Aprendizagem por reforço aplicada ao Frogger

Reinforcement learning applied to Frogger

A. N. Souza[1]; I. L. L. de Oliveira[2]; L. M. A. Santos[3]; M. V. R. Guimarães[4]; P. M. de J. Costa[5]

Departamento de Computação/CCET, UFS, 49100-000, São Cristóvão-Sergipe, Brasil

[1]xandnuness@academico.ufs.br

[2]isaacl@academico.ufs.br

[3]yazonira@academico.ufs.br

[4]mvrguimaraes@academico.ufs.br

[5]periclesmaikon@academico.ufs.br

Este estudo tem como objetivo explorar a aplicação de técnicas de Aprendizado por Reforço em um agente treinado para jogar o jogo Frogger do Atari, essa escolha foi baseada na dinâmica simples do jogo, mas que apresenta desafios adequados para o aprendizado e avaliação de algoritmos de aprendizado por reforço. Para isso o modelo foi treinado utilizando a biblioteca , que implementa o algoritmo *Deep Q-Network*, dessa forma o agente foi capaz de aprender a maximizar suas recompensas ao interagir com o ambiente. As técnicas escolhidas para a avaliação do agente foram redes neurais convolucionais e perceptrons multicamadas.

Palavras-chave: CNN, MLP, Frogger.

This study aims to explore the application of Reinforcement Learning techniques in an agent trained to play the Frogger game from Atari. This choice was based on the game's simple dynamics, which nonetheless presents appropriate challenges for learning and evaluating reinforcement learning algorithms. To achieve this, the model was trained using the library, which implements the Deep Q-Network algorithm, allowing the agent to learn how to maximize its rewards through interaction with the environment. The techniques chosen for evaluating the agent were Convolutional Neural Networks and Multi-Layer Perceptrons.

Keywords: CNN, MLP, Frogger.

1. INTRODUÇÃO

O aprendizado por reforço (RL) é uma área da inteligência artificial em que agentes são introduzidos em ambientes onde aprendem a tomar decisões com base em recompensas que são

acumuladas ao longo do tempo. Aplicações de RL vem ganhando destaque devido ao sucesso de modelos desenvolvidos pela DeepMind, de acordo com Silver et al. (2017), o AlphaGo foi o primeiro modelo a conseguir derrotar o campeão mundial no jogo Go.

Neste estudo, um agente foi treinado para jogar Frogger, um clássico da plataforma Atari, para isso foi utilizado redes neurais convolucionais (CNN) e perceptrons multicamadas (MLP) para avaliar o desempenho de ambas as tecnologias, buscando entender as vantagens que cada abordagem possui em cenários dinâmicos do Frogger.

O jogo escolhido possui uma dinâmica simples, mas ideal para aprendizagem por reforço; Bellemare et al. (2013) cita a importância dos jogos Atari para a avaliação de agentes RL, pois são eficazes em medir o desempenho deles devido a uma série de cenários com diferentes níveis de complexidade e habilidades exigidas. O objetivo do Frogger é o agente atravessar uma pista movimentada ou um rio com muitos obstáculos sem ser atingido. É um ambiente desafiador o suficiente para o agente aprender a tomar decisões, com a necessidade de lidar com tempo, obstáculos e interações com o cenário.

Como afirmado por LeCun et al. (2015) as CNNs funcionam melhor para lidar com dados visuais, como imagens e vídeos, pois são capazes de identificar padrões por meio de filtros convolucionais. Essa é uma rede ideal para o jogo, pois o agente pode receber quadros das partidas e tomar suas decisões baseando-se em padrões de obstáculos aprendidos.

As MLPs são redes tradicionais que possuem múltiplas camadas com neurônios totalmente conectados, são eficazes em inúmeras tarefas, porém não possuem capacidade de interpretar imagens e vídeos. Esse estudo visa comparar como um modelo mais simples lida em cenários com entrada visual e dinâmica.

2. MATERIAL E MÉTODOS

Ambiente de Experimentação

O ambiente utilizado foi o *Frogger-v5* da plataforma Atari, registrado via *Gymnasium* e ALE (*Arcade Learning Environment*). Para capturar a temporalidade das ações, aplicou-se o pré processamento *VecFrameStack*, empilhando 4 quadros consecutivos do jogo. O ambiente foi vetorizado para permitir treinamento eficiente, com um único ambiente paralelo (*n_envs=1*).

O ambiente de hardware utilizado no treinamento e avaliação dos modelos foi:

- CPU: Ryzen 7 5800X;
- GPU: NVIDIA GeForce RTX 4070;
- RAM: 32GB.

Modificação das Recompensas

Um *wrapper* personalizado, *PunicaoFrogger*, foi implementado para ajustar a função de recompensa. Quando o episódio é finalizado (ex: o agente é atingido), uma penalização de *1* é aplicada à recompensa original, incentivando o agente a evitar falhas prematuras.

Arquiteturas dos Modelos

Foram testadas duas arquiteturas de redes neurais, utilizando o algoritmo *DQN* (*Deep QNetwork*) do *Stable Baselines 3*:

1. *CNN* (Rede Neural Convolucional): Implementada através da política *CnnPolicy*, projetada para processar imagens diretamente. A arquitetura é baseada na *nature CNN, e* inclui 3 camadas convolucionais para extração de características visuais (ex: padrões de obstáculos), uma camada de padding e, por fim, uma camada para linearização (Mazyavkina et al., 2021), ilustrada na figura abaixo.

Layer	input	output	kernel	stride
Conv-1	4	32	8	4
Conv-2	32	64	4	2
Conv-3	64	64	3	1
Padding	—	$121 * 64$	—	—
Linear	$121 * 64$	512	—	—

Figura 1: Descrição da nature CNN

2. *MLP* (*Perceptron* Multicamadas): Baseada na política *MlpPolicy*, com 2 camadas de 64 neurônios completamente conectados. Assume-se que as observações foram achatadas (*flattened*) para adequar-se à entrada tabular.

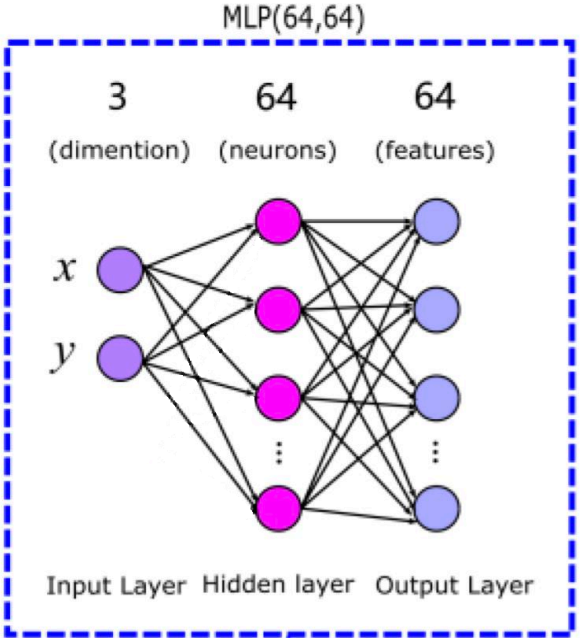

Figura 2: Descrição da MlpPolicy

Hiperparâmetros de Treinamento
Ambos os modelos compartilharam os seguintes parâmetros:

- **Taxa de Aprendizado** (*learning_rate*): 1×10^{-4}.
- **Tamanho do Batch** (*batch_size*): 64.
- **Buffer de Replay** (*buffer_size*): 125.000 experiências.
- **Passos Iniciais** (*learning_starts*): 20.000 (fase de coleta de dados antes do treinamento).
- **Dispositivo de Aceleração**: *GPU* (*CUDA*), quando disponível, via *PyTorch*.

O treinamento foi realizado por 1.000.000 *timesteps* (20 iterações de 50.000 *timesteps* cada), com *checkpoints* salvos a cada 50.000 *timesteps* na pasta *models/CNN* ou *models/MLP*.

Avaliação de Desempenho

Após o treinamento, o modelo foi avaliado em 10 episódios independentes, utilizando a função *evaluate_policy* do *Stable Baselines 3*. As métricas calculadas incluem:

- **Média das Recompensas**: Indicador central de sucesso.
- **Média da Duração**: mostra a taxa de sobrevivência do agente.
- **Desvio Padrão**: Medida da consistência do agente.
- **Renderização em Tempo Real**: Visualização do comportamento do agente via *env.render("human")*.

Ferramentas e Bibliotecas

- ***Stable Baselines 3***: Para implementação do *DQN* e avaliação.
- ***Gymnasium*** e ***ALE-py***: Para criação e gerenciamento do ambiente.
- ***PyTorch***: *Backend* para execução em GPU.
- ***TensorBoard***: Monitoramento de métricas (*ep_rew_mean*, *ep_len_mean*).

Reprodutibilidade

Todos os códigos, hiperparâmetros e configurações estão disponíveis publicamente, garantindo a reprodutibilidade do estudo. Os scripts de treinamento (*frogger_mlp.py*, *frogger_cnn.py*) e avaliação (*gameplay.py*) foram desenvolvidos em ***Python 3.10***, com dependências documentadas no *README.md*.

Aspectos Éticos

Por tratar-se de um experimento computacional sem envolvimento de humanos ou animais, não foi necessária aprovação de comitês de ética.

3. RESULTADOS E DISCUSSÃO

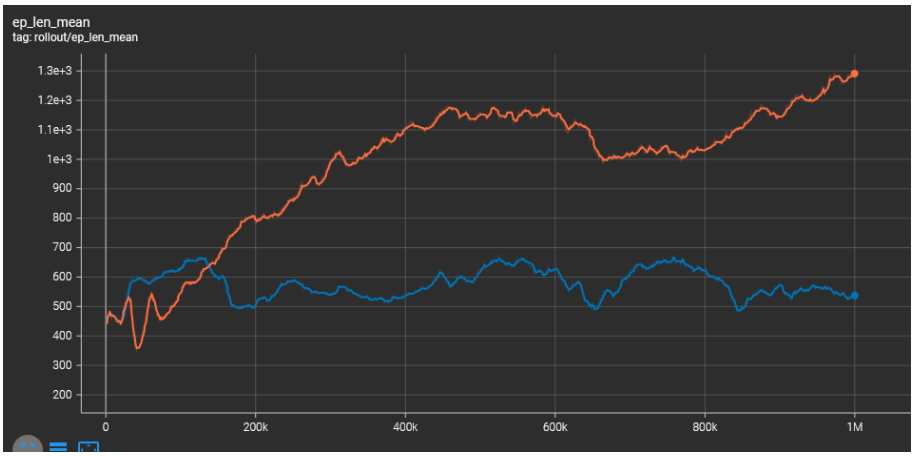

Figura 3: Média de sobrevivência por episódio ao longo do treinamento.

Os achados durante a fase de treinamento e avaliação dos modelos de Rede Neural Convolucional (CNN) e Perceptron Multicamadas (MLP) para o jogo Frogger revelaram diferenças notáveis no desempenho de ambas as estruturas. Nos estágios iniciais, o modelo MLP revelou uma taxa média de sobrevivência por episódio superior à do CNN, indicando que, nas fases iniciais de treinamento, o MLP teve um desempenho mais eficaz em evitar falhas precoces (ver Figura 3). Entretanto, com o avanço do treinamento, o CNN ultrapassou o MLP por uma quantidade significativa, evidenciando um aprendizado mais sólido e adaptável em situações dinâmicas.

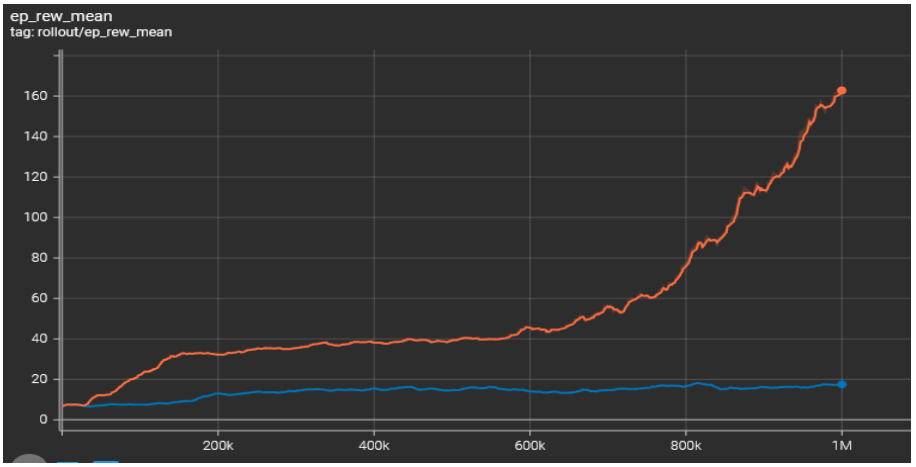

Figura 4: Média de recompensa por episódio ao longo do treinamento.

No que se refere à média de recompensas por episódio, ambos os modelos mostraram desempenhos equivalentes no começo do treinamento, com uma aprendizagem significativa iniciando-se a partir de 20.000 timesteps (ver Figura 4). Contudo, após esse marco, o CNN se destacou, obtendo recompensas consideravelmente superiores ao MLP. Enquanto o CNN continuava a melhorar seu desempenho, o MLP ficou estagnado, com médias de recompensas abaixo de 20 pontos após 700.000 timesteps. Isso indica que o MLP chegou a um ponto de saturação em seu desempenho, enquanto o CNN mostrou potencial para continuar evoluindo.

Um destaque significativo aconteceu com cerca de 700.000 timesteps, quando a CNN demonstrou um grande avanço no aprendizado, elevando substancialmente sua média de recompensas (ver Figura 4). Esse fenômeno pode ser explicado pela habilidade do CNN de identificar padrões visuais complexos no ambiente, o que é fundamental em jogos como Frogger, onde a interpretação visual e a detecção de obstáculos em tempo real são essenciais. Em contrapartida, o MLP, que não conta com a mesma capacidade de processamento de imagens, provou ser ineficaz frente à complexidade do ambiente, resultando em um desempenho inferior e estagnado.

A análise comparativa entre os dois modelos apoia a ideia de que as CNNs são mais proficientes em ambientes de jogos Atari, como Frogger, devido à sua aptidão inerente para processar dados visuais e reconhecer padrões espaciais. Uma vez que tanto MLP quanto CNN foram expostos a condições de treinamento idênticas (mesmo ambiente, parâmetros e recompensas), a superioridade do CNN pode ser atribuída à sua arquitetura, especialmente desenhada para manipular entradas visuais com eficiência. Em contrapartida, o MLP, apesar de ser competente em tarefas com dados tabulares, mostrou menor capacidade de adaptação a situações que necessitam de interpretação visual.

Esses resultados estão de acordo com as previsões teóricas, conforme apontado por LeCun et al. (2015), que afirmam a superioridade das CNNs em tarefas que envolvem dados visuais, como imagens e vídeos. A habilidade do CNN de continuar a melhorar seu desempenho ao longo do tempo, enquanto o MLP se mantinha estagnado, sugere que a arquitetura convolucional possui um potencial maior para aplicações em ambientes dinâmicos e visuais, como é o caso dos jogos Atari.

Em síntese, os achados indicam que, ainda que o MLP tenha mostrado um desempenho aceitável no início, o CNN o ultrapassou amplamente em relação à recompensa média e habilidade de aprendizado contínuo. Isso destaca a relevância de selecionar a arquitetura

apropriada para cada situação, principalmente em contextos que demandam processamento de imagem e ajuste a ambientes em mudança.

4. CONCLUSÃO

A MLP apresentou uma curva de aprendizado inicial rápida tanto que sua média de sobrevivência por episódio supera a do CNN nos primeiros estágios. Contudo, sua capacidade de adaptação a cenários dinâmicos foi limitada.

A CNN se mostrou muito mais eficaz a longo prazo, com uma média de sobrevivência por episódio muito maior que a MLP. No entanto, isso veio ao custo de um maior tempo de treinamento e necessidade de mais recursos computacionais.

É possível reforçar essa diferença analisando a média de recompensa por episódio com a CNN após 1.000.000 de interações passar dos 160 pontos enquanto a MLP após os mesmos 1.000.000 de episódios não conseguiu ultrapassar os 20 pontos.

Uma vez que nosso resultado está longe do estado da arte, para melhorias futuras, é imprescindível o modelo ser submetido a mais rodadas de treinamento, após a análise dos resultados continuamos a treinar apenas o CNN por mais 2.000.000 de *timesteps*, o que nos levou até os 300 pontos (ver Figura 5), uma pontuação razoável para um jogador intermediário. Também é importante ressaltar que outros modelos do *Stable Baselines 3* possuem métodos de priorização da experiência de replay aplicados ao DQN, o que os faz ultrapassar o DQN tradicional em resultados e atinge um novo estado da arte em 41 de 49 jogos de Atari segundo Schaul et al. (2015).

Portanto, a escolha entre MLP e CNN deve levar em consideração o compromisso entre eficiência de aprendizado, custo computacional e a complexidade da tarefa a ser resolvida.

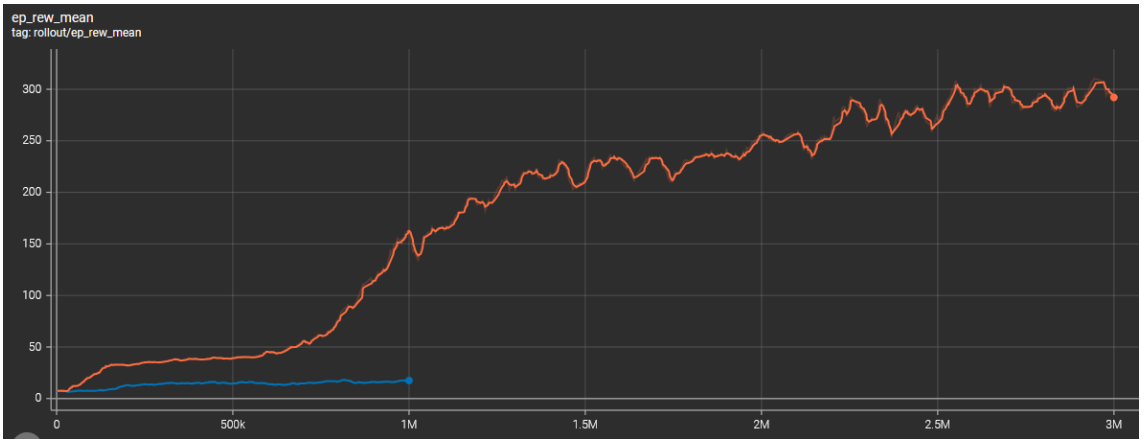

Figura 5: Média de recompensa por episódio da continuação do treinamento.

5. AGRADECIMENTOS

Gostaríamos de agradecer aos desenvolvedores e mantenedores do *Stable Baselines 3* por disponibilizar ferramentas fundamentais para o desenvolvimento deste experimento. Também expressamos nossa gratidão ao professor Hendrik Teixeira Macedo pelo suporte e orientação ao longo deste trabalho, contribuindo significativamente para nossa compreensão e aprimoramento na área de inteligência artificial.

6. REFERÊNCIAS BIBLIOGRÁFICAS

1. Silver D, Schrittwieser J, Simonyan K, et al. Mastering the game of Go without human knowledge. Nature. 2017 Oct;550(7676):354-359, doi:10.1038/nature24270.

2. Mnih V, Kavukcuoglu K, et al. Playing Atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602. 2013.

3. Guo X, Singh S, et al. Deep learning for real-time Atari game play using offline Monte Carlo tree search planning. Adv Neural Inf Process Syst. 2014;27.

4. Mnih V, Kavukcuoglu K, et al. Human-level control through deep reinforcement learning. Nature. 2015 Feb;518(7540):529-533, doi:10.1038/nature14236.

5. LeCun Y, Bengio Y, Hinton G, et al. Deep learning. Nature. 2015 May;521(7553):436-444, doi:10.1038/nature14539.
6. Bellemare MG, Naddaf Y, Veness J, Bowling M, et al. The arcade learning environment: An evaluation platform for general agents. Proc 24th Int Conf Mach Learn (ICML). 2013;214-224.

7. Ye W, Liu S, Kurutach T, et al. (2021). Mastering atari games with limited data. Advances in neural information processing systems, 34, 25476-25488.

8. Bellemare MG, Naddaf Y, Veness J, Bowling M, et al. The arcade learning environment: An evaluation platform for general agents. Proc 24th Int Conf Mach Learn (ICML). 2013;214-224.
9. Mazyavkina, N. & Moustafa, Samir & Trofimov, Ilya & Burnaev, Evgeny. (2021). Optimizing the Neural Architecture of Reinforcement Learning Agents. 10.1007/978-3-030-80126-7_42.
10. Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2015). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.