1. Que algoritmo deve ser utilizado para desenvolver um agente jogador de *Connect4 PopOut* vencedor? Deve-se utilizar uma implementação de Min-Max com poda alpha-beta? Se sim, qual a profundidade que deverá ser utilizada para evitar processamentos superiores a 10 segundos por jogada? Qual a função de utilidade que deve ser utilizada?

A minmax de fato é uma abordagem com alto desempenho em profundidades adequadas. A poda alfa beta também permite com que se aumente a profundidade do algoritmo, uma vez que diminui a quantidade de nodos que se abrem ao longo da busca. Considerando um hardware com as seguintes configurações: i7 7th com 4 cores e 8 threads, 16GB RAM, SSD M2 240G e GTX 1050ti, a profundidade 7 apresenta alguns casos em que o processamento supera 10 segundos por jogada.

2. O seu jogador faz uso de alguma base de conhecimento? Se sim, como ela é utilizada durante o processo de tomada de decisão?

Para o treinamento da CNN foi utilizado dados de 20 jogos simulados com o BarthPlayer com profundidade 7 e um RandomPlayer. Objetivo é que a CNN consiga simular uma profundidade mais elevada com processamentos abaixo de 10 segundos.

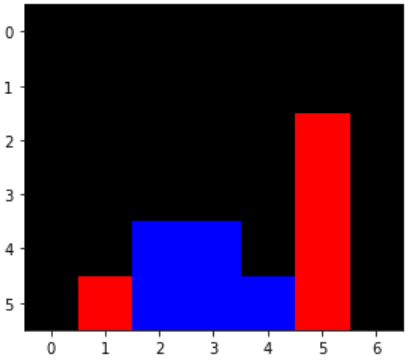
3. Foi utilizada alguma função de utilidade não definida manualmente, por exemplo, alguma função de utilidade gerada a partir de um processo de aprendizagem de máquina supervisionado? Se sim, como é que foi o treinamento desta função de utilidade? Como foi feita a integração desta função de utilidade com o restante do código?

Foi desenvolvido uma Rede Neural Convolucional, com a seguinte topologia:

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 6, 7, 64)          1792

 max_pooling2d_3 (MaxPooling  (None, 3, 3, 64)         0
 2D)

 conv2d_4 (Conv2D)           (None, 3, 3, 128)         73856

 max_pooling2d_4 (MaxPooling  (None, 1, 1, 128)        0
 2D)

 flatten_2 (Flatten)         (None, 128)               0

 dense_4 (Dense)             (None, 256)               33024

 dense_5 (Dense)             (None, 7)                 1799

=================================================================
Total params: 110,471
Trainable params: 110,471
Non-trainable params: 0
_________________________________________________________________
```

Segundo Géron [1], as Redes Neurais Convolucionais emergiram do estudo do córtex no cérebro humano, e são especializadas na análise de imagem, visto que são baseadas no processamento de dados visuais dos neurônios nos humanos. Em alguns casos, atingem desempenho sobre-humano. A CNN efetua o cálculo de acordo com o operador matemático de convolução, assim, essa operação permite obter o grau de influência de uma função em relação a outra função, com características semelhantes à correlação. Portanto, a topologia da rede possui camadas convolucionais, de forma a capacitar o recebimento da matriz do jogo que representa o tabuleiro do jogo.

Para auxiliar a CNN a diferenciar as peças dos dois jogadores do tabuleiro, transformou-se as matrizes em imagens RGBs através da biblioteca OpenCV, abaixo um exemplo de uma das imagens formadas.



O dataset gerado contém 10202 situações (imagens RGB) e para cada imagem, há o movimento do BarthPlayer com profundidade 7. Assim, o dataset foi dividido através do train_test_split do sklearn para realizar o shuffle e a separação de trainset e testset. O tamanho do testset é de 10%, portanto há 9181 situações para o treinamento da CNN.

O treinamento foi feito com 30 Epochs, e um validation_split de 20%. Abaixo a acurácia obtida após o treinamento:

A acurácia convergiu em torno de 45%. O que é considerável, dado que a probabilidade de acertarmos uma jogada é de 1/7 ou 14%.

4. Qual a sua expectativa com relação ao desempenho do seu agente? Você acredita que ele irá desempenhar bem na competição? Por que? Você executou testes contra outros jogadores? Qual foram os resultados?

Embora a acurácia seja considerável, ainda há mais chances da CNN classificar errado o movimento. Portanto acredito que não haverá um bom desempenho na competição, dado que o objetivo dela é emular um minmax com profundidade elevada.

Infelizmente não pude ir ao Insper durante a semana, portanto não consegui utilizar o supercomputador da instituição para gerar um grande número de dados com o BarthPlayer com profundidade acima de 7. Acredito que ao aumentar o volume do conjunto de treino e gerá-lo com base em uma profundidade alta, o desempenho do jogador seria notável.

Os testes feitos foram contra o Random, ganhou na maioria dos casos.
Contra o BarthPlayer com profundidade 5 não conseguiu ganhar.

5. Quais foram as principais referências utilizadas para a implementação do seu jogador?

O Jogador desenvolvido foi baseado na implementação do BarthPlayer, fornecida pelo professor. E conhecimentos prévios da Iniciação Científica [2] e da disciplina de Machine Learning. Para decidir se as CNNs eram de fato recomendadas para o problema, consultei o Hands on Machine Learning do Géron [1]. Embora a abordagem com CNNs não seja usual, foi um projeto de muito aprendizado e pude colocar em prática meus conhecimentos de machine learning e o que aprendi na disciplina de Agentes até o momento.

6. Existem diferenças significativas entre um jogador de *Connect4* e um jogador de *Connect4 PopOut* em termos de árvore de busca e função de avaliação? É possível utilizar o jogador implementado para o *Connect4 PopOut* em competições de *Connect4* sem muitas modificações?

Sim, a árvore de busca é maior, dado que a quantidade de jogadas possíveis aumenta caso haja uma peça de sua cor na última linha, portanto, em um minmax o processamento tende a demorar mais, mesmo com a mesma profundidade de um Connect4 convencional. Em competições de Connect4 podemos utilizar o jogador PopOut, modificando a forma com que a árvore é aberta, ou não considerar que o pop seja uma jogada aceita.

[1] Gerón, Aurelien, Hands on Machine Learning with Scikit-learn & TensorFlow, Alta Books, 2017.

[2] Yamashiro, Eiki, Contagem de Multidões através de Redes Neurais Convolucionais, Insper - Instituto de Ensino e Pesquisa, 2021.a