Agente capaz de jogar Don't Touch the Spikes usando Imitation Learning

Brian Clark Zanfelice, Guilherme Schwinn Fagundes, Lucas Balen Cardozo
Instituto Tecnológico de Aeronáutica
São José dos Campos
18 de julho de 2022

Resumo

Don't Touch The Spikes é um jogo inspirado no famoso jogo Flappy Bird e que requer muito reflexo e velocidade de ação do jogador, fortes atributos de uma inteligência artificial. Com o intuito de testar as capacidades de uma inteligência artificial baseada em imitation learning, fora implementado uma versão do jogo em python usando a biblioteca pygame e uma rede neural simples com 1633 parâmetros no total. O agente obteve resultados abaixo do esperado, obtendo uma média de 10 pontos por partida e um máximo de 16 pontos, porém notou-se que o agente tinha conhecimento de como o jogo funcionava, o que é considerado um êxito.

I. Introdução teórica

A criação de um algoritmo que seja capaz de jogar jogos de forma similar a um ser humano ainda é um desafio no contexto atual dos pesquisadores de *machine learning* mundialmente. O grupo *Deep Mind* (2022), nos últimos anos, desenvolveu diversas pesquisas e estudos na área de aprendizado por imitação e é considerado uma referência nessa área. Um exemplo de um programa realizado pela *Deep Mind* foi o *AlphaZero*, versão de inteligência artificial para jogar xadrez teve resultados impressionantes, vencendo 155 mil partidas contra a versão TCET 2016 do stockfish (plataforma mais forte até então de *engine* para xadrez), perdendo apenas seis e empatando o restante.

No contexto do artigo, utilizou-se a técnica de *Imitation Learning*, um procedimento de *machine learning* na qual copia-se uma política de jogo baseado em um modelo com dados já existentes, e utiliza essa política para fazer a predição de um evento. Para o exmeplo usado nesse artigo de criação de um agente para jogar o jogo Don't Touch the Spikes, foi efetuado uma rede neural com *imitation learning* usando os dados de agentes humanos para se basear.

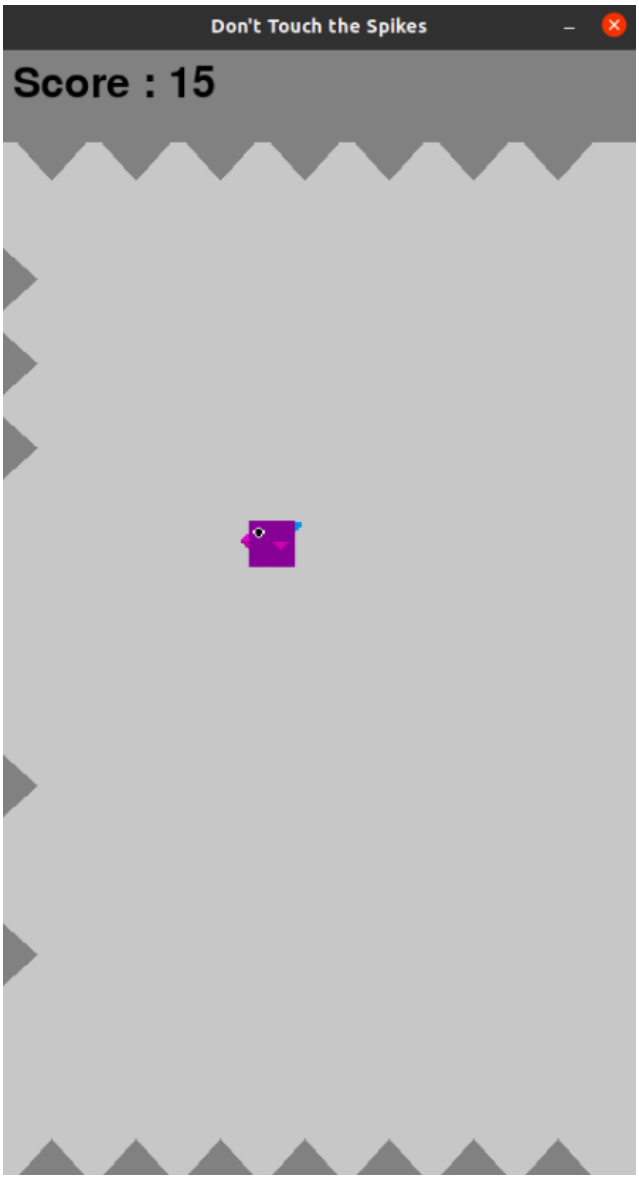

Figura 1: *Imagem da implementação do jogo usando pygame*

II. Implementação

A implementação tanto do jogo utilizado quanto da rede neural e de seu treinamento foi feita utilizando-se a linguagem *Python*. Em relação ao jogo, foram utilizadas as bibliotecas *Pygame* e *Time*, enquanto que a rede neural foi feita utilizando-se o *framework Tensorflow*. Foram utilizadas, também, bibliotecas auxiliares durante a implementação dos códigos, como o *Pandas*, para armazenamento e resgate de dados, e *Numpy*, para utilização de funções matemáticas.

O jogo utiliza uma taxa de 60 *frames* por segundo e foi baseado fortemente no jogo já existente com o nome *Don't Touch the Spikes*. Para aumentar a dificuldade do jogo gradativamente, o número de espinhos e a velocidade do agente aumenta a cada 10 pontos obtidos pelo mesmo, sendo que isso para de ocorrer quando é atingido o valor de 50 pontos. Os valores de velocidade máxima e mínima e o número máximo e mínimo de espinhos no jogo foram determinados empiricamente, por meio de diversos testes pelos desenvolvedores. Além disso, há uma probabilidade de o número de espinhos presentes na parede ser maior que o determinado naquela faixa, com o objetivo de gerar uma aleatoriedade maior para o jogo.

Na execução do jogo, há 6 informações necessárias armazenadas sobre o agente em cada *frame*: sua posição no eixo X e no eixo Y; sua posição no eixo X e no eixo Y no *frame* anterior; um vetor referente à posição dos espinhos no ambiente e o dado se o agente pulou ou não naquele instante. A rede neural utiliza as 5 primeiras informações como *input* e retorna como *output* para o jogo a informação sobre o pulo. Assim, a base de dados foi obtida por meio das cinco primeiras informações citadas obtidas pelo jogo dos três integrantes do grupo. A última informação foi utilizada para como *outputs* esperados utilizados para treinar a rede.

Com isso, a entrada para a rede neural possui dimensão 16, pois o vetor de entradas possui dimensão 12. A rede neural utilizada possui 3 camadas. A primeira camada possui 32 neurônios com função de ativação *Leaky ReLU*, já a segunda camada possui 32 neurônios com função de ativação *Leaky ReLU* e a última camada possui 1 neurônio com função de ativação *sigmoid*.

III. Resultados e Discussões

Em relação aos dados obtidos pelo jogo dos integrantes do jogo, a média da pontuação de cada integrante do grupo está expressa na seguinte tabela:

Tabela 1: *Média da pontuação de cada integrante e do agente.*

Integrante	Pontuação
1	50.3
2	21,3
3	42,8
Agente	10.2

Observa-se que os jogadores apresentaram relativa dificuldade para obter pontuações acima de 50, de forma que a dificuldade do jogo aumenta de forma que torna-se difícil, mas não impossível, jogar na dificuldade mais elevada. Além disso, os valores determinados experimentalmente para características do agente estão reunidas na seguinte tabela:

Tabela 2: *Valores das constantes determinadas.*

Constante	Valor
Velocidade inicial	4
Velocidade final	7
Número de espinhos inicial	3
Número de espinhos final	8

Com estes valores determinados, atingiu-se a dificuldade necessária para o jogo. Dessa forma, percebe-se que a implementação do jogo e as considereções feitas para a composição do mesmo apresentaram resultados satisfatórios.

Com relação à rede neural, para cada frame dado no input, de posição, posição um frame atrás e disposição de espinhos na parede, a rede retorna um número *float* entre 0 e 1, podendo ser encarada como uma probabilidade de efetuar um pulo ou não.

O agente teve como resultados uma média de 10,2 pontos por partida e a maior marca alcançada como 16 pontos. O dataset usado foi de 6mb e a rede neural foi treinada com 200 epochs. O resultado foi abaixo do esperado, pois foi abaixo da média de pontos de todos os integrantes, apesar de ser treinado com base nos dados dos jogos dos integrantes. Observou-se também que o jogo sofria um grande desaceleramento com o agente jogando, causado provavelmente por se ter uma rede muito grande para um jogo que roda a 60 frames por segundo.

Uma das possíveis causas do resultado abaixo do esperado foi um dataset pequeno se comparado com as possibilidades dentro do jogo, no caso em que um dataset maior beneficiaria em muito o aprendizado da rede. Outra possível causa foi o jogo usar uma taxa de 60 fps(frames

por segundo), pois se em média um jogador ”pula”a cada um segundo, teríamos que a cada 60 linhas de dados apenas 1 seria com um output 1, o que dificultaria para a rede entender quando pular, pois ela considera não pular sempre como uma opção viável por ser a coisa mais comum. Na tentativa de mitigar essa causa foi-se diminuído o valor esperado de output, que normalmente seria de 0,5 para 0,2. Esse valor foi encontrado empiricamente. Uma terceira possível causa seria um baixo número de epochs no treinamento da rede neural, porém se o número aumentasse demais encontraríamos uma situação de overfitting, também indesejada, porém é porvável que houvesse algum número melhor de epochs para que o treinamento da rede neural fosse otimizado.

IV. Conclusão

Baseado nos resultados discutidos acima, pode-se concluir que o artigo cumpriu sua proposta, a qual era de obter um agente gerado através de um aprendizado por imitação que fosse capaz de jogar o jogo Don’t Touch the Spikes sozinho. Em relação à criação do jogo, foi obtido um jogo simples que fosse funcional e capaz de poder jogar com facilidade. A rede neural implementada também teve sucesso na sua criação e no seu uso durante o jogo, apesar de ser um pouco pesado para rodar durante a execução do jogo, para que a rede neural retornasse uma resposta a cada frame. Por fim, em relação ao resultado do agente no jogo, pode-se considerar que ele obteve um bom resultado dado que teve pouco tempo de treinamento e poucos dados para treinar.

Referências

1 [1] - ALPHAGO THE MOVIE; Direção: Kohs, G. Produção de Moxie Pictures. 2017.