Uma abordagem de algoritmos de reinforcement learning para atingir performance próxima do nível humano em jogos de atari

João Murackami , Samuel Heinrichs
Faculdade de Tecnologia Bandeirantes – BandTec
Rua Haddock Lobo, 595 – 01414-905 – São Paulo – SP – Brasil
{jnkami@gmail.com, ti.samuelh@gmail.com}

Resumo:

Diversos avanços têm sido feitos na área de inteligência artificial e um dos grandes marcos foi a vitória de um programa de computador, AlphaGo Zero, contra um campeão mundial do jogo de Go. Isso serve como motivação para aprender sobre reinforcement learning e a sua aplicação em outros jogos e ambientes.

Com isso, este artigo tem como objetivo apresentar o tema de reinforcement learning e reproduzir um de seus algoritmos em dois jogos de Atari, a fim de alcançar pontuações consideráveis e mostrar o quão poderoso esses algoritmos são. E também, compara os resultados com a performance humana, como autores expressivos nesse meio já fizeram.

Palavras-chave: Reinforcement Learning, RL, Deep Learning, DQN, Atari, Breakout, Pong, Machine Learning, Deep Q Learning.

1. Introdução

Reinforcement Learning é uma das categorias de algoritmos de machine learning o qual um agente através de observações de um ambiente aprende a maximizar as recompensas de suas ações.

Neste trabalho tem-se como objetivo demonstrar a capacidade dessa categoria de algoritmos de aprender atividades complexas sem ter antes dados anotados como acontece na categoria de algoritmos supervisionados.

Para isso, irá ser desenvolvido agentes utilizando o modelo DQN, o qual utiliza redes neurais, para alcançar uma pontuação próxima ao nível humano em dois jogos de atari (Breakout e pong) através dos frames desses jogos.

No capítulos 2 será apresentado os conceitos e a ligação com a biologia, no capítulo 3 será demonstrado a metodologia bem como as ferramentas utilizadas, no capítulo 4 será mostrado a performance desse agente nesses dois jogos escolhidos em comparação ao nível humano, no capítulo 5, a conclusão sobre os resultados obtidos bem como melhorias e dificuldades encontradas, no último capítulo a bibliografia utilizada.

2. Referencial Teórico

2.1. Reinforcement Learning

Reinforcement Learning (RL) é aprender o que fazer e como mapear situações para ações, com o objetivo de maximizar a recompensa numérica. As ações que devem ser tomadas não são ditas a quem está aprendendo, mas ao invés disso, este deve tentar e descobrir as ações que levam a recompensas maiores.

Neste paradigma, ações podem não só afetar recompensas imediatas, mas também situações e recompensas subsequentes diferentes. Sendo que, as características de descobertas por tentativa e erro e a existência de recompensas adiadas são os mais importantes fatores que distinguem os atributos de RL. (Sutton e Barto, 2018)

Ainda conforme os mesmos autores, o problema de RL é formalizado usando ideias de controle do Markov Decision Processes (MDP), que será ainda discutido neste artigo. Mas, basicamente, refere-se ao ponto central de capturar os principais aspectos que existem no aprendizado de um agente ao agir e interagir em determinado ambiente a fim de completar um objetivo. Sendo assim, MDP tem a pretensão de ter apenas três aspectos em sua forma mais simples: sensação, ação e objetivo, sendo que qualquer método que é feito para resolução desse problema pode ser considerado um método de Reinforcement Learning.

2.1.1. Diferenças entre Reinforcement Learning, Supervised Learning e Unsupervised Learning

Ainda para Sutton e Barto, Supervised Learning é um dos campos de pesquisa mais estudados dentro de Machine Learning, sendo este um tipo de aprendizado que provém de um treinamento com exemplos etiquetados por um supervisor, ou seja, cada exemplo é uma descrição de uma situação em conjunto com uma identificação, que comumente, é categoria na qual essa situação citada pertence.

Em geral, esse tipo de aprendizado é muito importante, porém sozinho não é adequado para o aprendizado por interação, pois nesse tipo de problema, é impraticável obter exemplos de comportamentos que sejam tanto corretos quanto representativos para diversas situações em que o agente precisa agir, basicamente, um agente precisar ser capaz de aprender a partir de sua própria experiência.

RL também se diferencia em relação a Unsupervised Learning, sendo que este tem o objetivo de encontrar estruturas escondidas dentro de dados sem classificação anterior. Os termos Supervised Learning e Unsupervised Learning parecem ser exaustivamente classificados como os paradigmas do Machine Learning, mas eles não são. Sendo assim, Sutton e Barto consideram RL como sendo o terceiro paradigma de Machine Learning em conjunto com Supervised Learning e Unsupervised Learning.

2.1.2. Aplicações de Reinforcement Learning

Conforme os autores acima citados, é preciso também saber alguns exemplos para melhor entendimento do conceito, sendo assim, é possível citar alguns exemplos de RL como: jogos de tabuleiro, como o xadrez; robôs com algum objetivo programado, como robôs de limpeza e robôs de setores industriais; controladores de energia, como otimizar e controlar gastos com refrigeração de sistemas; finanças, com sistemas de *trading*; mídia e propaganda com sistemas de recomendações, entre outros.

2.2. Ligação com a Biologia

A biologia tem sido fonte de inspiração para criação de algoritmos de *machine learning* (ML), os quais têm sido utilizados por pesquisadores para materializar as suas descobertas e fonte de estudo ao passo que cada descoberta está impactando direta e indiretamente todas as áreas da atuação humana.

Um dos exemplos mais comuns desse fato são as redes neurais que são baseadas diretamente no sistema nervoso dos animais onde por uma série de *inputs* que são os estímulos dados pelos sentidos passam por uma gama de neurônios que decodificam esses sinais e associam com experiência passadas para determinar quais são as melhores ações possíveis para aquele cenário.

Na psicologia o behaviorismo é a área de estudo do comportamento fundado por John B. Watson (1878-1958) o qual contrariava os geneticistas da época mostrando que as pessoas eram definidas pelo ambiente e não somente pela sua carga genética. Para provar a sua teoria Watson realizou um experimento controverso conhecido como *Little Albert*: para iniciar mostrou ao bebê Albert uma série de objetos e animais o qual ele não demonstrou medo, logo após sempre que o bebê tentava alcançar um rato um barulho estridente acompanhava a ação, até que então ao ver qualquer animal peludo o pequeno Albert chorava com medo. O experimento é controverso e é tido como anti-ético visto que o bebê carregou as consequências do experimento durante a sua vida.

O trabalho de Watson é fundamentado pelos trabalhos de Ivan P. Pavlov (1849-1936) principal autor na teoria do condicionamento clássico que mostra como as reações a estímulos podem ser aprendidas. No seu experimento conhecido como Cão de Pavlov mostra como um estímulo neutro pode ser transformado num estímulo aprendido e com reações fisiológicas. O experimento consiste em tocar uma campainha ao alimentar o cão até que após algumas repetições o cão salivava apenas de ouvir a campainha.

O estudo moderno da neurociência mostra que a dopamina é neurotransmissor com papel fundamental no sistema de recompensa / punição cerebral e através disso tem sido possível entender até como alguns estímulos geram dependência (como drogas):

> Cada droga de abuso tem o seu mecanismo de ação particular, mas todas elas atuam, direta ou indiretamente, ativando uma mesma região do cérebro: o sistema de recompensa cerebral. Esse sistema é formado por circuitos neuronais responsáveis pelas ações reforçadas positiva e negativamente. Quando nos deparamos com um estímulo prazeroso, nosso cérebro lança um sinal: o aumento de dopamina , importante neurotransmissor do sistema nervoso central (SNC), no núcleo accumbens , região central do sistema de recompensa e importante para os efeitos das drogas de abuso.
>
> O termo reforço, bastante usado na área de Neurobiologia, refere-se a um estímulo que fará com que um determinado comportamento ou resposta se repita normalmente devido ao prazer que causa (reforço positivo) ou ao desprazer e desconforto que é aliviado por meio desse comportamento (reforço negativo). Por exemplo, quando você come uma comida deliciosa como um um bombom de chocolate, mesmo sem estar com fome, o alimento é um reforço positivo. Quando você come uma comida de que não gosta, somente porque está com muita fome e aquela é a única comida disponível, a comida é um reforço negativo, porque alivia uma sensação ruim, de desconforto – a fome. (Formigoni et al, 2017)

2.3. Finite Markov Decision Processes

Finite Markov Decision Processes ou simplesmente MDPs Finitas é uma formulação clássica de processos de decisão sequencial, onde ações influenciam não somente recompensas imediatas, mas também situações subsequentes, como estados e futuras recompensas. (Sutton, Barto 2018)

Em tal interface, quem aprende e toma as decisões é chamado de agente, enquanto que a entidade que interage com esse agente é chamada de ambiente. Essas duas entidades interagem entre si continuamente, com o agente selecionando ações e o ambiente respondendo a essas ações e apresentando novas situações para o agente. O ambiente também é responsável por apresentar recompensas, que são valores especiais que o agente tenta maximizar a partir das escolhas de suas ações. (Sutton, Barto 2018)

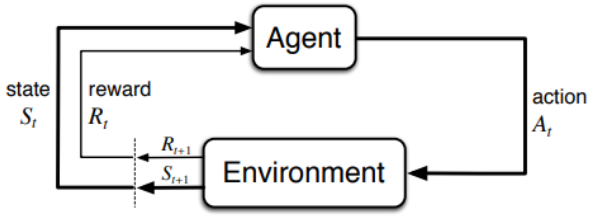

Figura 1 - Interface agente-ambiente no MDP

Para se entender melhor qual a formulação matemática e lógica usada por algoritmos de RL que fazem uso de MDPs, é necessário também entender alguns elementos chave, como a função retorno, função de valor, equações de *Bellman* e função Q, porém, neste artigo tais conceitos não serão apresentados. Contudo, é necessário definir alguns fatores que também fazem parte dessa interface agente-ambiente e que são base para o algoritmo utilizado neste trabalho e muitos outros existentes nesse meio. Basicamente, MDP é uma tupla (S, A, P, R, γ):

- S é um conjunto finito de estados
- A é um conjunto finito de ações
- P é a matriz de probabilidade de transição (probabilidade de a partir de um estado S e uma ação A, chegar-se no estado S')
- R é a função recompensa
- γ é fator de desconto, sendo o quanto de desconto será feito para recompensas futuras, onde $\gamma \in [0, 1]$

2.4. Deep Q Network (DQN)

DQN ou Deep Q Network é, basicamente, um algoritmo que conecta os algoritmos de RL com *Deep Neural Networks* (Minih et al, 2013).

Uma técnica bem estabelecida relacionada ao DQN é o *Q-learning*, onde utiliza as funções Q para ter sucesso, já a DQN utiliza *deep neural networks* como funções Q e assim representar o par estado ação. (Lin et al, 2018).

Fatores como *target network e experience replay,* são componentes importantes para o algoritmo da DQN, porém os detalhes não serão explicados neste artigo, mas o *pseudocode* que resume a aplicação utilizada na metodologia deste artigo será apresentado abaixo:

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Figura 2 - Algoritmo DQN e *pseudocode* do mesmo apresentado por Minih et al. 2013

Ainda para Lin, outras melhorias ao DQN surgiram com o tempo, como: *Double-DQN* (DDQN) (Van Hasselt et al., 2016), *Prioritized experience replay* (Schaul et al., 2016) e *Dueling networks* (Wang et al., 2016).

3. Metodologia

Nesta parte do artigo, será apresentado como serão feitas as execuções e modelagens do algoritmo apresentado no referencial teórico (DQN) usando dois jogos de Atari (Pong e Breakout), visando treinar um agente para que seja capaz de ter performance próxima à humana em ambos os ambientes.

Isso será feito a partir do treinamento de agentes, recebendo pixels do estado de determinado jogo como entrada de uma rede convolucional e, a partir disso, escolhendo a melhor ação que se encaixe nesse estado, com o objetivo de maximizar a pontuação do jogo.

O agente recebe os sinais de recompensa (seja positivo ou negativo) a partir do próprio ambiente do jogo, sem que necessariamente tenha que ser feita uma programação para tal, ou seja, é necessário criar a interface de treinamento e de ação do agente dentro do ambiente, para poder receber os sinais do ambiente como resultado, conseguindo assim, treinar o agente.

3.1. Ferramentas

Neste trabalho, a execução dos procedimentos será dada a partir da linguagem de programação Python e uso de *frameworks* como *Tensorflow* e *Keras*, esses em conjunto da biblioteca *gym*, sendo que esta última oferece uma gama de jogos e ambientes no qual é possível aplicar soluções de RL com Python.

3.2. Ambientes (Jogos de Atari)

Como citado acima, os jogos de Atari que serão utilizados neste trabalho, a partir da biblioteca *gym* são: o jogo Pong e o jogo Breakout. Ambos são jogos simples e possuem o objetivo de adquirir a maior quantidade de pontos possíveis. No pong, a

maior pontuação é 21 enquanto que no Breakout existe a pontuação máxima de 448 pontos por tela.

Os estados (dado por pixels), recompensas e ações disponíveis para cada jogo, estão já intrinsecamente disponíveis no ambiente do jogo em questão, teoricamente não seria necessário a alteração de nenhum parâmetro desses para o agente aprender a jogar. Porém, para o jogo de Breakout, foi alterado a parte do código em questão, para considerar uma recompensa de -10 pontos a cada vez que o agente perde uma vida (em treinamento apenas). Isso foi feito para facilitar o entendimento do agente que perder uma vida é um comportamento indesejável nesse ambiente, pois inicialmente no ambiente original, não existe recompensas negativas (abaixo de zero) para qualquer ação, somente recompensas positivas.

As imagens de ambos os jogos estão representadas abaixo, Pong e Breakout respectivamente:

Figura 3 - Imagens retiradas da biblioteca gym

3.3. Pré-Processamento e Processamento

Para o processamento dos pixels dos jogos, primeiro, é necessário realizar um pré processamento na imagem de cada jogo. Basicamente são realizados os seguintes passos:

- É cortado partes que não serão necessárias para o agente aprender (como por exemplo o número de vidas e a pontuação);
- É feito também a escala da imagem para 84 x 84;
- Alterar a imagem para a escala de cinza;
- Finalmente, normalizado a matriz de pixels dividindo tudo por 255.

Após isso, é realizado o agrupamento de quatro imagens consecutivas e processadas como um estado único do jogo, isso é feito recorrentemente para o agente saber para qual direção os componentes do jogo se deslocam. Enfim, esse estado processado é passado como entrada para uma rede convolucional, cujo resultado é o valor Q de cada ação possível no jogo. Uma representação da estrutura é mostrada na imagem abaixo:

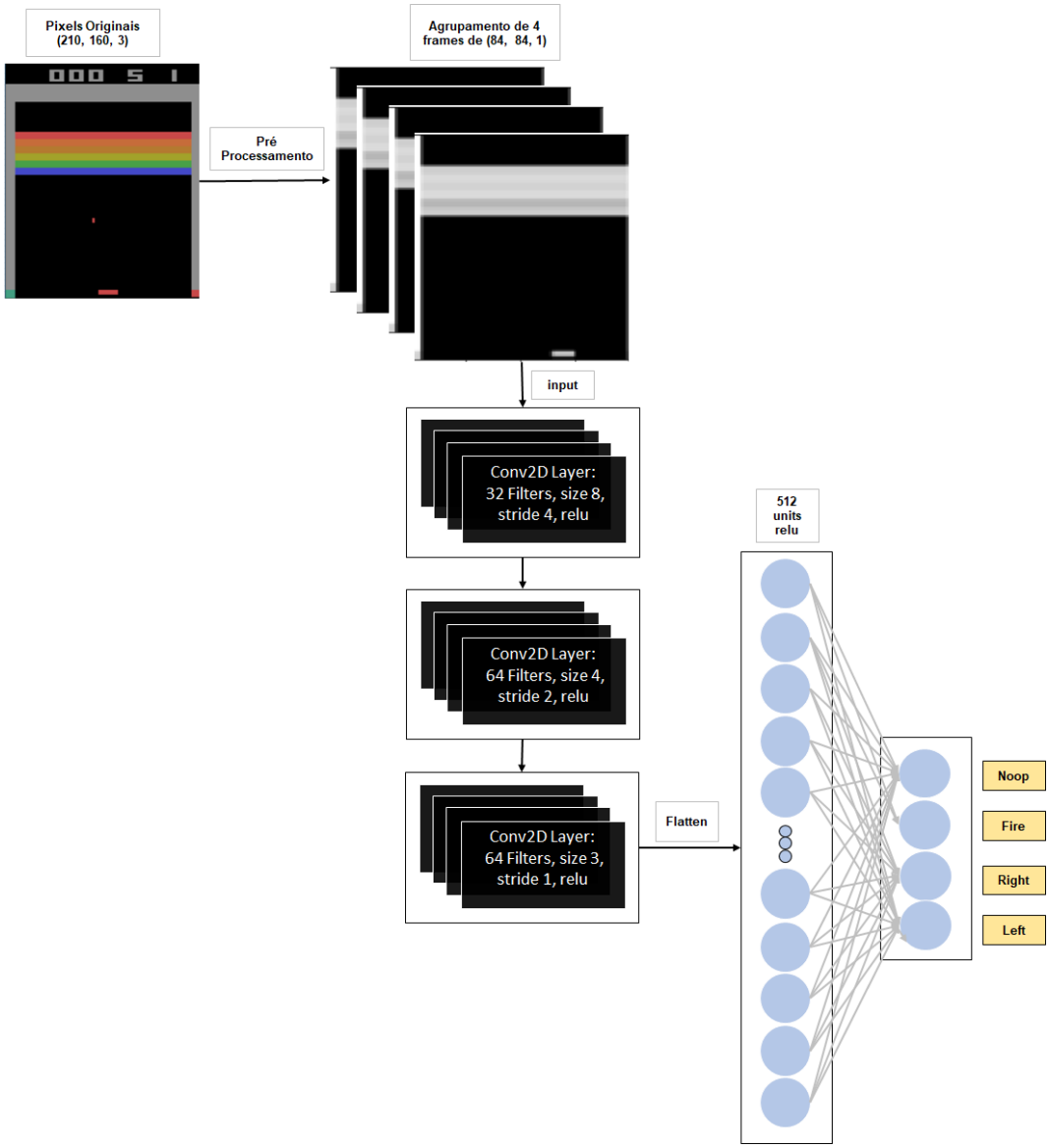

Figura 4 -
Imagem do pré processamento e rede convolucional

Nas camadas de convolução, também foram utilizados os parâmetros: *no bias* e inicialização variada para todas as camadas da rede.

Essa rede processa todo estado que é encontrado nos jogos, conforme o *pseudocode* apresentado na figura 2, e a partir de cada estado é escolhido uma ação a ser feita, dependendo do valor encontrado.

O algoritmo feito para o agente aprender é o mesmo para os dois jogos, ou seja, a figura 4 representa o mesmo esquema de estrutura para o jogo Breakout e o jogo Pong, contudo, a única mudança para que diferencia os dois jogos são alguns hiperparâmetros que serão exemplificados na parte de resultados deste artigo.

3.4. Nível Humano

O nível humano que será considerado neste trabalho não é a capacidade total ou o recorde já feito por um ser humano, mas sim, a partir da performance de pontuação média depois de duas horas jogando cada jogo (Minih et al. 2013). Os valores para cada jogo são os seguintes:

Jogo	Pontuação
Pong	-3
Breakout	31

Logo, o objetivo principal é reproduzir o *Deep Q Network* e alcançar resultados próximos ao que o nível humano foi capaz de realizar. Fazendo com que o agente aprenda a jogar o jogo sem que seja necessário programar ele para tal, fazendo com que ele aprenda de acordo com as experiências passadas por ele, tanto positivas quanto negativas.

4. Resultados

Com o ambiente e estruturas preparadas, só é necessário a realização do treinamento do agente nos dois diferentes jogos e apurar os resultados

4.1. Pong

Além de toda estrutura apresentada acima, o treinamento do agente no Pong foi feito a partir de 100.000 iterações, sendo que para cada iteração, 25 ações no jogo foram realizadas e um treinamento em batch da rede convolucional. Além disso, a cada 500 iterações o hiperparâmetro *epsilon* era multiplicado por 0.99, sendo reduzido conforme o tempo, sendo que este, iniciou o treinamento com o valor de 0.5. O *experience replay* possuía tamanho máximo de 50.000 tuplas.

Conforme o treinamento foi sendo realizado, foi avaliada a performance do agente em três jogos a cada 100 iterações, os resultados se deu pela média da pontuação desses três jogos, conforme o gráfico abaixo:

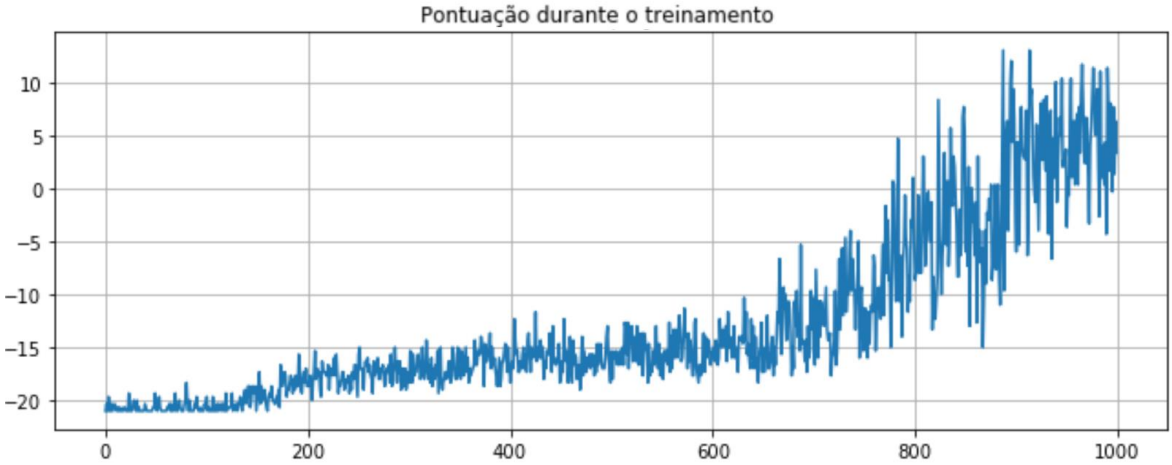

Figura 5 - Média de pontuação em Pong conforme andamento do treinamento

Após o treinamento, os pesos da rede convolucional são salvos e é possível realizar o teste do agente no ambiente em questão. Testando o agente em 500 jogos de Pong com epsilon de 0.01, foi obtido os seguintes resultados:

Jogo	Média	Desvio	Máximo	Mínimo
Pong	19.51	0.88	21.00	14.00

A distribuição seguiu-se conforme o gráfico abaixo:

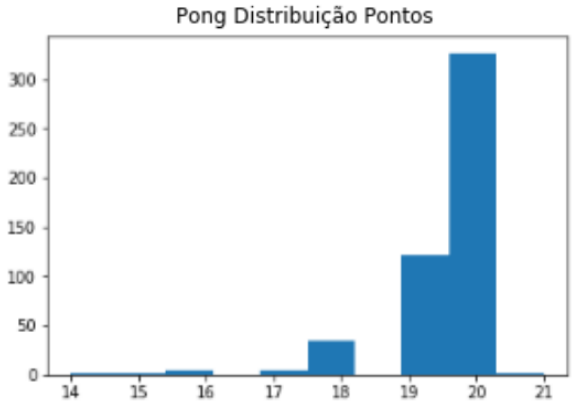

Figura 6 - Histograma da pontuação do agente em 500 jogos

Percebe-se então, que o agente de Pong conseguiu uma performance superior ao nível humano caracterizado neste artigo. É possível verificar também que o agente não possui tanta variação de resultados, possuindo uma média alta e um desvio pequeno, e também, a pontuação mínima encontrada ainda sim conseguiu ser superior ao nível humano, que foi de -3.

O agente nesse caso, conseguiu aprender a jogar Pong e ter uma boa performance no mesmo.

4.2. Breakout

Já para o agente de breakout, o treinamento se seguiu parecido com o agente de Pong, porém, o *epsilon* foi iniciado a 0.65 e o *experience replay* tinha espaço máximo de 70.000 tuplas. Além disso, o número de iterações era de 200.000 e o número de ações por iteração aumentava em 0.5 a cada 100 iterações, iniciando-se em 10. Como no Pong, o agente era testado também a cada 100 iterações, jogando 3 jogos e capturando a média desses jogos, como é possível verificar no gráfico abaixo:

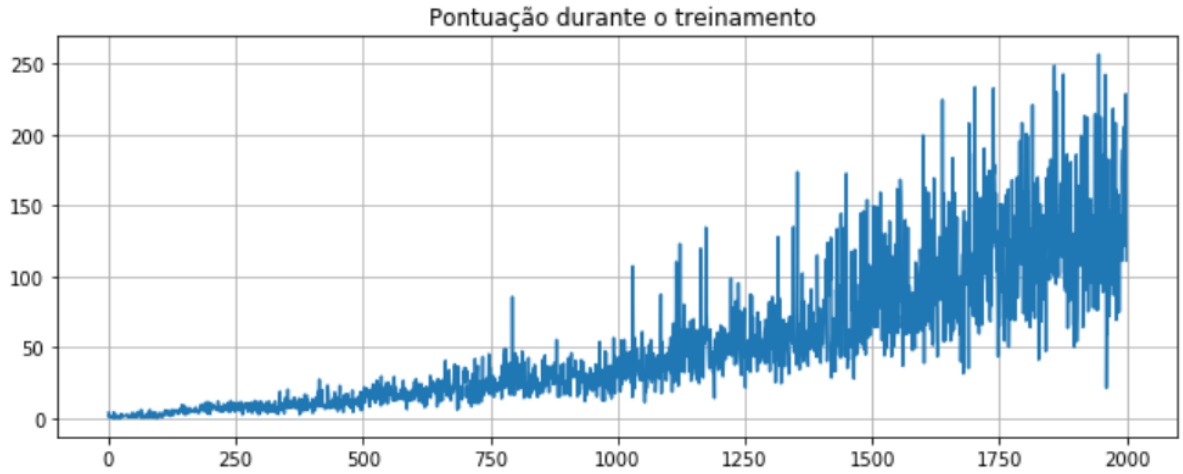

Figura 7 - Média de pontuação em Breakout conforme andamento do treinamento

Depois de realizado o treinamento, foi feito o teste do agente no jogo Breakout, jogando 500 jogos com *epsilon* de 0.01. Os seguintes resultados foram encontrados:

Jogo	Média	Desvio	Máximo	Mínimo
Breakout	208.48	59.89	361.00	41.00

A distribuição seguiu-se conforme o gráfico abaixo:

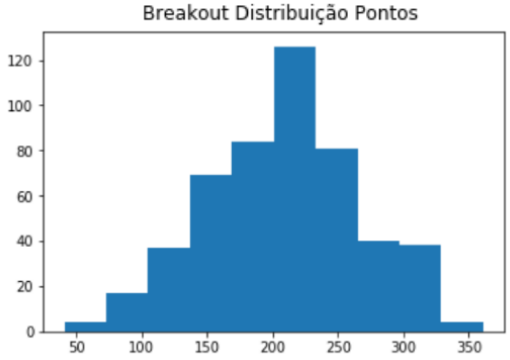

Figura 8 - Histograma da pontuação do agente em 500 jogos

Conforme os resultados é possível perceber que o agente conseguiu aprender a jogar Breakout, com bons resultados, conseguindo a performance máxima de 361 pontos e uma média de 208 pontos aproximadamente.

Porém, é notória que esse agente possui uma maior variação que o agente de Pong, com um desvio de aproximadamente 28% da média. Mesmo possuindo uma variação maior, ainda assim, o agente de Breakout conseguiu um resultado superior ao nível humano (31), conseguindo superar consideravelmente nos seus melhores resultados, inclusive nos mais próximos à média. Como é um jogo mais complexo que Pong, o mesmo algoritmo não conseguiu chegar em uma performance que conseguisse chegar próximo a pontuação máxima, mas mesmo assim, conseguiu atingir bons resultados.

Como resumo dos resultados, segue a tabela abaixo:

Jogo	**Pong**	**Breakout**
epsilon inicial	0.5	0.65
Iterações	100.000	200.000
Experience replay tamanho	50.000	70.000
Ações por iteração	25	10 + (0.5 * iterações/500)
Pontuação Média	19.51	208.48
Nível humano	-3	31

5. Conclusão

Após a execução do código e avaliação dos resultados, é possível verificar que o agente conseguiu aprender a jogar ambos os jogos apresentados neste artigo, conseguindo resultados superiores ao nível humano apresentado, sendo que para o jogo Pong, o agente consegue uma pontuação quase perfeita e no Breakout, o agente consegue chegar a pontuações até acima de 300 pontos, com uma média de 208, consideravelmente acima do nível humano. Ainda assim, é possível aprimorar os agentes que foram treinados com a *Double-DQN, Dueling Networks* e *Prioritized Experience Replay*, sendo essas algumas sugestões para não só melhorar a estabilidade e performance do agente, como também talvez reduzir a quantidade de iterações necessários para se atingir determinada pontuação.

Ademais, é preciso citar que muitos desafios foram encontrados para a execução desta tarefa, como todo o aprendizado de um tema complexo e a execução do mesmo para atingir o objetivo e resultados encontrados. A leitura e estudo de diversos autores que realizaram diferentes abordagens de algoritmos de RL, tanto em outros jogos como também em jogos de Atari foi fundamental. Um dos maiores problemas foi lidar com o tempo de treinamento desses algoritmos, que é consideravelmente longo, sendo que para se encontrar uma boa configuração de hiperparâmetros e descobrir o porquê o agente não está conseguindo aprender de maneira satisfatória também é igualmente difícil.

De forma geral, a partir dos resultados obtidos foi possível perceber o quão poderoso esses algoritmos de RL são, pois é possível realizar o treinamento e fazer com que um agente efetivamente aprenda a realizar tarefas complexas apenas com imagens como entradas de suas rede neurais, o que abre uma possível gama de outras aplicações que estes algoritmos são capazes de auxiliar e executar.

6. Bibliografia

[1] FORMIGONI, M. **Neurobiologia: Mecanismos De Reforço E Recompensa E Os Efeitos Biológicos E Os Efeitos Comuns Às Drogas De Abuso.** Aberta, 2017.

[2] LORICA, B. **Practical applications of reinforcement learning in industry**. Disponível em <https://www.oreilly.com/ideas/practical-applications-of-reinforcement-learning-in-industry> Acesso em: 20 setembro 2018.

[3] SUTTON, R.; BARTO, A. **Reinforcement Learning: an Introduction**, second edition, The MIT Press, 2018.

[4] ALPHAYDIN, E. **Introduction To Machine Learning**, The MIT Press, 2014.

[5] SILVER, D. **Lecture 2: Markov Decision Processes**. Disponível em <http://www.aberta.senad.gov.br/medias/original/201704/20170424-094615-001.pdf> Acesso em: 20 setembro 2018.

[6] MINIH, V. et al. **Playing Atari with Deep Reinforcement Learning**, DeepMind Technologies, 2013.

[7] LIN, L. **Self-Improving Reactive Agents Based On Reinforcement Learning,** Planning and Teaching, Kluwer Academic Publishers, 1992.

[8] HASSELT, H. GUEZ, A. SILVER, D. **Deep Reinforcement Learning with Double Q-learning,** Google DeepMind, 2016.

[9] SILVER, D. Hassabis, D. **AlphaGo Zero: Learning From Scratch**. Disponível em <https://deepmind.com/blog/alphago-zero-learning-scratch/> Acesso em: 25 setembro 2018.