CIS 520 Fall 2020, Project Report

Structured Representations for Visual Reinforcement Learning

Team I Just Want To Pass

- Edward Hu; Email: hued@seas.upenn.edu
- Pratik Kunapuli; Email: pratikk@seas.upenn.edu
- Jianing Qian; Email: jianingq@seas.upenn.edu

Abstract

Visual reinforcement learning, where policies are learned over the image space, are a promising direction for acquiring general robotic skills. Due to the high dimensionality of images, researchers have investigated learning policies over mid-level image representations to make learning tractable. In this work, we investigate how a structured latent state representation can affect RL performance on robotic benchmarks in the OpenAI Gym. We evaluate the reinforcement learning performance of policies using either learned keypoints, true keypoints, or true joint angles on the HalfCheetah and Walker2d continuous control tasks, and find learned keypoints to be competitive with conventional state formats for policy learning.

1 Motivation

Model-free reinforcement learning has shown promising results for solving various robotic control tasks in recent years. However, such policies usually use predefined states, raw pixel representations or unstructured learned latent state representations as input, limiting their applicability and interpretablility. Thus, we propose to use unsupervised keypoint detection methods to learn a structured latent keypoint representation for reinforcement learning tasks, and empirically evaluate the performance on robotic locomotion tasks. In particular, we evaluate PPO on keypoints rather than pixels, and investigate the potential of keypoint representations for improving sample efficiency and asymptotic performance for RL methods operating on images.

2 Related Work

2.1 Unsupervised Keypoint Detection

Keypoint is a popular representation, in terms of video prediction and video understanding tasks. Recent unsupervised keypoint learning methods have shown competitive performance in computer vision tasks with supervised methods. [1] uses an autoencoder structure and imposes induction bias by reconstructing the original input image. On top of this method, [2] also couples the unsupervised network with a latent dynamics model. Similarly, [3] adopt the same reconstruction method, but has an additional refinement network to further refine the predicted keypoints.

2.2 Reinforcement Learning from Pixels

Reinforcement learning has seen success in learning control policies from predefined state representations, such as assuming access to oracle object poses, joint velocities, etc [4, 5, 6]. However, such state representations require sensors to be present, and may not capture all task relevant information. Images on the other hand, are a versatile representation capable of capturing large amounts of information about the environment. Therefore, reinforcement learning from pixels [7] is promising due to the potential for policies trained on images to better generalize than policies trained on predefined states formats.

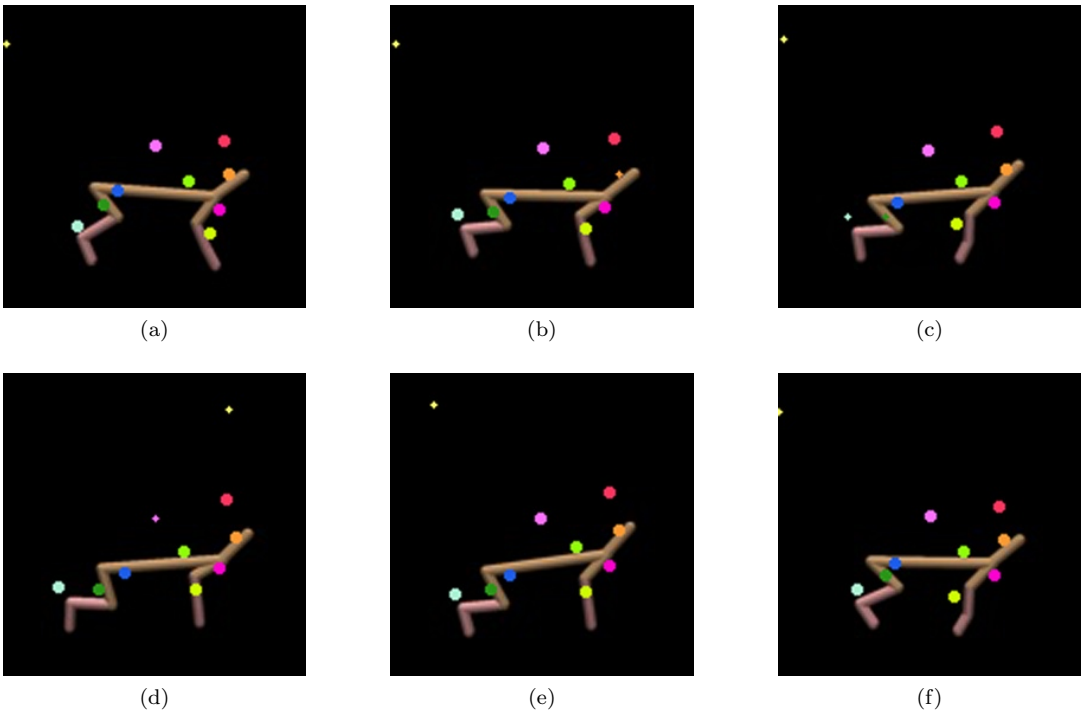


Figure 1: Keypoint detection on HalfCheetah environment. The color of each keypoints represent its index throughout the dataset. The size of the keypoints represent the confidence predicted by the network. With larger keypoints, the network is more confident that this point represent something meaningful.

3 Data Set

For our experiments, we intend to use the OpenAI Gym robotics environments [8]. Specifically, we intend to use some environments which are interfaced with a third party physics and contact simulation environment, MuJoCo [9]. This environment offers a number of continuous controls tasks with very fast, realistic physics simulation in a variety of simulation environments where things like background image, lighting, and models used can be controlled. The environments can be seen on OpenAI Gym's Website. Two of the environments we intend to use are the **HalfCheetah** and **Walker2d** environments. The HalfCheetah environment contains a 2D cheetah robot, and the reward is given by the horizontal distance the cheetah is able to cover. The Walker2d environment contains a 2d bipedal robot and the reward is given by the horizontal distance the robot is able to walk. The environments differ in the size of the action spaces associated with them, and the models between the environments also differ, but the camera viewpoint in each of the environments is the same. The images for the visualization of the environment are in the form of $128 \times 128 \times 3$, where there are 3 color channels (RGB) and each one is 128 pixels wide and 128 pixels high. Since this is a reinforcement learning task, there output of the models depend on the size of the action space which is dependent on the particular environment used. The action space for both the HalfCheetah and the Walker2d environments is 6, which represents 6 joint torques to be applied to the MuJoCo model. More details for each environment is given in the experiment section 6.2.

4 Problem Formulation

4.1 Unsupervised Keypoint Detection

For unsupervised keypoint detection, we use an autoencoder network to learn the keypoint representation. The input of the network is a single image frame $x_i^a \in \mathbb{R}^{h,w,3}$ that we want to detect keypoints on. The

information bottleneck in this autoencoder network is to output k heatmap, each representing a possible keypoint location. Then the decoder of this network is supposed to reconstruct the original image, only from these k heatmaps. Thus we are forcing the network to capture essential information only by these k heatmaps. Lastly, we train this network with and $L2$ reconstruction loss between the reconstructed frame and the original frame.

4.2 Model-Free Reinforcement Learning

Reinforcement learning problems represent a class of machine learning methods where algorithms are used to train an agent interacting within its environment. These problems are often represented as Markov Decision Processes (MDPs), where the environment is captured by the tuple $\{S, A, R, P, \rho_0\}$. In this formulation, S is the set of all valid states, A is the set of all valid actions for the agent to take, $R : S \times A \to \mathbb{R}$ is the reward function with $r_t = R(s_t, a_t, s_{t+1})$, $P : S \times A$ is the transition function where $P(s'|s, a)$ is the probability of transitioning to state s', and ρ_0 is the initial state distribution. Model-free reinforcement learning is a family of algorithms to solve reinforcement learning problems without the explicit use of a learned model in the learning process. Methods in this family learn a policy $\pi_\theta(a|s)$ which take a state s as input and produce either a stochastic probability over the action space or deterministic action to take. This policy π is parameterized by θ, and policy optimization is concerned with choosing the best θ such that the performance objective $J(\pi_\theta)$ is maximized. Often, this involves learning some $V_\phi(s)$, which is an approximation of the value function $V^\pi(s)$, parameterized by ϕ. Often, these parameters π and ϕ are represented as neural networks.

5 Methods

5.1 Unsupervised Keypoint Detection

Similar to [1], we designed our keypoint detection network to have an encoder Φ consists of convolutional layers. The input of the network would be one image $x_i^a \in \mathbb{R}^{h,w,3}$, representing the anchor frame x_i^a at time t. We want the encoder to capture the structures of the scene, in the format of K keypoints. This is done in three steps. First the encoder maps from image to K heatmaps:

$$\Phi(x) = Y, Y \in \mathbb{R}^{h',w',K}, Y = [Y_1, \ldots, Y_K]$$

where (h', w') is the size of one resulting heatmap. Next, each heatmap is normalized into a probability distribution using Softmax:

$$P_k = \frac{e^{Y_{k,i,j}}}{\sum_{i \in h', j \in w'} e^{Y_{k,i,j}}}$$

Next, each keypoint location u_k is extracted from these heatmaps P_k as the expected value

$$u_k = (\frac{\sum_{i \in h'} i * P_k[i,:]}{\sum_{i \in h'} P_k[i,:]}, \frac{\sum_{j \in w} j * P_k[:,j]}{\sum_{j \in w} P_k[:,j]}), k \in K$$

In the end we replace these keypoints u_k with 2D Gaussian functions centered at u_k with a fixed variance σ

$$\Phi_k(x) = exp(-\frac{1}{2\sigma^2}||x - u_k||^2)$$

$\Phi_k(x)$ will have the size of $\mathbb{R}^{H*W*K}$. Lastly we feed $\Phi_k(x)$ into a decoder network. This decoder network is supervised with a reconstruction loss of the original input image. Our implementation is based on an open source codebase [1].

[1] https://github.com/DuaneNielsen/keypoints

5.2 PPO: Proximal Policy Optimization

Proximal Policy Optimization (PPO) is a model-free reinforcement learning method that has seen success in many controls tasks [10]. PPO is an on-policy method that iteratively improves the policy over time using first-order methods. Concretely, policy update for PPO is done as a greedy search:

$$\theta_{k+1} = \arg\max_{\theta} \mathbb{E}_{s,a\sim\pi_{\theta_k}}[L(s, a, \theta_k, \theta)]$$

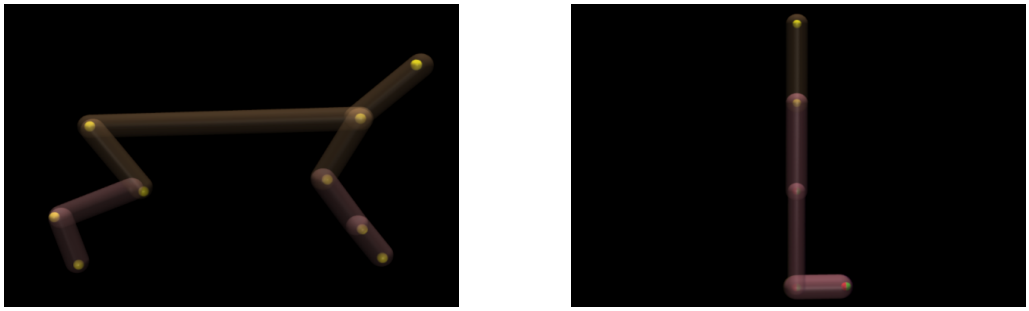
where the objective function is given by:

$$L(s, a, \theta_k, \theta) = \min[\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s, a), g(\epsilon, A^{\pi_{\theta_k}}(s, a))]$$

such that:

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A, & A \geq 0 \\ (1-\epsilon)A, & A < 0 \end{cases}$$

This clipped formation of PPO ensures that the policy iteration at every step does not vary significantly (given by ϵ) between iterations. A is the advantage calculated for the state and action under policy π_{θ_k}. We used an open source implementation of PPO [2] for our experiments.



(a) (b)

Figure 2: True 3D keypoints placed at the joints of each robot.

5.3 Baselines

In order to evaluate the effectiveness of the learned keypoint representations we used two different baseline methods. The first of which represents an oracle approach, where the ground truth joint angles were fed into the reinforcement learning algorithm. The second approach represented a visual reinforcement learning task, where the ground truth key points were fed into the reinforcement learning algorithm from the image of the agent. In order to generate the ground truth keypoints, we used the position of the joints of the MuJoCo model in 3D space. An example of the agents and the ground truth keypoints labeled over the agent can be seen in figure 2. From the true position coordinates of the keypoints it is possible to perform inverse kinematics and obtain the ground truth joint angles so both the ground truth keypoint and ground truth joint angle baselines are fair for the represented task.

6 Experiments and Results

In the experimental section, we aim to answer the following questions: (1) Can unsupervised keypoint learning algorithms extract meaningful keypoints from the scene of interest, and (2) Are learned keypoints competitive with conventional state representations for reinforcement learning? To answer such questions, we evaluate the visual reinforcement learning pipeline in two experimental phases. In the first phase, the quality of the learned keypoints are evaluated through both qualitative and quantiative measures. Next, the learned keypoints are used as the input for an RL policy in two robotic continuous control tasks.

[2] https://github.com/youngwoon/robot-learning

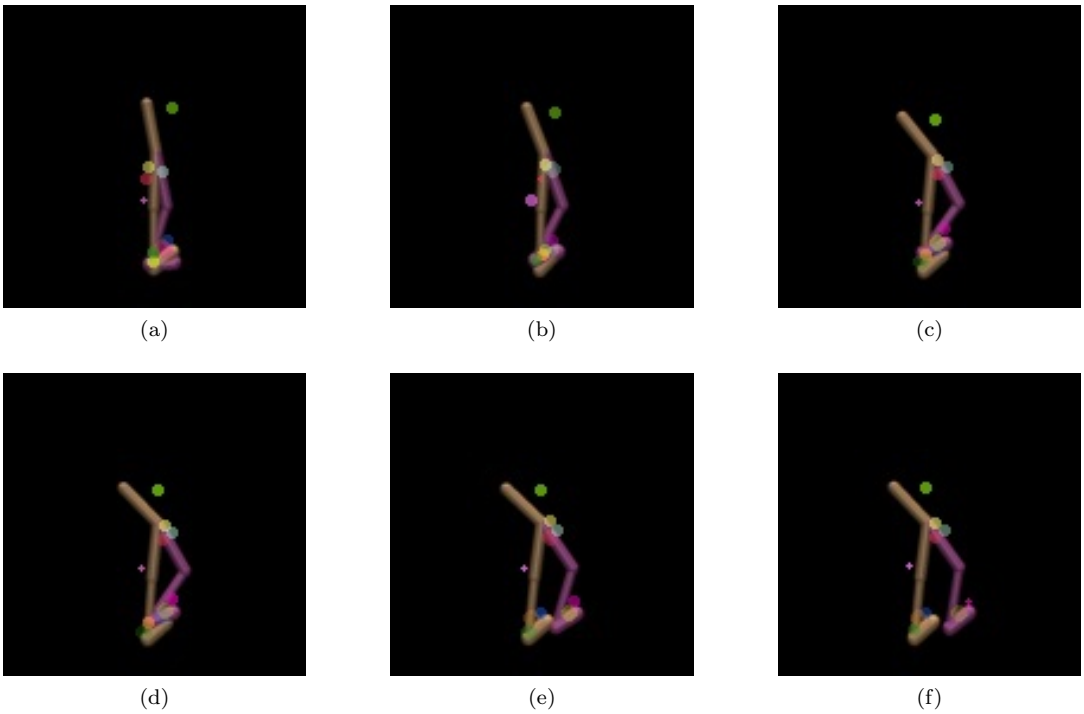


Figure 3: Keypoint detection on Walker2D environment.

6.1 Unsupervised Keypoint Detection Experiment

For the unsupervised keypoint detection experiments, we train our keypoint detection network on two dataset: Walker2D and HalfCheetah. For both of these dataset, we collected 10k images of the agents moving in the environments, based on a trained RL agent using SAC algorithm. To ease the training of our keypoint detection network, we make the background black during data collection. Since it is an unsupervised method, it doesn't require any labeling after the data collection. We choose $k = 10$ to be the number of keypoints we predict for each input images. The keypoint detection network adopts the common VGG16 network structure. We use Adam as the optimization algorithm with 1e-4 learning rate.

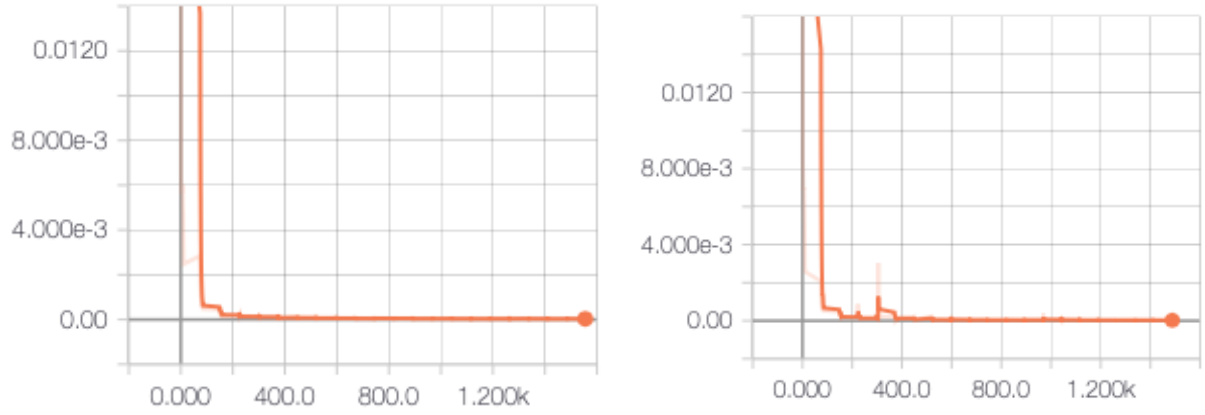
6.1.1 Qualitative Results on Keypoint Detection

We report the results from our trained keypoint detection algorithm here. We overlay learned keypoints on the original images, and the coloring of the keypoints are consistent throughout the entire dataset: each color correspond to the same keypoint in one of the ten keypoints learned. We can see in figure 1 and 3 the working keypoint detection encoding. This keypoint detection encoding is trained in an unsupervised manner as described in Section 5.1. Note that while the keypoints may seem arbitrarily scattered in the scene, they in fact track various joints of the agent as a result of the reconstruction loss. We can see that while the agent has varying pose throughout the environment, the keypoint tracking is robust such that if a particular joint is visible, the same keypoint is being mapped to the same part of the agents' body across the images. This consistency in keypoints would provide an efficient low-level representation for policy learning.

6.1.2 Quantitative Results on Keypoint Detection

Since we train the keypoint detection network in an unsupervised fashion, we don't have any ground-truth label for keypoints. Thus it's hard for us to directly evaluate the quantitative quality of learned keypoints. Instead we look at the reconstruction loss during testing. Recall that the training objective is to be able to reconstruct the input image from the learned keypoints, then if the reconstruction loss is low, it means that our learned keypoints do capture all the useful information in the current image, which is the object poses.

Fig 4 Shows the test reconstruction loss for both of the environment. We see that the reconstruction error converges to below 1e-4 in both environments. This means that the learned keypoints at the end of training are sufficient representations of the entire image, which indicates that it should be useful for any downstream control tasks.



(a) (b)

Figure 4: Test reconstruction loss for HalfCheetah and Walker2d

6.2 Reinforcement Learning Results

Once the keypoint network is trained, the next step is to evaluate the effectiveness of the keypoint representations for reinforcement learning. The main metric for comparison is the mean episodic reward, since a better performing agent will collect more reward in an episode. The max episode length is 1000. For each state representation type, we train the PPO policy for 5 million environmental steps. See the following table for the full hyperparameters.

Table 1: PPO Hyperparameters

Parameter	Value
Learning rate	3e-4
Discount factor (γ)	0.99
Number of hidden layers	2
Number of hidden units	256
Minibatch size	256
Nonlinearity	ReLU
Workers	3
Rollout steps per update	1333
Clipping value	0.2
Entropy penalty	1e-4
Observation normalization	True

6.2.1 HalfCheetah

This task requires the policy to control a 2d cheetah robot to sprint as far as possible. The joint angle observation space contains 8 dimensions for the joint angles, and 9 dimensions for the joint velocities for a total of 17 dimensions. The learned keypoint observation space contains 10 2D keypoints (20 dimensions), and 9 dimensions for the joint velocities for a total of 29 dimensions. The true keypoint observation space contains 9 3D keypoints (27 dimensions), and 9 dimensions for the joint velocities for a total of 36 dimensions.

(a) HalfCheetah

(b) Walker2d

Figure 5: The mean and standard deviation of the episodic reward for policies trained on different state representations in the environments. Each state representation was trained over 5 random seeds to account for PPO's variance in performance, and the shaded regions reflect the standard deviation. Videos of the policies and interactive graphs can be found on https://wandb.ai/pal/tckn-rl/reports/CIS520-RL-Experiments--VmlldzozNDIwNzE.

The action space is joint torque and is 6 dimensions. The reward function is the forward velocity of the robot, as well as a control penalty to discourage extreme actions.

6.2.2 Walker2d

The policy must control a 2d walker robot to walk as far as possible. The joint angle observation space contains 8 dimensions for the joint angles, and 9 dimensions for the joint velocities for a total of 17 dimensions. The learned keypoint observation space contains 10 2D keypoints (20 dimensions), and 9 dimensions for the joint velocities for a total of 29 dimensions. The true keypoint observation space contains 8 3D keypoints (24 dimensions), and 9 dimensions for the joint velocities for a total of 33 dimensions. The action space is joint torque and is 6 dimensions. The reward function is the forward velocity of the robot, a control penalty to discourage extreme actions, and a upright penalty to discourage leaning too far.

6.2.3 Learned Keypoints RL Performance

In the HalfCheetah environment, the learned keypoints are actually the optimal state format. It has lower seed variance, and higher mean reward than even the true joint state, which is the default observation space. The learned keypoints and the true joint state policies learn a dynamic sprinting behavior, which is promising. Interestingly, the true keypoints state underperformed by a wide margin of 2000 reward. When we inspected the videos, 2 out of the 5 true keypoint policies learned to flip over, a suboptimal behavior where the cheetah gains a large initial forward velocity reward, but low overall reward.

Next, in the Walker2d environment, the trends are more normal. The true joints and true keypoints both perform well, with the true keypoints having marginally higher mean reward as shown in Table 2. The learned keypoint performs worse than the true keypoints, but is still competitive with true keypoint and true joints as seen in the training curve 5b. When we watch the videos of the Walker robot, we notice a trend between the keypoint (learned / true) and joint angle states. The true joint policies have more flexible gaits, bending every joint, while the keypoint policies learn a stiff legged walking motion. Despite the true keypoint and true joint policies reaching similar rewards, the true joint state is perhaps more conducive for the policy to learn the kinematics of the robot, and hence have a more realistic walking behavior.

	HalfCheetah	Walker2d
True Joints	4085 ± 1017	3327 ± 1847
True Keypoints	2126 ± 1551	3505 ± 1812
Learned Keypoints (ours)	4346 ± 534	2844 ± 1907

Table 2: Mean episodic reward and 1 standard deviation at the end of 5 million environment steps for policies trained on each state representation. The learned keypoints outperform the true keypoints in the HalfCheetah environment, but are marginally worse than the true keypoints in the Walker2d environment.

7 Discussion and Conclusion

From the HalfCheetah and Walker experiments, we have seen that the type of state representation can result in different behaviors. Specifically, in the HalfCheetah experiment we showed that the learned keypoint representation performs better than the two other baseline methods, but in the Walker experiment we saw that the learned keypoint representation performs slightly worse than the other two baseline methods. We believe that keypoint, or positional based representations, may excel at object manipulation tasks due to object manipulation's focus on object positioning. On the other hand, the learned keypoints policy in the HalfCheetah environment learned dynamic sprinting behavior. This makes investigating the relationships between the final performance, given task, and state representation a promising direction to explore.

Next, the unsupervised keypoint network already outputs confidence maps for each keypoint. Currently, the policy cannot take advantage of the uncertainty information of the keypoints. We tried feeding in the confidence score for each keypoint as an additional feature, but this made the RL policy perform worse. Investigating how to properly incorporate keypoint uncertainty will improve the RL performance.

Finally, we trained the keypoint detector on black background scenes, which is an unrealistic assumption. Improving the keypoint detector's robustness to visual input is necessary for us to move to real world tasks.

We have demonstrated the effectiveness of unsupervised keypoints for visual reinforcement learning on the HalfCheetah and Walker2d tasks. First, we show that unsupervised keypoint detection can recover meaningful keypoints that reconstruct the scene. Next, the RL experiments suggest that learned keypoints are competitive with conventional state representations such as true joint angles and true keypoints in continuous control tasks. Incorporating uncertainty into policy learning and improving the keypoint detector are exciting directions for future work.

References

[1] Tomas Jakab et al. "Conditional Image Generation for Learning the Structure of Visual Objects". In: *CoRR* abs/1806.07823 (2018). arXiv: 1806.07823. URL: http://arxiv.org/abs/1806.07823.

[2] Matthias Minderer et al. "Unsupervised Learning of Object Structure and Dynamics from Videos". In: *ArXiv* abs/1906.07889 (2019).

[3] T. Kulkarni et al. "Unsupervised Learning of Object Keypoints for Perception and Control". In: *NeurIPS*. 2019.

[4] Timothy P Lillicrap et al. "Continuous control with deep reinforcement learning". In: *arXiv preprint arXiv:1509.02971* (2015).

[5] Sergey Levine et al. "End-to-end training of deep visuomotor policies". In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 1334–1373.

[6] Youngwoon Lee et al. "Composing Complex Skills by Learning Transition Policies with Proximity Reward Induction". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=rygrBhC5tQ.

[7] Pierre Sermanet et al. "Time-contrastive networks: Self-supervised learning from video". In: *IEEE International Conference on Robotics and Automation*. 2018, pp. 1134–1141.

[8] Greg Brockman et al. "Openai gym". In: *arXiv preprint arXiv:1606.01540* (2016).

[9] Emanuel Todorov, Tom Erez, and Yuval Tassa. "Mujoco: A physics engine for model-based control". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 5026–5033.

[10] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).