Visual Latent Space RL Stacking

Qingkai Pan
Columbia University
New York, NY 10025
qp2134@columbia.edu

Zhen Bang Tan
Columbia University
New York, NY 10027
zt2291@columbia.edu

Jobin Binoy George
Columbia University
New York, NY 10027
jb4512@columbia.edu

1 Introduction and Problem Setup

Deep reinforcement learning methods are predominantly used to help robots and other agents learn particular tasks, especially for object manipulation. Grasping and placing have been the most commonly observed manipulation tasks in both virtual and real worlds, where useful general RL approaches and potential challenges have been thoroughly explored and analyzed in recent years [11] [12]. Object stacking not only involves grasping and placing but also interactions among different objects, which inevitably makes this task more challenging and difficult to generalise. In recent years, there have been progress made to help alleviate the obstacles and shortcomings of the stacking tasks, but most of these approaches are based on deep learning and computer vision without a reinforcement learning setup, where stacking itself is a task involving significant time horizon and it's intuitive and subtle to postulate that deep reinforcement learning is one of the optimal frameworks to generalise the learning process.

Latent space representation is one of the most essential and practical methods in machine learning and deep learning, where it has the capability to play an significant role in pivotal information extraction and noise filtering in dataset so that the learning efficiency and stability would be significantly boosted. There are also several works that utilizes latent space representation within an RL framework, but none of the methods targets at robotic stacking and most of them are apparently not as challenging and rusty as our task.

In this paper, we intend to learn an actor critic model incorporated with an effective latent space representations from raw image data for the object stacking task so that our method can generalise cross different object arrangements with significant improvement in sample efficiency in contrast to end-to-end RL approach.

2 Literature Review

Actor critic methods The actor critic paradigm was originally proposed by [8], which was originally intended for trajectory based optimization of Markov decision processes. Because of its robustness and high rate of convergence, this learning paradigm has been widely used and adopted in other literature, where the job of the critic is to compute the approximation of projection of the value function to a lower dimensional subspace to update the policy in an approximate gradient direction. Since then, there have been many extensions to this highly generic learning paradigm, where Soft Actor-Critic(SAC) [6] and Proximal Policy Optimization(PPO) [15] are the most commonly used extensions in academia and industries. For this project in particular, we are interested in incorporating SAC with latent space representation and a recently proposed further extension of SAC called Stochastic Latent Actor-Critic(SLAC)[10], which separates the learning of planning and visual recognition via training the agent using induced latent space.

Representation learning in RL Using reinforcement learning in high dimensional data can be arduous and tedious, where the training may not make significant progress without excessive hyperparameter tuning. Indeed, conventional RL methods are demonstrated to be restricted by computational and data efficiency issues without a well-learned representation [16]. Therefore, it's not surprising that numerous work have been done in the past that utilises latent space representations, such as [18] [13] [5] [4]. Though many of the authors claimed that their method has the potential to generalise across various robotic tasks, none of them explicitly experimented their method on object stacking, which is certainly more complicated and difficult to control. Our method exploits effective latent space representation for this stacking task in particular, which is expected to significantly outperform prior works in terms of sampling efficiency and generalisation across different object shapes and arrangements.

Robotic Stacking The most recent work on object stacking is produced by Deepmind [9], where they managed extend the stacking task to different object shapes, where most earlier works such as [14] are only demonstrated to be working for cubic blocks. Another significant work that handles object stacking of different shapes is [3]. However, the shapes of the stones don't differ significantly on the whole and all the stones have wide support during experimentation. Our

approach not only aims to generalise across different object configurations but also demonstrate good performance across different object arrangements.

3 Methods and models

3.1 Reinforcement Learning

An reinforcement learning agent is embedded in an environment in such a way that it can discriminate state space S and implement actions from the action space$\mathcal{A}$[7]. $\forall s \in S$, the agent can choose an available action $a \in \mathcal{A}$ and gains a immediate reward $r(s, a)$ which follows a distribution R given $s \in S$ and $a \in \mathcal{A}$. The environment inherits transition probabilities $P(s', r|s, a)$ that determines the probability of the agent going to s' and gains r by taking a from s. Now the objective is to learn an optimal policy where the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decisions regardless of the initial state and decisions.[1] Hence, the agent takes a trajectory with joint probability distribution given by the optimal policy π^* so that it maximises the expected accumulative reward $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$, where the agent starts at s_0 and $\gamma \in (0, 1]$ is the discount factor. For this paper, the transition probabilities and reward functions are given by the stacking environment, where good rewards are gained by moving towards the object, picking up the object as well as successfully placing it on top of the chosen object to be stacked on.

3.2 Stochastic Latent Actor-Critic

Stochastic Latent Actor-Critic(SLAC)[10] is an enhanced method on top of Soft Actor-Critic(SAC)[6] that extends to partially observed Markov Decision Process(POMDP). The intuition is to maximize $p(s_{1:\tau+1}, \mathcal{O}_{\tau+1:T}|a_{1:\tau})$, which is joint likelihood of observation (state) space and the future optimality variables $\mathcal{O}_t$ that indicates whether time step t is optimal conditioned on past actions given by maximum entropy policies. The evidence lower bound(ELBO) of the logarithm of the joint likelihood is computed from the posterior of the factorization of from the variational distribution $q(z_{1:T}, a_{\tau+1:T}|s_{1:\tau+1}, a_{1:\tau})$, where z_t is the learned latent space at time step t, which is derived to be:

$$\begin{aligned}\mathbb{E}_{z_{1:T}, a_{\tau+1:T} \sim q}\Bigg[& \sum_{t=0}^{\tau} (\log(p(s_{t+1}|z_{t+1})) - D_{KL}(q(z_{t+1}|x_{t+1}, z_t, a_t) \| p(z_{t+1}|z_t, a_t))) \\ & + \sum_{t=\tau+1}^{T} (r(z_t, a_t) + \log(p(a_t)) - \log(\pi(a_t|s_{1:t, a_{1:t-1}})))\Bigg]\end{aligned} \tag{1}$$

Now SLAC learns to maximize ELBO, where the first and second summations correspond to maximization the likelihood of the observations via the latent variable model and the maximum entropy RL objective respectively. Consequently, we end up with training 3 networks including parameterized policy π_ϕ, critic Q_θ and latent variable model M_ψ. This frame elegantly combines latent space representation learning and maximum entropy RL, which is claimed to have higher sample efficiency than prior RL methods, so that SLAC would achieve desirable performance in expectation.

$$J_\pi(\phi) = \mathbb{E}_{z_{1:\tau+1} \sim q_\psi}\left[\mathbb{E}_{a_{\tau+1} \sim \pi_\phi}[\sum_{t=\tau+1}^{T} \alpha \log(\pi_\phi(a_t|s_{1:\tau+1, a_{1:\tau}})) + Q_\theta(z_{\tau+1}, a_{\tau+1})]\right] \tag{2}$$

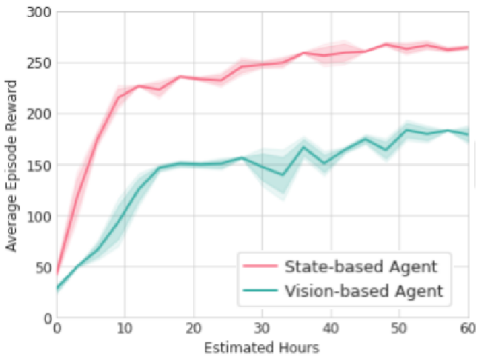
,

$$J_Q(\theta) = \mathbb{E}_{z_{1:\tau+1} \sim q_\psi}\left[\frac{1}{2}(Q_\theta(z_\tau, a_\tau) - (r_\tau + \gamma V_{\bar{\theta}}(z_{\tau+1})))^2\right], \tag{3}$$

$$J_M(\psi) = \mathbb{E}_{z_{1:\tau+1} \sim q_\psi}\left[\sum_{t=0}^{\tau} (\log(p_\psi(s_{t+1}|z_{t+1})) - D_{KL}(q_\psi(z_{t+1}|x_{t+1}, z_t, a_t) \| p_\psi(z_{t+1}|z_t, a_t)))\right]. \tag{4}$$

4 Experiment Setup

Our virtual dataset is extracted from rgb stacking environment [9] constructed from [17]. The shapes used for stacking are split into triplets that consist of sufficiently dissimilar geometric shapes, as provided by the environment. The



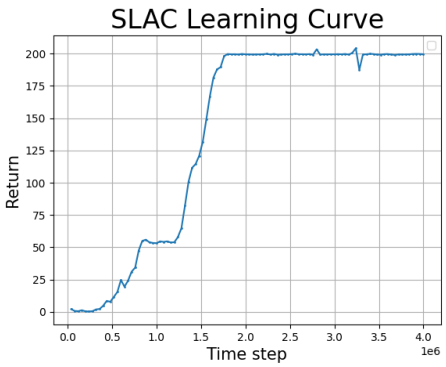


Figure 1: Left: Comparison of the State-Based MPO vs Vision-Based MPO Learning Curves; Right: SLAC Average Return Learning Curve

objects within the triplets are color coded between Red, Green and Blue to simply inform the agent which object needs to be stacked on top and which remains on the bottom. The final object acts as a distractor.

We are using two image observations from cameras placed around the environment (two front cameras) positioned appropriately to capture the required amount of spatial information. Although the MPO model, that we compare our implementation against, used proprioception data (joint angles, velocities, torques, gripper/ wrist poses) in addition to the image input, we opted for solely an image trained approach and use no additional true-state input as it would have little to no effect when mapped in the latent space. Therefore, as we further explain in next section, we use less state information than the approach being compared against.

We implement the SLAC with pixel inputs (64 x 64 x 3) that is resized from the concatenation of two front camera images, where both of them have the dimension of (128 x 128 x 3). The image is then resized to (64 x 64 x 3). We use a Gaussian policy because the action space is continuous. The latent dimensions for z1 and z2 are 32 and 256 respectively. The action space is a 5-dimensional continuous vector consisting of the Cartesian position of the gripper, the rotation with respect to the z-axis and the gripper width.

5 Results

As shown in Figure 1, SLAC is able to reach and maintain a high accumulative reward of 200 before 2 million steps, where the visual MPO reached 180 in 5 million steps(Every 1 million steps take approximately a hour). It's evident that SLAC has demonstrated obvious sample efficiency boost. Note that we didn't embed any physical information within the environment such as gripper poses, sawyer joint velocity etc. into our state space. In fact, we can achieve similar or better generalization with less state information. Figure 2 gives a clearer visual comparison between the performance of our approach and the state-based MPO, where both models are trained on triplet 1 and evaluated on triplet 4 and 5. For triplet 4, state-based MPO does pick up and move the red cube towards the destination, but the gripper doesn't drop it and remains hanging the red cube above the blue cuboid instead. Our approach does make the effort to drop the red cube but it unfortunately place it in a good angle where it eventually falls to the horizon. The distinction becomes clearer in dealing with triplet 5, where state-based MPO continuously fails to accomplish the basic task to pick up the red object, where SLAC picks it up with ease despite the eventual drop after clashing with the blue object. Figure 3 further justifies our claims numerically, where SLAC gives higher returns in most triplets even for state-based MPO and the results in the visual-based MPO column are significantly worse than SLAC especially in triplets different from the one being trained on, accentuating the enhanced and robustness of our approach.

Buffer size has large influence on the returns. We also tried training SLAC on a smaller buffer size of 10000 instead of 100000. However, this resulted in a significant decrease in performance from an average return of 200 to an average return of 10 after 4 million steps. We hypothesize that the interaction dynamics is very complex and the agent needs to understand multiple such interactions to successfully perform the task, thus requires a large buffer to store those trajectories.

Curiosity Driven Exploration. Naive Random Network Distillation(RND) [2] adds curiosity reward for unseen images to motivate better exploration. However, it gives neither faster convergence nor better final performance.

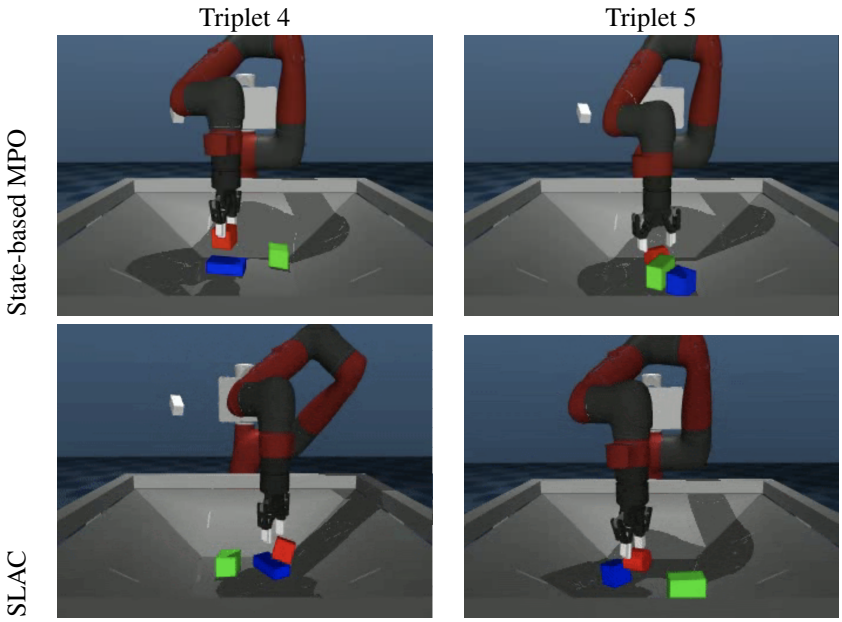


Figure 2: Comparative simulation of models trained on triplet 1

Env/Method	MPO (State Only)	MPO (Vision Only)	SLAC (Vision Only)
Triplet 1	215.25	180	199.59
Triplet 2	30.19	20.46	24.38
Triplet 3	27.53	17.19	60.17
Triplet 4	87.54	53.03	98.93
Triplet 5	54.01	31.92	73.19

Figure 3: Evaluation Return Table(Avg over 5 seeds)

6 Discussion

The outstanding performance by SLAC can be explained by its better generalisability to unseen states(from different triplets) as well as the adoption of latent space learning which is a legitimate way to improve sample efficiency as we claimed and justified in both theoretical and practical aspects. An interesting observation is that our method's performance plateaus at around 200 after 2 millions steps, and cannot reach the theoretical maximum. It is unclear whether the limit is caused by model capacity or the method itself is insufficient to complete the task. Furthermore, it is interesting to note that the trained agent greatly benefits when representation learning is separated from task learning. By explicitly learning latent representations, reinforcement learning from high dimensional images is accelerated especially for long-horizon continuous control tasks.

A reasonable explanation for RND not being helpful in learning is that the camera shots are all taken in the front of the gripper, where similar image sets might not correspond to similar gripper positions or strategies, camera shots from different points of view would induce a more comprehensive information capture for RND to play its role. Another possible reason is that the physical construction is very structured in this environment, where curiosity exploration of unseen states without any preference would not generate the correct inductive bias. Another possible adjustment is to feed the latent states into the RND network instead of raw image sets since the latent space extracts the most important features from the raw observations in theory so that curiosity exploration via the learned representation is potentially more effective.

Due to the time and computational resource limits, we have only experimented with training on triplet 1 and test generalization to other triplets. In future, we could attempt training on different triplets and generalize to different triplets to explore what parametric shapes of objects facilitates faster learning and better generalization.

References

[1] Richard Bellman. The theory of dynamic programming. Technical report, Rand corp santa monica ca, 1954.

[2] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[3] Fadri Furrer, Martin Wermelinger, Hironori Yoshida, Fabio Gramazio, Matthias Kohler, Roland Siegwart, and Marco Hutter. Autonomous robotic stone stacking with online next best object target pose planning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2350–2356. IEEE, 2017.

[4] Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G Bellemare. Deepmdp: Learning continuous latent space models for representation learning. In *International Conference on Machine Learning*, pages 2170–2179. PMLR, 2019.

[5] Dibya Ghosh, Abhishek Gupta, and Sergey Levine. Learning actionable representations with goal-conditioned policies. *arXiv preprint arXiv:1811.07819*, 2018.

[6] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[7] Leslie Pack Kaelbling. Learning to achieve goals. In *IJCAI*, pages 1094–1099. Citeseer, 1993.

[8] Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014, 2000.

[9] Alex X Lee, Coline Manon Devin, Yuxiang Zhou, Thomas Lampe, Konstantinos Bousmalis, Jost Tobias Springenberg, Arunkumar Byravan, Abbas Abdolmaleki, Nimrod Gileadi, David Khosid, et al. Beyond pick-and-place: Tackling robotic stacking of diverse shapes. In *5th Annual Conference on Robot Learning*, 2021.

[10] Alex X Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *arXiv preprint arXiv:1907.00953*, 2019.

[11] Andrew Lobbezoo, Yanjun Qian, and Hyock-Ju Kwon. Reinforcement learning for pick and place operations in robotics: A survey. *Robotics*, 10(3):105, 2021.

[12] Marwan Qaid Mohammed, Kwek Lee Chung, and Chua Shing Chyi. Review of deep reinforcement learning-based object grasping: Techniques, open challenges and recommendations. *IEEE Access*, 2020.

[13] Ashvin Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. *arXiv preprint arXiv:1807.04742*, 2018.

[14] Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Wiele, Vlad Mnih, Nicolas Heess, and Jost Tobias Springenberg. Learning by playing solving sparse reward tasks from scratch. In *International Conference on Machine Learning*, pages 4344–4353. PMLR, 2018.

[15] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[16] Evan Shelhamer, Parsa Mahmoudieh, Max Argus, and Trevor Darrell. Loss is its own reward: Self-supervision for reinforcement learning. *arXiv preprint arXiv:1612.07307*, 2016.

[17] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[18] Albert Zhan, Philip Zhao, Lerrel Pinto, Pieter Abbeel, and Michael Laskin. A framework for efficient robotic manipulation. *arXiv preprint arXiv:2012.07975*, 2020.

A Appendix

A.1 More results

Videos are available on https://drive.google.com/drive/u/4/folders/1mAIxmBkVhwO-93WMJP1rNuXadtt7JfvL

A.2 Teammates and work division

All members contributed equally. Jobin: Setup JAX framework, dependencies and the trainer optimization functions. Zhen: Setup GYM wrapper for the env, and PyTorch framework. Alvin: Setup the RND method and PyTorch framework.

A.3 DeepMind needs to put more effort into their "open-source" work

So we don't need to fix bugs for them. Full Stop.