OVERALL HYPERPARAMETERS :

- **actor_lr** and **critic_lr**: This controls the step size for updating the policy/value network. A higher learning rate can lead to faster learning but may cause instability if too large, resulting in fluctuating or diverging scores. A lower learning rate can provide stable learning but may slow down the convergence.
- **entropy_coef**: This term encourages exploration by adding the entropy of the policy to the loss function. Higher entropy coefficients promote more random actions, which can prevent premature convergence to suboptimal policies. However, too high a value may lead to excessive exploration, delaying convergence.
- **batch_size**: This defines the number of samples per mini-batch for policy and value function updates. Smaller batch sizes can introduce more variance in gradient estimates, which might lead to noisier updates and unstable training. Larger batch sizes provide more accurate gradient estimates, but they require more memory and computational resources.
- **num_epochs**: This parameter sets the number of times the entire dataset is used for training in each update cycle. More epochs mean more updates per batch of experiences, which can lead to more thorough learning from the same data but also increase the risk of overfitting to recent experiences.
- **vf_coef**: This term scales the value function loss in the total loss function. A higher coefficient places more emphasis on reducing value prediction errors, which can stabilize the training process but may slow down policy improvement. A lower value does the opposite.
- **GAE Lambda**: This parameter controls the bias-variance trade-off in the Generalized Advantage Estimation (GAE). Higher values (close to 1) reduce bias but increase variance, while lower values reduce variance but increase bias.

CARTPOLE, ACROBOT, MOUNTAINCARCONTINIOUS, PENDULUM

CARTPOLE

- **actor_lr=5e-4, critic_lr=3e-4, entropy_coef=0.005**: the higher learning rate for the actor and critic indicates faster learning, but the entropy coefficient is low, which means less exploration. After reaching the optimal policy, the agent maintains this performance without significant fluctuations, suggesting that the learning rates, while high, were not too high to cause instability after the optimal policy was found. **actor_lr=1e-4, critic_lr=1e-4, entropy_coef=0.005**: this curve shows gradual improvement without significant fluctuations, but it takes longer to reach high scores compared to the yellow curve. Eventually, it achieves high scores with stability, reflecting a conservative yet stable learning approach.
- **actor_lr=1e-4, critic_lr=3e-3, entropy_coef=0.01**: Shows significant improvement early in the training, similar to the blue curve. The scores reach high values but exhibit considerable fluctuations, indicating instability. The higher learning rate for the critic allows rapid value updates, but this, combined with the higher entropy coefficient, leads to instability in the learning process, the agent does to much exploration, even when it has reached its optimum.

ACROBOT

- impact of **rollout_len**: This determines the number of steps the agent takes before updating the policy. Longer rollouts can provide more stable gradient estimates but can also be computationally more expensive. In this comparison, 1028 versus 512 steps. The longer rollout length should, in theory, lead to better policy updates due to more data per update, but can also slow down the learning process because updates happen less frequently.
- Impact of **num_epochs**: epochs (12 versus 10) means more training iterations per batch, which can improve learning but also increase the risk of overfitting to the batch data.
- Impact of **batch_size**: Larger batch sizes (32 versus 16) provide more stable gradient estimates but require more memory and can be less responsive to recent changes in the data.
- **entropy_coef=0.01, rollout_len=1028, num_epochs=12, batch_size=32, actor_lr=1e-3, critic_lr=1e-3**: This configuration uses a higher rollout length, more epochs, larger batch size, and higher learning rates. It appears to struggle initially (low scores) likely due to the high learning rates causing instability and the larger batch size making it less responsive to recent changes. Over time, it shows some improvement, indicating that it eventually starts to learn a more stable policy.
- **entropy_coef=0.01, rollout_len=512, num_epochs=10, batch_size=16, actor_lr=1e-4, critic_lr=5e-4**: This configuration shows much better performance early on, suggesting that the lower learning rates and smaller batch size are providing a more stable and responsive learning process. The moderate entropy coefficient helps balance exploration and exploitation effectively.
- **entropy_coef=0.005, rollout_len=512, num_epochs=10, batch_size=16, actor_lr=1e-4, critic_lr=5e-4**: This configuration has similar parameters to the previous but with a lower entropy coefficient, which means less exploration. This results in better score and in the long-term performance as the policy becomes more refined and less exploratory.

MOUNTAINCARCONTINIOUS

- **entropy_coef =0.003, num_epochs=64, batch_size =512, 1e-3, 5e-3**: A lower entropy coefficient means less emphasis on exploration. The policy focuses more on exploiting known good actions. With a smaller batch size (256), we can get slightly more noisy updates but it requires less memory, and seems to perform better in this case.

PENDULUM

- **entropy_coef=0.005, vf_coef=0.5, rollout_len=2048, num_epochs=25, actor_lr=1e-4, critic_lr=5e-4, ActionNormalizer=True**: Leads to more exploitation of the known good actions, possibly resulting in faster initial learning but can lead to premature convergence to suboptimal policies.

For PPO, we used 2 hidden_layers for both actor and critic of dimensions 64. If not mentioned, the following hyperparameters where used: gamma = 0.99, lamda = 0.95, entropy_coef = 0.01, epsilon = 0.2, vf_coef = 0.5, rollout_len = 256, total_rollouts = 3000, num_epochs = 10, batch_size = 16.