DESENVOLVIMENTO DE UMA SIMULAÇÃO DE EVACUAÇÃO DE AMBIENTES UTILIZANDO REDES NEURAIS ARTIFICIAIS

Guilherme Manfroi* João Mário Lopes Brezolin†

20 de dezembro de 2022

Resumo

O comportamento humano é fortemente influenciado pelas relações estabelecidas com seus pares. Decisões racionais muitas vezes não são adotadas pelos indivíduos quando introduzidos a uma multidão por ter seu comportamento alterado em casos de situação de perigo. Este trabalho objetiva realizar a implementação de um ambiente virtual no qual estarão inseridos um conjunto de indivíduos que serão treinados para realizar a saída segura dos mesmos. Busca-se, dessa forma, avaliar as contribuições do uso de um sistema de simulação que realiza o treinamento do comportamento de indivíduos por meio de Redes Neurais Artificiais.

Palavras-chaves: Redes neurais. Controle de multidões. Unity ML-Agents.

Introdução

A formação de multidões pode ocorrer de forma voluntária ou involuntária, além disso pode levar ao surgimento de comportamentos coletivos que não aconteceriam de forma individual, podendo levar a situações imprevisíveis. Nesse contexto, existiram acontecimentos onde a segurança das pessoas foi comprometida devido a falta de possibilidade de sair de um local. Um exemplo recorrente desse tipo de situação é a evacuação de locais nos quais identificam-se emergências como enchentes ou incêndios, e para encontrar soluções para orientar indivíduos nessas situações é comum o uso de simulações, já que os ambientes virtuais permitem criar modelos de simulação seguros e economicamente viáveis.

Além disso, objetiva-se analisar situações sem necessidade de testes com pessoas reais, devido a ser inviável a utilização de pessoas para compor uma aglomeração e simular uma evacuação para averiguar a situação. Utilizando essas simulações e modelos que possuem uma semelhança

*<guilherme_gbm@outlook.com>
†<joaobrezolin@ifsul.edu.br>

com a realidade, há a possibilidade de eleger locais que de fato são seguros numa situação semelhante. Dessa forma, existe também o experimento de utilizar redes neurais nesse contexto, a fim de avaliar se a contribuição será efetiva para que testes mais complexos possam ser sugeridos.

Nesse sentido, a presente pesquisa objetiva criar um ambiente de simulação para avaliar o comportamento de indivíduos que desejam realizar a evacuação segura de locais nos quais identificam-se situações de perigo. O ambiente de simulação apresentado neste artigo foi implementado utilizando da plataforma Unity [1], juntamente com a biblioteca ML-Agents Toolkit [2] para criar os agentes inteligentes. Esses agentes são capazes de implementar um processo de aprendizado por meio de uma Rede Neural Artificial (RNA) que visa estabelecer a sequência de passos necessária para que seja realizada a evacuação segura do ambiente. Este artigo detalha o processo de implementação desse ambiente e apresenta os resultados do processo de aprendizado desses indivíduos.

A seção 1 retoma conceitos basilares sobre simulação e controle de multidões como também descreve as tecnologias utilizadas no desenvolvimento do ambiente de simulação. A seção 2 apresenta a aplicação criada e suas funcionalidades. Os resultados dos testes e validações do sistema são apresentando na seção 3. E, para finalizar, são abordadas as considerações finais e as perspectivas de trabalhos futuros.

1 REFERENCIAL TEÓRICO E TECNOLOGIAS

Dois principais conceitos são destacados para a implementação deste projeto. A primeira abordagem deve ser o estudo sobre controle de multidões, o qual traz embasamento teórico e conceitos sobre o comportamento de indivíduos, destacando a análise de pessoas em grupos. O segundo conceito basilar é o estudo de redes neurais e sua importância na implementação dos indivíduos no sistema de simulação. Além disso, é necessário conhecimento prático da plataforma Unity, tanto para modelagem do projeto como para implementação da biblioteca ML-Agents que será crucial neste projeto.

1.1 CONTROLE DE MULTIDÕES

Existe uma pesquisa muito grande no que diz respeito ao comportamento humano, muito porque há uma fonte inacabável de dados para analisar e comparar situações que possam explicar as diferenças de comportamentos, expressões e ideais que caracterizam cada ser humano.

Em um contexto em que há grande concentração de pessoas, dados de percepção de cada um desses indivíduos podem nos mostrar como acontece o comportamento espacial entre estes independente de qualquer outro hábito individual (ARAUJO, 2020). Discutido por alguns autores, é dito que ao compor uma multidão os indivíduos têm o seu comportamento alterado diminuindo a sua capacidade de buscar soluções racionais para as situações de perigo, ou seja, possuíam uma espécie de transformação psicológica ao estarem no meio de um grupo de pessoas. Um conceito introduzido por Gustave LeBon em 1895, a partir da publicação da obra “The Crowd: a study of the popular mind”, diz que, uma multidão mesmo que composta por pessoas normais, possuíam uma espécie de transformação psicológica ao estarem no meio de um grupo de pessoas. Para LeBon, a multidão transformava o indivíduo, diminuindo ou eliminando suas habilidades para controlar o seu comportamento de forma racional (BARBOSA, 2014).

[1] <https://unity.com/pt>
[2] <https://github.com/Unity-Technologies/ml-agents>

Realizar testes reais em situações com multidões pode ocasionar riscos para as pessoas envolvidas e nesse sentido a simulação com indivíduos virtuais tende a ser a melhor maneira. Em uma simulação é possível criar indefinidas quantidades de indivíduos sem qualquer risco, além disso, há a possibilidade de colocá-los em situações perigosas e deixá-los adotar seu comportamento de emergência (MUSSE, 2013). Apesar de existirem pesquisas no sentido de encontrar padrões (ARAUJO, 2020), observa-se que o comportamento humano é idiossincrático levando em conta elementos como o humor, proximidade com as pessoas, seu temperamento, entre outros, o que torna impossível prever com exatidão as decisões que poderão ser tomadas. Nesse contexto, o presente projeto busca analisar o comportamento de forma espacial entre os agentes, onde sua percepção será apenas em proximidade local com o outro indivíduo, excluindo qualquer descontrole emocional ou manifestação humana que poderia ser gerado dentro de um contexto realista ou em uma situação de conflito, como ajuda voluntária entre indivíduos.

1.2 REDES NEURAIS ARTIFICIAIS

Segundo Rauber (2005), a inteligência do ser humano é a mais avançada dentro do universo das criaturas e o local dessa inteligência dentro do corpo humano é o cérebro. As entidades básicas são os neurônios, interconectados em redes, o que permite a troca de informação entre eles, criando a inteligência biológica. Observou-se que a essa estrutura poderia ser mapeada para uma estrutura artificial, por exemplo uma combinação de hardware e software, assim transformando as redes neurais biológicas em Redes Neurais Artificiais (RNA) (RAUBER, 2005). “Baseando-se nas características de seres biológicos, as pesquisas em RNAs e em Inteligência Computacional buscam por gerações completas de novos sistemas computacionais, muito mais eficientes e inteligentes que os sistemas atuais” (GOLDSCHMIDT, 2010).

Inspirados pela sofisticada funcionalidade do cérebro humano onde bilhões de neurônios interconectados processam informações em paralelo, pesquisadores tentaram com sucesso demonstrar certos níveis de inteligência no silício (WANG, 2003). Conforme Goldschmidt (2010): “Trata-se de uma área de grande importância para a computação e suas aplicações, responsável pela solução de inúmeros problemas complexos.”.

Nas redes neurais artificiais, a ideia é realizar o processamento de informações tendo como princípio a organização de neurônios do cérebro. Em uma RNA os neurônios são arrumados em camadas com conexões entre elas. Consiste em uma primeira camada de neurônios que recebe os dados e, por convenção, é chamada de camada de entrada. Uma ou mais camadas internas onde ocorre o processamento interno da rede, denominadas camadas escondidas, e uma camada de neurônios de saída. Cada camada possui uma quantidade determinada de neurônios, e todos os neurônios de todas as camadas estão conectados com a camada em paralelo à ela (GOLDSCHMIDT, 2010). A plataforma Unity permite a implementação de ambientes de simulação e agentes inteligentes (indivíduos) por meio da biblioteca ML-Agents (JULIANI et al., 2018). Essa biblioteca permitiu estabelecer o treinamento dos agentes utilizados no ambiente de simulação.

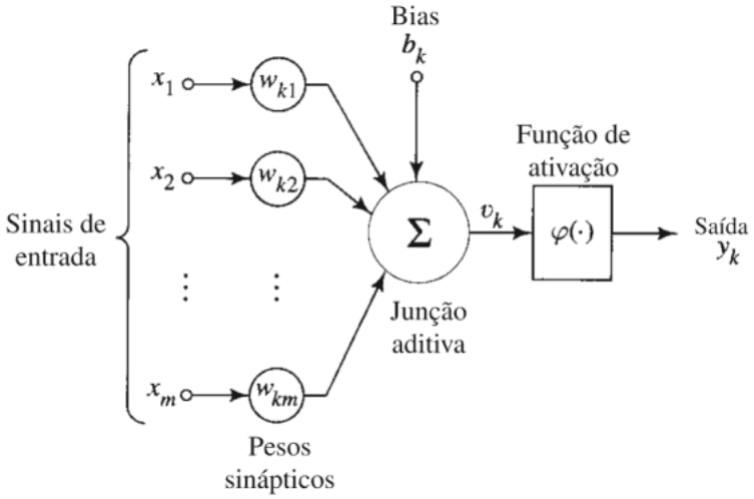
1.2.1 Neurônio artificial

Segundo Rauber (2022), o modelo de neurônio utilizado tenta simular as realidades biológicas que ocorrem dentro de uma célula no sistema nervoso. No neurônio acontecem os cálculos necessários que geram a saída para a próxima camada ou a saída final da rede.

Na Figura 1 é possível observar todos os atributos que compõem os cálculos dentro do neurônio, onde x_j representam as entradas, x_k os pesos sinápticos que serão ponderadores das entradas e a junção aditiva. Um somador que constitui um combinador linear de todas as entradas, representado pela equação:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j$$

Figura 1 – Componentes de uma RNA



Fonte: (HAYKIN, 2001)

A representação b_k constitui o *bias* aplicado pela camada, o qual tem efeito de aumentar ou diminuir a entrada líquida da função de ativação, e a representação da função de ativação que faz a restrição dos valores de saída dependendo da função utilizada (HAYKIN, 2001), também representada pela função:

$$y_k = \varphi(u_k + b_k)$$

Resumidamente, os cálculos se iniciam a partir das informações fornecidas por outros neurônios, que entram em N entradas x_j no neurônio processador. A cada entrada está associado um peso w_{kj} que reflete a importância da entrada x_j. Ou seja, cada entrada será multiplicada por um peso correspondente (w_{kj}), e após são somadas todas essas entradas multiplicadas, gerando um valor. Esse valor por sua vez, será acrescentado a um valor *bias* da camada em questão e após, será restringido por uma função de ativação, determinando se o neurônio será ativado caso o valor ultrapasse o limite de ativação (GOLDSCHMIDT, 2010).

A função de ativação de um neurônio artificial determina o novo valor do estado de ativação deste neurônio, a partir de seu potencial de ativação u_k, determinando a saída efetiva de um neurônio (GOLDSCHMIDT, 2010). O modelo de cada unidade da rede pode incluir uma não-linearidade na sua saída, a qual deve ser reduzida (FLECK et al., 2016).

1.3 Unity ML-Agents Toolkit

A Unity em sua grande distribuição de bibliotecas disponibiliza esse projeto open source que permite que pesquisadores e desenvolvedores consigam ter facilidade na utilização de machine learning (ML) dentro da plataforma Unity (JULIANI et al., 2018). Os agentes podem ser treinados

com o uso de técnicas de *Machine Learning* (ML), *Reinforcement Learning*, *Imitation Learning* (IL), *Curriculum Learning* (CL), entre outros (SILVA et al., 2020). Com essa biblioteca é possível criar ambientes simulados a partir do Unity Editor e, através de uma API em Python, consegue se comunicar com esse framework o qual contém todas funcionalidades necessárias para criar uma inteligência com aprendizado nesse ambiente (JULIANI et al., 2018). Esse framework pode ser integrado com a Unity, porém como descrito por (SILVA et al., 2020), alguns conceitos básicos são requeridos.

> "Visto que o pacote ML-Agents possui inerentemente intimidade com áreas da Inteligência Artificial é recomendada uma noção básica dos conceitos que o compõem, tais como redes neurais e aprendizado de máquina, de modo que o usuário possa compreender com totalidade suas funções e fundamentos" (SILVA et al., 2020).

Um dos recursos da biblioteca é o framework TensorFlow o qual é utilizado para pesquisas na área de Inteligência Artificial que auxilia no aprendizado de máquina (SILVA et al., 2020). Com esse framework é possível fazer análise do treinamento que está acontecendo a partir do ML-Agents Toolkit e verificar tudo através de gráficos disponibilizados, podendo fazer comparativos com diferentes treinos executados e observar a diferença de aprendizado em algumas situações.

As entidades principais no treinamento com ML-Agents são os Sensores (Sensors), Agentes (Agents) e Academia (Academy). O componente Agente está diretamente ligado a um GameObject e determina que aquele objeto pode coletar observações, realizar ações e receber recompensas (JULIANI et al., 2018), os quais estão descritos a seguir (SILVA et al., 2020):

- **Observações (*observations*):** pode ser realizada a partir de dados obtidos de várias formas, como um raio disparado a partir do objeto, distâncias, posições e reconhecimento de imagem. É aquilo que o agente percebe sobre o ambiente.

- **Ações (*actions*):** são as ações e funções que um agente pode executar. Podem ser discretas ou contínuas, dependendo da complexidade do ambiente e do agente.

- **Recompensa (*reward*):** em algum momento do treinamento é necessário dar uma recompensa ao agente, geralmente quando o agente executa uma ação boa ou ruim. É um valor escalar que sintetiza o desempenho do agente.

É possível fazer a alteração dos parâmetros utilizados pelo aprendizado de máquina para atingir diferentes níveis durante o treinamento. Também podem ser estudados esses parâmetros para aplicar a melhor estratégia de aprendizado para objetivos específicos. Após realizado o treinamento, utiliza-se o TensorFlow para visualizar os treinamentos executados através de gráficos, além de ser possível obter a rede neural treinada para fazer a aplicação desejada, como é descrito na documentação da biblioteca.

Além disso, a biblioteca inicialmente nos permite escolher o tipo de algoritmo que será utilizado para o treinamento, dando a opção de dois algortimos de reforço, os quais são *Proximal Policy Optimization* (PPO) e *Soft-Actor Critic* (SAC). Escolha deve ser realizada com base no tipo de treinamento desejado.

1.4 Trabalhos relacionados

O trabalho desenvolvido por MUSSE (2013), a partir dos seus estudos realizados desde 1996 junto com pesquisas de alunos da PUCRS, apresenta uma simulação de multidões de humanos virtuais. Assim como nesse projeto, o trabalho do autor necessitou do desenvolvimento de um ambiente virtual para fazer a realização de testes de multidões, então para isso foi desenvolvido um software chamado *CrowdSim* o qual foi usado para criação da simulação. Nessa simulação o objetivo foi recriar virtualmente um estádio de futebol a aplicar a quantidade total de pessoas, nesse caso foram 47000, e verificar o procedimento de evacuação com diferentes estratégias ou ocorrências. Apesar da similaridade, o desenvolvimento da simulação não apresenta uso de inteligência artificial, podendo não ser possível avaliar determinadas estatísticas pelo comportamento dos indivíduos ser totalmente linear e objetivo. Por outro lado, esse projeto de pesquisa tende avaliar a evacuação de locais simulados com alguma ocorrência específica, porém adotando a análise de uso de redes neurais nos indivíduos.

Similarmente a esse projeto, GÖLLNER (2022) desenvolveu um jogo competitivo de voleibol utilizando aprendizado por reforço com o Unity ML-Agents. O trabalho descreve a integração de aprendizado de máquina para o desenvolvimento do jogo a partir da utilização de vários recursos que a biblioteca *ML-Agents Toolkit* proporciona dentro da plataforma Unity. Inicialmente são introduzidos os principais conceitos por trás da implementação e também do treinamento que seria aplicado aos agentes em cena para o aprendizado do jogo em questão. Após isso é feita a explicação da implementação apresentando os objetos em cena, onde seriam dois agentes um em cada lado do cenário separados por uma parede que simulava a rede do voleibol. A percepção dos agentes nesse caso é sua rotação no seu eixo y, o seu vetor de velocidade (x, y, z), o vetor de direção até a bola (x, y, z) e a velocidade da bola (x, y, z). As ações exercidas pelos agentes baseados nas observações seriam a movimentação para frente e para trás, esquerda e direita, rotação para os dois lados e pulo.

No projeto da autora, um dos principais pontos era o aprimoramento dos parâmetros para utilizar no treinamento, o qual serviria para otimizar o processo de treinamento. Os parâmetros que podem ser alterados são vários e dependendo do projeto pode ser necessário testar diferentes estratégias para ter mais benefícios. Nesse caso alguns dos parâmetros alterados foram: *max_steps*, *buffer_size*, *batch_size*, *learning_rate* e outros mais específicos. Outro destaque do trabalho foi a explicação de como foram distribuídas as recompensas (rewards) aos agentes, a estratégia utilizada e a dificuldade até encontrar o melhor método. Ao final do projeto é apresentado os resultados utilizando o TensorBoard e verificado o desempenho dos treinamentos.

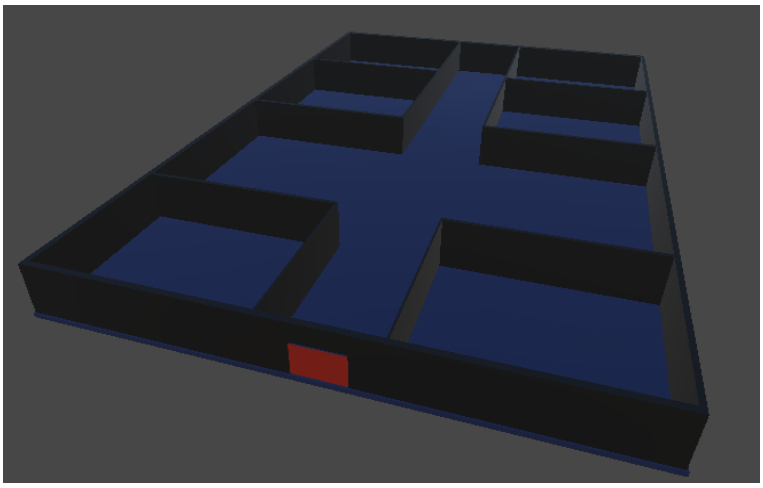
Apresentado por Kayser (2018), um trabalho utilizando sistemas neurais foi desenvolvido com o propósito de treinar esse agente para resolver o problema de CartPole. Resumidamente, o objetivo do treinamento é ter uma rede neural capaz de manter um pêndulo inverso equilibrado em 90 graus em relação a sua base. Para tal feito, foram utilizadas redes neurais artificiais e algoritmos genéticos, formando assim um sistema neural híbrido. Foram utilizadas estratégias e técnicas de algoritmos genéticos para encontrar o melhor desempenho de treinamento, como também, encontrar a melhor estrutura de rede neural para o objetivo. Neste trabalho, foi utilizada uma biblioteca chamada Gym para realizar a simulação e treinamento, contudo, não foi necessário criar modelos em outras plataformas para obtenção do cenário. Em contrapartida, esse projeto de pesquisa necessitará da criação do modelo 3D e implementação dos agentes na Unity para seu treinamento.

2 Desenvolvimento do ambiente de simulação

Este trabalho tem como objetivo analisar as contribuições do uso de uma rede neural artificial no aprendizado de agentes em um ambiente simulado. O ambiente será modelado na plataforma Blender[3] e importado no Unity para a criação da simulação. Por meio da biblioteca ML-Agents serão utilizados os treinadores em Python para realizar o treinamento, e a rede neural será adaptada conforme o aprendizado do agente.

O primeiro passo dessa pesquisa foi a criação do ambiente virtual o qual será utilizado como cenário para a distribuição dos indivíduos, implementação e treinamentos dos agentes (Figura 2).

Figura 2 – Ambiente virtual



Fonte: Autor (2022)

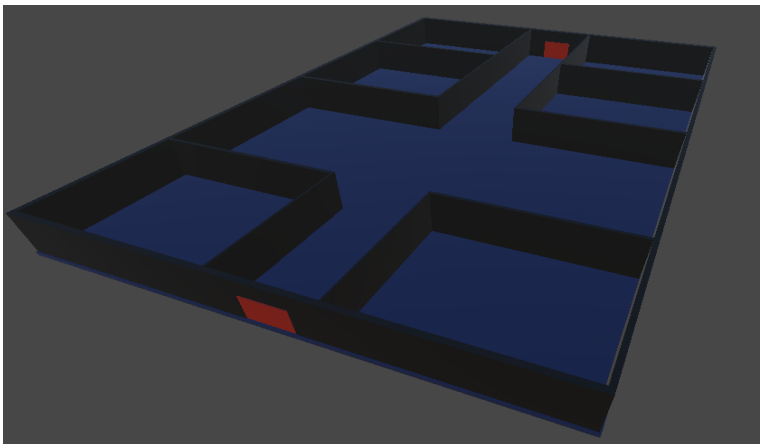
2.1 Diferença entre modelos

Uma questão muito importante neste projeto está relacionada à avaliação do local, o que remete a eleição de locais seguros. Neste contexto, vários cenários poderiam ser criados a fim de avaliar, dentro do modelo, a melhor forma de compor um local com a segurança adequada. Desta forma é possível atribuir diferentes características ao ambiente para avaliar o quão diferente uma evacuação poderia acontecer após analisar essas diferentes situações. Tais características podem estar relacionadas a qualquer exercício de segurança, e nesse caso a diferença na quantidade de saídas será o foco a ser analisado.

Os cenários utilizados serão compostos pelas mesmas dimensões e a mesma quantidade de indivíduos serão alocados em cena. No primeiro modelo temos um ambiente padrão com uma saída, representada em vermelho nas imagens, (Figura 2) e no segundo modelo foi adicionado mais uma saída (Figura 3). A intenção durante o treinamento será a análise de aglomeração formada nas saídas, além de ser possível verificar o tempo que seria tomado em diferença aos dois modelos.

[3] <https://www.blender.org/>

Figura 3 – Ambiente virtual (segundo modelo)



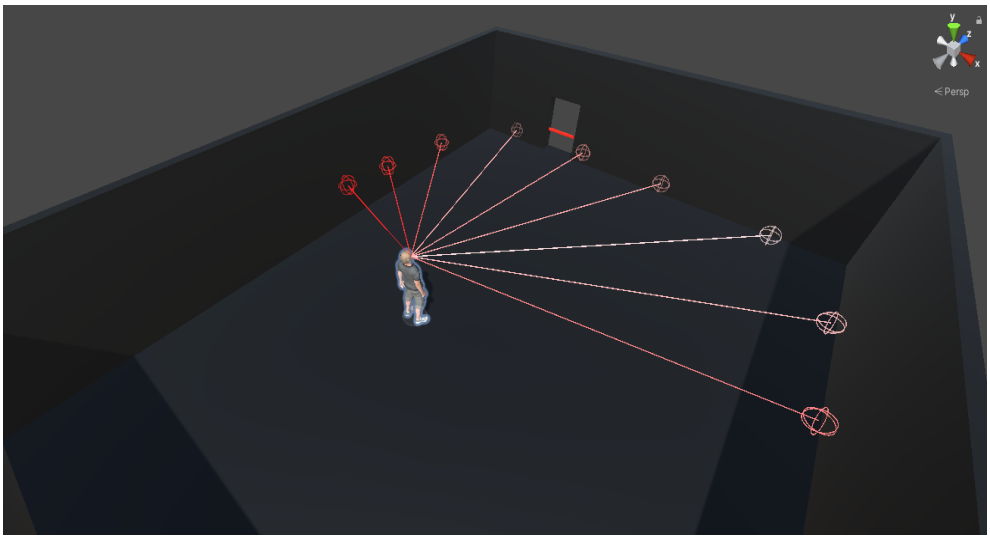
Fonte: Autor (2022)

2.2 Percepção e ações do agente

Para o agente atingir as saídas do local ele tomará as suas decisões baseadas na sua percepção com o ambiente. De forma aleatória, o agente se movimenta pelo ambiente enquanto sua recompensa é acrescida ou decrescida dependendo da estratégia utilizada para obtenção do objetivo.

Ao ingressar no ambiente o agente recebe um conjunto de dados sobre a sua localização atual e a localização da saída, que são armazenadas em um vetor de três dimensões (eixo x, eixo y e eixo z). Além disso, o agente recebe outras informações sobre o ambiente a partir de sensores de lasers 3D que partem do agente com uma angulação de 75° até atingir algum objeto (outro indivíduo) ou parede, como é possível visualizar na Figura 4.

Figura 4 – Ilustração de raios 3D para percepção do agente



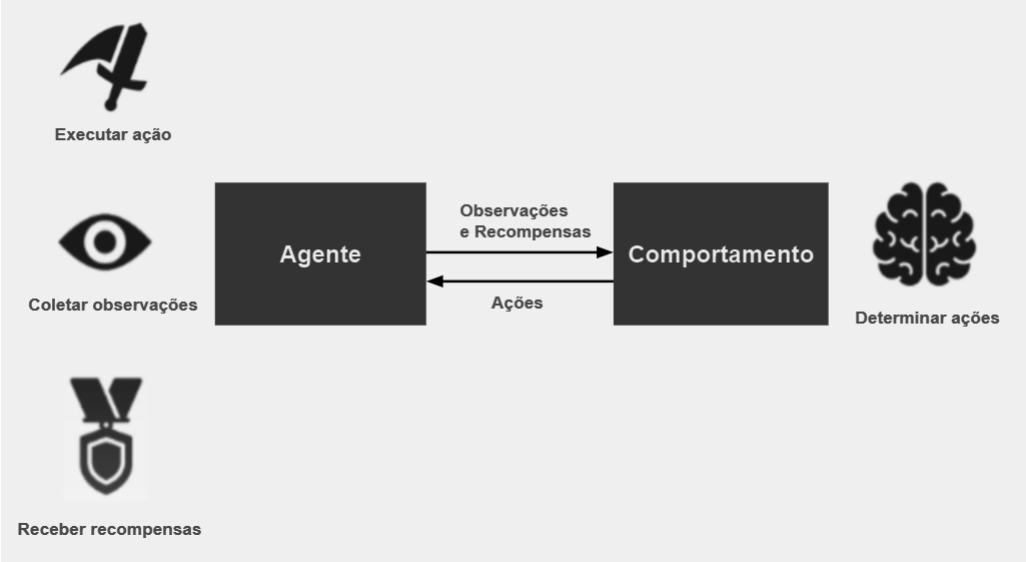
Fonte: Autor (2022)

A sua tomada de decisão acontece a partir dos dados de saída da rede neural, sendo esses valores representados por um vetor de duas posições, que vão determinar as próximas ações do agente.

O vetor, que é resultado do cálculo realizado na rede neural, é composto por duas posições, onde a primeira posição pode resultar em 3 valores, determinando a rotação do agente. O valor 0 mantém a rotação atual, ou seja, não rotaciona, o valor 1 rotaciona para a esquerda, e o valor 2 rotaciona para a direita. Já na a segunda posição do vetor é possível obter dois valores, sendo 0 para ficar parado e 1 para se movimentar para frente.

Com as duas principais características determinadas, a percepção do ambiente e as ações do agente, é possível visualizar a integração da forma de aprendizado. A Figura 5 ilustra a troca de informações entre os componentes que compõem o treinamento.

Figura 5 – Ilustração da comunicação entre os componentes do ML-Agents



Fonte: Autor

O Agente, está comprometido em captar as informações, executar as ações e receber recompensas conforme a ação exercida. Por consequência, ele precisa de um componente que receberá as informações e ditará as suas ações, determinado pelo Comportamento, o qual será a sua rede neural que está sendo treinada para aquele agente. Durante o treinamento dos agentes será possível observar a sua condução aleatória até que passe por suas gerações e defina a melhor estratégia de locomoção para atingir o objetivo. Para facilitar no treinamento é possível replicar os ambientes e os agentes dentro da plataforma para acelerar o processo.

2.3 Recompensas

Dentro da estrutura da rede neural com aprendizado por reforço, existe a necessidade de demonstrar ao agente o que é certo ou errado. Para tanto, faz-se necessário o uso de alguma estratégia que beneficia as boas escolhas da rede neural. Cada implementação exige um diferente método de aplicação dessas recompensas.

Neste cenário, como já dito, o objetivo é que os agentes treinados saiam do espaço, assim, seria interessante recompensá-los quando suas decisões tomem esse rumo. Outro fator observado a

partir de alguns trabalhos relacionados é a quantidade de recompensas que seriam entregues, onde foi mencionado que o melhor desenvolvimento era trabalhar com recompensas na escala de -1 a 1. Dentro dessa lógica, essa metodologia propõe o seguinte.

O indivíduo ao se aproximar da saída pode ser recompensado por estar se dirigindo ao objetivo, e nesse caso o valor aplicado é 0,005, multiplicado por uma variável com valor inicial de 0,00001 que era acrescida pelo mesmo valor cada iteração da execução do programa. Ao atingir a saída o objetivo era cumprido, tendo como recompensa o valor de 1,0. Recompensas negativas eram aplicadas em colisões, sendo a colisão com a parede o valor de -1,0 e ao colidir com outros indivíduos a recompensa era -0,01.

Porém, utilizando essa sistemática já foi possível analisar o desenvolvimento do treinamento com alguns resultados positivos. Além disso, existe em vários casos a utilização de uma recompensa limite, onde ao alcançar tal valor a rede neural seria considerada competente para resolver o problema, porém nessa primeira implementação isso não foi abordado. Outro detalhe levando em consideração a evacuação de ambientes, é que a colisão entre as pessoas que estão no local é um problema para o deslocamento do agente, nesse sentido faz-se necessário estimular o melhor comportamento tentando atingir o objetivo com segurança. Então, como pôde ser observado serão aplicadas recompensas negativas à colisão entre indivíduos, crendo que a rede neural possa aprender a partir disso.

2.4 Parâmetros de otimização da Rede Neural

O algoritmo utilizado neste trabalho será o PPO, que consiste em um método de aprendizado de máquina onde a metodologia utilizada é o aprendizado por reforço. Nessa situação, é determinado uma política (*policy*[4]), baseado no ambiente em que os agentes serão implementados, e através de sinais de recompensas é possível calcular a perda, ou erro, da rede neural por um método chamado de gradiente descendente estocástico, e a partir disso é propagado no sentido contrário pela rede neural para corrigir os pesos e *biases*. Utilizando esse algoritmo será possível obter mais resultado dentro da implementação.

Assim como em toda rede neural, existem configurações que podem ser feitas para obter um melhor resultado a partir das otimizações realizadas. Todo cenário de utilização de redes neurais pode ter uma validação diferente do que é o melhor método de aprendizado dos agentes e afins que estão integralizando o ambiente. A partir disso, é essencial testar e verificar quais as alterações necessárias para que se obtenha os resultados desejados na aplicação da rede neural.

O conceito de otimização se chama *Hyperparameters Optimization*, e a biblioteca ML-Agents proporciona isso de forma simples desde que haja conhecimento das alterações efetuadas. Para que isso seja feito, é necessário incluir um arquivo de configuração *.yaml* seguindo os parâmetros descritos pela biblioteca.

Existem alguns principais parâmetros que podem ser ajustados para melhor performance dos agentes. O **learning_rate** define a taxa de atualização dos pesos e *biases* da rede neural durante o treinamento. Valores mais baixos podem ser mais eficazes devido a acontecer muita atualização durante o treinamento, então assim não é perdido o conhecimento anterior que a rede neural adquiriu, e o treino acontece mais consistentemente. O **learning_rate_schedule** define se o valor do *learning_rate* se mantém constante ou linear tendendo a 0. Em caso onde se tenha um ambiente que possui muita alteração ou uma IA que requer um treinamento contínuo, o valor *constant* é mais interessante.

[4] método ou função que vai determinar uma ação para um determinado problema

O parâmetro **buffer_size** corresponde a depois de quantas experiências observadas deve ser feito alguma atualização no modelo da rede neural. É recomendado que seja múltiplas vezes maior que o **batch_size**, dizendo quantas vezes a experiência do agente deve ser levada a rede neural antes de acontecer uma atualização. O **num_epoch** é um parâmetro que determina quantas vezes as experiências coletadas pelo *batch_size* serão passadas pela rede neural. Quanto maior o valor mais vezes os pesos da rede neural serão atualizados, podendo perder estabilidade.

Os próximos parâmetros são específicos para o algoritmo que será utilizado, tendo o **beta** que determina o quão randômico é o início do treino, o que oferece mais exploração do ambiente, e também esse valor é reduzido linearmente até 0 conforme a progressão do treino. O *epsilon* está relacionado com a atualização da *policy* que acontece a cada determinada quantidade de experiências, sendo essa quantidade o valor do parâmetro *buffer_size*. O valor representa a porcentagem de quanto a *policy* pode mudar.

A partir da colocação do significado dos principais itens e para que eles servem dentro da rede neural, é necessário verificar com base no desenvolvimento do projeto, quais os melhores valores a serem colocados que resultem em um melhor treinamento. Sendo assim, foi determinado que a rede neural possui 3 camadas escondidas com 128 neurônios cada camada, e os valores alocados para os *Hyperparameters* podem ser observados na Tabela 1.

Tabela 1 – Hyperparameters.

Parâmetros	Valor
learning_rate	0.0003
learning_rate_schedule	linear
buffer_size	5120
batch_size	512
num_epoch	5
beta	0.01
epsilon	0.2

Fonte: Autor (2022)

3 Resultados

Além de ser possível observar a efetividade do treinamento a partir dos gráficos apresentados, é preciso avaliar a utilização de redes neurais nesse contexto, pois um grande detalhe neste projeto é justamente a adoção de inteligência artificial, já que vários projetos sobre simulação de evacuação não levantam essa questão. O benefício dessa implementação é a análise do ambiente durante o treinamento e também com a rede neural treinada, a fim de ser coerente com um fato levantado no projeto que é sobre a dificuldade do indivíduo tomar as melhores decisões quando há uma necessidade rápida de evacuação, geralmente relacionado a algum perigo no local.

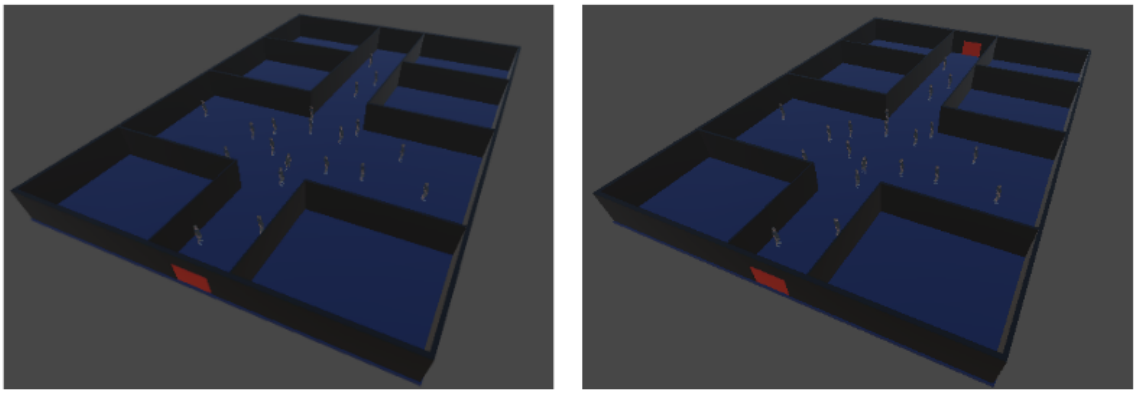
Ademais, a implementação realizada juntamente com o framework TensorFlow permite acompanhar os resultados do treinamento dos agentes. Todos os passos realizados dos agentes até a obtenção das recompensas e finalização do objetivo são contados e os resultados são disponibilizados em gráficos.

O primeiro treinamento e obtenção de resultados foi executado no modelo onde seria o cenário padrão (imagem (a) da Figura 6). A otimização de parâmetros e randomização de posição dos indivíduos durante o treino trouxe melhorias significativas para conclusão do objetivo. O treino

possuía dois milhões de *steps* percorridos pelos agentes e levou cerca de duas horas e meia para conclusão dos resultados.

O segundo treinamento foi concebido pela ideia de diferença entre locais, onde poderia então ser visualizada algumas diferenças na evacuação dos agentes pelo ambiente possuir mais rotas de saída. Como é possível observar na imagem (b) da Figura 6, foi adicionada uma segunda porta, o que resultou em uma significativa diferença nos resultados e além de possuir as demais características do treino anterior.

Figura 6 – Treinamentos

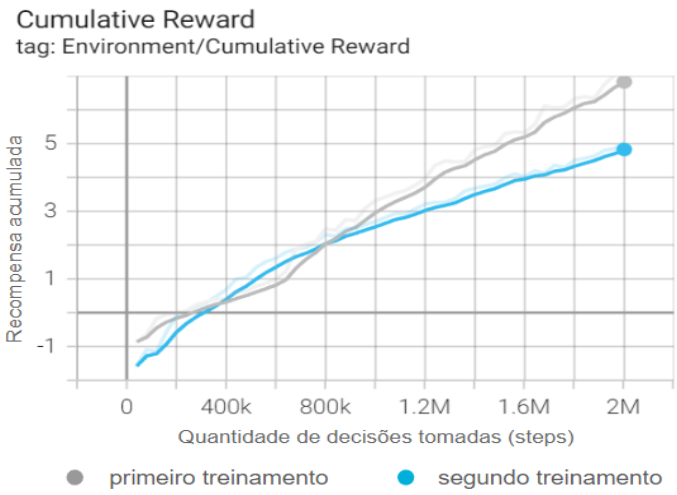


(a) primeiro treinamento (b) segundo treinamento

Fonte: Autor (2022)

Durante o treinamento foi possível observar alguns comportamentos que a rede neural adotou para evitar alguns conflitos, já que uma das formas de perder recompensa era a colisão entre os agentes. Assim, quando a rede neural teve a capacidade de perceber essas situações, foi adotado alguns desvios do trajeto linear para que os indivíduos pudessem esperar os outros à sua frente, o que de certa forma resultou em um trajeto padrão para os demais que estavam atrás.

Figura 7 – Gráfico relacionando recompensa acumulada e episódios



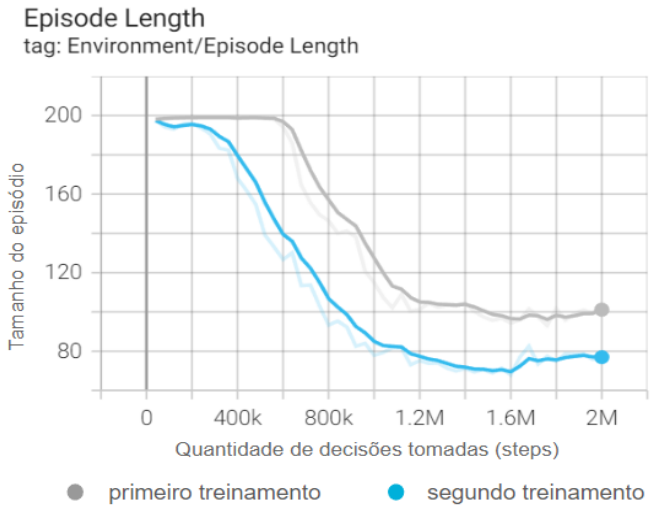
Fonte: Autor (2022)

No gráfico da Figura 7 é possível observar o acúmulo de recompensa (eixo y) do treinamento ao longo dos episódios (eixo x). Duas coisas são importantes entender nessa análise, sendo a quantidade de recompensa acumulada significando o quão mais próximo do objetivo os agentes chegaram, ou seja, quanto maior a recompensa acumulada melhor o treinamento ocorreu. E o outro ponto são os episódios, que são determinados por um ciclo de desenvolvimento do agente, em suma, é uma sequência de estados do ambiente, ações e recompensas, que terminam em um estado final. No caso do *script* aqui criado, o episódio termina ao encontrar a saída ou até atingir 1000 *steps* (quantidade de decisões tomadas).

Ainda no gráfico da Figura 7, os dois resultados apontam que ambos treinamentos tomaram rumo correto nas decisões, acumulando cada vez mais recompensa ao longo do treinamento devido ao sucesso de alcançar a saída.

No gráfico da figura 8 é apresentado um dos resultados mais importantes nessa análise, isso porque é mostrado o quão mais rápido os objetivos foram atendidos até os últimos passos do treinamento. O tamanho do episódio foi sendo reduzido conforme o treino ocorria, destacando que o episódio estava sendo concluído mais rápido em ambos treinamentos, o que significa que a saída estava sendo alcançada. Uma conclusão que pode ser ressaltada é que de fato ao incluir uma porta a mais no ambiente simulado (segundo treinamento), a evacuação tende a ser executada mais rapidamente por não terem tantos indivíduos com grande distância da saída mais próxima.

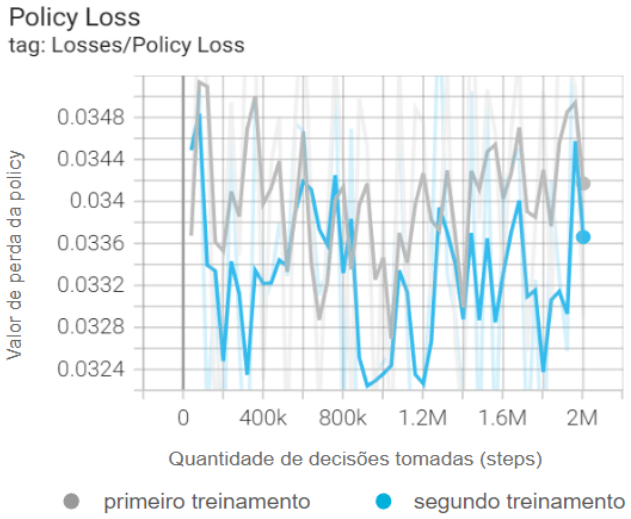
Figura 8 – Gráfico relacionando o tamanho dos episódios ao longo de cada episódio



Fonte: Autor (2022)

A *policy* pode ser descrita como um padrão de execução, ou uma sequência de regras que determina como deve ser o comportamento em dada situação. O agente tenta determinar qual seria a melhor ação na situação atual, então, a *policy loss* mostra o quanto isso foi errado. Dessa forma, o que é representado no gráfico da Figura 9 é como a rede neural alterou a sua forma de tomar decisões para maximizar o seu ganho de recompensa. Assim, é possível observar em ambos treinamentos como aconteceu os ajustes nesse padrão ao longo dos episódios, que pode ter sido bastante, porém com pouca alteração no valor, o que significa um treino estável. A valor não exatamente indica o quanto a *policy* mudou, mas sim o quanto foi a perda da previsão da melhor decisão a ser tomada.

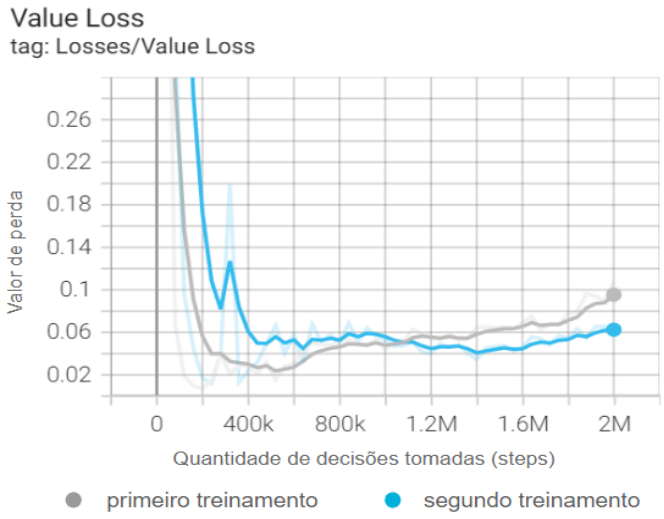
Figura 9 – Gráfico relacionando a alteração da policy ao longo de cada episódio



Fonte: Autor (2022)

O gráfico da Figura 10 determina a perda média da atualização da função de valor. Correlaciona o quão bem o modelo é capaz de prever o valor de cada estado. Isso deve aumentar enquanto o agente está aprendendo e diminui quando a recompensa se estabilizar. Esses valores aumentam à medida que a recompensa aumenta e, em seguida, devem diminuir quando a recompensa se estabiliza. Nesse caso, o gráfico mostra que desde o início a obtenção de recompensa foi estável, o que significa que a perda média foi reduzida devido a discrepância na atualização não ser tão grande em relação à predição feita anteriormente sobre a recompensa que receberia nos próximos passos. Ou seja, o gráfico se torna estável quando a rede neural tem maior capacidade de prever a recompensa que será recebida nas decisões que se sucedem.

Figura 10 – Gráfico relacionando a predição de recompensa ao longo de cada episódio



Fonte: Autor (2022)

Assim que foi possível obter as redes neurais e ter ciência da qualidade do treinamento e a verificação de evacuação dos agentes, pode-se entrar em outra questão onde os resultados podem ser interessantes.

Retomando um dos objetivos deste trabalho, onde era pensado em avaliar um ambiente e sua segurança, além de verificar possíveis melhorias no local para ter uma evacuação mais eficiente, é visto que no mundo real e em um ambiente não simulado, o tempo influencia muito na questão de segurança em locais onde existem acontecimentos perigosos, já que quanto mais rápido for feita a retirada das pessoas dentro da aglomeração, menor será a exposição a riscos de vida. Sendo assim, uma das avaliações nesses dois ambientes utilizados para o treinamento, é a utilização de tempo para avaliar o quão benéfico pode ser uma alteração dentro do ambiente. Ou seja, avaliar, nesse caso, a diferença que ocorreu de um ambiente para outro com diferenças de características.

As redes neurais utilizadas são as apresentadas nos gráficos acima, os agentes são posicionados da mesma forma, e a execução acontece de forma separada, um modelo por vez, cronometrando o tempo necessário para que todos os vinte indivíduos atingissem a saída do local.

Em um primeiro momento, houve a verificação do modelo com apenas uma saída, onde levou cerca de 13,3 segundos para que os agentes fizessem a evacuação total do ambiente. Em contrapartida, a verificação no modelo com mais de uma saída, levou 8,2 segundos para o mesmo objetivo. Possuindo assim, uma eficácia de aproximadamente 38% da segunda execução em relação à primeira.

É necessário entender também que as proporções não são realistas, a velocidade foi adaptada para o modelo, como também as medidas são fictícias. O resultado observado deve ser maior na questão de diferença de segurança adotada nos ambientes, trazendo a informação de que esses agentes conseguem fazer a evacuação na metodologia que foi aplicada nesse contexto. Além disso, a utilização de redes neurais aborda o diferencial na questão de haver uma semelhança maior com a realidade no comportamento dos agentes.

Considerações finais

Este projeto teve como objetivo o desenvolvimento de um ambiente para simular evacuações através de agentes utilizando redes neurais artificiais. O intuito era poder analisar e verificar como acontecia uma evacuação sem a utilização de pessoas reais para testes. Mesmo que em um cenário não realista e com poucas características aplicadas ao cenário, com os testes e resultados apresentados é possível concluir que o objetivo foi alcançado, conseguindo desenvolver uma simulação que trouxe resultados que podem ser debatidos e comparados com futuros testes e projetos equivalentes. Embora houvesse conhecimento prévio na plataforma utilizada, foi necessário buscar conhecimento sobre o desenvolvimento e alguns conceitos que seriam importantes para aplicação. Ainda assim, foram encontradas dificuldades durante a modelagem e em algumas lógicas para a rede neural, que foram superadas através de testes e pesquisas.

Este trabalho também aponta uma nova aplicação de redes neurais e inteligência artificial, o qual possui grande desenvolvimento, abrangendo uma causa que possivelmente pode ser utilizada no mundo real assim que tivermos parâmetros adequados para tal experimento. Como trabalhos futuros, pretende-se adequar ainda mais com a realidade, aplicando treinamentos em locais mais complexos e adotar diferentes formas de treinamento para que mais informações possam ser analisadas. Junto a isso, há interesse em conseguir fazer a determinação de quantidade de pessoas em locais realistas, para que dentro da simulação seja concluído se haverá segurança para evacuação naquele local.

DEVELOPMENT OF AN ENVIRONMENT EVACUATION SIMULATION USING ARTIFICIAL NEURAL NETWORKS

Guilherme Manfroi[¶] João Mário Lopes Brezolin[□]

20 de dezembro de 2022

Abstract

The human behavior is highly influenced by relations with their couples. By composing a crowd population, individuals have a different behavior decreasing their capacities to find rational solutions in danger situations. This paper aims to implement a virtual environment in which a set of individuals will be trained to perform their way out safely. In this way, we seek to evaluate the contributions from using a simulation system that perform the training of individuals behavior through Artificial Neural Networks.

Key-words: Artificial Neural Networks. Crowd control. Unity ML-Agents

Referências

ARAUJO, V. F. d. A. Study and evaluation of human perception on virtual humans and crowds. 2020. Citado 2 vezes nas páginas 2 e 3.

BARBOSA, P. D. V. O comportamento humano nas multidões e seus reflexos na gestão de segurança e operações do sistema metroferroviário. *AEAMESP*, 2014. Citado na página 2.

FLECK, L. et al. Redes neurais artificiais: Princípios básicos. *Revista Eletrônica Científica Inovação e Tecnologia*, v. 1, n. 13, p. 47–57, 2016. Citado na página 4.

GOLDSCHMIDT, R. R. Uma introdução à inteligência computacional: fundamentos, ferramentas e aplicações. *Rio de Janeiro Brasil: IST-Rio*, v. 1, p. 32, 2010. Citado 2 vezes nas páginas 3 e 4.

[¶]<guilherme_gbm@outlook.com>
[□]<joaobrezolin@ifsul.edu.br>

GÖLLNER, S. A competitive 3d-volleyball-game using reinforcement learning with unity ml-agents. 2022. Citado na página 6.

HAYKIN, S. *Redes neurais: princípios e prática*. [S.l.]: Bookman Editora, 2001. Citado na página 4.

JULIANI, A. et al. Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627*, 2018. Citado 3 vezes nas páginas 3, 4 e 5.

KAYSER, C. Implementação de um agente utilizando sistemas neurais híbridos para o ambiente cartpole. 2018. Disponível em: <https://painel.passofundo.ifsul.edu.br/uploads/arq/20190221150556933040472.pdf>. Citado na página 6.

MUSSE, S. R. *Simulação de Multidões de Humanos Virtuais*. [S.l.]: Conhecimento em Rede, 2013. Citado 2 vezes nas páginas 3 e 6.

RAUBER, T. Redes neurais artificiais. 07 2022. Citado na página 3.

RAUBER, T. W. Redes neurais artificiais. *Universidade Federal do Espírito Santo*, v. 29, 2005. Citado na página 3.

SILVA, V. A. et al. Aprendizagem profunda em unity com ml-agents. *Sociedade Brasileira de Computação*, 2020. Citado na página 5.

WANG, S.-C. Artificial neural network. In: *Interdisciplinary computing in java programming*. [S.l.]: Springer, 2003. p. 81–100. Citado na página 3.