CONSTRUINDO UM JOGADOR AUTÔNOMO: IA de aprendizado em jogos usando JavaScript

Larissa de Oliveira Sousa[1]

RESUMO

O presente trabalho apresenta um relato de experiência sobre a construção de uma Inteligência Artificial de aprendizado em jogos, vivenciado no contexto de tecnologia, demonstrando o aprendizado de máquina de um jogador autônomo durante os meses fevereiro e março de 2024. O objetivo da pesquisa consiste em apresentar aprendizado de máquina, que é um campo da inteligência artificial que se concentra nos desenvolvimentos de algoritmos e técnicas que permitem computadores aprenderem a partir de dados, utilizando ferramentas que permitem visualizar este aprendizado. Através desta narrativa, foram desenvolvidos um jogador autônomo para demonstrar a evolução do aprendizado de máquina de um jogo, isso aconteceu a partir de ferramentas utilizadas no desenvolvimento web. Foram analisados e descritos a construção de uma inteligência artificial onde demonstrou seu aprendizado. A experiência baseou-se teoricamente em ensinar “joaninhas" a seguir em frente desviando-se de “predadores” usando sensores frontais. De acordo Endo e Tanaka (2023), estudos recentes mostram que estão testando métodos de IA em jogos da atualidade, devido à vasta incerteza e complexidade. A complexidade é um fator relevante nos estudos de algoritmos de desenvolvimento de inteligências artificiais, pois quanto mais complexo o jogo maior é a dificuldade da pesquisa e a inclusão de inteligência artificial. Além disso, o aprendizado de máquina tem contribuído para melhoria em diversos campos de atuação. Esse é o resultado dos avanços nas últimas décadas que permitiram projetos inovadores baseando-se em inteligência artificial. Acredita-se que os resultados deste relato possam contribuir para o debate e reflexões sobre inteligência artificial e seu impacto na aprendizagem de máquinas em jogos.

Palavras-chave: IA. Robôs. Inteligência Artificial. Aprendizado de Máquina .

[1] Graduando(a) em Bacharelado em Sistemas de Informação no Centro Universitário do Sul de Minas.

1 INTRODUÇÃO

Segundo Feneda (2016), o cérebro é composto por bilhões de neurônios, estruturas celulares que apresentam três seções distintas - corpo, dendritos e axônios; cada uma com suas funções específicas e complementares. Os dendritos são responsáveis por captar os estímulos recebidos ao longo do tempo e transmiti-los ao corpo do neurônio, onde são processados. Quando estes estímulos atingem um determinado limiar, o corpo do neurônio emite um novo impulso que se propaga pelo axônio e é transmitido a células vizinhas através de sinapses. Este processo pode ocorrer em várias camadas de neurônios, resultando no processamento da informação de entrada e eventualmente levando o cérebro a comandar reações físicas.

Desta forma conceito de rede neural artificial pode ser compreendido como um modelo computacional que se assemelha ao funcionamento dos neurônios do cérebro humano. A construção de uma rede neural também está diretamente relacionada a princípios matemáticos, sendo fundamental a escolha da linguagem de programação que melhor atenda aos objetivos estabelecidos. Assim, é possível integrar linguagem de alto nível para resolver problemas matemáticos com o objetivo de aprendizado de máquina.

As redes neurais artificiais são um subcampo da inteligência artificial (IA) que é um conceito difundido na área da ciência da computação, no qual algoritmos são desenvolvidos com o propósito de aprendizado e reconhecimento de padrões, coletando informações suficientes para solucionar diversos problemas para a tomada de decisões.

O desenvolvimento do trabalho depende de um conhecimento sólido na teoria que descreve o que é uma rede neural artificial, sendo necessário aplicar esse conhecimento teórico por meio da implementação de um algoritmo em JavaScript.

Ao aplicar a teoria é fundamental validar o processo de aprendizagem por meio de representação visual em um jogo onde a experiência baseia-se teoricamente em ensinar “joaninhas" a seguir em frente desviando-se de “predadores” com o apoio de sensores frontais utilizando ferramentas de desenvolvimento web. Sendo assim, o principal desafio é

ser capaz de visualizar de maneira eficaz o processo de aprendizagem do jogador autônomo e seus neurônios, acompanhando a sua evolução.

Então, este trabalho tem como objetivo geral descrever o processo de desenvolvimento de uma rede neural artificial desde sua criação até o acompanhamento do seu aprendizado. Sendo assim o objetivo da pesquisa consiste em apresentar aprendizado de máquina, que consiste em um campo da inteligência artificial que se concentra nos desenvolvimentos de algoritmos e técnicas que permitem computadores aprenderem a partir de dados. Como objetivos específicos, serão apresentados os conceitos teóricos sobre rede neural artificial, o desenvolvimento de um jogo em JavaScript, HTML e CSS, a construção do algoritmo da rede neural em JavaScript, a conexão dos neurônios à dinâmica do jogo e monitoramento do aprendizado da joaninha ao desviar dos obstáculos corretamente.

Uma vez que o jogo é iniciado, a proposta será acompanhar o processo de aprendizado de máquina a partir do jogo, monitorando seus erros e acertos. Além disso, será possível observar quais neurônios estão ativos durante o processo de aprendizado. Dessa forma, será possível acompanhar o processo de aprendizagem de uma IA.

2 REFERENCIAL TEÓRICO

Apesar dos modelos de IA que temos hoje em dia serem relativamente recentes, a inteligência artificial já estava em evolução há muito tempo. Conforme Vasco(2020), a história da Inteligência Artificial é datada em 1950, onde Alan Turing propôs o teste de Turing. Este teste tinha como objetivo verificar a inteligência de uma máquina, sendo considerada inteligente caso o ser humano não fosse capaz de distinguir entre a máquina e outro ser humano.

Com a evolução da inteligência artificial, ela está se tornando cada vez mais relevante em diversos setores, o que resulta em uma aplicação cada vez mais ampla. De acordo com a revista EXAME (2023), no cotidiano e no ambiente de trabalho dos brasileiros, observa-se uma crescente adoção de ferramentas de inteligência artificial.

Uma rede neural sendo uma técnica dentro do campo de inteligência artificial é capaz de assemelhar ao sistema neural que temos no corpo humano. De acordo com IBM (2023), as redes neurais cuja estrutura é modelada com base no funcionamento do cérebro humano, têm

como objetivo reproduzir o processo biológico dos neurônios, destacando a importância da transmissão de sinais entre essas unidades para o funcionamento do sistema. Esse padrão de processamento, inspirado na complexa rede de neurônios no cérebro, é essencial para a capacidade das redes neurais artificiais de aprender e realizar tarefas complexas de maneira eficiente. Além disso, Ferneda (2006) aponta que a capacidade dos seres humanos de executar funções complexas, especialmente sua habilidade de aprender, é atribuída ao processamento paralelo e distribuído pela rede de neurônios do cérebro.

De acordo com o estudo de Ribeiro (2019); Lucchese (2019); Rocha (2019) e Figueiredo (2019), uma rede neural pode ser descrita como uma estrutura de processamento, que pode ser implementada em dispositivos eletrônicos. Ela é composta por unidades interconectadas, chamadas neurônios artificiais, onde cada unidade possui um comportamento específico.

Conforme Vasco(2020), no contexto de aprendizagem de máquina, os algoritmos são treinados através da exposição a uma grande quantidade de dados, permitindo assimilar e compreender melhor as informações processadas. À medida que adentramos na era do *Big Data* somos confrontados com uma imensa quantidade de dados disponíveis para alimentar aplicações de aprendizado.

De acordo com Ludermir (2021), uma rede neural é uma técnica de inteligência artificial que possibilita verificar o processo de aprendizagem. Um exemplo notável do poder das redes neurais é a competição ImageNet Challenge, onde o objetivo é reconhecer diferentes objetos em um conjunto de dados com cerca de 14 milhões de imagens .Um dos desafios enfrentados em relação às redes neurais profundas, uma arquitetura composta por múltiplas camadas de neurônios, foi superado pela AlexNet durante essa competição, alcançando um erro de apenas 15,3%, representando uma significativa melhoria em relação a outros sistemas. Em 2014, o erro dos sistemas de inteligência artificial e aprendizado de máquina diminuiu para 7,3%, e em 2016, chegou a apenas 3,6%, superando até mesmo a precisão humana que era de 5,1%. Ainda de acordo com Ludermir (2021), o avanço das capacidades dos computadores contemporâneos é atribuído em parte ao desenvolvimento de técnicas de aprendizado de máquina.

Conforme Gregori (2009), inicialmente, os jogos eram concebidos em ambientes limitados, geralmente em hardware especializados e sua percepção era de mero entretenimento, mais próximo de um brinquedo comum do que de uma forma sofisticada de

expressão artística. No entanto, ao longo do tempo aconteceu uma transformação relevante na indústria. Atualmente os jogos eletrônicos estão em uma posição de destaque como uma das indústrias mais influentes e lucrativas do mundo. Esse crescimento exponencial é em grande parte atribuído ao desenvolvimento tecnológico e à evolução das práticas de desenvolvimento.

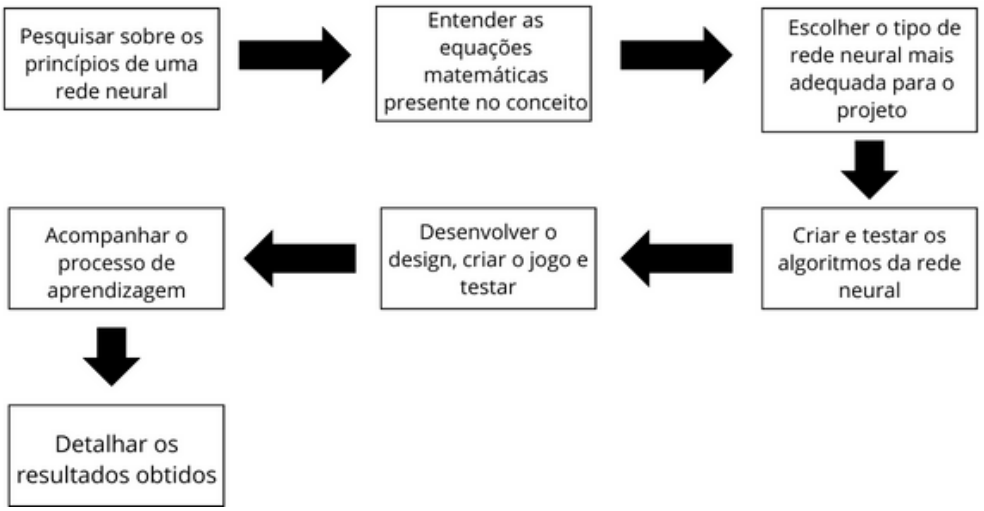
Os jogos podem ser utilizados para acompanhar o processo de aprendizagem. De acordo com CARVALHO(2015), dentro do universo dos jogos existe uma estrutura organizada na qual os jogadores se envolvem para superar objetivos, sendo como o último objetivo alcançar a vitória . Além disso, de acordo com Ribeiro (2019); Lucchese (2019); Rocha (2019) e Figueiredo (2019), entender como os seres humanos percebem o mundo ao seu redor é fundamental para criar a ilusão de atenção em um agente artificial. Isso significa que os algoritmos estão sendo desenvolvidos a partir de modelos que imitam os nossos sentidos como visão, audição e olfato, processando informações para que os agentes artificiais possam interagir com o ambiente de uma forma semelhante a nossa. .

3 MATERIAIS E MÉTODOS

Neste estudo foi utilizada uma metodologia experimental para analisar a capacidade de aprendizado da rede neural durante a execução de um jogo.

Dentro do conceito de rede neural, foi necessário detalhar aspectos que ajudam a definir o conceito de rede neural, que são: Aprendizado de máquina, um algoritmo que possui como foco aprendizado contínuo com o passar do tempo; Redes Neurais Artificiais que são algoritmos inspirados no cérebro humano; Processamento de Linguagem Natural que são algoritmos capazes de compreender texto de fala humanos; e Aprendizado profundo onde as redes neurais são capazes de acessar dados de forma mais eficaz. Para concretizar os objetivos propostos neste trabalho, seguiram-se as seguintes etapas:

Figura 1: Etapas de desenvolvimento do trabalho



Fonte: a autora

O jogo tem como objetivo levar uma joaninha a completar uma fase, desviando dos seus predadores. Foi desenvolvido em HTML e CSS, onde foram incluídos também os seguintes arquivos essenciais para alcançar os resultados desejados no projeto. São eles:

1. **index.html:** responsável pela exibição dos elementos da rede neural na interface.
2. **style.css**: responsável pelos estilos que serão exibidos na tela durante a execução do jogo.
3. **main.js:** encarregada por receber todas as classes e funções presentes no projeto.
4. **network.js:** responsável pelas classes e funções para construção da rede neural.
5. **road.js:** responsável pelo local de exibição do processo de aprendizagem.
6. **sensor.js:** onde se encontram as funções que configuram os sensores utilizados no jogo.
7. **utilis.js:** onde se encontram funções auxiliares para o desenvolvimento eficiente da rede neural.
8. **visualizer.js:** onde se encontram as funções para visualização dos neurônios da rede neural;
9. **joaninha.js:** Agrupa classes e funções essenciais para criação de joaninhas e predadores, elementos utilizados para demonstrar o funcionamento da aprendizagem.
10. **controls.js:** Gerencia os movimentos das joaninhas durante a execução do jogo.

Essa composição de arquivos garante a integração dos elementos visuais, lógicos e funcionais, contribuindo para o pleno funcionamento e compreensão do jogo, principalmente em relação à simulação do processo de aprendizagem da rede neural.

3.2 Princípios de uma rede neural desenvolvido em javaScript

Este projeto será capaz de empregar técnicas de javaScript para o desenvolvimento de um objeto autônomo capaz de ser construído sem bibliotecas externas. Os objetos autônomos podem realizar tarefas e tomar decisões de forma independente, fazendo com que algoritmos possam interagir com o ambiente de forma autônoma e eficaz.

O uso do JavaScript para desenvolver um objeto autônomo faz com que a flexibilidade e a versatilidade da linguagem possibilitem a criação de algoritmos complexos que resolvam o objetivo esperado.

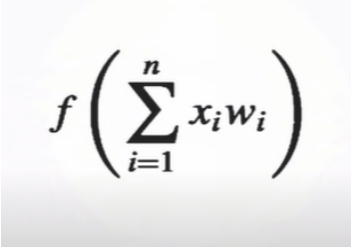
Uma rede neural é formada por três camadas que são: neurônios, camada de entrada, camada oculta e a camada de saída.

3.3 Equações matemáticas presentes no conceito de uma rede neural

Uma rede neural pode ser considerada uma função universal de aproximação, pois tudo que ela faz é aproximar entradas e saídas e o resultado chegar cada vez mais próximo da saída. Desta forma ela busca constantemente melhorar a precisão desses elementos. O processo consiste em diminuir gradualmente a discrepância entre os resultados obtidos e as saídas desejadas, visando uma proximidade cada vez maior entre o que é imputado e sua respectiva saída.

Como observamos na figura 2, podemos visualizar a função que define entradas se aproximarem de suas saídas. Esta função retorna zero para valores negativos e o próprio valor para valores positivos permitindo que a rede aprenda padrões mais complexos.

Figura 2 Função que define entradas se aproximarem da saída



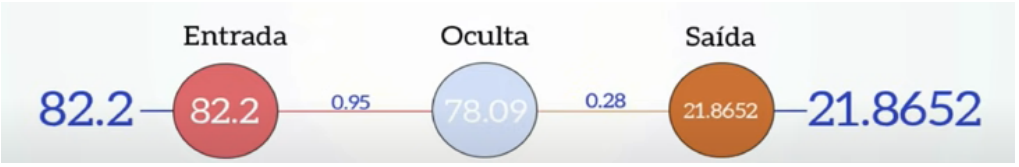
$$f\left(\sum_{i=1}^{n} x_i w_i\right)$$

Fonte: José Bezerra (2019)

Como exemplo de uma função *Feedforward*, a princípio temos a entrada de 82.2, essa entrada está conectada com um nó oculto, este tem peso de 0,95, a conexão entre entrada e oculta possui 0,95, então essa entrada de 82.2 será multiplicada pelo peso 0,95 que dá o resultado 78,09. Esse valor será multiplicado para dar o valor da saída 21,86, porque a camada oculta está conectada à saída. A importância desse são os pesos que são responsáveis pelo aprendizado, pois toda vez que der 82,2 e ela tem como saída 21,86. Digamos que a resposta esperada seja 30, como 81,2 entrou e a resposta 21,86 está distante 30, então o valor de saída 21,86 está longe do que deveria ser. Portanto ele vai calcular esse erro fazendo 30 - 21,86 e ajusta os pesos para que chegue o mais próximo de 30.

Podemos notar na figura 3, o exemplo de uma função *feedforward*, que podemos definir como uma arquitetura onde a informação flui em uma direção específica, sem retornar para estágios anteriores do processo.

Figura 3 Exemplo da função Feedfoward



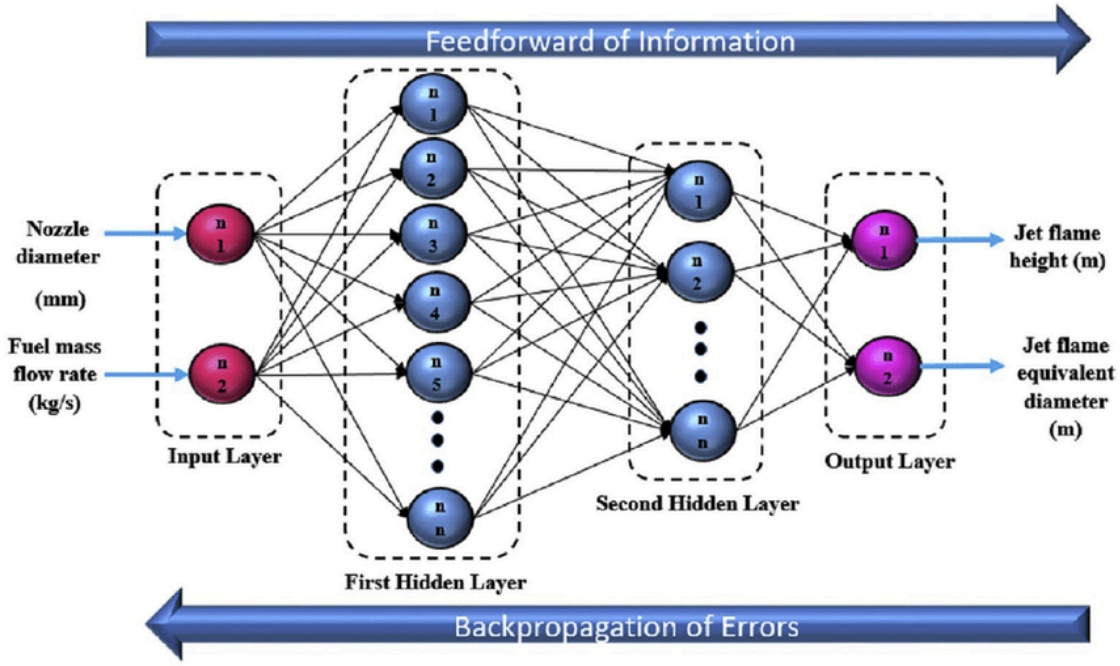
Fonte: José Bezerra (2019)

3.4 Tipo de rede neural mais adequada para trabalho

Para este projeto, o tipo de rede mais apropriado é o *FeedForward*. Essa escolha se baseia na capacidade dessa arquitetura de atender as necessidades específicas do projeto, permitindo que o processo alcance o objetivo desejado de autonomia em relação ao objeto observado. A estrutura unidirecional do *FeedForward* facilita a manipulação eficiente dos dados, oferecendo uma base sólida para a realização dos objetivos do projeto.

O modelo de rede neural escolhido para o projeto é a Recurrent Neural Network (RNN), onde o processo de aprendizagem se desdobra em duas fases distintas. O *Backpropagation* desempenha um papel crucial, direcionando os dados desde a camada de saída e realizando correções em direção à camada oculta, para, em seguida, ajustar a camada de entrada. Simultaneamente, o *Feedforward* assume a responsabilidade de transmitir os dados desde a entrada até a camada oculta e, finalmente, até a saída. Este processo propaga informações em direção a cada camada de saída. A figura 4 demonstra o processo de *Feedforward* e *Backpropagation.*

Figura 4 Recurrent neural network



Fonte: Hossein Mashhadimoslem (2020)

Nas diversas camadas da rede neural, a primeira é a camada de entrada, onde os dados são inseridos. A camada oculta assume a responsabilidade pelo processamento desses dados, enquanto a camada de saída fornece a resposta, representando a predição da rede neural. Cada unidade é denominada um "nó" e está conectada ao primeiro nó da camada oculta. Cada conexão entre nós possui um peso associado. A entrada, que é um número, é multiplicada pelo peso da conexão com um nó na camada oculta. Esse resultado, por sua vez, é multiplicado pelo peso associado à conexão com a camada de saída. Todos esses nós são, essencialmente, representações numéricas que encapsulam o aprendizado da rede neural.

3.5 Criando e testando os algoritmos da rede neural

O primeiro passo é construir três arquivos: um arquivo chamado main.js, onde ficarão as principais classes e funções do projeto, index.html onde exibiremos os elementos na tela e outro style.css onde definimos os estilos da página. Primeiro criamos um retângulo utilizando CSS, com a largura 600 pixels e altura de 647 pixels, este retângulo será responsável pela limitação da área, que será incluído posteriormente as joaninhas e seus predadores, onde estes deverão respeitar a limitação do retângulo. Ainda trabalhando com estilos vamos definir a cor do retângulo como verde, representando a grama por onde passarão as joaninhas e os predadores.

Depois adicionaremos o arquivo joaninha.js, neste momento será criado uma classe chamada joaninha, e o construtor para essa classe terá quatro parâmetros diferentes, um parâmetro de largura e outro de altura para definirmos o tamanho do espaço que a joaninha irá ocupar, outro dois parâmetros x e y que serão utilizados posteriormente para definir um polígono para movimentação das joaninhas. O x será o centro da joaninha e o valor de y será a metade do tamanho da joaninha.

Outro arquivo que será adicionado é controls.js, este arquivo terá uma classe para controles. E o construtor, o objeto de controles, terá quatro atributos diferentes, um deles saberá se está indo para frente, um para a esquerda, um para a direita e um para o reverso. Depois será definido um método para verificar as entradas do teclado, sendo assim, se pressionado o botão de esquerda no teclado, o objeto deverá ir para esquerda, o mesmo acontece com outras direções.

Voltando para o arquivo joaninha.js, será escrito um método de atualização onde será implementado a animação para a joaninha, detalhando como os controles serão usados para mover a joaninha para frente ou para trás, ajustando sua posição vertical. Além disso, é criada uma função de animação para atualizar continuamente a posição da joaninha na tela, criando a ilusão de movimento. Será adicionado também um parâmetro para velocidade máxima e atrito para controlar a velocidade da joaninha e impedir que ela acelere indefinidamente.

Neste momento será definido a localização das joaninhas e dos predadores no retângulo criado anteriormente. São ajustados a distância entre os predadores usando interpolação linear para que sejam distribuídos uniformemente. É criado um novo arquivo, utils.js, para que a função de interpolação linear seja movida para o arquivo utils.js já que será usada muitas vezes. Os arquivos .js são movidos para index.html para que seja possível visualizar o que foi construído até agora.

Depois, será implementado o arquivo sensor.js, o sensor que será anexado a joaninha, inicialmente definindo a estrutura do sensor e seus métodos de atualização e desenho. Também é vinculado o sensor a joaninha permitindo que ele seja atualizado e desenhado junto a joaninha. O sensor neste momento depende da implementação de melhorias. Primeiramente é ajustado o sensor para se mover junto com a joaninha, também é ajustado os limites do sensor, definindo como será identificado possíveis colisões com os predadores.

Neste momento é criado a matriz de predadores. Os predadores são identificados como *dummy*, que se movem a uma velocidade fixa e não possuem sensores. O código é ajustado para que a joaninha interaja com os predadores detectando colisões. Além disso, é necessário garantir que o sensor da joaninha reconheça não apenas o limite do retângulo mas também os predadores.

Agora será implementada a rede neural, ela será dividida em níveis, onde cada nível terá uma camada de entrada, uma camada de saída e conexões entre elas. O primeiro nível da rede neural consiste em definir as camadas de entrada e saída, bem como os pesos e viés das conexões entre elas. Estes pesos e viés são inicialmente atribuídos aleatoriamente para criar uma “rede cerebral” inicial. Desta forma é possível desenvolver gradualmente a rede neural, adicionando mais camadas e refinando os pesos e viés ao longo do processo. A rede neural é conectada aos sensores da joaninha, permitindo que ela tome decisão de direção com base nas entradas sensoriais.

Neste momento, será descrito o processo de definição dos pesos e vieses de uma rede neural. Os pesos representam o peso ou a importância das conexões entre neurônios em diferentes camadas da rede, enquanto vieses são adicionados aos neurônios de saída para controlar sua ativação. Estes parâmetros são essenciais para o funcionamento da rede, pois determinam como os sinais são processados e transformados durante a propagação através da rede. Os pesos são inicialmente atribuídos aleatoriamente a partir de 0, pois a inicialização aleatória ajuda a evitar vieses indesejados na rede. Essa aleatoriedade é importante para garantir que a rede neural possa aprender e se adaptar aos padrões nos dados de entrada. Este é apenas o primeiro nível de uma rede neural artificial.

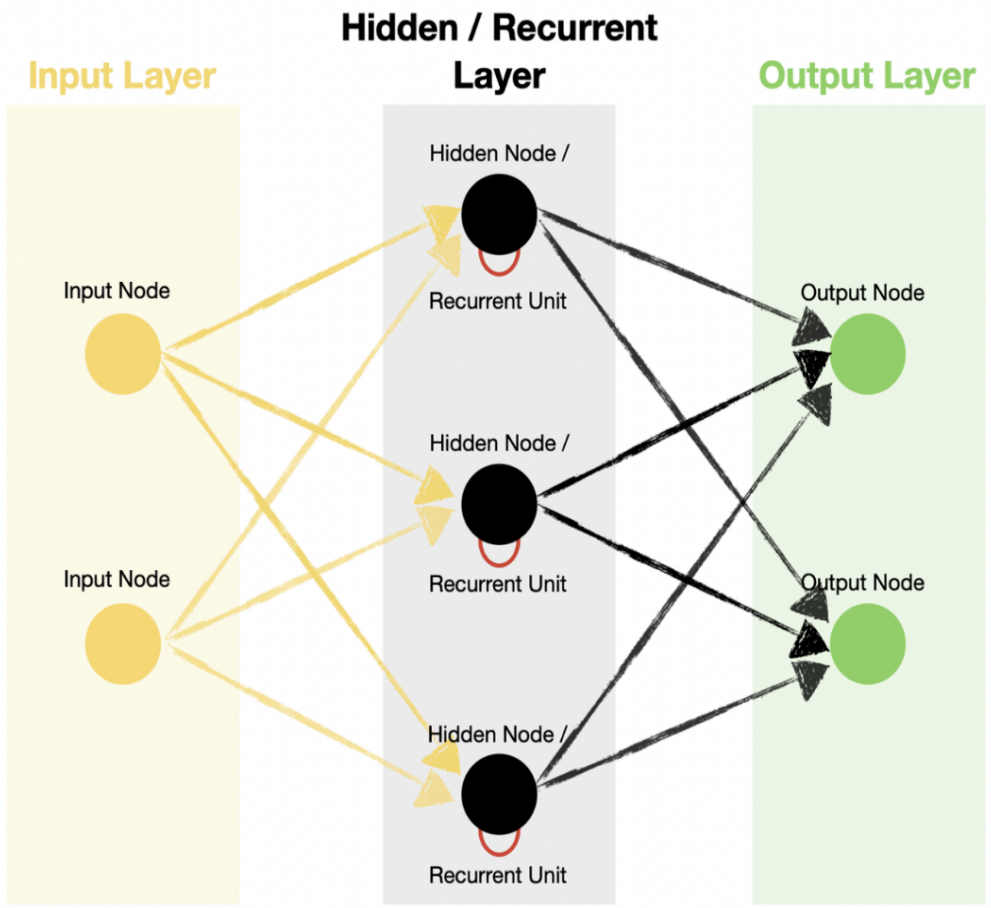
Neste momento cada neurônio na camada de *input* representa um sensor da joaninha, e cada neurônio na camada de *output* representa uma ação de controle (avançar, retroceder, virar para a esquerda ou para a direita). Os valores dos sensores da joaninha são usados como entrada para rede neural, os valores são processados pela rede neural que retorna as saídas.

Agora é realizada a configuração da visualização, onde é definido a interface gráfica dividindo a tela em dois setores: um para visualizar o ambiente da joaninha e predadores e outro para mostrar a rede neural em ação. Depois, é escrito o código para criar a estrutura da rede neural, incluindo camadas de entrada, ocultas e de saída. Também é inicializado os pesos e os viés da rede neural com valores aleatórios. O algoritmo de *feedforward* na rede neural é incluído permitindo que ela tome decisões de direção com base nas leituras dos sensores virtuais do carro. As decisões são calculadas usando os pesos e os viés da rede neural. Desta forma, é expandido o projeto para incluir múltiplas joaninhas controladas por diferentes redes neurais. Isso permite a comparação e análise de várias estratégias de controle.

Desta forma a joaninha que se aproxima do seu predador tem o sensor ativado, indicando que a joaninha deverá ir para outra direção. Nesta etapa é possível também definir a quantidade de sensores, estes sensores precisam se dispersar, no caso do projeto faremos uma dispersão de 45° entre os sensores.

Conforme a figura 5, ao construir uma rede neural podemos dividi-las em três partes: uma camada de neurônios na entrada, outra camada para tendências e outra para saídas.

Figura 5 Recurrent neural network



Fonte: Vivek Bhagat (2022)

4 RESULTADOS E DISCUSSÕES

O principal propósito deste projeto foi a elaboração de uma rede neural, proporcionando o acompanhamento do processo de aprendizagem de máquina, incluindo visualizações que evidenciam o projeto da rede.

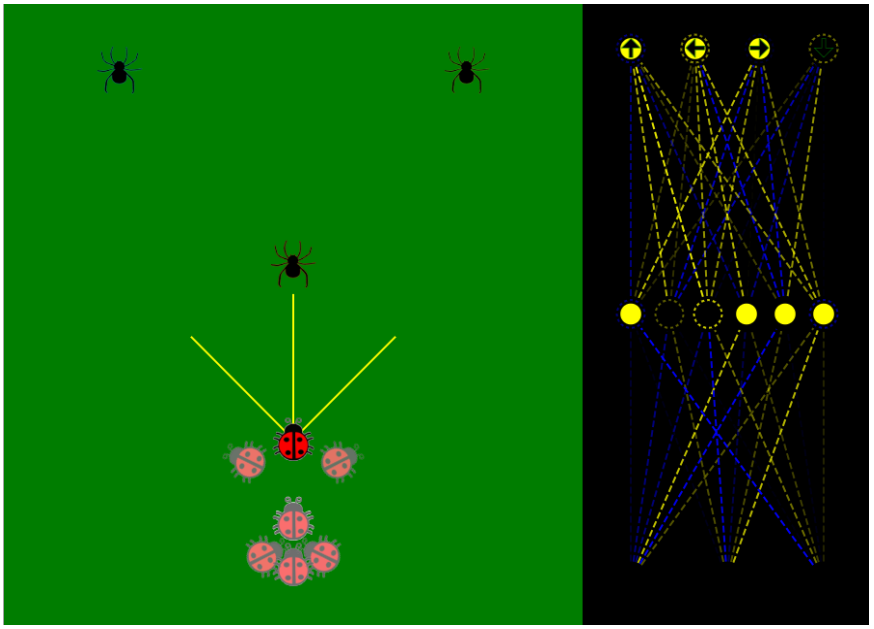
Conforme a figura 6, com os sensores configurados, foram aplicados uma regra a estes sensores, a cada vez que se aproximar de um predador a joaninha deverá ir para outra direção, a cor de colisão também deverá ser alterada para preto, identificando ali uma possível colisão. Neurônios na primeira colisão serão conectados aos sensores, e enviarão sinais, algumas vezes a última camada será conectada aos controles da joaninha para que ela tome uma nova decisão.

Figura 6 Funcionamento do sensor

Fonte: a autora

A figura 7 apresenta o lançamento de gerações da joaninha, cada geração é uma aprendizagem, neste caso, conforme as limitações do hardware foram gerados 100 gerações de joaninhas, onde cada uma delas aprendeu a evitar determinadas direções.

Figura 7 Lançamento das gerações da joaninha



Fonte: a autora

Como resultado do código que gera e rede neural, tem-se a figura 8 que apresenta a classe NeuralNetwork, a classe em um contexto de rede neural é responsável por representar a própria rede neural e coordenar suas operações, como inicialização, propagação direta e mutação. Seu construtor (neuronCounts), inicializa a rede neural com base no número de neurônios em cada camada. O método estático feedForward(givenInputs, network), implementa o algoritmo de feedforward, já o método mutate(network, amount=1) é estático, e tem a função de alterar os pesos e os vieses da rede, recebendo como parâmetro a rede neural (network) e opcionalmente uma quantidade (amount) que controla o tamanho da mutação.

Figura 8 Classe NeuralNetwork

Fonte: a autora

Na figura 9 temos a classe Level, em um contexto de rede neural é responsável por representar uma camada da rede. Ela é fundamental para processar entradas e produzir saídas durante a fase de propagação direta (feedforward) da rede neural. Na classe Level, o construtor(inputCount, outputCount) é responsável por inicializar uma camada da rede neural com o número de entradas e saídas especificado. inputCount representa o número de neurônios na camada de entrada, enquanto outputCount representa o número de neurônios na camada de saída. Além disso, o método estático randomize(level) é uma função privada que inicializa aleatoriamente os pesos e os vieses da camada. Por fim, o método estático feedForward(givenInputs, level) implementa a propagação direta (feedforward) em uma camada específica da rede neural. Ele recebe uma entrada (givenInputs) e a camada (level) e calcula a saída da camada para a entrada dada.

Figura 9 Classe Level

Fonte: a autora

Por fim, o código-fonte está disponível publicamente no GitHub (https://github.com/oliveiralar/joaninha-sem-bibliotecas.git).

5 CONSIDERAÇÕES FINAIS

O objetivo desta pesquisa foi de criar um jogador autônomo, desta forma destacamos a importância do aprendizado de máquina como um componente essencial da inteligência artificial, permitindo que um jogador aprenda a partir de dados e realize tarefas de forma autônoma. Este estudo destacou o papel essencial das ferramentas de visualização no processo de aprendizagem das máquinas, impulsionando o desenvolvimento de jogos autônomos que interagem de forma inteligente com o ambiente. É esperado que este trabalho motive pesquisas adicionais e avanços contínuos nesse campo dinâmico e em constante evolução.

Apesar das múltiplas possibilidades, a principal observação da joaninha foi a preferência de permanecer atrás do predador a ser evitado, desta forma conseguimos cumprir o objetivo de visualizar um jogador autônomo em ação, mostrando diferentes resultados a cada geração lançada. Como consequência, várias gerações subsequentes optaram por seguir o mesmo comportamento.

Este trabalho tem potencial para ser base para estudos futuros sobre jogadores autônomos desenvolvidos por meio da inteligência artificial. Contribuindo para o desenvolvimento de novas tecnologias e aplicações inovadoras no âmbito de jogos autônomos e da inteligência artificial.

BUILDING AN AUTONOMOUS PLAYER: Learning AI in Games Using JavaScript

ABSTRACT

The present work presents an experience report on the construction of an Artificial Intelligence for learning in games, experienced in the context of technology, demonstrating the machine learning of an autonomous player during the months of February and March 2024. The objective of the research consists of introduce machine learning, which is a field of artificial intelligence that focuses on the development of algorithms and techniques that allow

computers to learn from data, using tools that allow you to visualize this learning. Through this narrative, an autonomous player was developed to demonstrate the evolution of machine learning in a game, using tools used in web development. The construction of artificial intelligence was analyzed and described, demonstrating its learning. The experience was theoretically based on teaching “ladybugs” to move forward by avoiding “predators” using frontal sensors. According to Endo and Tanaka (2023), recent studies show that they are testing AI methods in current games, due to to vast uncertainty and complexity. Complexity is a relevant factor in studies of artificial intelligence development algorithms, as the more complex the game, the greater the difficulty of research and the inclusion of artificial intelligence. Furthermore, machine learning has contributed for improvement in various fields of activity. This is the result of advances in recent decades that have allowed innovative projects based on artificial intelligence. It is believed that the results of this report can contribute to the debate and reflections on artificial intelligence and its impact on machine learning in games.

Keywords: AI. Robots. Artificial intelligence. Machine Learning**.**

REFERÊNCIAS

CARVALHO, Carlos. **APRENDIZAGEM BASEADA EM JOGOS**. Publicado em 2023. Disponível em: https://www.ibm.com/br-pt/topics/neural-networks. Acesso em:19 de setembro de 2023.

EXAME. **Gastos dos brasileiros com inteligência artificial cresceram 120% nos cinco primeiros meses de 2023**. Publicado em 2023. Disponível em: https://exame.com/inteligencia-artificial/transacoes-com-inteligencia-artificial-120/. Acesso em:19 de setembro de 2023.

ENDO, Wellington; Tanaka, Simone. **Métodos Evolutivos de Inteligência Artificial Para Jogos** Publicado em 2023. Disponível em: https://canaltech.com.br/inteligencia-artificial/o-que-e-aprendizado-de-maquina/. Acesso em:20 de março de 2024.

FERNEDA, Edberto. (2006). **Redes neurais e sua aplicação em sistemas de recuperação de informação**. Publicado em 2006. Disponível em: https://www.scielo.br/j/ci/a/SQ9myjZWLxnyXfstXMgCdcH/. Acesso em: 25 de março de 2024.

GREGORY, Jason. Game Engine Architecture. Publicado em 2009. E-book. Disponível em: http://www.latexstudio.net/wp-content/uploads/2014/12/Game_Engine_Architectureen.pdf. Acesso em: 26 de Março de 2024

IBM. **O que são redes neurais?**. Publicado em 2023. Disponível em: https://www.ibm.com/br-pt/topics/neural-networks. Acesso em:19 de setembro de 2023.

LUDERMIR, Tereza. **Inteligência Artificial e Aprendizado de Máquina: estado atual e tendências.** Publicado em 2021. Disponível em : https://www.scielo.br/j/ea/a/wXBdv8yHBV9xHz8qG5RCgZd/?lang=pt&format=html#. Acesso em: 25 de março de 2024.

RIBEIRO, Bruno; LUCHEESE, Fabiano; ROCHA, Maycon; FIGUEIREDO, Vera. **Inteligência Artificial em Jogos Digitais**. Publicado em 2019. Disponível em: http://www. dca. fee. unicamp. br/~ martino/disciplinas/ia369/trabalhos/t4g3. Acesso em: 5 de abril de 2024..

VASCO, Lucas. **Um Estudo de Redes Neurais Recorrentes no Contexto de Previsões no Mercado Financeiro**. Publicado em 2020. Disponível em: https://repositorio.ufscar.br/bitstream/handle/ufscar/13730/Trabalho_de_Conclusao_de_Curso_Lucas_Vasco.pdf?sequence=2&isAllowed=y. Acesso em: 14 de setembro de 2023