INSTITUTO FEDERAL DO ESPÍRITO SANTO
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA – PPCOMP

GUSTAVO AMORA BASÍLIO

TRABALHO DA DISCIPLINA DE INTELIGÊNCIA ARTIFICIAL

Serra
2023

GUSTAVO AMORA BASÍLIO

TRABALHO DA DISCIPLINA DE INTELIGÊNCIA ARTIFICIAL

3º Trabalho Avaliativo da Disciplina Inteligência Artificial do curso de Pós-Graduação em Computação Aplicada – PPCOMP do Instituto Federal do Espírito Santo.

Professor: Dr. Sérgio Nery Simões

Serra
2023

LISTA DE FIGURAS

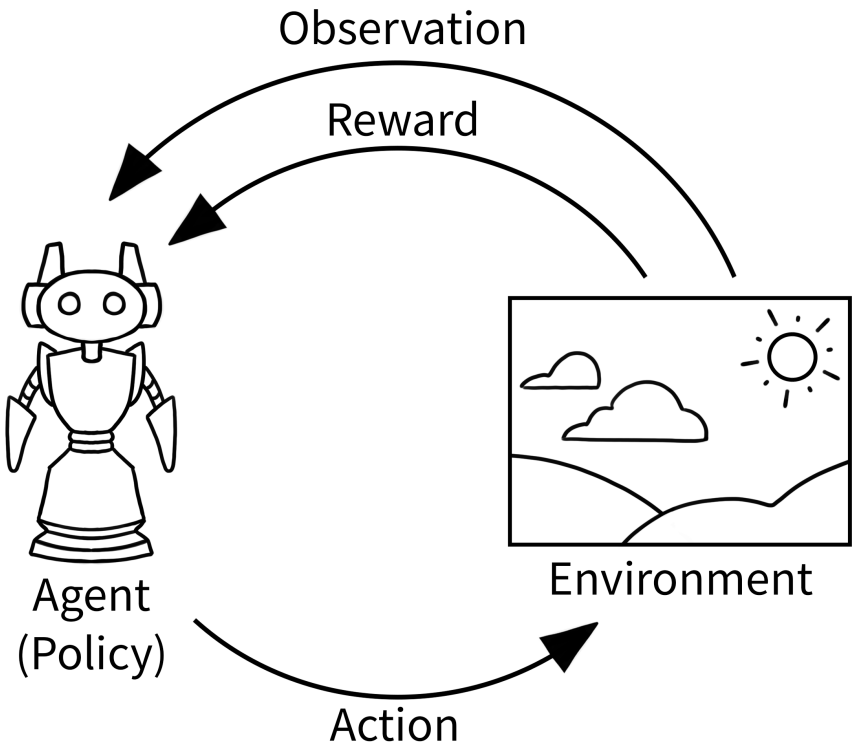
SUMÁRIO

1 INTRODUÇÃO

O fluxo clássico do modelo de aprendizado por reforço pode ser visto na Figura 1.

Gymnasium (Gym) é um projeto da OpenAI que visa fornecer um ambiente padronizado e flexível para o treinamento de agentes de inteligência artificial (IA) por meio de aprendizado por reforço (RL). É uma biblioteca de software de código aberto que oferece uma ampla gama de ambientes de treinamento para desenvolvedores e pesquisadores explorarem e aprimorarem algoritmos de IA.

Figura 1 – The classic "agent-environment loop".



Fonte: <https://gymnasium.farama.org/content/basic_usage/>

O projeto Gym oferece APIs para alguns ambientes de RL, como por exemplo: *cartpole, pedulum, moutain-car, mujoco, atari* entre outros.

Para treinar o modelo é utilizada a biblioteca de aprendizagem por reforço denominada *Stable Baselines3* (SB3), está foi escolhida pela sua compatibilidade com o a Gym. Além disso, a SB3 é definida como a implementação de algoritmos de aprendizado por reforço em *Pytorch* (RAFFIN et al., 2021), este, por sua vez, é uma biblioteca de tensores otimizada para *deep learning* usando GPUs e CPUs (DOCUMENTATION,).

Uma descrição dos paramêtros base oferecidos pela SB3 para treinamento dos modelos pode ser acessada em <https://stable-baselines3.readthedocs.io/en/master/modules/base.html>.

Alguns dos quais acreditamos ter maior relevância para os problemas abordados neste trabalho, são:

1. n_timesteps: Número de exemplos a serem treinados. Esse parâmetro determina a quantidade de exemplos de treinamento dedicada ao agente. Geralmente, quanto maior o parâmetro fornecido, mais oportunidades o agente tem para aprender e melhorar seu desempenho.

2. policy: Política aplicada ao treinamento. A escolha da política de aprendizado pode ter um impacto significativo no desempenho do agente. Diferentes políticas possuem arquiteturas e técnicas de aprendizado específicas, e a escolha correta depende das características do problema e do ambiente.

3. learning_rate: Taxa de aprendizado. A taxa de aprendizado controla o quão rápido o agente ajusta seus parâmetros com base nas informações recebidas do ambiente.

4. gamma: O *gamma* determina o quão importante o agente considera as recompensas futuras em relação às recompensas imediatas. Um valor de *gamma* mais alto indica maior importância para as recompensas futuras, incentivando o agente a buscar recompensas a longo prazo.

5. batch_size: Tamanho do lote. Esse parâmetro determina quantos exemplos de treinamento são usados em cada atualização dos parâmetros do agente. Valores maiores podem fornecer uma estimativa de gradiente mais precisa, mas também podem exigir mais recursos computacionais.

6. buffer_size: Tamanho do buffer de replay . É o tamanho do buffer que armazena as experiências passadas do agente para a técnica de replay de experiência. Valores maiores permitem um maior histórico de experiências para o agente aprender, mas também requerem mais memória.

2 AMBIENTES

Neste trabalho desenvolvemos agentes por RL em dois ambientes: Mountain Car e Car Racing.

2.1 Mountain Car

2.1.1 Descrição

O ambiente *Mountain Car* (MountainCar-v0) apresenta um problema do tipo *Markov Decision Process* (MDP) determinístico e discreto.

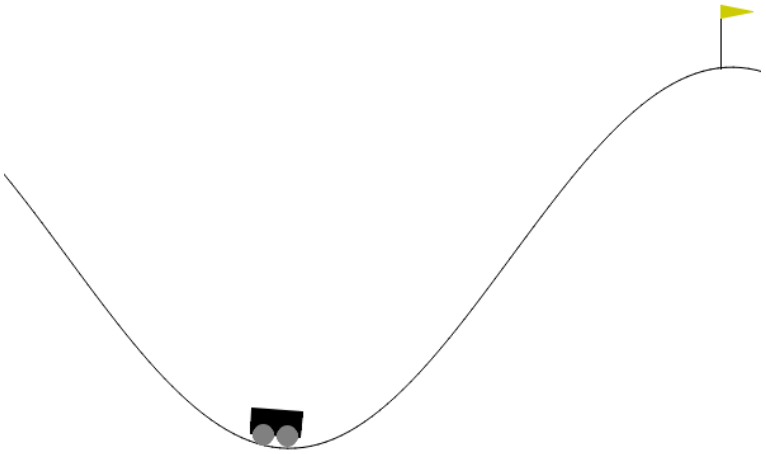
Em ambiente determinístico não há aleatoriedade ou incerteza envolvida nas interações com o ambiente. O agente pode tomar decisões com base em uma regra fixa e esperar resultados consistentes (RUSSELL; NORVIG, 2009).

Um MDP é um modelo matemático usado para tomar decisões em situações estocásticas. Ele é baseado na teoria de processos de Markov, que considera que o futuro depende apenas do estado presente, não do histórico completo. Em um MDP, existem estados, ações, probabilidades de transição e recompensas. O objetivo é encontrar uma política ótima, que maximize a recompensa esperada ao longo do tempo. (PUTERMAN, 1990).

Já um problema de RL dito discreto está relacionado ao espaço de estados e/ou ações sendo discretos e finitos, ou seja, composto por um conjunto limitado de opções (RUSSELL; NORVIG, 2009).

No *MoutainCar-v0* um carro é posicionado estocasticamente dentro de um vale, com as únicas ações possíveis acelerar para a esquerda ou para a direita. O objetivo é atingir o topo da direita. A representação do problema pode ser vista na Figura 2.

Figura 2 – Mountain Car.



Fonte: <https://gymnasium.farama.org/environments/classic_control/mountain_car/>

2.1.2 Espaço de observação

O espaço de observação do problema é definido conforme Tabela 1.

Tabela 1 – Espaço de observação Mountain Car

Num	Obervação	Min.	Max.	Unidade
0	posição do carro do eixo x	-1.2	0.6	posição (m)
1	velocidade do carro	-0.07	0.07	velocidade (v)

2.1.3 Espaço de ação

Existem 3 ações discretas e determnísticas possíveis:
Acelerar para esquerda, não acelerar ou acelerar para direita.

2.1.4 Recompensas

O objetivo é alcançar a bandeira no topo direito do vale o mais rápido possível. Enquanto não alcança o agente recebe uma recompensa = -1 para cada *timestep*.

2.1.5 Fim do episódio

O episódio acaba em duas situações:
i) *Termination*: a posição do agente é >= 0.05 (posição da bandeira).
ii) *Truncation*: o agente não atinja o objetivo até o tamanho do episódio = 200.

As especificações completas podem ser acessadas em <https://gymnasium.farama.org/environments/classic_control/mountain_car/>

2.2 Car Racing

2.2.1 Descrição

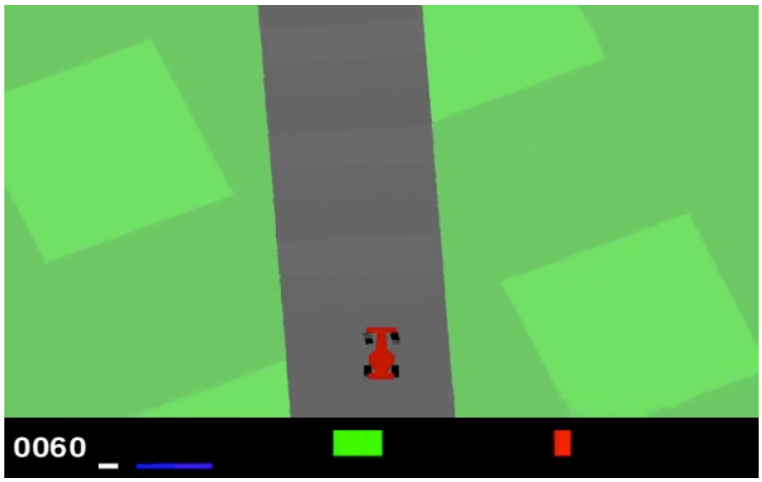
O problema consiste em um ambiente *top-down racing*. As pistas são geradas aleatoriamente e o objetivo é um carro aprender a andar na pista efetuando curvas, acelerando e freando.

É apresentada uma visualização da pista e do carro. Além disso, são apresentados indicadores na parte de baixo da tela. Da esquerda para a direita: velocidade, sensores ABS, direção e o giroscópio.

2.2.2 Espaço de observação

Uma imagem *top-down* de um carro de corrida em uma pista de corrida. Sua resolução é 96x96 e utilizando o padrão de cores RGB.

Figura 3 – Car Racing.



Fonte: Elaborada pelos autores.

2.2.3 Espaço de ação

Se contínuo, há 3 ações: Onde a direção pode variar de -1 a +1 em que 0 é seguir em frente, -1 é totalmente à esquerda, +1 é totalmente à direita. Acelerar, frear.

Se discreto, há 5 ações: não fazer nada, virar à esquerda, virar à direita, acelerar, frear.

2.2.4 Recompensas

O mecanismo de recompensa funciona descontando 0.1 por cada *frame* e há uma soma de 1000/N por todos *tiles* visitados, em que N é o número de *tile* visitados.

2.2.5 Fim do episódio

O episódio termina quando o carro visita todo os *tiles*. O carro também pode sair da pista, sendo assim ele receberá -100 de recompensa e morrerá.

3 RESULTADOS

3.1 Mountain Car

3.1.1 Treinamento

Para o problema do *mountain car* o código foi executado 3 vezes (DQN-1, DQN-2, DQN-3) mantendo todos os parâmetros iguais para as 3 execuções, exceto o *n_timesteps*. Os parâmetros utilizados em cada uma das execuções podem ser vistos na Tabela 2.

Tabela 2 – Parâmetros utilizados no problema Mountain Car

Parâmetro	DQN-1	DQN-2	DQN-3
n_timesteps:	3e5	1e5	5e4
policy:	MlpPolicy	MlpPolicy	MlpPolicy
learning_rate:	4e-3	4e-3	4e-3
batch_size:	128	128	128
buffer_size:	10000	10000	10000
learning_starts:	1000	1000	1000
gamma:	0.98	0.98	0.98
target_update_interval:	600	600	600
train_freq:	16	16	16
gradient_steps:	8	8	8
exploration_fraction:	0.2	0.2	0.2
exploration_final_eps:	0.07	0.07	0.07
policy_kwargs:	net_arch=[256, 256]	net_arch=[256, 256]	net_arch=[256, 256]

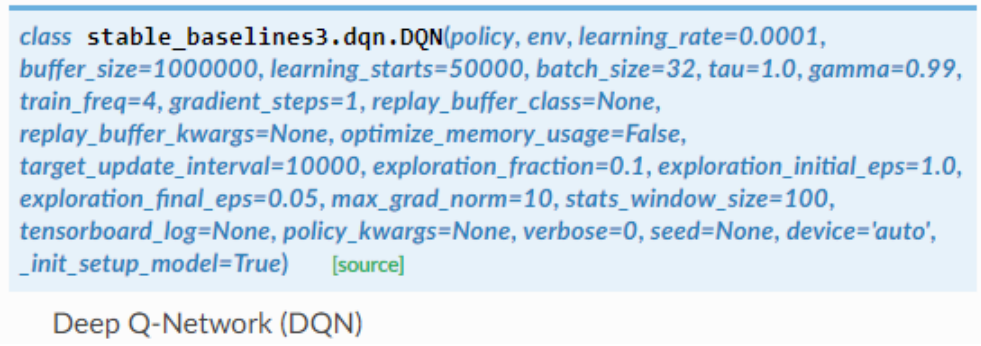
Sobre os principais parâmetros que se mantiveram iguais nos testes:

i) policy: foi escolhida a Mlpolicy, uma política estruturada em Deep Q Network (DQN) e adequada para problemas discretos. De acordo com sua documentação <https://stable-baselines3.readthedocs.io/en/master/modules/dqn.html> ela se utiliza de diversos "truques"para estabilizar o aprendizado através de redes neurais.
ii) learning_rate: a taxa de aprendizado foi reduzida da padrão oferecida pela SB3. O objetivo foi evitar oscilações indesejadas e manter o modelo mais estável.
iii) gamma: valor alto, próximo a 1 para dar mais peso a recompensas futuras e assim incentivar o agente a recompensas de longo prazo.
iv) batch_size: valor foi aumentado a fim de fornecer uma estimativa de gradiente mais precisa já que os recursos computacionais para o problema não eram de alta exigência.

Sobre os demais parâmetros, alguns foram reduzidos do padrão SB3 com o objetivo de adequar aos valores de n_timesteps, como no caso do buffer_size e do taget_uptade_interval. Os demais ficaram iguais ou próximos dos parâmetros base SB3, descritos na Figura 4

Sobre os valores de n_timesteps, parâmetro que foi alterado em cada uma das três execuções: Na primeira execução, DQN-1, foi testado o valor deste parâmetro com 300.000

Figura 4 – Parâmetros padrão DQN.



Fonte: <https://stable-baselines3.readthedocs.io/en/master/modules/dqn.html>

obtendo um resultado satisfatório de convergência para o objetivo. Depois, foram testados valores menores, 100000 e 50000 para os modelos DQN-2 e DQN-3, respectivamente. O objetivo foi testar se a convergência dos agentes para o objetivo se manteria com bons resultados mesmo com menos episódios de treinamento.

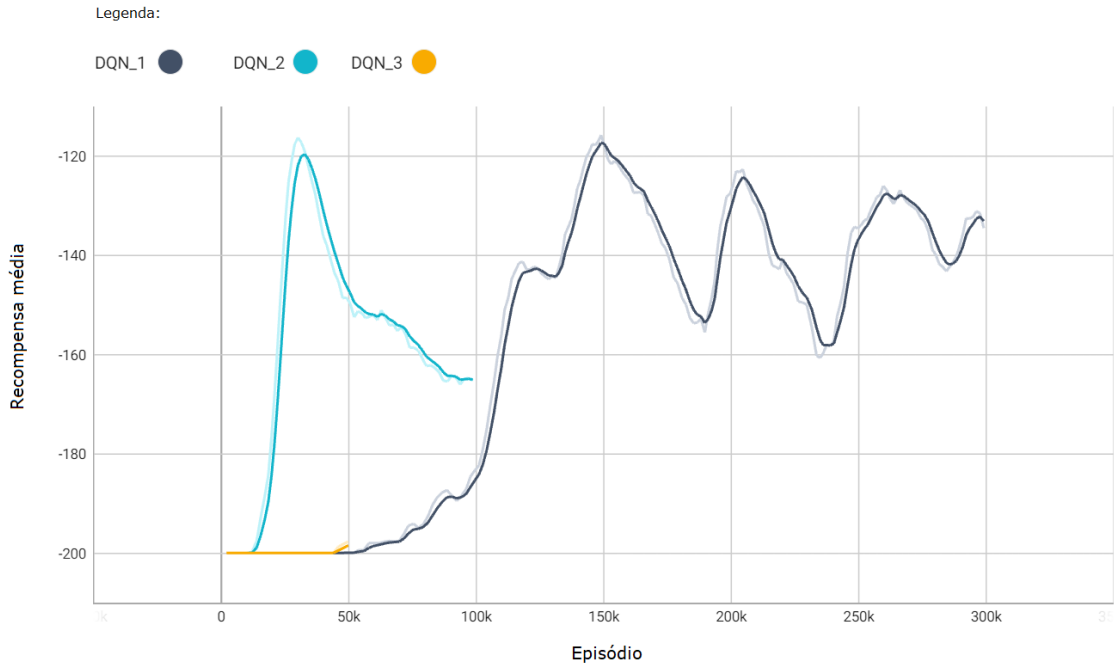
Os resultados do desempenho dos agentes ao longo do tempo na etapa de treinamento podem ser observados na Figura 5 que apresenta um gráfico com o número de episódios no eixo x e a recompensa média de cada um deles durante o treinamento no eixo y.

Como o episódio termina em 200 passos e o agente recebe uma recompensa = -1 para cada passo dado, logo, quanto mais rápido ele encontrar o objetivo maior será a recompensa. Na figura podemos observar que o modelo DQN-1 não convergiu para o objetivo até os 50k episódios. Contudo, a partir de 100k episódios de treinamento ele recebeu recompensas entre -160 e -120, o que para o problema parece ser um bom resultado. Já o modelo DQN-2 atingiu sua maior recompensa rapidamente, por volta do episódio número 30000, momento em que os demais ainda não tinham convergido para o objetivo. Porém, depois o agente caiu o seu desempenho já que não foi usada a função callback para os treinamento. Uma hipótese para o DQN-2 seria de que, caso a função callback estivesse ativada, o agente poderia melhorar a sua recompensa a partir do mínimo encontrado no episódio 30000, convergindo para uma solução melhor que todos os modelos em um tempo de treinamento mais rápido. Sobre o modelo DQN-3, o agente não convergiu para o objetivo em quase todos os 5e4 episódios de treinamento, apenas em alguns episódios perto do fim do seu ciclo de aprendizado e mesmo assim com uma recompensa baixa comparada aos outros modelos, o que indica que demorou para convergir nos episódios que obteve sucesso.

3.1.2 Teste

Os resultados dos testes com a média de recompensa de cada modelo e seu desvio-padrão pode ser observado na Tabela 3. Já o link para acessar os vídeos dos testes gravados podem ser encontrados na Tabela 4.

Figura 5 – Resultados Treinamento Mountain Car - Recompensa média ao longo dos episódios.



Fonte: Elaborado pelos autores

Dentre os modelos testados o que obteve melhor resultado foi o DQN-1. O agente conseguiu convergir para a solução em todos os testes com menor tempo. A média das recompensas de teste deste modelo foi bem acima dos outros comparados e seu desvio-padrão o menor encontrado, indicando uma menor dispersão em torno da média. No modelo DQN-2 o agente também convergiu para a solução do problema nos exemplos de teste, porém de forma mais lenta em relação ao DQN-1. Já o modelo DQN-3 obteve a menor média de recompensa entre os modelos e maior desvio-padrão. Nos vídeos gravados deste agente, em 3 exemplos ele convergiu apenas uma vez ao objetivo.

Tabela 3 – Resultados Teste Mountain Car - Recompensa média e desvio-padrão.

Modelo	Recompensa média	Desvio-padrão
DQN-1	-108.70	+/- 16.48
DQN-2	-168.20	+/- 20.96
DQN-3	-186.80	+/- 23.08

Tabela 4 – Links para vídeos com desempenho dos agentes

Modelo	Vídeo
DQN-1	<https://encurtador.com.br/fgLU5>
DQN-2	<https://encurtador.com.br/bdDEL>
DQN-3	<https://encurtador.com.br/DKPU8>

3.1.3 Código fonte

O notebook com o código do problema pode ser acessado em <https://colab.research.google.com/drive/1FPwrIreu2rcSuIbYVoxcTAKjyroxbmNe?usp=sharing>

3.2 Car Racing

3.2.1 Treinamento

Para o problema do *Car Racing* o código foi executado 4 vezes, das quais 3 foram alterando apenas o *n_timesteps* para as instâncias foram utilizadas o *n_timesteps* 25.000, 50.000, 100.000 denominados PPO-1, PPO-2 e PPO-3, respectivamente. E uma última instância foi executada para efetivamente verificar um cenário com apenas alguns parâmetros alterados em relação à configuração padrão do modelo PPO denominada PPO-4. A Tabela 5 apresenta os parâmetros escolhidos para cada instância executada.

Tabela 5 – Parâmetros utilizados no problema *Car Racing*

Parâmetro	PPO-1	PPO-2	PPO-3	PPO-4
n_timesteps:	1e5	5e4	2.5e4	1e5
policy:	CnnPolicy	CnnPolicy	CnnPolicy	CnnPolicy
learning_rate:	3e-4	3e-4	3e-4	3e-4
batch_size:	128	128	128	64
gamma:	0.99	0.99	0.99	0.99
gae_lambda:	0.95	0.95	0.95	0.95
use_sde:	True	True	True	False
vf_coef:	0.5	0.5	0.5	0.5
policy_kwargs:	net_arch= [256, 256]	net_arch= [256, 256]	net_arch= [256, 256]	net_arch= [256, 256]

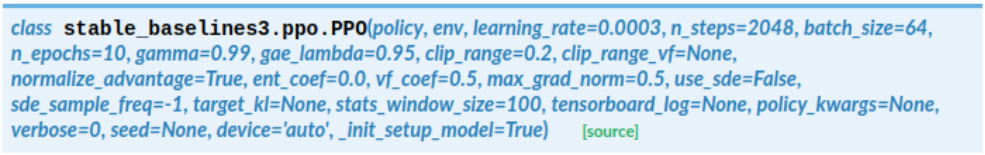
Sobre os principais parâmetros que se mantiveram iguais nos testes:

i) policy: foi escolhida a CnnPolicy, uma política que consegue lidar com múltiplos episódios e sua atualização são em *minibatchs*. De acordo com sua documentação <https://stable-baselines3.readthedocs.io/en/master/modules/dqn.html> essa política junta o A2C, o qual utiliza múltiplos *workes*, e o Trust Region Policy Optimization (TRPO) que usa a região de confiança para melhorar o ator.
ii) learning_rate: a taxa de aprendizado foi mantida a padrão do modelo.
iii) batch_size: valor foi aumentado a fim de fornecer uma estimativa de gradiente mais precisa.
iv) gamma: valor alto, próximo a 1 para dar mais peso a recompensas futuras e assim incentivar o agente a recompensas de longo prazo.
v) use_sde: o Exploração Dependente do Estado (SDE) é uma técnica utilizada no aprendizado por reforço para adaptar a taxa de exploração com base no estado atual do ambiente. Por padrão no PPO ela não é utilizada, porém, no problema de a recomendação

é utilizar para equilibrar a troca entre exploração e explotação.
vi) vf_coef: ajusta o equilíbrio entre a exploração da função de valor e a exploração da função de política durante a otimização, influenciando assim o comportamento do algoritmo de aprendizado por reforço. A decisão de aumentar esse parâmetro foi para aumentar a importância da função de valor o que significa que a política será mais influenciada pelos valores estimados dos estados.
vii) policy_kwargs: foi escolhida a CnnPolicy, uma política que consegue lidar com múltiplos episódios e sua atualização são em *minibatchs*. De acordo com sua documentação <https://stable-baselines3.readthedocs.io/en/master/modules/dqn.html> essa política junta o A2C, o qual utiliza múltiplos *workes*, e o Trust Region Policy Optimization (TRPO) que usa a região de confiança para melhorar o ator.

Um recurso utilizado para avaliar o melhor resultado do modelo foi o *callback*, disponibilizado pela biblioteca do SB3. A medida que o modelo vai aprendendo ele é avaliado com uma frequência e armazenando o melhor resultado. Os demais ficaram iguais ou próximos dos parâmetros base SB3, descritos na Figura 6

Figura 6 – Parâmetros padrão PPO.



Fonte: <https://stable-baselines3.readthedocs.io/en/master/modules/ppo.html>

3.2.2 Teste

A Tabela 6 apresenta os resultados médios e seus respectivos desvios-padrão. O que pode se observar é que ainda não convergiu, porém, a medida que aumenta o parâmetro *n_timestep* mais próximo de convergir ele fica. Entretanto os três primeiros modelos tiveram uma recompensa média baixa em relação ao quarto modelo bem como os seus desvios-padrão.

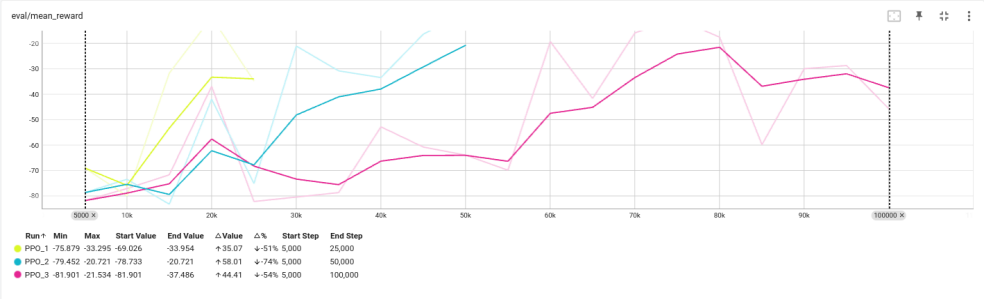
Tabela 6 – Resultados Teste Car Racing - Recompensa média e desvio-padrão.

Modelo	Recompensa média	desvio-padrão
PPO-1	-5.52	+/- 30.49
PPO-2	32.34	+/- 46.66
PPO-3	-40.72	+/- 6.11
PPO-4	520.63	+/- 240.59

A figura 7 apresenta os resultados e evolução da média dos modelos PPO-1, PPO-2, e PPO-3. O objetivo é maximizar a função objetivo, pelo comportamento da curva o PPO-3 não conseguiu sair de um máximo local e sua explotação ficou restrita, fazendo assim, mesmo com maior quantidade de passos que os outros cenários analisados não conseguiu

ficar com a maior média. Porém, o seu desvio-padrão foi menor que em relação ao PPO-1 e PPO-2.

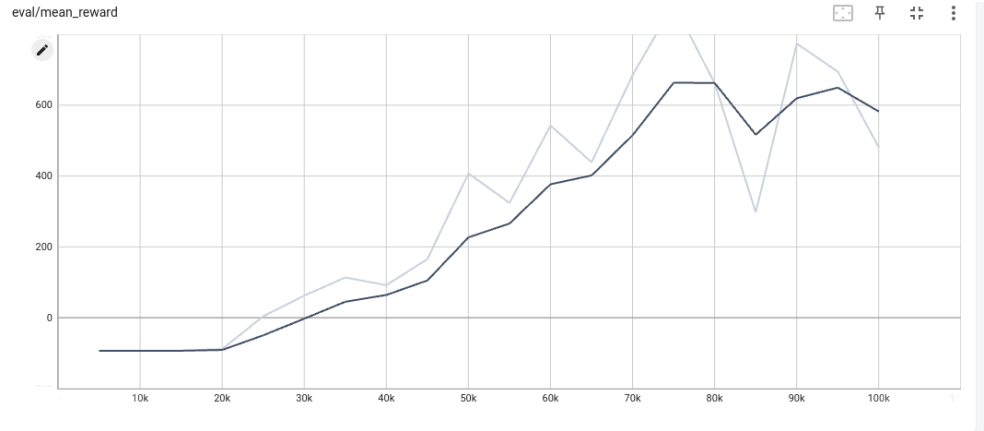
Figura 7 – Resultado *Car Racing* PPO-1, PPO-2 e PPO-3.



Fonte: Elaborado pelos autores.

A figura 8 apresentou o melhor resultado. Diferente do PPO-3 na 30000 iteração já havia saído da recompensa negativa, o que fez com que o agente evoluísse o aprendizado. E assim obtendo melhores resultados. Outro ponto a se analisar é que por volta da iteração 80000 o agente busca estabilizar e, por consequência, inicia um processo de oscilação na recompensa. Portanto, uma quantidade maior de passos é necessário para o agente convergir.

Figura 8 – Resultado do melhor Modelo do *Car Racing* - PPO-4.



Fonte: Elaborado pelos autores.

A Tabela 7 disponibiliza o link dos principais testes executados. Na primeira coluna informa sobre qual instância está se referindo e a coluna 2 são os links do Vídeo. Apenas o PPO-4 conseguiu andar na pista de maneira que completasse a volta.

Tabela 7 – Links para vídeos com desempenho dos agentes

Modelo	Vídeo
PPO-1	<https://encurtador.com.br/isLM2>
PPO-2	<https://encurtador.com.br/qFR02>
PPO-3	<https://encurtador.com.br/cABIW>
PPO-4	<https://encurtador.com.br/cdEQZ>

3.2.3 Código Fonte

O notebook com o código do problema pode ser acessado em <https://colab.research.google.com/drive/1Qi-nsQx01SPPRUVbLsMMf-ZvAuaeFidU#scrollTo=e4cfSXIB-pTF>

REFERÊNCIAS

DOCUMENTATION, Pytorch. *Stable-Baselines3 Docs - Reliable Reinforcement Learning Implementations*. <https://pytorch.org/docs/stable/index.html>. Acesso em: 06 Jul 2023.

PUTERMAN, Martin L. Chapter 8 markov decision processes. In: *Stochastic Models*. Elsevier, 1990, (Handbooks in Operations Research and Management Science, v. 2). p. 331–434. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0927050705801720>.

RAFFIN, Antonin et al. *Stable-Baselines3 Docs - Reliable Reinforcement Learning Implementations*. 2021. <http://jmlr.org/papers/v22/20-1364.html>. Acesso em: 06 Jul 2023.

RUSSELL, Stuart J.; NORVIG, Peter. *Artificial Intelligence: a modern approach*. 3. ed. [S.l.]: Pearson, 2009.