Aprendizagem por Reforço Profundo aplicado a veículos autônomos

Gabriel Henrique Serapião Tavares[1], **Antonio Luiz Basile**[1]

[1]Faculdade de Computação e Informática – Universidade Presbiteriana Mackenzie
São Paulo – SP – Brazil

{contato.ghtavares, albasile2}@gmail.com

Abstract. *The automotive industry has a big market and receives a lot of investments, that is why this is a field of great innovations. A technology that has been growing in this area is Artificial Intelligence applied to vehicle control, in a way that one day it becomes fully independent. This would bring many improvements for humanity, in terms of mobility access and also the many accidents that could be avoided, taking into consideration that the majority of car accidents are caused by human error. The present study has the objective of explore and deepening the methods of Deep Reinforcement Learning applied to autonomous vehicles.*

Resumo. *O setor automotivo possui um grande mercado e recebe grandes investimentos, por esse motivo é uma área de grandes inovações. Uma tecnologia que vem ganhando cada vez mais espaço nessa área é a Inteligência Artificial aplicada ao controle do veículo, de forma que este se torne cada vez mais independente. Isso traria diversos avanços para a humanidade quando se trata de maior acesso à mobilidade, além dos vários acidentes que poderiam ser evitados, tendo em vista que a maior parte dos acidentes de trânsito são oriundos de falhas humanas. Com isso, o presente estudo tem como objetivo explorar e aprofundar os métodos de Aprendizagem por Reforço Profundo aplicados aos veículos autônomos.*

1. Introdução

A ideia de se construir carros autônomos não é nova e recebe grandes investimentos a muitos anos. Um dos mais conhecidos foi o Grand Challenge organizado pela Defense Advanced Research Projects Agency (DARPA) lançado em 2003.

O objetivo do desafio era desenvolver um carro autônomo capaz de dirigir no deserto em altas velocidades, devido ao sucesso da competição a DARPA continuou incentivando o desenvolvimento dos veículos autônomos nos anos seguintes, trazendo a competição para cenários urbanos, involvendo a interação com outros veículos.

Reduzir o número de acidentes é um dos objetivos da construção de um veículo autônomo, tendo em vista que o fator humano é o principal motivo dos acidentes de trânsito [Ruffo 2018] e até 2017 os acidentes de trânsito custaram R$ 36 bilhões por ano no Brasil [Observatório Nacional de Segurança Viária 2018].

Mais preocupante que os valores são os resultados de tais acidentes, em 2018 foram 1,35 milhões de vítimas fatais ao redor do mundo e é a principal causa de morte de crianças e jovens entre 5 e 29 anos [World Human Organization 2018].

Além da redução no número de acidentes, esses veículos facilitariam a mobilidade permitindo até mesmo que crianças e pessoas com deficiência se locomovessem de forma mais independente sem a necessidade de um responsável pela direção.

2. Objetivo

Um veículo totalmente autônomo trata-se de um veículo de transporte equipado com sensores que auxiliam um sistema de controle em seu objetivo de navegar autonomamente até um local desejado, ou seja, sem a necessidade de atuação direta ou indireta de um condutor humano. Esse sistema deve ser capaz de dirigir de maneira segura e consistente ao longo de todo seu percurso.

Por se tratar de uma área muito abrangente, o foco deste estudo é na aplicação das técnicas de Aprendizagem por Reforço no treinamento dos sistemas utilizados pelos veículos autônomos, com o objetivo de manter a direção do veículo dentro de uma pista em um ambiente simulado.

3. Referencial Teórico

Veículos autônomos podem ser classificados de formas diferentes a depender do nível de responsabilidade imposto ao sistema, existem no total seis níveis de automação, variando de zero a cinco, aos quais um veículo pode ser enquadrado [Lavaimaa 2018], essa taxonomia mostra a complexidade de se solucionar o problema, tendo em vista que até hoje nenhuma empresa foi capaz de atingir o nível 5 (total automação). Para alcançar o objetivo de produzir um sistema inteligente capaz de controlar de forma autônoma um veículo é preciso cumprir uma série de tarefas, as tarefas podem ser representadas por três principais: percepção, planejamento e controle [Pendleton et al. 2017].

Percepção engloba as tarefas de localizar e entender o ambiente com o qual o sistema está interagindo, isso inclui a detecção de semáforos, pedestres, vias, outros veículos, etc. Isso é possível utilizando sensores que captam informações a respeito do estado atual do veículo e o seu entorno, os sensores mais conhecidos e mais utilizados para essas tarefas são as câmeras e os LIDARs. Métodos de computação visual para a percepção utilizando câmeras envolvem muitas vezes o uso de Redes Neurais Convolucionais (CNNs), como é o caso do detector de objetos YOLO, cuja proposta é realizar a detecção de objetos com baixa latência [Redmon et al. 2016]. Tesla é uma das empresas que trazem grandes avanços para o mundo dos veículos autônomos através do seu Autopilot, sua peculiaridade em relação aos competidores do mercado é o fato da empresa não optar pela utilização dos LIDARs, possuindo uma forte ênfase no processamento das imagens das 8 câmeras espalhadas pelos veículos da marca [Templeton 2019].

Planejamento refere-se a definição das atitudes a serem tomadas a longo e curto prazo, como é o caso da rota necessária para se chegar à determinado local ou das ações necessárias para desviar de um obstáculo, usando das informações adquiridas pela percepção. Em casos menos complexos o planejamento de rota até o destino pode ser feito por algoritmos de busca em grafos dirigidos como o algoritmo de Dijkstra [Dijkstra 1959] e o A^* [Hart et al. 1968].

O controle é onde são realizadas as intenções geradas pelo planejamento, traduzindo para um comando entendível pelo hardware. Os algoritmos responsáveis pela tarefa direta de controle estão fora do escopo do presente estudo.

Apesar da possibilidade de separar as tarefas de um veículo autônomo em sistemas ultraespecialistas existe também a possibilidade da criação de um sistema fim-a-fim, onde é deixado para o próprio sistema aprender a lidar com as peculiaridades de cada tarefa. Um grande estudo que mostrou a possibilidade de criar um sistema fim-a-fim foi realizado pela NVIDIA [Bojarski et al. 2016], nele foram utilizadas CNNs para mapear de forma supervisionada as capturas das câmeras de um veículo para a direção realizada pelo motorista. Essa abordagem é limitada apenas a manter o veículo na pista, sendo assim outros estudos ampliando a capacidade de tal método foram realizados, para isso adicionaram como entrada ao sistema, além dos pixels das câmeras, a direção desejada para seguir [Codevilla et al. 2018].

3.1. Aprendizagem por Reforço

Outra forma com muito potencial de alcançar a aprendizagem fim-a-fim é utilizando da Aprendizagem por Reforço, seus métodos se mostraram capazes de realizar diversas tarefas associadas principalmente a jogos e ambientes simulados, ultrapassando a performance humana em muitas delas [Mnih et al. 2013] e [Silver et al. 2017]. Além de aprendizagem no ambiente simulado [OpenAI et al. 2019] mostrou a capacidade do método para aplicações no mundo real treinando um braço robótico para ser capaz de manusear objetos.

Aprendizagem por Reforço é uma forma de Aprendizagem de Máquina inspirada na forma que os animais aprendem, através do recebimento de recompensas (reforço). Dessa forma a AR é usada para problemas onde existe a necessidade de aprender uma política de tomadas de decisão dentro de um ambiente por meio de tentativa e erro.

Para formular matematicamente os problemas de AR é geralmente utilizado o Processo de Decisão de Markov (PDM), ele estabelece um *framework* para mapear as interações entre um agente e um ambiente, ou seja, o PDM é usado para formular uma sequência de tomadas de decisão, onde as ações influenciam tanto as recompensas imediatas, mas também as situações subsequentes, e dessa forma impactando também as recompensas futuras [Sutton and Barto 2018].

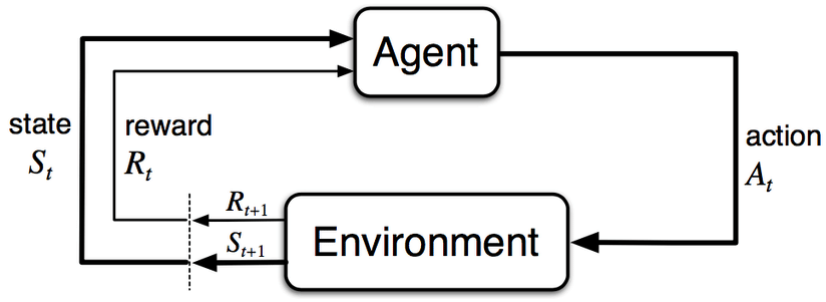

Figura 1. Relação Agente-Ambiente em um PDM. Fonte: [Sutton and Barto 2018]

A Figura 1 apresenta a clássica interação agente-ambiente em um Processo de Decisão de Markov. Em intervalos discretos de tempo (t) o agente recebe uma representação de seu estado ($S_t \in S$) e com base nisso realiza uma ação ($A_t \in A$) no ambiente, recebendo como resposta uma nova recompensa ($R_{t+1} \in R \subset \mathbb{R}$) e um novo estado ($S_{t+1}$).

Para um PDM finito, a função que define a probabilidade de transição é dada por:

$$p(s', r|s, a) \doteq P(S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a) \tag{1}$$

Para medir o desempenho do agente no cumprimento de um objetivo é utilizado da somatória dos retornos dentro de um período de tempo (G_t), que é definido da seguinte forma:

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+2} + ... + R_T \tag{2}$$

Para tarefas contínuas, que não possuem um estado final, é necessário acrescentar um fator de desconto, $\gamma \in \mathbb{R} \mid 0 <= \gamma <= 1$, nas recompensas:

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \tag{3}$$

o fator de desconto define a importância dada pelo agente a recompensas futuras, quanto maior o valor de γ, maior a relevância das recompensas futuras.

3.2. Q-Learning

Q-Learning é um dos algoritmos usados para resolver problemas de Aprendizagem por Reforço, ele é considerado *off-policy* devido ao algoritmo conseguir aprender com ações tomada fora de sua política atual, sendo capaz de aprender mesmo com ações aleatórias [Data Science Academy 2021].

A função *valor-ação* (q_π), também conhecida como *Função Q*, é a função a qual o algoritmo busca aproximar, ela determina o valor de uma ação a em um estado s sob uma política π:

$$q_\pi(s, a) \doteq \mathbb{E}_\pi[G_t|S_t = s, A_t = a] \tag{4}$$

Como a função mostrada na Fórmula 4 indica qual o maior valor para uma ação em um determinado estado, temos que uma política ótima (q_*) pode ser alcançada utilizando os maiores valores para as combinações de estados e ações:

$$q_*(s, a) \doteq \max_\pi q_\pi(s, a) \tag{5}$$

Assim como em muitos métodos de AR, o Q-Learning aproxima a função *valor-ação* utilizando a *Equação de Bellman*:

$$Q(s, a) = r(s, a) + \gamma \max_{a'} Q(s', a') \tag{6}$$

Para aplicar o algoritmo é utilizado uma tabela denominada *Q-Table*, nela são mapeados os valores de $Q(s, a)$, dessa forma para a utilização da *Q-Table* os estados s e as ações a devem ter valores discretos.

```
1 Iniciar a Q-Table Q(s,a) de forma arbitrária
2 Iniciar o Agente para interagir com o Ambiente
3
4 Por N episódios, faça:
```

```
    Inicializar estado s
    Enquanto s não for um estado terminal:
        Escolher uma ação a usando a política derivada de Q
        Executar ação a, observar recompensa r e próximo estado s'
        Q(s,a)+ = α * (r + γmax_{a'∈A}Q(s',a') − Q(s,a))
        s = s'
    Fim
Fim
```

Exemplo de código 1. Pseudocódigo do Q-Learning

Um episódio representa uma sequência de interações feitas dentro do ambiente, a variável α representa a taxa de aprendizagem do agente, s' representa o novo estado recebido após executar a ação a e $max_{a' \in A} Q(s', a')$ é qual o maior valor esperado para s' de acordo com a política atual.

3.3. Deep Q-Learning

O *Deep Q-Learning* é um método de Aprendizagem por Reforço Profundo (APR), *off-policy* e aprende a função *valor-ação* utilizando a *Equação de Bellman*, assim como o *Q-Learning*. A diferença entre os métodos acontece na forma como a *Função Q* é representada, enquanto o *Q-Learning* utiliza da *Q-Table*, o *Deep Q-Learning* faz o mapeamento através de uma rede neural [Data Science Academy 2021].

A entrada para a rede neural é a representação do estado do agente e como saída os valores da *Função Q* para cada ação. Devido ao fato das redes neurais aceitarem números reais como entrada, o algoritmo não possui a mesma limitação dos estados necessitarem de valores discretos como é o caso do *Q-Learning*. Diversas arquiteturas podem ser usadas para a rede neural, para lidar com imagens como entrada é muito comum a utilização de CNNs.

Experience Replay é uma técnica utilizada para aumentar a estabilidade do algoritmo e evitar o esquecimento das situações já passadas. Nela os fatores vivenciados são armazenados em uma memória no formato da 4-tupla (s_t, a_t, r_t, s_{t+1}), podendo ser uma 5-tupla armazenando também se s é um estado terminal, e dentro de cada iteração o processo de atualização dos valores da *Função Q* são feitos utilizando tais amostras [Mnih et al. 2013].

Atualizar os pesos da rede neural para aproximar a *Função Q* utilizando valores gerados pela própria política que tomou a decisão pode criar correlações indesejadas, gerando uma instabilidade no treinamento. É possível diminuir esse potencial erro na estimativa da política ótima separando a rede neural que realiza a tomada de decisão (*Policy Network*) da rede neural que atualiza seus pesos para aproximar a função *valor-ação* (*Target Network*) [Torres 2020].

Em Aprendizagem por Reforço o termo *Exploration vs Exploitation* é utilizado para comparar quando o agente deve seguir somente sua política para a tomada de decisão ou tomar ações que não são baseadas na maior estimativa de valor, a ideia por trás disso é que para o agente aprender sobre novos estados que levariam a um maior valor que o atual estimado, ele deve explorar o ambiente para adquirir experiências. Uma política $\epsilon-greedy$ utiliza de uma probabilidade ϵ para que uma ação aleatória seja tomada, dessa forma, é comum começar com uma probabilidade maior de exploração e atualizar o valor

de ϵ gradativamente para que cada vez mais ações sejam tomadas baseadas na política do agente.

O Apêndice A possui um pseudocódigo demonstrando a aplicação dos conceitos mencionados acima.

4. OpenAI Gym

A plataforma *OpenAI Gym* oferece um conjunto de ambientes para treinar e comparar os mais variados algoritmos de Aprendizagem por Reforço. Os ambientes possuem uma interface simples e compartilhada, o que colabora com a criação de agentes genéricos capazes de aprender múltiplos cenários [Brockman et al. 2016]. Além disso a biblioteca dispõe de ferramentas para a criação de ambientes personalizados.

4.1. CarRacing-v0

O ambiente da plataforma avaliado pelo estudo é o *CarRacing-v0* ele possui uma vista *top-down* de uma pista de corrida gerada aleatóriamente e um carro de corrida é fornecido como agente.

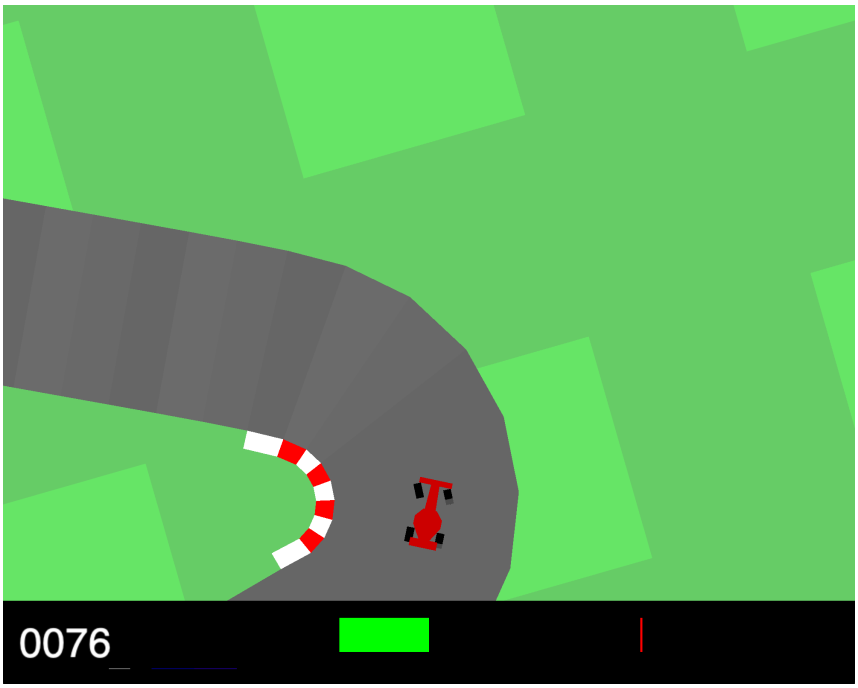

Figura 2. Ambiente CarRacing-v0 da plataforma OpenAI Gym. Fonte: Autor

No ambiente o estado consiste de uma imagem com 3 canais de cores (RGB), no formato $96 \times 96 \times 3$. A recompensa é $-0,1$ por frame e o agente é recompensado para cada tile visitado (parte da pista com coloração diferente) em $1000/N$, sendo N o número total de tiles na pista. O episódio termina quando o agente é capaz de coletar todos os tiles ou quando vai muito além do limite da pista, quando isso acontece ele recebe uma recompensa de -100 pontos. Os indicadores no inferior da Figura 2 representam: pontuação, aceleração, freio, direção e giroscópio.

As ações que controlam o movimento do agente são passadas para a plataforma no formato de um vetor com três valores:

- Direção: valores reais entre -1 e 1
- Aceleração: valores reais entre 0 e 1
- Freio: valores reais entre 0 e 1

5. Metodologia

O estudo iniciou com a pesquisa das técnicas de Aprendizagem por Reforço e outros métodos de Inteligência Artificial usados para os veículos autônomos e problemas análogos. Optou-se por uma abordagem fim-a-fim com a técnica denominada Deep Q-Learning, uma variação do Q-Learning.

Para a aplicação da técnica foi escolhido a utilização da linguagem de programação Python, a biblioteca *PyTorch* para a criação e o treinamento do modelo, o ambiente virtual utilizado foi o *CarRacing-v0*, parte da biblioteca *Gym*.

Os parâmetros utilizados para o treinamento e os dados dos resultados gerados foram armazenados e utilizados para posterior análise.

6. Aplicação do Deep Q-Learning

O algoritmo utilizado foi baseado na implementação contida no Apêndice A, utilizando do *Experience Replay*, junto a política $\epsilon-greedy$ e a utilização de duas redes neurais *Policy Network* e *Target Network*.

Com o intuito de diminuir o custo computacional e remover informações desnecessárias para o agente, foi feita uma etapa de pré-processamento dos estados, onde as imagens que representam os estados foram passadas de RGB para escala de cinza e seu tamanho reduzido para 66×66 pixels.

Foi utilizado também a técnica de *frame-skipping*, nela o agente escolhe uma ação a cada k frames e essa ação é replicada para os próximos frames [Mnih et al. 2013]. O valor de k utilizado foi 4 e os estados do agente foram representados pelo empilhamento dos estados obtidos através do *frame-skipping*.

As ações foram discretizadas para 4 possíveis combinações:

- Esquerda: (-1, 0,25, 0)
- Direita: (1, 0,25, 0)
- Acelerar: (0, 1, 0)
- Frear: (0, 0, 1)

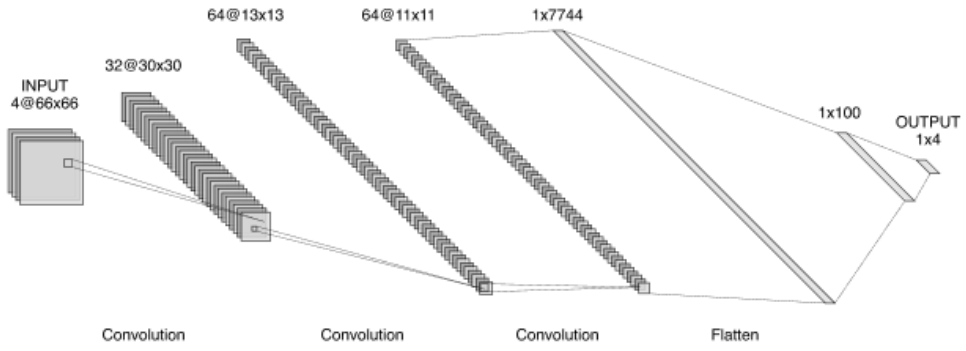

Figura 3. Modelo da rede neural utilizado. Fonte: Autor

O modelo de rede neural aplicado foi o CNN com entrada $66 \times 66 \times \times 4$ e o número de saídas representando o *Valor Q* para cada conjunto de ações. O RMSprop foi o otimizador usado para o treinamento.

A especificação do modelo utilizado, apresentado na Figura 3, foi a seguinte:

- Convolução 1: 32 filtros, kernel de tamanho 8×8, stride 5;
- Convolução 2: 64 filtros, kernel de tamanho 5×5, stride 2;
- Convolução 3: 64 filtros, kernel de tamanho 5×5, stride 1;
- Redimensionamento: transormação do formato resultante das convoluções $64 \times 11 \times 11$ para o formato 1×7744;
- Camada completamente conectada 1: saída 1×100;
- Camada completamente conectada 2: saída 1×4;

Para os hiperparâmetros da Aprendizagem por Reforço Profundo foi utilizado 0,995 para o γ, atualização da *Target Network* a cada 10 episódios, ϵ de 0,05 e taxa de aprendizagem de 0,0001.

7. Resultados

O algoritmo demonstrou grande evolução no recebimento de recompensas com o passar dos episódios quando comparado com a abordagem do agente executando apenas ações aleatórias.

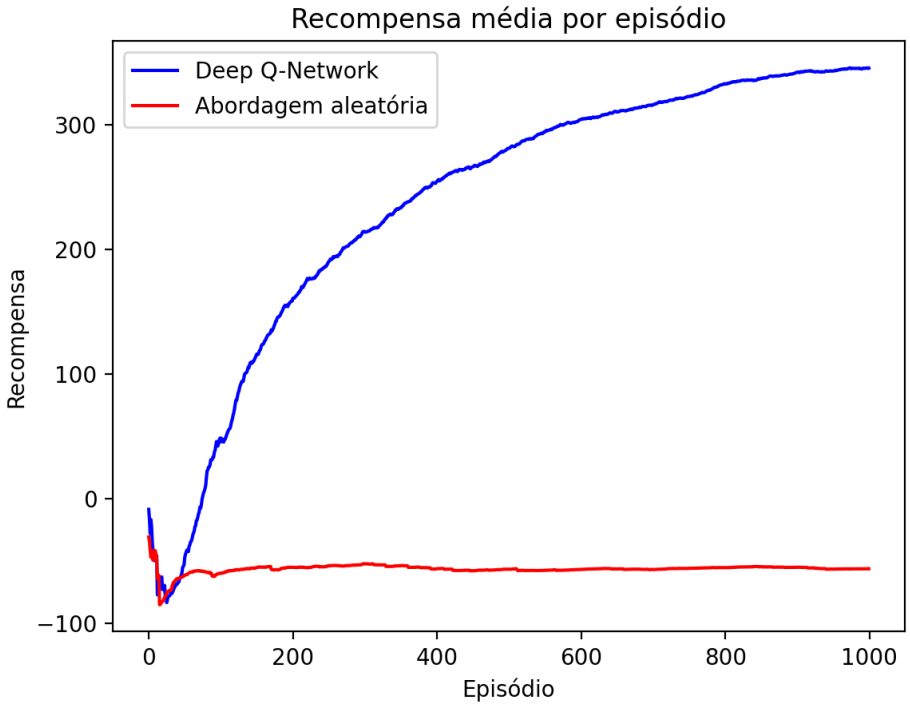

Figura 4. Comparação das recompensas médias após 1000 episódios. Fonte: Autor

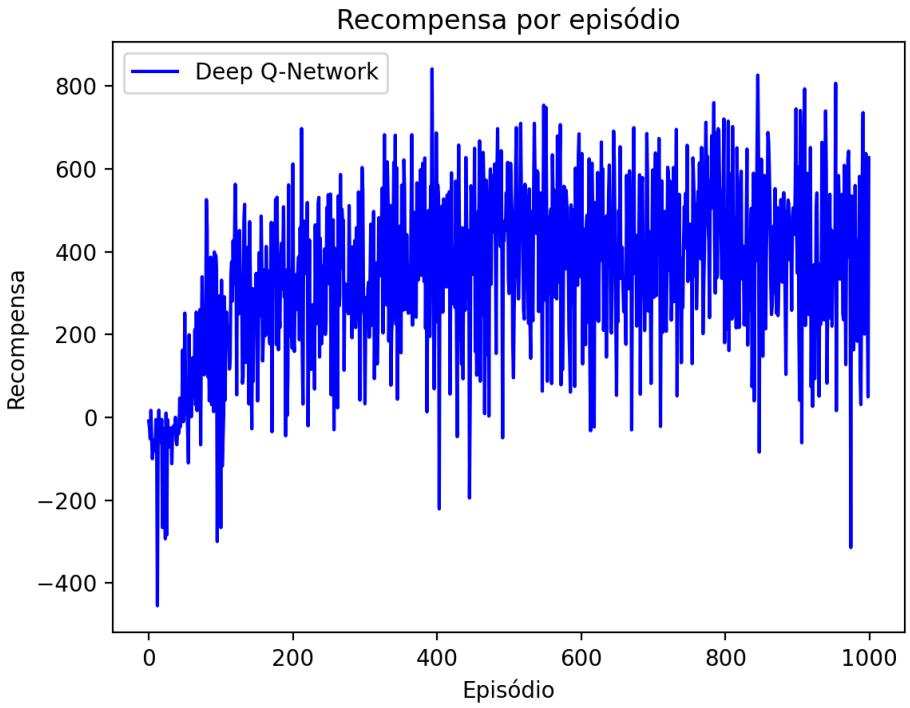

Figura 5. Gráfico das recompensas do algoritmo *Deep Q-Learning*. Fonte: Autor

Como apresentado na Figura 5, ainda com um aumento das recompensas ganhas, o agente não conseguiu se manter estável entres os episódios, registrando máximas superiores a 800 e mínimas inferiores a -300, mesmo após cerca de 900 episódios de treinamento. É possível notar que as recompensas aumentam até em torno do episódio 400, mas após isso elas se mantém em um platô de crescimento.

8. Conclusão

Os resultados mostraram que o agente foi capaz de aprender uma política para tomadas de decisão que aumentasse seus ganhos de recompensas. Embora satisfatórios, aplicar um algoritmo para veículos autônomos necessita de uma estabilidade muito grande no cumprimento dos objetivos e aprender a lidar com situações adversas.

A Aprendizagem por Reforço Profundo demonstra grande capacidade de aprender a solucionar problemas que envolvem a execução de ações em um ambiente e por conta disso possui potencial aplicação para os veículos autônomos. O algoritmo utilizado neste estudo é apenas um entre muitos outros que possuem como base os conceitos de ARP.

Estudos futuros envolverão a tentiva de estabilização do algoritmo com a aplicação de uma rede neural mais complexa e um método *policy based* para lidar com as ações sem a necessidade de discretização.

Referências

Bojarski, M., Testa, D. D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., and Zieba, K. (2016). End to end learning for self-driving cars.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym.

Codevilla, F., Müller, M., López, A., Koltun, V., and Dosovitskiy, A. (2018). End-to-end driving via conditional imitation learning.

Data Science Academy (2021). *Deep Learning Book*. https://www.deeplearningbook.com.br.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numer. Math.*, 1(1):269–271.

Hart, P. E. et al. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107.

Lavaimaa, J. (2018). Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. https://www.sae.org/standards/content/j3016_201806. SAE.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning.

Observatório Nacional de Segurança Viária (2018). 20 anos do ctb acidentes de transito custam r$36 bilhoes/ano. https://www.onsv.org.br/20-anos-do-ctb-acidentes-de-transito-custaram-r-36-bilhoes-por-ano.

OpenAI, Andrychowicz, M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., Schneider, J., Sidor, S., Tobin, J., Welinder, P., Weng, L., and Zaremba, W. (2019). Learning dexterous in-hand manipulation.

Pendleton, S. D. et al. (2017). Perception, planning, control, and coordination for autonomous vehicles. *Machines*.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection.

Ruffo, G. H. (2018). Maio amarelo: "90% dos acidentes são causados por fator humano". https://quatrorodas.abril.com.br/especial/maio-amarelo-90-dos-acidentes-sao-causados-por-fator-humano. Quatro Rodas.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D. (2017). Mastering the game of go without human knowledge. *Nature*, 550.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. The MIT Press, second edition.

Templeton, B. (2019). Elon musk's war on lidar: Who is right and why do they think that? https://www.forbes.com/sites/bradtempleton/2019/05/06/elon-musks-war-on-lidar-who-is-right-and-why-do-they-think-that. Forbes.

Torres, J. (2020). Deep q-network (dqn)-ii. https://towardsdatascience.com/deep-q-network-dqn-ii-b6bf911b6b2c.

World Human Organization (2018). *GLOBAL STATUS REPORT ON ROAD SAFETY*. https://www.who.int/publications/i/item/9789241565684.

Apêndice A - Pseudocódigo do Deep Q-Learning com Experience Replay e Target Network

```
Iniciar a Policy Network Q
Iniciar a Target Network Q̂
Iniciar a memória de experiências D
Iniciar o Agente para interagir com o Ambiente

Repetir para cada episódio:
    Iniciar estado s

    Enquanto s não é terminal, faça:
        Escolher uma ação a para o estado s usando a política ϵ−greedy(Q)
        Executar a ação a, observar a recompensa r, e o próximo estado s′
        Armazenar a transição (s, a, r, s′, terminal) em D
        s = s′
        Atualizar ϵ utilizando ϵ−decay

        Se D possui experiências suficientes, então:
            Fazer uma amostragem com N transições de D
            Para cada transição (s_i, a_i, r_i, s_i′, terminal_i) na amostra, faça:
                Se terminal_i, então:
                    y_i = r_i
                Senão:
                    y_i = r_i + γ max_{a′∈A} Q̂(s′, a′)
                Fim
            Fim

            Calcular a função de custo ℒ = 1/N Σ_{i=0}^{N−1} (Q(s_i, a_i) − y)^2
            Otimizar Q minimizando a função de custo ℒ
            A cada C iterações, copiar os pesos de Q para Q̂
        Fim
    Fim
Fim
```