REINFORCEMENT LEARNING - PROJECT 3
PPO and Policy optimization with Penalised Point Probability Distance

Aman Johar : 50320278
Mudita Sanjive : 50322660

The objective of this project is to explore advanced methods and/or applications in reinforcement learning. In this project, we used Proximal Policy Optimisation and a variant of Trust Region Policy Optimisation (TRPO) - Policy Optimisation with Penalised Point Probability Distance (POP3D) to train the LunarLander and CartPole environment from the OpenAI Gym. We also tried to implement the algorithms on Super Mario Bros from OpenAI Gym, however, it wasn't as successful.

1 Introduction

In the following sections we introduce the two algorithms that we are using in this project

1.1 Proximal Policy Optimisation (PPO)

The central idea of Proximal Policy Optimisation is to avoid having large policy updates. This is achieved by taking a ratio that gives the difference between the new and old policy, and then clip this ratio, so the value doesn't go beyond a specific range. Doing this ensures that the policy update will not be too large. Furthermore, to improve performance, PPO trains the agent by running K epochs of gradient descent over a sampling mini batches. Policy Gradient update, taken by calculating gradient ascent on the loss function, suffers from the following drawbacks :

- When step size is too small, training becomes too slow
- When step size is too large, there is too much variability in the training

PPO overcomes these drawbacks as it improves the stability of the Actor training by limiting the policy update at each training step. In order to achieve this PPO introduced a new objective function called "Clipped surrogate objective function" that constraints the policy change in a small range using a clip.

Clipped Surrogate Objective Function :
Instead of using log pi to trace the impact of the actions, the ratio between the probability of action under current policy divided by the probability of the action under previous policy is used :

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}, \text{ so } r(\theta_{\text{old}}) = 1.$$

rt(θ) denotes the probability ratio between the new and old policy:

- If rt(θ) >1, it means that the action is more probable in the current policy than the old policy.
- If rt(θ) is between 0 and 1: it means that the action is less probable for current policy than for the old one.

Consequently, the new objective function becomes :

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right] = \hat{\mathbb{E}}_t\left[r_t(\theta)\hat{A}_t\right]$$

However, without a constraint, if the action taken is much more probable in the current policy than in the former, this would lead to a large policy gradient step and lead to an excessive policy update.

Consequently, we need to constraint this objective function by penalising changes that lead to a ratio that will sway too much from 1 (in the paper ratio can only vary from 0.8 to 1.2). This ensures that there will not be too large policy updates because the new policy can't be too different from the older one. PPO clips probability ratio directly in the objective function with its Clipped surrogate objective function.

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(\underbrace{r_t(\theta)\hat{A}_t}_{\text{L CPI}}, \underbrace{\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t}_{\text{Modifies the surrogate objective by clipping the prob ratio. --> Which removes the incentive for moving rt outside of the interval [1 - e, 1 + e]}})\right]$$

The minimum of the clipped and non clipped objective is taken, so the final objective is a lower bound (pessimistic bound) of the unclipped objective. If $\hat{A}t > 0$**,** it means that the action is better than the average of all the actions in that state. Therefore, we should encourage our new policy to increase the probability of taking that action at that state. If $\hat{A}t < 0$, the action should be discouraged because negative effect of the outcome. Consequently, rt will be decreased (because action is less probable for current policy than for the old one).
The final Clipped Surrogate Objective Loss is : [3]

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t\left[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)\right]$$

c1 and c2 are coefficients.

Add an entropy bonus **to ensure sufficient exploitation.**

Squared-error value loss: $(V_\theta(s_t) - V_t^{targ})^2$

1.2 Policy Optimisation with Penalised Point Probability Distance (POP3D)

POP3D can be seen as a variant of Trust Region Policy Optimisation(TRPO). It keeps almost all positive spheres of proximal policy optimisation (PPO) such as easy implementation, fast learning and high score capability. As PPO, it also uses a single surrogate objective without constraints, where a penalised item based on point probability distance is included to prevent update step from growing too large.

KL Divergence, used in TRPO, is an asymmetric distance measurement for two probability distributions. Hence, in POP3D, mean square error is used to define the point probability distance. In discrete domain, when the agent takes action 'a', the point probability distance between πθold(·|s) and πθnew(·|s) is defined as :

$$D_{pp}(\pi_{\theta_{old}}(\cdot|s), \pi_{\theta_{new}}(\cdot|s)) = (\pi_{\theta_{old}}(a|s) - \pi_{\theta_{new}}(a|s))^2$$

Unlike KLD, Dpp is symmetric by definition. Since 0≤ πθ(a|s) ≤1 holds for discrete action space, Dpp has lower and upper boundary: 0 ≤ Dpp ≤ 1. Moreover, Dpp is less sensitive to action space dimension than KLD, which has a similar effect as PPO's clipped ratio to increase robustness and enhance stability.

The new surrogate function for POP3D can be defined as :

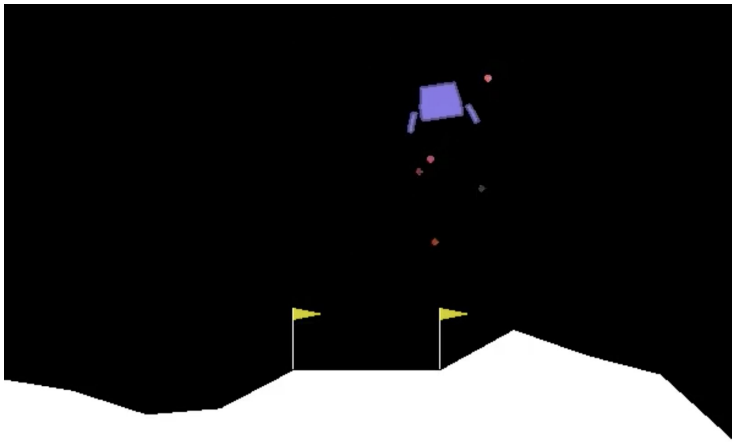
$$\max_{\theta} E_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t - \beta D_{pp}(\pi_{\theta_{old}}(\cdot|s), \pi_\theta(\cdot|s))\right],$$

where β is the penalised coefficient. In implementation, generalised advantage estimates to calculate At are used.[4]

2 Environments Used

We used three environments for this project - LunarLander, CartPole and Super Mario Bros.

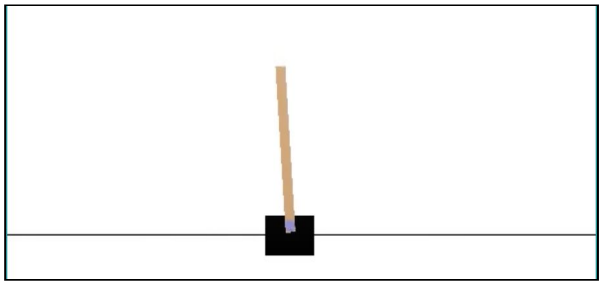
2.1 LunarLander-v2



The goal of the agent - LunarLander - is to successfully land on the landing pad. If lander moves away from landing pad it loses reward back. Episode finishes if the lander crashes or comes to rest, receiving additional -100 or +100 points. The environment consists of the following state, action and reward space :

- State space : Landing pad is always at coordinates (0,0). Coordinates are the first two numbers in state vector.
- Action space : The action space consists of four discrete actions - do nothing, fire left orientation engine, fire main engine, fire right orientation engine.
- Reward space : Reward for moving from the top of the screen to landing pad and zero speed is about 100-140 points. Episode finishes if the lander crashes or comes to rest, receiving additional -100 or +100 points. Each leg ground contact is +10. Firing main engine is -0.3 points each frame. Solved is 200 points.[1]

2.2 CartPole-v0



The game consists of a pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The pendulum starts upright, and the goal is to prevent it from falling over by increasing and reducing the cart's velocity. Given below is the environment description of the game [2] :

- Observation

Num	Observation	Min	Max
0	Cart Position	-2.4	2.4
1	Cart Velocity	-Inf	Inf
2	Pole Angle	~ -41.8°	~ 41.8°
3	Pole Velocity At Tip	-Inf	Inf

- Action

Num	Action
0	Push cart to the left
1	Push cart to the right

- Reward - Reward is 1 for every step taken, including the termination step

- Starting State - All observations are assigned a uniform random value between ±0.05

- Episode Termination

  1. Pole Angle is more than ±12°
  2. Cart Position is more than ±2.4 (centre of the cart reaches the edge of the display)
  3. Episode length is greater than 200

2.3 SuperMarioBros-v0

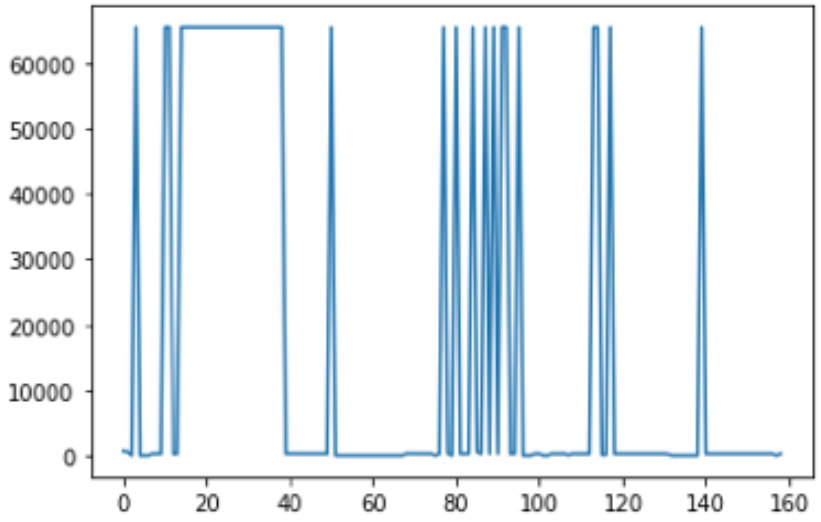
The game consists of a single agent - Mario. The goal of Mario is to begin on the left edge of a level and navigate through the map and reach the flagpole without dying, while trying to maximise the score.In order to achieve success, Mario generally must move right. However, in order to reach an optimal level of gameplay, it must also learn how to avoid/kill its enemies, jump over pipes and spaces, and adjust its speed to improve the score. The following is how the environment is defined:

- State space : represented as a 13×16 tile grid of the numbers 0-3. Number 0 represents empty space, 1 represents an object such as a coin, ground or pipe, 2 represents enemies
  and 3 represents Mario.
- Action space : the representation on the action space is dependent on the FCEUX emulator aka “Nintendo Controller”. There are 6 actions that can be either pressed (1) or not pressed (0) in the game environment. This leads to possible actions. However, there are only 9 actions that make logical sense and have an impact on the game. For example, pressing left and right is an action on the emulator but has no effect within the environment.
- Reward space : The Gym Super Mario environment by Philip Paquette provides a default reward function, which changes in respect to Mario’s distance among the level. Mario gets +1 reward if it moves right, -1 if it moves left and 0 if it stays in place. Mario also gets rewards for killing enemies, jumping over pipes and avoiding obstacles.

3 Experiments and Results

3.1 SuperMarioBros-v0

We tried to train Mario using PPO, however did not achieve great results :

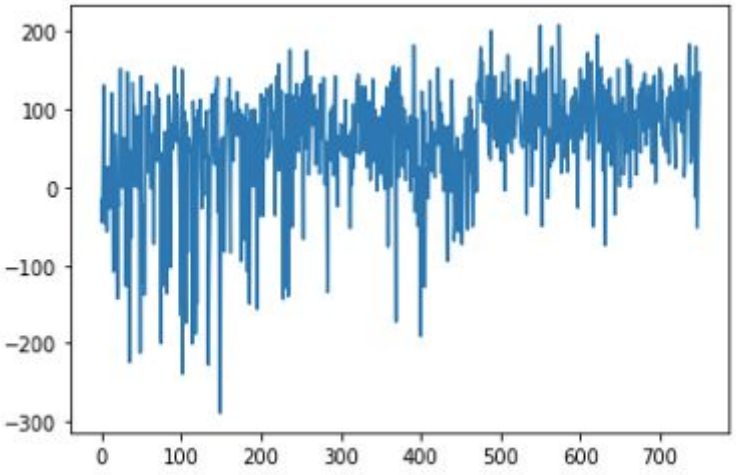


The agent, did learn to achieve the maximum score (65535) a little, however, did not exploit its learning enough. In many instances, the agent (Mario), chose to take no action, and stayed in one place, in order to avoid negative rewards.

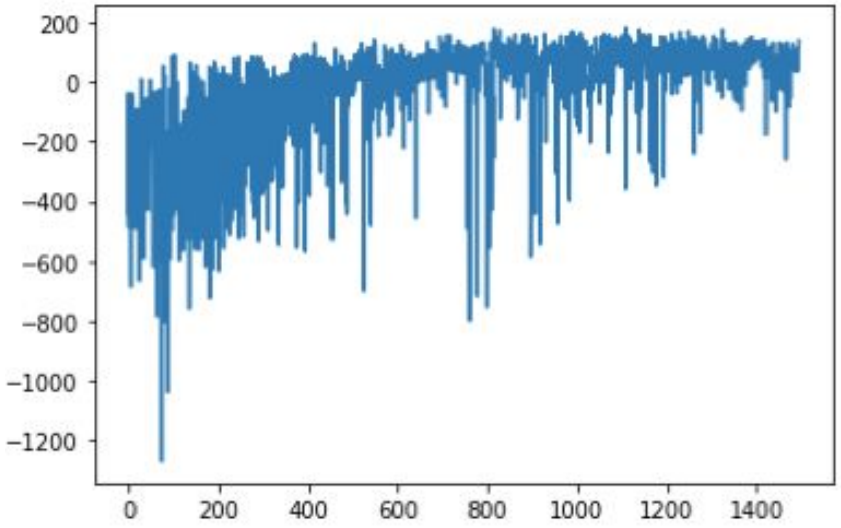
3.2 LunarLander-v2

3.2.1 PPO

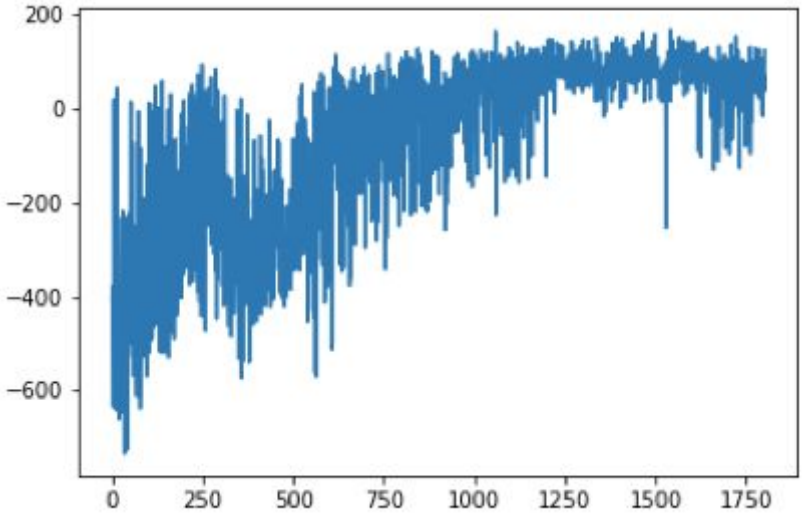
We kept the learning rate to 0.0001 and varied gamma and the entropy coefficient to get different results



LunarLander-v2 on PPO (gamma = 0.95, entropy coefficient = 0.01)



LunarLander-v2 on PPO (gamma = 0.99, entropy coefficient = 0.01)

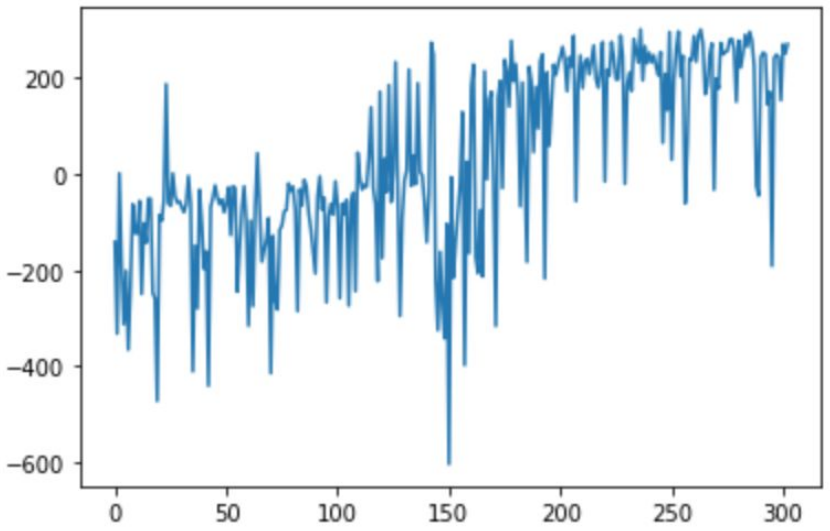


LunarLander-v2 on PPO (gamma = 0.99, entropy coefficient = 0.05)

The best results were achieved with gamma = 0.99 and entropy coefficient = 0.05 as evident from the above graphs. We stopped the training based on when we saw evidence of successful training or not so good training therefore the number of episodes can vary.

3.2.2 POP3D

We kept the learning rate to 0.0001, gamma = 0.99 , entropy coefficient = 0.05 and point probability distance coefficient to 5.0.

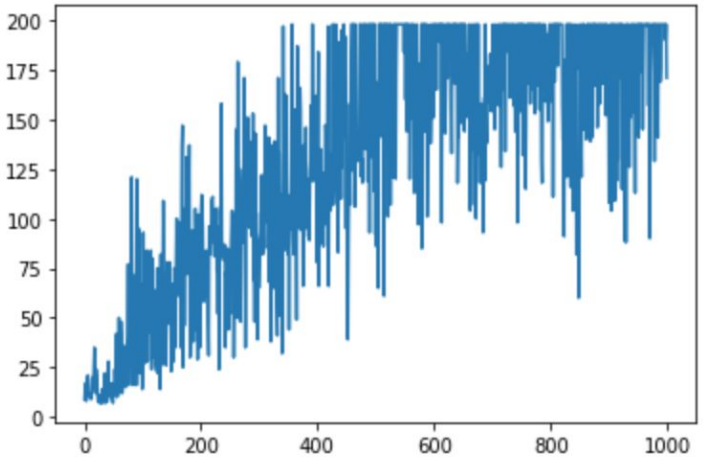


Lunar Lander-v2 on POP3D

We see a slight uptick around episode 160-170 and after that the score stays consistently high.

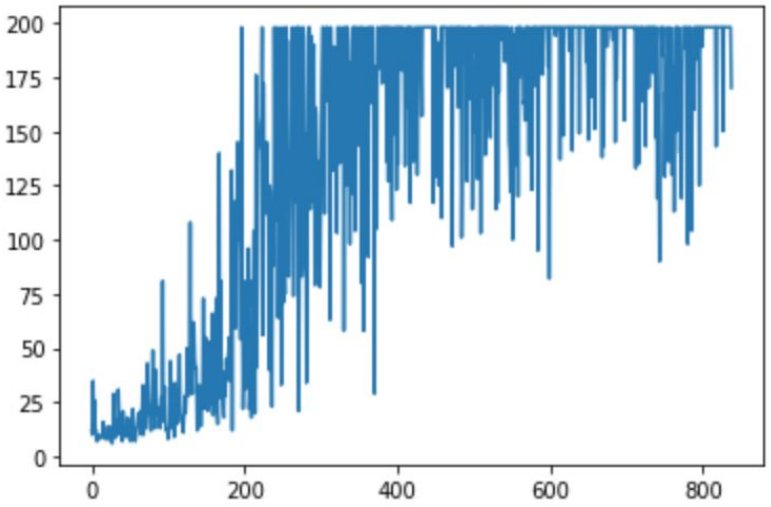
3.3 CartPole-v0

3.3.1 PPO



CartPole-v0 on PPO (gamma = 0.95, entropy coefficient = 0.01)

3.3.2 POP3D



Lunar Lander-v2 on POP3D (gamma = 0.95, entropy coefficient = 0.01, point probability coefficient = 5)

4 Conclusion

POP3D has proved to be a solid alternative to PPO. We see that it enjoys all the advantages of PPO such as fast learning ability and training efficiency while maintaining an optimistic surrogate objective. PPO on the other hand with a slightly pessimistic objective due to the clipped probability ratios can be outperformed on environments that may require an optimistic approach. This can be observed in case of LunarLander-v2 where in PPO requires slightly more training to start achieving the higher rewards as compared to POP3D. In case of CartPole-v0, the performance of the two algorithms is almost identical.

References

[1] https://towardsdatascience.com/solving-lunar-lander-openaigym-reinforcement-learning-785675066197
[2] https://github.com/openai/gym/wiki/CartPole-v0
[3]https://towardsdatascience.com/proximal-policy-optimization-ppo-with-sonic-the-hedgehog-2-and-3-c9c21dbed5e
[4]https://www.groundai.com/project/policy-optimization-with-penalized-point-probability-distance-an-alternative-to-proximal-policy-optimization/1
[