.12	6.23	6.12
RANDOM	15.11	15.35	6.21	6.09
Ensembles				
SUPERVISED	4.95	4.53	3.82	3.33
REINFORCE	5.69	5.57	5.11	4.99
DQN	6.15	5.97	5.86	5.77
Our Approach	**3.78**	**3.14**	**2.83**	**2.07**

Table 3: Mean and median (Med.) test results.

Reflects problem with limited data