Learning to Manipulate Object Collections Using Grounded State Representations

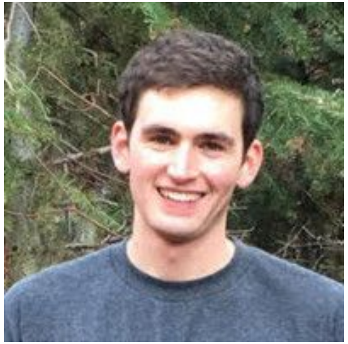
Matthew Wilson
University of Utah
United States
matthew.b.wilson@utah.edu

Tucker Hermans
University of Utah & NVIDIA
United States
thermans@cs.utah.edu

M. Yunus Seker

Colors Lab Weekly Presentations - Week 10

Authors



Matthew Wilson
University of Utah
United States
matthew.b.wilson@utah.edu

Tucker Hermans
University of Utah & NVIDIA
United States
thermans@cs.utah.edu

Introduction

- Humans regularly manipulate object groups as collections

- Learning to simultaneously manipulate many objects. To overcome this challenge:
  - Use raw object poses and images from a simulator to learn a latent space
  - Use this learned representations to train an image-based policy

- Learning policies directly from RGB images
  - Feature extraction
  - Hard to reach task relevant information without wasting encoder capacity to textures, backgrounds, etc.

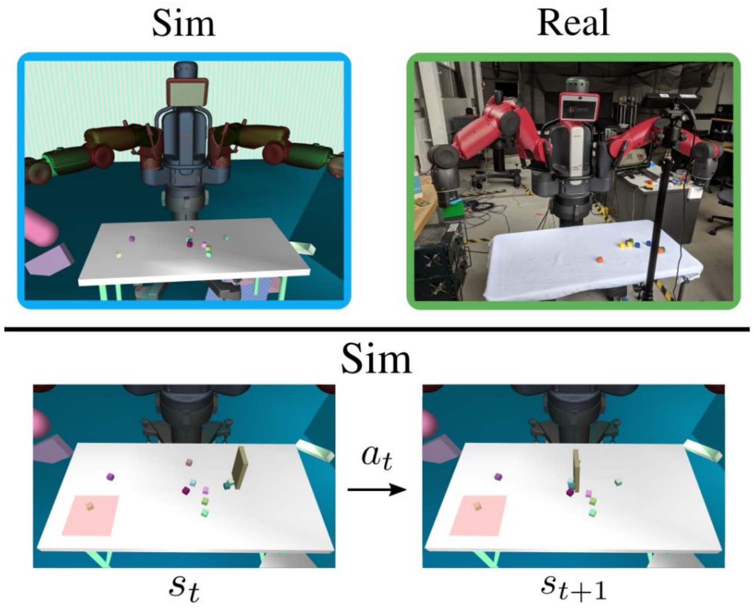
- Simulation-based training is a promising route to sidestep these difficulties
  - Exploiting raw state information to calculate rewards
  - However, all existing work rely on **fixed-length** vector inputs

    “Ours is the first work we are aware of in this space capable of handling a variable number of multiple objects”

  - Main contribution: proposed method scales to multiple and variable object settings

Task Overview

- Pushing a collection of 1-20 homogeneous objects to desired areas on a tabletop
- The agent’s state is factored into a set of 2D coordinates of object centers {x1, x2, ...}.
- Actions are parameterized by 4 continuous values: (x, y, θ, d)
- Kinect Camera
- Data collection -> scripted policy
- Two Phases
    - Representation Learning
        - state representations from images and poses
    - Reinforcement Learning
        - images -> actor network, policy estimation
        - poses -> critic networks, value estimation



Representation Learning

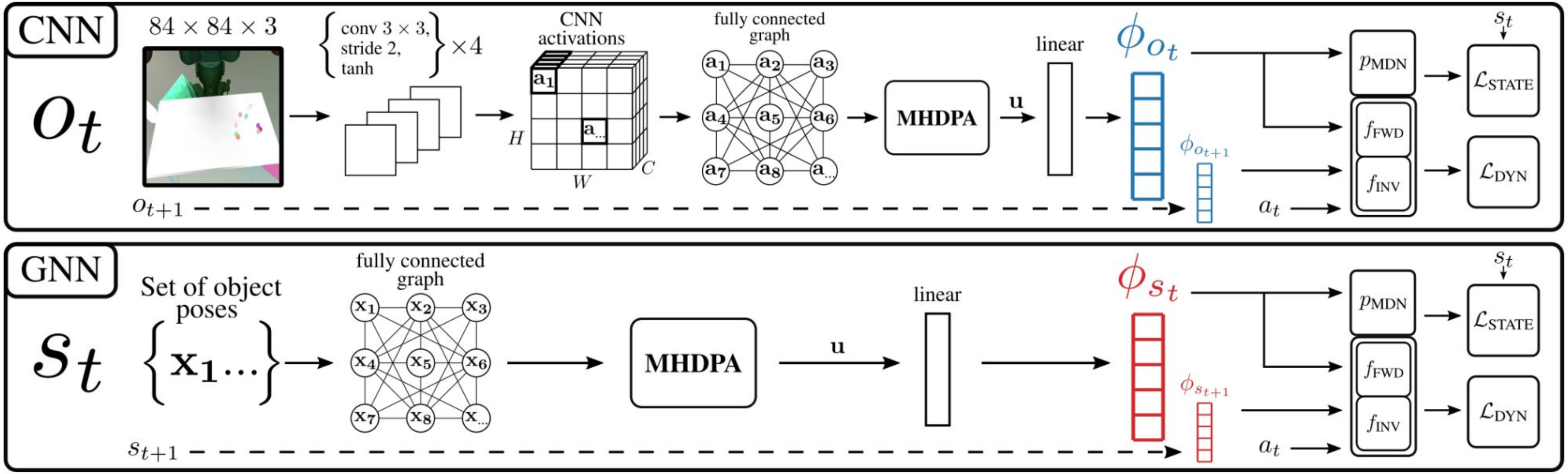
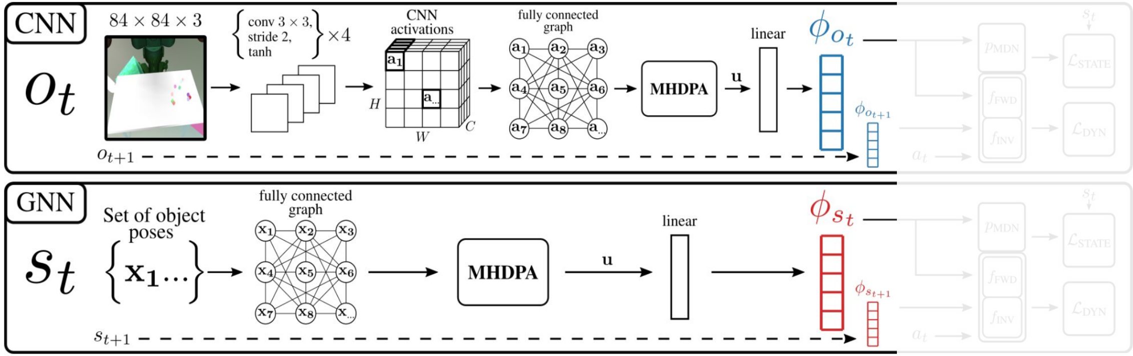


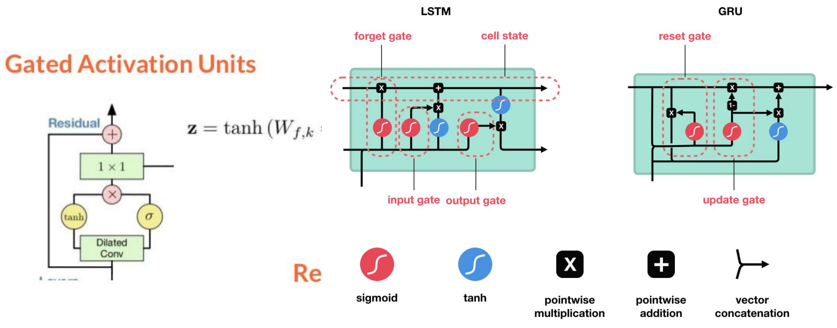
Figure 3: CNN and GNN architectures and losses for self-supervised representation learning

- Two different encoder, CNN and GNN.
- Learning representations by predicting
  - state, with mixture density network (neural network version of GMM)
  - forward and inverse dynamics (with action and next state), with simple multi-layer perceptron
    - $f_{fwd}(\varphi_t, a_t) = \varphi_{t+1}$
    - $f_{inv}(\varphi_t, \varphi_{t+1}) = a_t$

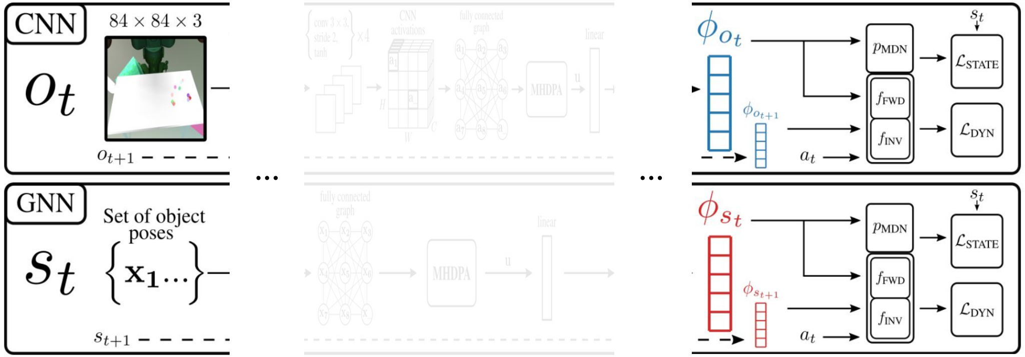
The Input Side - Encoding Raw State

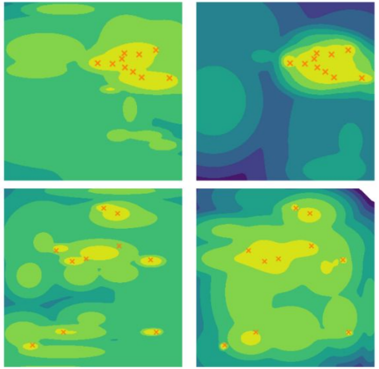


- Should be permutation invariant and scale to multi-objects
- Convert the state into a fully connected graph G associating a vertex, v_i , with each object centroid, x_i .
- G’ = MHDPA(G) #multi-head dot product attention
- $\mathbf{u} = \sum_{i=1}^{n} \tanh(W_1 \mathbf{v}'_i) \odot \sigma(W_2 \mathbf{v}'_i)$
- linear(u) = φ_t
- to avoid shortcomings of vanilla CNN models in tasks that require tracking object counts and reasoning about relative positions of objects, *and we also find it helps.*



The Output Side - Learning State Representations





- State Prediction:
  - should also scale to multi objects, and be permutation invariant
  - how we even count objects in an image to know how many predictions we should make?
- Mixture Density Networks

$$p_{\text{MDN}}(\mathbf{p}|\phi) = \sum_{k=1}^{K} \alpha(\phi)_k \, \mathcal{N}\Big(\mathbf{p}\Big|\mu(\phi)_k,\ \Sigma(\phi)_k\Big)$$

$\mathcal{L}_{\text{STATE}} = -\frac{1}{n}\sum_{i=1}^{n} \log p_{\text{MDN}}(\mathbf{p} = \mathbf{x}_i|\phi)$ for all ground truth $x_i \in (x_1, x_2, .. \, x_n)$

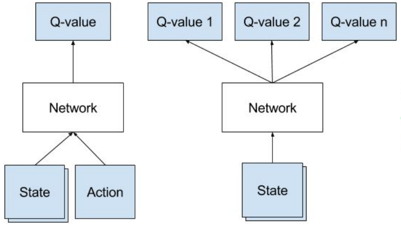
- Dynamics Prediction:
  - to condition representation to actions
  - two simple MLPs

$$\mathcal{L}_{\text{DYN}} = \mathcal{L}_{\text{FWD}} + \mathcal{L}_{\text{INV}} = \frac{1}{2}\left\|f_{\text{FWD}}(\phi_t, a_t) - \phi_{t+1}\right\|_2^2 + \frac{1}{2}\left\|f_{\text{INV}}(\phi_t, \phi_{t+1}) - a_t\right\|_2^2$$

$$\mathcal{L}_{\text{FULL}} = \mathcal{L}_{\text{STATE}} + \mathcal{L}_{\text{DYN}}$$

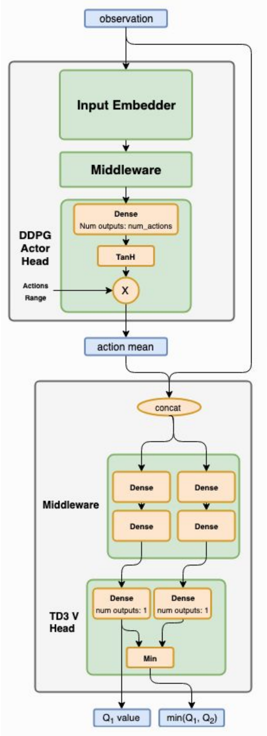
Soft Actor Critic (openAI spinning up)

Deep Q-Network (DQN): discrete action space



$$\underbrace{Q(s,a)}_{\text{Q target}} = \underbrace{r(s,a)}_{\text{Reward of taking that action at that state}} + \underbrace{\gamma max_a Q(s',a)}_{\text{Discounted max q value among all possibles actions from next state.}}$$

TD3 (Twin Delayed DDPG):



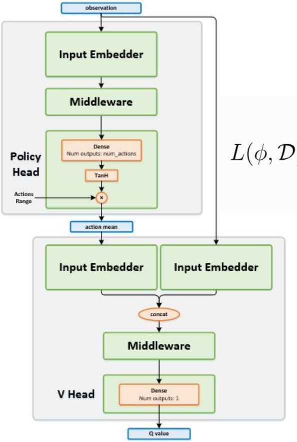
$$y(r,s',d) = r + \gamma(1-d)\min_{i=1,2} Q_{\phi_{i,\text{targ}}}(s', a'(s')),$$

Deep Deterministic Policy Gradient:

continuous action space

actor -> policy

critic -> value estimation



$$L(\phi, \mathcal{D}) = \underset{(s,a,r,s',d)\sim\mathcal{D}}{\mathrm{E}}\left[\left(Q_\phi(s,a) - \left(r + \gamma(1-d)Q_{\phi_{\text{targ}}}(s', \mu_{\theta_{\text{targ}}}(s'))\right)\right)^2\right]$$

Soft Actor Critic (openAI spinning up) cont.

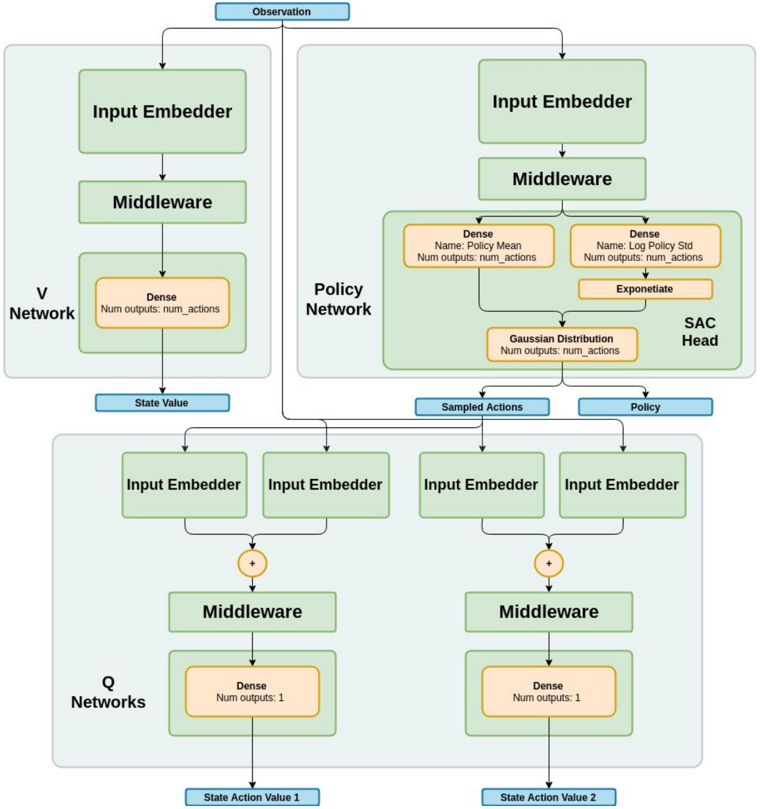
- Bridge between stochastic policy optimization and DDPG-style approaches
- Central feature of SAC is *entropy regularization*

$$H(P) = \underset{x \sim P}{\mathrm{E}} \left[-\log P(x) \right]$$

$$V^{\pi}(s) = \underset{\tau \sim \pi}{\mathrm{E}} \left[\left. \sum_{t=0}^{\infty} \gamma^t \bigg(R(s_t, a_t, s_{t+1}) + \alpha H\left(\pi(\cdot|s_t)\right) \bigg) \right| s_0 = s \right]$$

$$Q^{\pi}(s,a) = \underset{\tau \sim \pi}{\mathrm{E}} \left[\left. \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) + \alpha \sum_{t=1}^{\infty} \gamma^t H\left(\pi(\cdot|s_t)\right) \right| s_0 = s, a_0 = a \right]$$

$$V^{\pi}(s) = \underset{a \sim \pi}{\mathrm{E}} \left[Q^{\pi}(s,a) \right] + \alpha H\left(\pi(\cdot|s)\right)$$



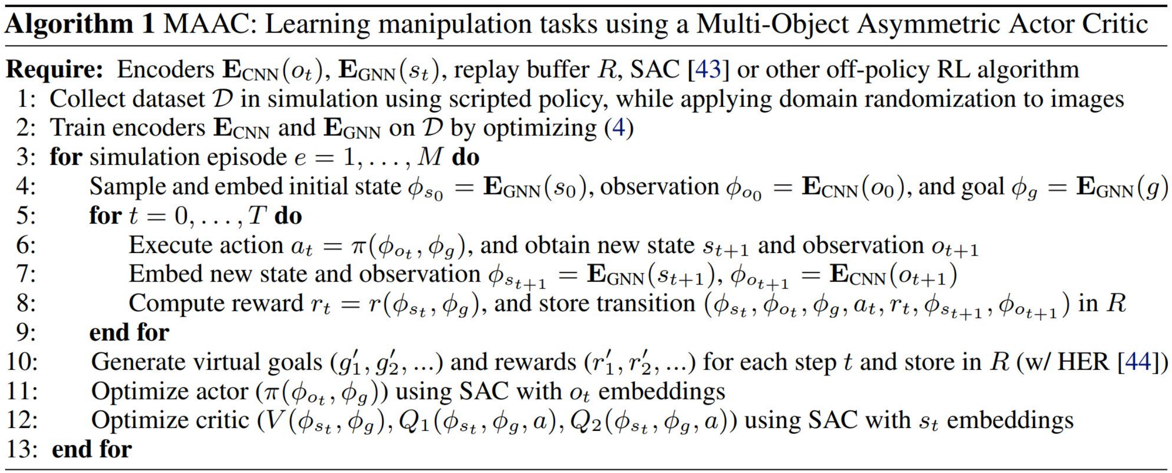
- SAC concurrently learns one policy, two Q, and a V

Q
$$L(\phi_i, \mathcal{D}) = \underset{(s,a,r,s',d) \sim \mathcal{D}}{\mathrm{E}} \left[\Bigg(Q_{\phi_i}(s,a) - \left(r + \gamma(1-d) V_{\psi_{\text{targ}}}(s')\right) \Bigg)^2 \right]$$

V
$$L(\psi, \mathcal{D}) = \underset{\substack{s \sim \mathcal{D} \\ \tilde{a} \sim \pi_\theta}}{\mathrm{E}} \left[\Bigg(V_{\psi}(s) - \left(\min_{i=1,2} Q_{\phi_i}(s, \tilde{a}) - \alpha \log \pi_\theta(\tilde{a}|s) \right) \Bigg)^2 \right]$$

pi
$$\underset{a \sim \pi}{\mathrm{E}} \left[Q^{\pi}(s,a) - \alpha \log \pi(a|s) \right]$$

Reinforcement Learning



Algorithm 1 MAAC: Learning manipulation tasks using a Multi-Object Asymmetric Actor Critic

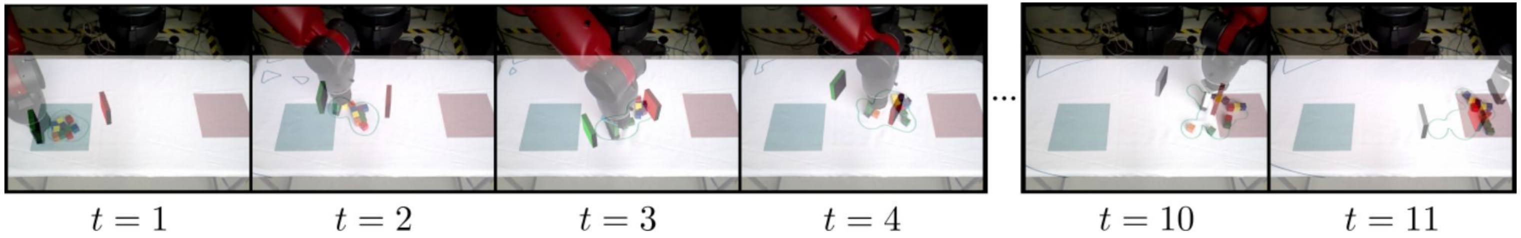
```
Require: Encoders $\mathbf{E}_{\text{CNN}}(o_t)$, $\mathbf{E}_{\text{GNN}}(s_t)$, replay buffer $R$, SAC [43] or other off-policy RL algorithm
Collect dataset $\mathcal{D}$ in simulation using scripted policy, while applying domain randomization to images
Train encoders $\mathbf{E}_{\text{CNN}}$ and $\mathbf{E}_{\text{GNN}}$ on $\mathcal{D}$ by optimizing (4)
for simulation episode $e = 1, \ldots, M$ do
    Sample and embed initial state $\phi_{s_0} = \mathbf{E}_{\text{GNN}}(s_0)$, observation $\phi_{o_0} = \mathbf{E}_{\text{CNN}}(o_0)$, and goal $\phi_g = \mathbf{E}_{\text{GNN}}(g)$
    for $t = 0, \ldots, T$ do
        Execute action $a_t = \pi(\phi_{o_t}, \phi_g)$, and obtain new state $s_{t+1}$ and observation $o_{t+1}$
        Embed new state and observation $\phi_{s_{t+1}} = \mathbf{E}_{\text{GNN}}(s_{t+1})$, $\phi_{o_{t+1}} = \mathbf{E}_{\text{CNN}}(o_{t+1})$
        Compute reward $r_t = r(\phi_{s_t}, \phi_g)$, and store transition $(\phi_{s_t}, \phi_{o_t}, \phi_g, a_t, r_t, \phi_{s_{t+1}}, \phi_{o_{t+1}})$ in $R$
    end for
    Generate virtual goals $(g'_1, g'_2, \ldots)$ and rewards $(r'_1, r'_2, \ldots)$ for each step $t$ and store in $R$ (w/ HER [44])
    Optimize actor $(\pi(\phi_{o_t}, \phi_g))$ using SAC with $o_t$ embeddings
    Optimize critic $(V(\phi_{s_t}, \phi_g), Q_1(\phi_{s_t}, \phi_g, a), Q_2(\phi_{s_t}, \phi_g, a))$ using SAC with $s_t$ embeddings
end for
```

- Encoder networks frozen in this phase
- MLPs with two hidden layers, standart SAC losses
  - Policy with image embedding
  - Value networks, and reward calculation with state embeddings
- $f_g(s_t) = \text{cos\_dist}(\varphi_s , \varphi_g) < E$
  - not reached the goal, −1 reward.
  - reaches the goal, +1 reward.
  - Episodes ends with −1 reward if goal condition not met
  - *Extra reward +0.1 when the agent moves any of the objects more than 2 cm*

Results

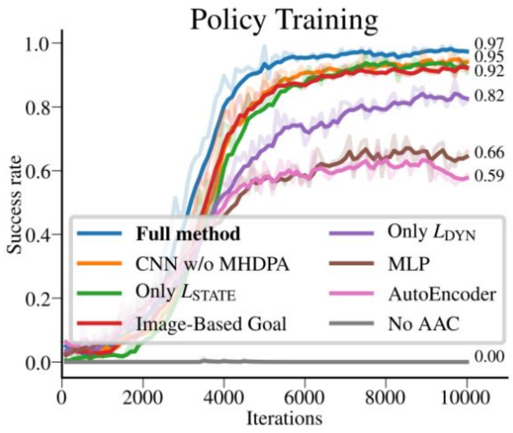
- How does the full method compare with alternative training formulations in simulation? (Sec. 5.1)
- What is the effect of domain randomization and using φ_s in policy training? (Sec. 5.2)
- How do the learned policies compare and generalize in real-world experiments? (Sec. 5.3)

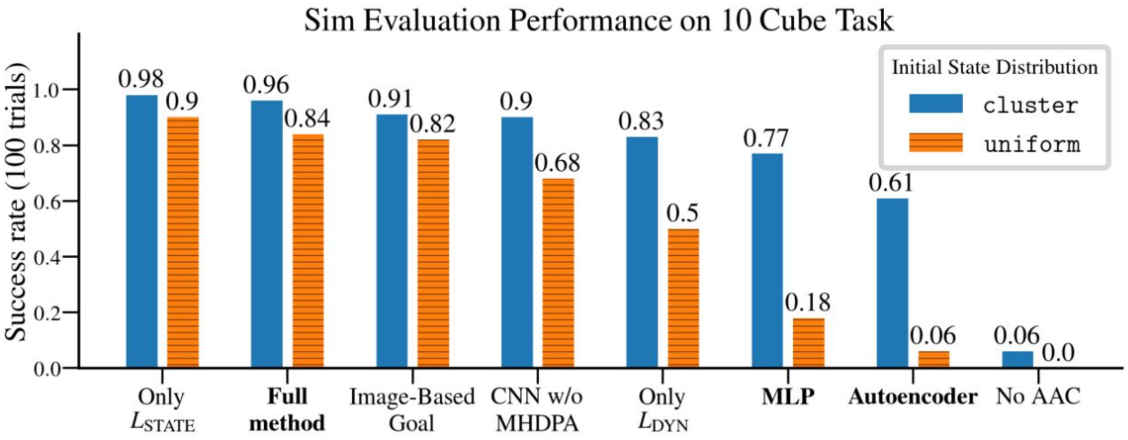
- Two sampling state
  - uniform: uniformly sampled
  - cluster: objects are sampled from 25cm x 25cm square areas on the table
  - initial state with both, goal state with cluster only
- Trained with 10 cubes, but evaluated on various objects in the real world
- Compared with
  - MLP (static network nx2 inputs)
  - Autoencoder (reconstruction loss)
  - image based goal instead of pose based goal
  - CNN without MHDPA
  - model trained with only L_{state}
  - model trained with only L_{dyn}



Results

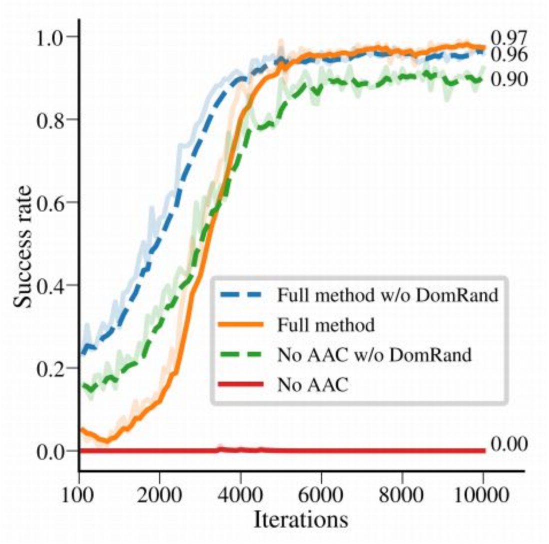
- How does the full method compare with alternative training formulations in simulation? (Sec. 5.1)





Results

- What is the effect of domain randomization and using φ_s in policy training? (Sec. 5.2)



Results

- How do the learned policies compare and generalize in real-world experiments? (Sec. 5.3)

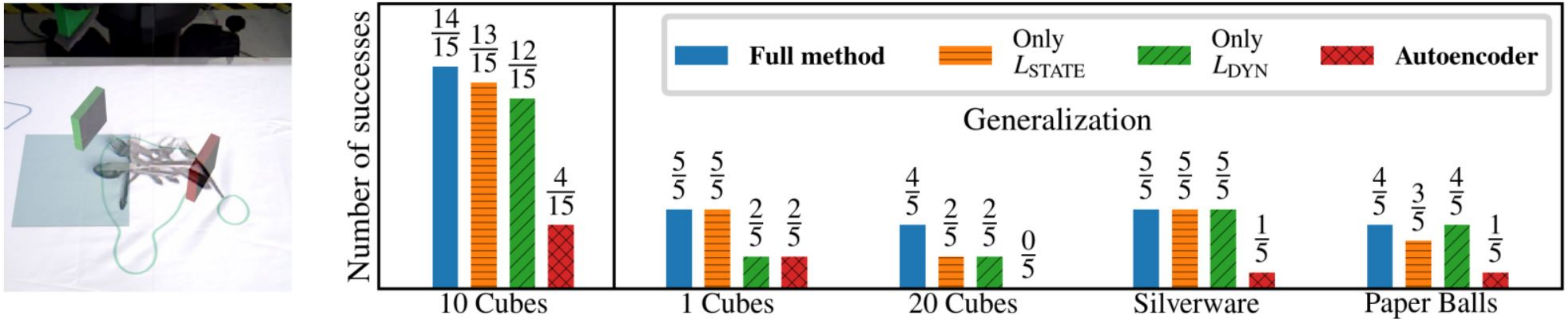


Figure 8: **Left:** Silverware task. **Right:** Real world evaluation results. We evaluate on 15 different start states and goal locations for the 10 Cube training task, and on 5 for the other generalization tasks. We terminate after successes or 15 push actions. We hold start states and goals constant across different methods. The full method produces the greatest performance and generalizability, especially outperforming the autoencoder approach.