ASEN 6519 Final Project REPORT

Sai Chikine

TOTAL POINTS

(not graded)

QUESTION 1

1 Final Project Report

- 0 pts Correct

State Action Learned Embeddings (SALE) for PPO

Sai Chikine
Aerospace Engineering
CU Boulder
Boulder, USA
sai.chikine@gmail.com

I. Introduction

Model-free deep reinforcement learning algorithms have seen much success in recent years. They are often preferred to model-based methods, frequently thought of as offering more stable training. However, it is easy to see the theoretical benefits of model-based methods: by learning model dynamics, they can be much more sample efficient than model-free methods. In fact, model-free methods are often considered very sample inefficient [5]. Recent work in representation learning [1] offers options to increase sample efficiency in model free methods. One such option is the concept of State Action Learned Embeddings (SALE), which was applied by Fujimoto et. al. to TD3 to create the TD7 algorithm [1]. SALE learns the dynamics of a model by encoding them into a pair of neural network encoders, which are then used to provide additional information to the actor and critic networks within TD3/TD7. The authors show significant performance benefits to applying SALE to TD3, even without the additional components comprising TD7, suggesting learning dynamics information can improve training efficiency in model-free methods. However, SALE has not yet been applied to any other state-of-the-art model-free algorithms, so it is unclear how much benefit this approach when used outside of TD3/TD7. Therefore, this project proposes to implement SALE for PPO to evaluate its efficacy and, if so, under what conditions.

II. Background

A. PPO

PPO (Proximal Policy Optimization) is an extremely popular model-free deep reinforcement learning algorithm proposed by Schulman et. al. [2]. It is an actor critic method which incorporates a clipped surrogate loss function to restrict the size of policy changes in order to avoid unstably large training updates. The most relevant parts of PPO to consider for this project are its actor-critic structure, its use of GAE (Generalized Advantage Estimation), and its loss computation. In PPO, the actor is a neural network which maps states to actions, while the critic is a neural network which maps states to values. The actor's loss function is $L(CLIP)(\theta) = \hat{\mathbb{E}}\big[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\big]$; note that this requires estimation of the advantage $\hat{A}$. To do so, GAE is used [3], which estimates the advantage at time t as $\hat{A}_t = \sum_{l=0}^{\infty}(\gamma\lambda)^l(r_{t+l} + \gamma V(s_{t+l+1}) - V(s_{t+l}))$. In PPO, the advantage is computed recursively backwards after each episode, starting with the last action take in the episode and proceeding backwards, using the value function critic to compute each $V(s)$. Meanwhile, to train the critic, a squared-error loss is used: $L_t^{VF} = (V_\theta(s_t) - V_t^{\text{target}})^2$, where the V^{target} values come from the actual returns in each episode.

B. State Action Learned Embeddings

State Action Learned Embeddings (SALE) consists of a set of encoders which learn the transition dynamics of a system and feed the outputs of these encoders into policy and value function computations. The encoders consist of $f(s) = z^s$, which encodes the state into the *state embedding*, and $g(z^s, a)$, which encodes the state embedding and action into the *state-action embedding*. The encoders are trained concurrently with the policy and critic using the loss function $L(f, g) = (z^{sa} - |z^{s'}|_x)^2$, where the $|\cdot|_x$ denotes the stop-gradient operation (the next-state encoding should be held fixed when training the encoders). This loss function essentially tries to construct encoders which know how an action maps a state to its next action. In the TD7 algorithm, these embeddings are input to the Q and π networks, making them $Q(s, a, z^s, z^{sa})$ and $\pi(s, z^s)$, respectively, to provide transition information to them, leading to higher sample efficiency and, in most problems, higher achieved rewards [1].

III. Incorporating SALE into PPO

As can be seen in its policy loss function, PPO requires estimation of the advantage. To obtain this estimate, it uses GAE (Generalized Advantage Estimation) [3], which requires evaluation of the the value function network $V(s)$. On the other hand, SALE was conceived for use with TD3, which uses its critic network to learn the state-action value function $Q(s, a)$. Since $Q(s, a)$ aims to estimate the value of a particular state and action combination, it is intuitive to see how learning how that action affects the state would help $Q(s, a)$ obtain a better estimate. When applied to PPO, where there is no Q function as typically implemented, it is not immediately obvious how to incorporate SALE. In fact, there are multiple options, which can be broken up into whether to estimate $V(s)$ or $Q(s, a)$ within PPO.

A. PPO - State Value Critic

One option is to disregard incorporating the state-action value function $Q(s, a)$ into PPO, and simply feed the encoder outputs z^s and z^{sa} into the existing policy and value function

networks. The encoders are trained as in TD7, and the value function, instead of being $V(s)$, is $V(s, z^s)$ (PPO-V-ZS) or $V(s, z^s, z^{sa})$ (PPO-V-ZS-ZSA); Option 1 attempts to match the pure state input to $V(s)$, while Option 2 adds more explicit transition information. Note that since the computation of z^{sa} requires both the state embedding z^s and the action a, an action needs to be selected to compute $V(s, z^s, z^{sa})$. In this option, actions are selected according to the actual actions taken during training episodes. Since $Q(s, a)$ needs to incorporate both state and action information into a value computation, incorporating information about how actions affect states into Q makes intuitive sense. Upon first glance, adding this additional information to $V(s)$ does not present the same obvious benefit. However, the value function within PPO is used in the context of Generalized Advantage Estimation (GAE) , whose goal is estimate the *advantage* $A(s, a)$. Therefore, there may still be some merit to including encoded dynamics information in the input to the value network.

B. PPO - State-Action Value Critic

Alternatively, PPO could be modified to include an estimation of the state-action value function instead of the value function; its critic network would predict $Q(s, a)$ instead of $V(s)$. However, this presents issues for the GAE computation, which needs $V(s)$ to compute the advantage. One method to do this would be to set $V_t(s) \leftarrow Q_\theta(s_t, a_t)$ within the GAE loop, using the recorded actions during the training episode similarly to PPO-V-ZS-ZSA; this is denoted PPO-Q in this project. From here, PPO's policy and value function losses are unchanged, and, since we know have a critic network that ingests s and a, we may modify it to be $Q(s, a, z^s, z^{sa})$ as in the original SALE implementation; this is denoted PPO-Q-ZS-ZSA. However, notice that the value function $V(s)$ is defined in terms of $Q(s, a)$ as $V(s) = \max_{a \in A} Q(s, a)$, which does *not* match the above computation of V. Furthermore, since this project focuses on continuous action spaces, one cannot maximize Q over all actions to obtain V. However, one option to rectify this would be to maximize over *all actions actually taken in the episode*. With this scheme, within the GAE computation, $V(s) \leftarrow \max_{a \in \text{episode}} Q(s, a)$, where $Q(s, a)$ is computed via the critic network. This algorithmic option is denoted PPO-Q-MAX, and the associated option with added embeddings is denoted PPO-Q-MAX-ZSA.

IV. Numerical Experiment Setup

The goal of this project is to gain some insight into how SALE benefits (or hinders) PPO; this behavior will be determined by running each of the previously identified options on several continuous action-space test problems, which are: (a) Pendulum-v1, MountainCarContinuous-v0, CartPoleContinuous, and PlanarCR3BPLyap. Multiple algorithm options were tested on each of these problems, denoted *PPO-V*, *PPO-V-ZS*, *PPO-V-ZS-ZSA*, *PPO-Q*, *PPO-Q-ZS-ZSA*, and finally: *PPO-Q-MAX*, *PPO-Q-MAX-ZSA*; these are all detailed below. For each environment, the same seed and machine were used to run all algorithms.

A. Pendulum-v1

This environment is part of the well known classic control problem suite in Farama's gymnasium [4]. The problem task consists of balancing a pendulum in an inverted position. The action space consists of the angular velocity of the pendulum, $\dot{\theta}$, and the position of the pendulum's end, which is assumed to be 1 meter in length (given by $cos(\theta), sin(\theta)$. The action space is continuous and consists of a single torque τ applied to the pendulum, which can range between -2 and 2 Nm. The reward function is $r = (-\theta^2 + 0.01\dot{\theta}^2 + 0.001\tau^2$, which rewards the pendulum staying upright with minimum control effort.

B. ContinuousMountainCar-v0

This environment is also part of the Gymnasium classic control suite, and consists of a cart in a valley between two peaks. The cart must be accelerated to the top of the right peak by strategically 'rocking' it back and forth in the valley. The state consists of the cart's (1D) position and velocity, and there is a 1-dimensional continuous action space, consisting of a directional force. The reward consists of $r = -0.1a_t^2$ (where a_t is the action at time t), plus an additional reward of $+100$ if it reaches its goal.

C. CartPoleContinuous

This environment is a custom modification of the typically discrete CartPole problem. In the discrete version of the problem, an agent must balance an inverted pendulum on a (1-dimensional) cart by moving the cart, and can take a left or right push action; transitions are provided by an Euler integration step. In this custom continuous version, the agent provides a force on the cart in the range of $(-1, 1)N$, and the full fidelity equations of motion are integrated forward (with a fixed time step of 0.2s; the control frequency is therefore 5Hz) using a variable order, variable step numerical integrator. As compared to the discrete version, the dynamics here are more complex, and the agent must learn a continuous action to take at each step. The state space is left unchanged, consisting of the cart's position and velocity as well as the angular position and velocity of the pendulum. The reward structure is also the same as the discrete version: the agent receives a reward of $+1$ at each time step if observation bounds are not exceeded (the goal is to keep the cart upright for as long as possible).

D. PlanarCR3BPLyap

The final environment tested here is a Planar Earth-Moon CR3BP (Circular Restricted Three Body Problem) environment. The goal is to transfer from one L_1 Lyapunov orbit to another, larger L_1 Lyapunov orbit. It begins at a deterministic state (no stochasticity in the starting state) and must reach a *fixed* final state. The agent is allowed to make an impulsive ΔV maneuver at each time step, with a maximum magnitude of ΔV_{max}. There is a fixed time step of Δt and a fixed episode length of n_{steps}. Additionally, the episode terminates if the final state is within ϵ of the target state. The transitions are deterministic, and are dictated by numerical integration of the (planar) CR3BP equations of motion:

$$\ddot{x} = 2\dot{y} + x - \frac{(1-\mu)(x+\mu}{r_1^3} - \frac{\mu(x-1+\mu)}{r_2^3}$$
$$\ddot{y} = -2\dot{x} + y - \frac{(1-\mu)y}{r_1^3} - \frac{\mu y}{r_2^3} \quad (1)$$

where r_1 and r_2 are the distances from the first and second primary bodies, respectively. The problem setup is depicted in Figure 1; the agent must transfer from the initial point to the final point. To simplify the problem, there is no penalty applied for control magnitude; the sole goal is to simply reach the target. The agent receives rewards equal to $R = -||X_f - X_t||^2 - ||R_f - R_t||^2$ when the episode ends, either due to truncation or termination. Observation bounds are also shown in the figure; if the state exceeds these boundaries, the episode truncates. It should be noted that this environment is *significantly* more complex than the others, and a successful policy was not obtained.

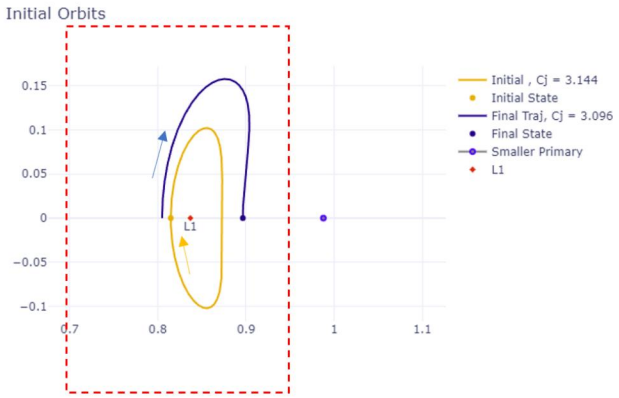

Fig. 1: Earth-Moon CR3BP Lyapunov orbit transfer setup. Red dotted line indicates valid observation box.

E. Algorithm Options

Several algorithm options were tested, each implemented as described in Section III. They are described in Table I, showing each algorithm's name, brief description, and inputs to the critic function. Each algorithm was based off the CleanRL implementation of PPO, and was augmented with SALE using components from Fujimoto et al.'s TD7 code [1].

F. Experiment Setup

For each environment, a baseline run was established using the vanilla continuous action PPO implementation in CleanRL. Hyperparameters were tuned manually based on 'normal' PPO, including number of total training timesteps, and were then kept constant across all of the other PPO variations (with the exception of parameters specific to SALE). Then, the other algorithm options were also run on these environments, keeping the hyperparameters the same as normal PPO where possible. Note that not all algorithm options were tested on all environments; this was due to speed issues with some of the options presented. Hyperparameters are listed in the Appendix, *except* where noted in the experiment results.

	Description	Inputs to Critic Function
PPO-V	Nominal, normal PPO	s
PPO-V-ZS	PPO with value function critic + state encoding	s, z^s
PPO-V-ZS-ZSA	PPO with value function critic + state, state-action encoding	s, z^s, z^{sa}
PPO-Q	PPO with $Q(s,a)$ critic, using episode actions	s, a
PPO-Q-ZSA	PPO with $(Q(s,a)$ critic + state, state-action encoding, using episode actions	s, a, z^s, z^{sa}
PPO-Q-MAX	PPO with $Q(s,a)$ critic, maximizing over episode actions within GAE	s, a
PPO-Q-MAX-ZSA	PPO with $Q(s,a)$ critic + state, state-action encoding, maximizing over episode actions within GAE	s, a, z^s, z^{sa}

TABLE I: Algorithm options

V. Results

For each environment, algorithm curves are colored according to Figure 2.

Fig. 2: Algorithm option legend.

A. Pendulum-v1

This experiment intended to test all algorithm options on a simple continuous control problem and served as a testbed for investigation into the behavior of the various embedding-enabled algorithm options. Though the original implementation of SALE with TD7 used a dimension of 256 for the embedding dimensions and for the encoders' hidden layers, a dimension of 64 was chosen here in the interest of training speed. Episodic rewards and losses over training are shown in Figures 3 and 4.

In this case, PPO-V-ZS-ZSA performs best here, despite possessing the largest encoder loss of all the SALE-enabled

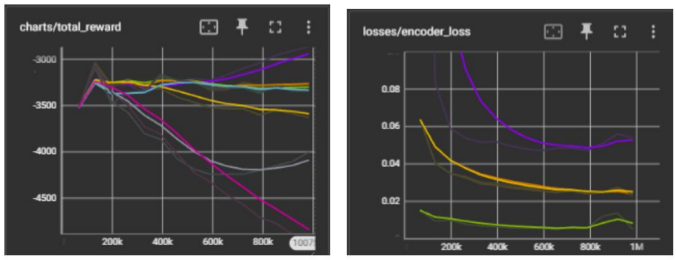

Fig. 3: Reward and encoder loss for Pendulum environment.

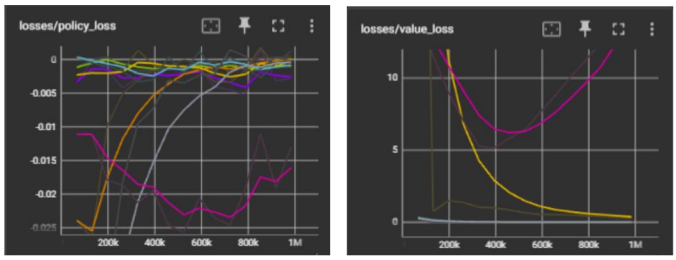

Fig. 4: Policy and value losses for Pendulum environment.

options. PPO-Q-MAX, which does not use SALE and computes $V(s)$ within the GAE loop using the max over actions taken in each episode, performs worst by a large margin. All of the SALE-enabled algorithms perform close to or better than vanilla PPO.

1) Pendulum-v1: Large Encoders: To see how the originally conceived larger encoder dimension would work, cases were run with the larger embedding dimension as well. The reward curve and encoder loss curve for this experiment are shown below. These results show that the encoder loss diverges towards the end of the training, but the agent rewards go up dramatically, which is a counterintuitive observation. Figures 5 and 6 compare the PPO-V variations and the PPO-Q variations, respectively, between the small and large encoder configurations. In each comparison, the pink-colored curves represent the larger encoder models, while the blue-colored curves represent the smaller encoder models.

Note that for the state value critic options, the larger encoder models exhibit higher encoder loss, but also higher rewards, which is a counterintuitive result. However, for the state-action value critic algorithms, the larger encoder models exhibit an increase in encoder loss, which corresponds with a lower total reward; as the encoder model is better able to predict the dynamics, better performance is achieved. The fact that the SALE-PPO variations which predict $Q(s, a)$ behave more in line with intuition may initially suggest that they are the better option for implementing SALE with PPO, but results in other environments are not always consistent with this expectation.

B. MountainCarContinuous-v0

This experiment tested all algorithms on a continuous version of Mountain Car, which has a more complex reward structure than the inverted pendulum problem. The dimension of the embeddings as well as the encoder hidden layers was

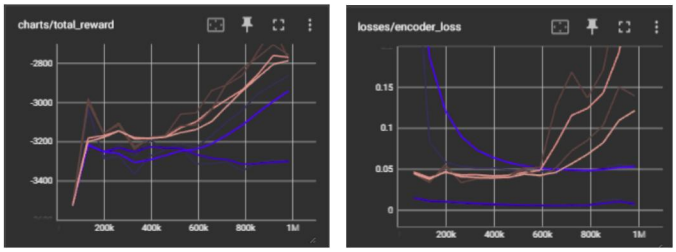

Fig. 5: Reward and encoder loss comparison between the small and large encoder models for the PPO-V algorithm options.

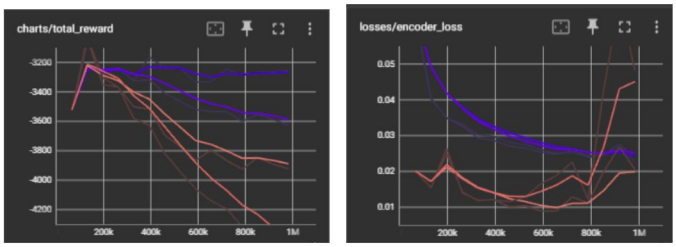

Fig. 6: Reward and encoder loss comparison between the small and large encoder models for the PPO-Q algorithm options.

set to 64, and a relatively low encoder learning rate of 5e5 was used.

In this environment, the results show that the Q-MAX variations performed particularly poorly. Also note the behavior of PPO-Q-ZSA here, which performed best. The encoder loss jumps and then settles back down, which also coincides with a jump in reward obtained. This type of behavior could indicate that the encoders have moved between local minima in their dynamics modeling. The agent being able to achieve a higher reward upon improving/changing its encoder models is evidence that the encoders' dynamics modeling is providing tangible benefits to the algorithm.

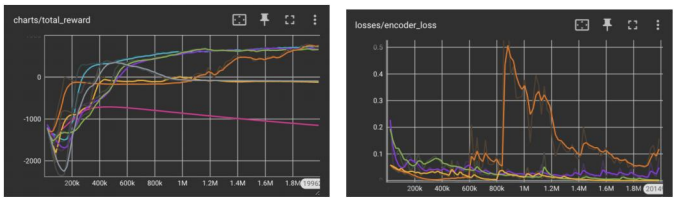

Fig. 7: Reward and encoder loss for continuous mountain car environment.

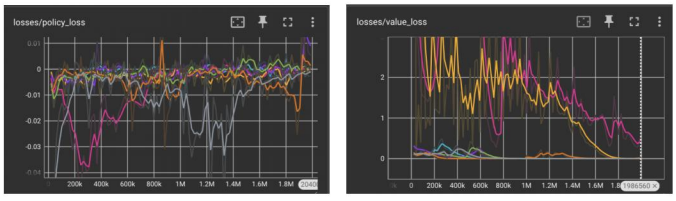

Fig. 8: Policy and value losses for continuous mountain car environment.

C. CartPoleContinuous

All algorithm options were tested on the continuous cart-pole environment; these results are shown below. Of particular interest here is the behavior of the PPO-Q-MAX variations, which is the opposite off the previous Mountain Car environment. Here, they perform best, and the PPO-Q-MAX-ZSA algorithm far outperforms the others despite having the poorest encoder models as indicated by the encoder loss. Meanwhile, the $V(s)$ based SALE-enabled PPO variations perform the worst. This complete reversal of the algorithm behavior shows that the benefits of applying SALE to PPO depend very strongly on the particular environment.

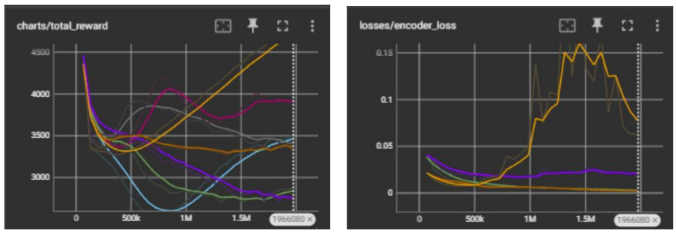

Fig. 9: Reward and encoder loss for continuous cart pole environment.

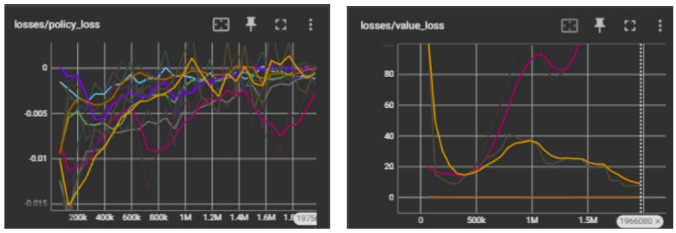

Fig. 10: Policy and value losses for continuous cart pole environment.

D. PlanarCR3BPLyap

Due to the much more complicated dynamics of the CR3BP environment, only the PPO-V, PPO-V-ZS, and PPO-V-ZS-ZSA options were tested here in the interest of runtime. In this environment, though none of the options were successful in reaching the target state, the SALE-enabled options did in fact perform better than vanilla PPO here, suggesting that these embeddings are in fact useful for environments with complex dynamics, as suggested by Fujimoto et al. [1]. Though a successful policy which could reach the target state was never found, a rollout using the best training result, obtained by PPO-V-ZSA, in shown in Figure 13.

E. Overall Tabular Results

Results, in terms of final reward at the end of training, for each algorithm option and environment, are shown below in Table II.

Examining these results, we can observe that the best performing algorithm in each environment *used the state-action embedding z^{sa} as an input to the critic*. Initially,

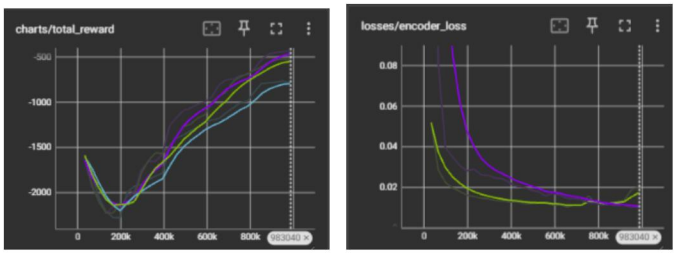

Fig. 11: Reward and encoder loss for planar CR3BP environment.

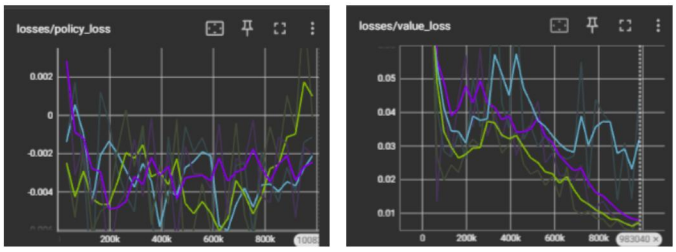

Fig. 12: Policy and value losses for planar CR3BP environment.

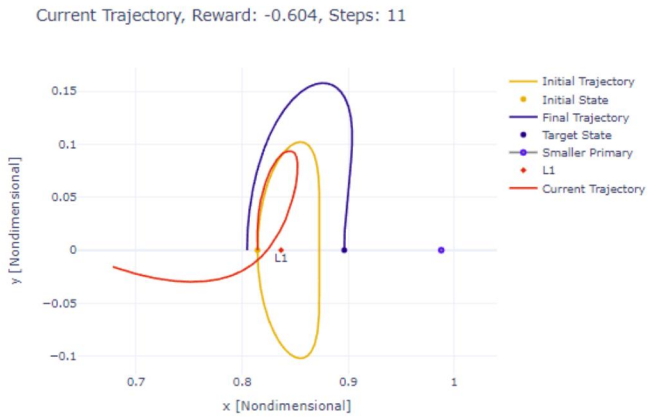

Fig. 13: Best CR3BP policy result, obtained using PPO-V-ZSA

	Pendulum	MountainCar	CartPole	CR3BP
PPO-V	-3344	726.2	3461	*-785*
PPO-V-ZS	-3294	657.1	2856	-529
PPO-V-ZS-ZSA	**-2862**	665.9	*2732*	**-466**
PPO-Q	-4001	-94.4	3406	
PPO-Q-ZSA	-3265	**748.2**	3329	
PPO-Q-MAX	*-4840*	-1161	3891	
PPO-Q-MAX-ZSA	-3589	*-121.1*	**4829**	

TABLE II: Algorithm results; best result in **bold** and worst in *italic* for each environment.

this suggests that the SALE embeddings do provide valuable additional information during the training process. However, it is important to note that the specific manner of incorporation of z^{sa} that results in the best performance differs based on the environment. In fact, in some cases, the best performing algorithm for one environment performs the *worst* for another — although both still incorporate z^{sa}. Because of this variability, it cannot be concluded from these experiments whether to use z^{sa} in $V(s)$ or $Q(s,a)$. Indeed, it cannot even be concluded which method of computing V based on Q — either using the action history during the episode, or maximizing over actions take in the episode — is better, despite the latter option matching more closely with the correct way of computing $V(s)$.

Furthermore, notice that the value loss when the using PPO-Q-MAX variations are much larger than the others. This anomalous behavior perhaps points towards an incorrect implementation of these options, or a potential theoretical issue with this approach.

Though wall-clock results (in terms of SPS, steps-per-second) are only shown in the Appendix, another key result is that more complex augmentations to PPO always take longer to run, in some cases showing a slowdown of a factor of 3 or more. Because of this, it may be the case that simply continuing to train PPO is more efficient in terms of real time despite lower sample efficiency.

Lastly, examining the entropy curves of the various algorithms reveals that the options which include both embeddings tend to reduce the entropy of the policies much faster than not doing so; this may be a benefit in some cases, for example, if the reward structure allows for quick movement towards an optimal policy, but could also hinder performance if a sparse reward structure necessitates a large amount of exploration to find a good policy. This entropy behavior may suggest that the performance increase or decrease caused by the embedding information is tied to its influence on the exploration vs exploitation tradeoff.

VI. Conclusions

The major algorithmic conclusions that can be drawn from this project are that the inclusion of the embeddings, specifically the state-action embedding z^{sa}, can improve the performance of PPO, either through incorporation into $V(s)$ or $Q(s,a)$, though the determination of whether the critic should estimate $V(s)$ or $Q(s,a)$ seems highly dependent on the environment. The initial goal was to determine if SALE can improve the performance of PPO applied to the CR3BP. Towards this end, though a converged CR3BP transfer policy was not recovered with any method, it seems clear that including embedding information does in fact improve upon vanilla PPO's performance in the CR3BP transfer environment.

VII. Future Work

Although z^{sa} does improve the performance of PPO, generally, it is still unclear how this tendency interacts with the choice of a $V(s)$ critic or a $Q(s,a)$ critic, and, if using the latter option, how to extract a value from the state-action value prediction. This choice seems to be critical to the performance of the algorithm, and future work should perform more experiments to understand this behavior. Specifically, an understanding of the theoretical implications of the $Q(s,a)$ critic option(s) on GAE. Further numerical experiments should also be conducted examining a wider range of environments, with special attention paid to whether the behavior of the different algorithm options depends on some particular property of the environment.

VIII. Contributions and Release

Sai Chikine performed all of the tasks described here. The authors grant permission for this report to be posted publicly.

References

[1] S. Fujimoto Kayser, W. Chang, E.J. Smith, S.S. Gu, D. Precup, D. Meger "For SALE: State-Action Representation Learning for Deep Reinforcement Learning," arXiv.2306.02451 [cs.LG], Jun. 2023

[2] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov "Proximal Policy Optimization Algorithms," arXiv:1707.06347 [cs.LG], Jul. 2017

[3] J. Schulman, P. Moritz, S. Levine, M. Jordan, P. Abbeel "High-Dimensional Continuous Control Using Generalized Advantage Estimation," arXiv:1506.02438 [cs.LG], Jun. 2015

[4] M. Towers et. al, "Gymnasium," https://gymnasium.farama.org/

[5] D. Valencia, J. Jia, R. Li, A. Hayashi, M. Lecchi, R. Terezakis, T. Gee, M. Liarakapis, B.A. MacDonald, H. Williams "Comparison of Model-Based and Model-Free Reinforcement Learning for Real-World Dexterous Robotic Manipulation Tasks," Presented at IEEE International Conference on Robotics and Automation (ICRA), 2023.

IX. Appendix

A. Hyperparameters

Hyperparameters for all environments, as well as for PPO and SALE parameters, are specified here.

1) Continuous Cart Pole: The cart pole was adapted from the discrete cart-pole environment provided in Gymnasium. A custom environment was created to allow for a continuous range of force applied to the cart. The dynamics are integrated with SciPy's 'dopri853' integrator.

Continuous Cart Pole	
Parameter	**Value**
g	$9.8m/s^2$
m_{cart}	$1kg$
m_{pole}	$0.1kg$
l_{pole}	$1m$
Time step	$0.02s$
F_{max}	$10N$
θ_{reward}	$+/-12°$
θ_{max}	$90°$

TABLE III: Hyperparameters for custom continuous action-space cart pole environment.

2) Planar CR3BP: The planar CR3BP environment is a transfer scenario between two fixed states corresponding to two L_1 Lyapunov periodic orbits in the Earth-Moon CR3BP. The system parameter, C_J values of the orbits, and the initial and target states are shown below.

3) PPO: Below are the hyperparameters used for PPO, except where specifically specified otherwise in other sections.

Planar CR3BP Lyapunov Transfer	
Parameter	**Value**
μ	0.012150585
$C_{J,i}$	3.144
$C_{J,f}$	3.096
$\vec{X}_i$	8.14664463e-01, - 5.23426638e-30, 2.20440878e-31, - 3.12529628e-17, 2.21724494e-01, 2.70666852e-31
$\vec{X}_f$	8.96461834e-01, 7.66894693e-06, 1.14124823e-33, - 3.07077066e-05, - 3.84964481e-01, 3.84836007e-31

TABLE IV: Parameters for planar CR3BP environment.

PPO	
Hyperparameter	**Value**
Learning Rate	1e-4
Value Loss Coefficient	0.5
Entropy Loss Coefficient	0.01
γ	0.95
λ_{GAE}	0.95
Number of Minibatches	32
Update Epochs	10
Steps per Environment	2048
Number of Environments	32

TABLE V: PPO Hyperparameters

4) SALE: Below are the hyperparameters used for SALE, except where specifically specified otherwise in other sections.

SALE	
Hyperparameter	**Value**
Learning Rate	5e-4
z^s dim	64
Encoder hidden layer dimension	64
Number of encoder hidden layers	3

TABLE VI: SALE Hyperparameters

B. Full Results

In addition to the results shown in each section in the main part of the paper, other performance metrics are presented here.

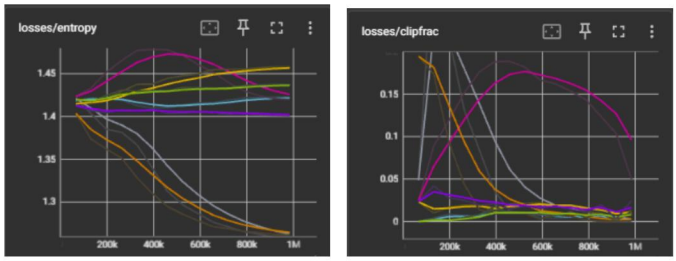

Fig. 14: Entropy and Clip Fraction for Pendulum-v1

1) Pendulum:

2) MountainCar: In addition to the results shown in each section in the main part of the paper, other performance metrics are presented here.

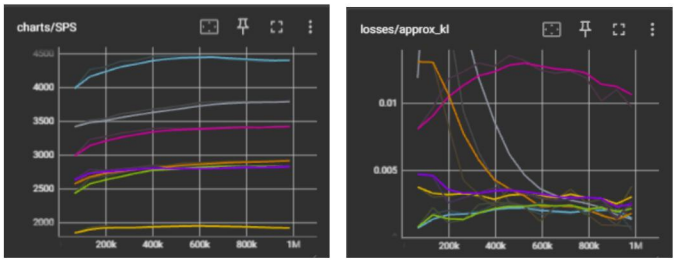

Fig. 15: Steps-per-Second and KL Divergence for Pendulum-v1

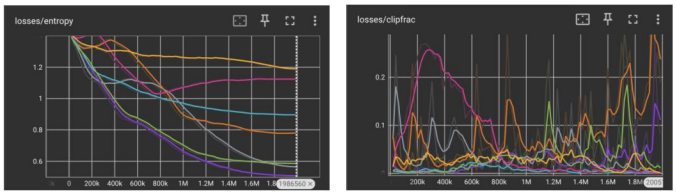

Fig. 16: Entropy and Clip Fraction for MountainCarContinuous-v0

3) CartPole: In addition to the results shown in each section in the main part of the paper, other performance metrics are presented here.

4) Planar CR3BP: In addition to the results shown in each section in the main part of the paper, other performance metrics are presented here.

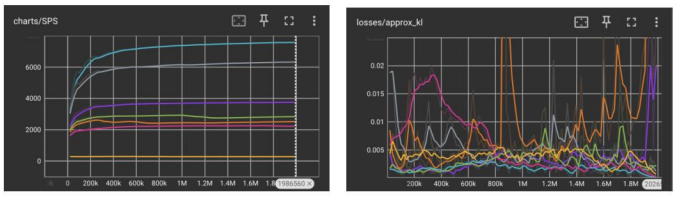

Fig. 17: Steps-per-Second and KL Divergence for MountainCarContinuous-v0

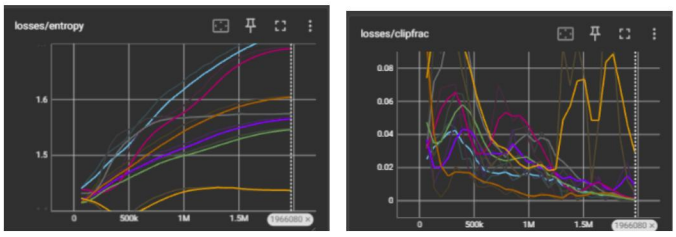

Fig. 18: Entropy and Clip Fraction for CartPoleODE-v0

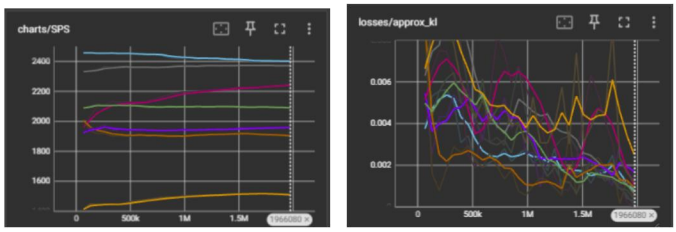

Fig. 19: Steps-per-Second and KL Divergence for CartPoleODE-v0

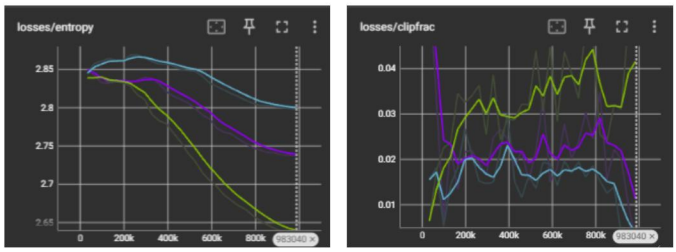

Fig. 20: Entropy and Clip Fraction for CR3BPLyap-v0

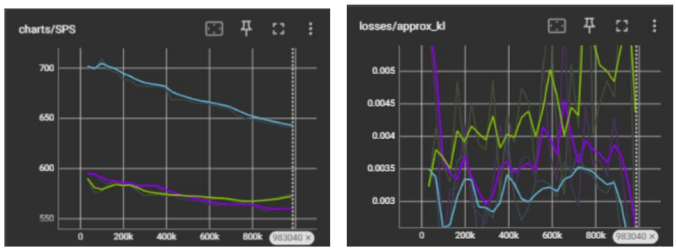

Fig. 21: Steps-per-Second and KL Divergence for CR3BPLyap-v0

1 Final Project Report

- 0 pts Correct