DEPARTAMENTO DE COMPUTAÇÃO
CURSO DE CIÊNCIA DA COMPUTAÇÃO

Eduardo Schmidt Lohmann

Técnica de Redes Neurais Artificiais aplicada ao Reconhecimento de Imagens para a interação com um Jogo Computacional

Santa Cruz do Sul
2016

Eduardo Schmidt Lohmann

Técnica de Redes Neurais Artificiais aplicada ao Reconhecimento de Imagens em um Jogo Computacional

Trabalho de Conclusão II apresentado ao Curso de Ciência da Computação da Universidade de Santa Cruz do Sul para obtenção do título de Bacharel em Ciência da Computação.

Orientadora: Profª. Dra. Rejane Frozza

Santa Cruz do Sul

2016

AGRADECIMENTOS

Quero agradecer primeiramente a minha mãe Marlene e meu pai Rudi por todos os ensinamentos, apoio, dedicação e carinho que me deram desde o início da minha vida.

Meus amigos e irmão, que entenderam e perdoaram minhas faltas devidas aos trabalhos e provas, que sempre estiveram ao meu lado, dando conselhos e causando gargalhadas mesmo nos tempos mais estressantes.

A minha família que sempre me apoiou e incentivou a manter a concentração nos estudos.

Aos colegas de curso por tornarem as aulas e trabalhos momentos divertidos em meio ao conteúdo maçante, e por vezes ajudarem com as dúvidas mais cruciais no entendimento das matérias.

A minha orientadora e amiga Profª. Drª. Rejane Frozza, que me acompanhou e instruiu durante o desenvolvimento do trabalho, e forneceu todo apoio e contribuição necessária para sua realização.

Aos professores do Departamento de Computação da UNISC, por todos ensinamentos e experiências compartilhados.

De maneira geral, a todos que de alguma maneira contribuíram para a realização desta graduação.

RESUMO

Redes Neurais Artificiais (RNAs) são uma área da inteligência artificial com a principal característica de ser utilizada no reconhecimento de padrões, como, por exemplo, o reconhecimento de imagens. A junção de mais áreas do conhecimento é algo comum e implícito para a utilização de RNAs, assim, a proposta deste trabalho possui esta premissa, a junção de RNAs para reconhecimento de imagens em jogos computacionais. O objetivo é desenvolver um sistema baseado em RNAs para reconhecer padrões desenhados por um jogador e a semelhança destes com as imagens originais apresentadas em um jogo computacional. As RNAs são capazes de executar os processos de aprendizado e reconhecimento de padrões, por este motivo, foi a técnica escolhida para o desenvolvimento desta aplicação. Como resultados obtidos, destaca-se a implementação da RNA que é capaz de treinar e reconhecer padrões de desenho, assim como o armazenamento dos dados coletados sobre seus treinamentos, em um jogo de aventura.

Palavras-chave: Redes Neurais Artificiais. Reconhecimento de Imagens. Jogos Computacionais.

ABSTRACT

Artificial Neural Networks (ANNs) are a field in Artificial Intelligence with the main characteristic of being used in patter recognition, as in, for instance, recognizing images. The union of more fields of study is something common and implicit in the use of ANNs, the proposal of this paper has this fundament, the junction of Artificial Neural Networks in image recognition with computer games. The goal is the creation of a system based on ANN that evaluates drawn patters by a player and recognize the resemblance between these and the original images presented by a computer game. ANN are capable of recognizing patterns, that is why, it was the technique chosen to develop this application. As achieved results it's highlighted the ANN implementation which is capable of training and recognizing drawn patterns, as well as the stored data collected about its trainings, in an adventure game.

Keywords: Artificial Neural Networks, Image Recognition, Computational Games.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

O uso de Redes Neurais Artificiais (RNA) está cada vez mais em evidência nos mais diversos sistemas utilizados no dia a dia, o que demonstra o quão flexível e eficaz é esta técnica. Diferentes aplicações logicamente requerem diferentes características da RNA, sendo que, em muitos casos, essas características são principalmente rapidez e precisão. Como exemplo destes casos, há alguns tipos de jogos computacionais, onde o reconhecimento de padrões é algo essencial para a concepção, desenvolvimento e aproveitamento do jogo.

A relevância dos jogos vem se tornando cada vez mais forte com o passar do tempo, sendo que desde o início dos anos 2000, a indústria de jogos vem superando a indústria de outras formas de entretenimento, como música, DVDs e cinema (VIDEO, 2015).

A utilização de RNAs é eficaz por sua flexibilidade no desenvolvimento de aplicações. A tarefa de Reconhecimento de Padrões é algo que pode ser aplicado a diversas áreas do conhecimento por servir a um propósito geral, aos quais os humanos estão habituados e realizam sem muitas vezes nem perceber, como por exemplo, reconhecimentos de sons, imagens, cheiros, entre outros.

O reconhecimento de imagens é, fundamentalmente, um reconhecimento de padrões; com isso em mente, o objetivo do trabalho foi desenvolver o reconhecimento de imagens, na forma de desenhos, aplicado a um jogo computacional. O trabalho foi desenvolvido com base em pesquisar um modelo de rede adequado para o reconhecimento de imagens com eficiência (rapidez). Os resultados obtidos na utilização do modelo e características da RNA utilizada neste trabalho justificam o seu desenvolvimento, pois problemas com requisitos similares poderão utilizar a RNA proposta e desenvolvida na solução parcial, ou até total do problema.

Neste contexto, "O uso de RNAs é uma estratégia adequada para avaliar (aprender e reconhecer) as ações de jogadores em jogos computacionais" é o problema de pesquisa deste trabalho.

Os objetivos específicos compreenderam o estudo dos diferentes modelos de RNAs conhecidos, determinando qual é o mais adequado para reconhecimento de imagens com as características dessa aplicação; o desenvolvimento de uma RNA com desempenho satisfatório para reconhecimento de imagens com detalhes; e o desenvolvimento de um

sistema de entradas para um jogo computacional de plataforma e aventura que utilizou a RNA desenvolvida para, assim, validar o seu desempenho.

O trabalho está organizado da seguinte maneira: O capítulo 2 apresenta a fundamentação teórica dos principais assuntos envolvidos nesta pesquisa, Redes Neurais Artificiais e Jogos computacionais, além de trabalhos relacionados. O capítulo 3 é uma explicação da metodologia. O capítulo 4 descreve o desenvolvimento da RNA e do jogo e o capítulo 5 relata os resultados obtidos com a implementação do sistema.

2. FUNDAMENTAÇÃO TEÓRICA

Nos seguintes capítulos serão descritos os fundamentos estudados nas áreas de Redes Neurais Artificiais, Jogos Computacionais, um comparativo entre os trabalhos relacionados, finalizando com algumas considerações concluídas a partir destes assuntos.

2.1. REDES NEURAIS ARTIFICIAIS

As Redes Neurais Artificiais (RNAs) são uma técnica de inteligência artificial cuja característica é trabalhar com processamento de dados visando simular o comportamento do cérebro humano. Elas são definidas como um modelo computacional que compreende um conjunto de unidades de processamento densamente interconectadas, que possuem uma natureza adaptativa, que é resultado de uma característica muito importante: O aprendizado por exemplos. Este ponto substitui, muitas vezes, a programação nas RNAs, fazendo com que elas sejam um modelo muito interessante de se utilizar quando não se possui um conhecimento completo sobre o problema a ser resolvido (MOHAMAD, 1995, p.1). Para que sejam compreendidas, é indispensável a definição das partes de um neurônio, conceitos retirados diretamente da rede neural biológica (LODISH, BERK and ZIPURSKY, 2000):

- *Corpo Celular:* Contém o núcleo do neurônio e é responsável pela síntese de quase todas as proteínas e membranas utilizadas nele.

- *Dendritos:* São ramificações que se encontram no início do corpo celular, estes são responsáveis por receber os impulsos e conduzi-los ao corpo celular.

- *Axônio* (fibra nervosa): São as ramificações que se estendem do corpo celular em direção à próxima célula, sendo responsável por conduzir os impulsos em direção aos próximos neurônios.

- *Sinapse:* É a conexão entre o final de um axônio com os dendritos de uma próxima célula. Esta conexão é a parte que transmite efetivamente o impulso de um neurônio ao próximo fazendo uma conversão para um tipo diferente de sinal, que será utilizado pelo mesmo. Um detalhe importante aqui relacionado, é que as sinapses podem ser excitatórias ou inibitórias, que ativam ou inativam, respectivamente, o próximo neurônio da rede.

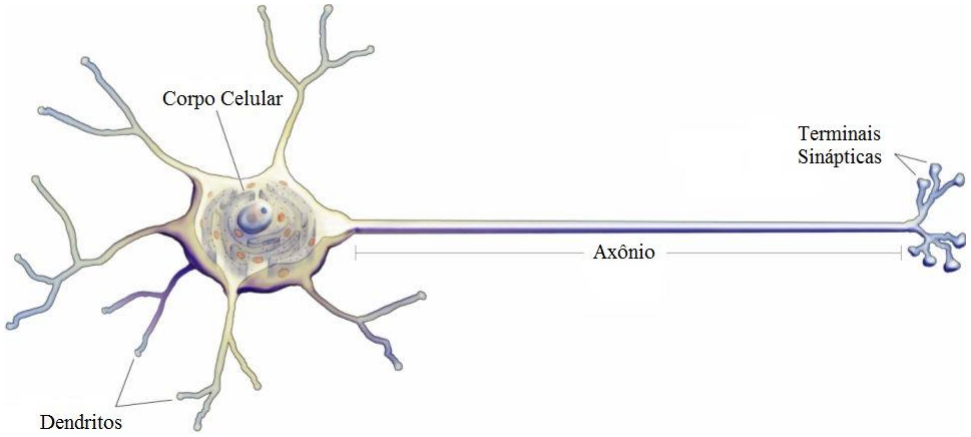

Figura 1– Neurônio Biológico: Adaptado de
http://bio3520.nicerweb.com/Locked/chap/ch03/3_11-neuron.jpg

Com base nessas informações básicas, é possível extrair os fundamentos para a criação do modelo de neurônio a ser utilizado em uma RNA. Assim, em uma RNA, o corpo celular refere-se aos elementos de processamento (EP); os axônios e dendritos referem-se às conexões; e as sinapses são as funções para determinar a ativação ou não dos próximos elementos de processamento.

Uma RNA tem como seu objetivo fundamental a simulação do funcionamento de uma rede neural natural, ou seja, aprender, a partir da implementação de um sistema envolvendo o modelo de neurônio artificial. Para que isso seja possível, padrões devem ser apresentados à rede para que seja realizada a etapa de treinamento. Uma vez que um padrão é inserido na rede, este deve ser convertido para um modelo que seja processado pela RNA.

Quando a informação chega a um EP, ocorre o processamento de funções de avaliação. Estas funções dependem fundamentalmente do modelo de RNA utilizado, tópico tratado nas seguintes seções. Após processada pelas funções de transferência e ativação, a informação é devolvida na forma de um número que vai indicar a ativação ou inibição do próximo EP (KUMAR, 2004).

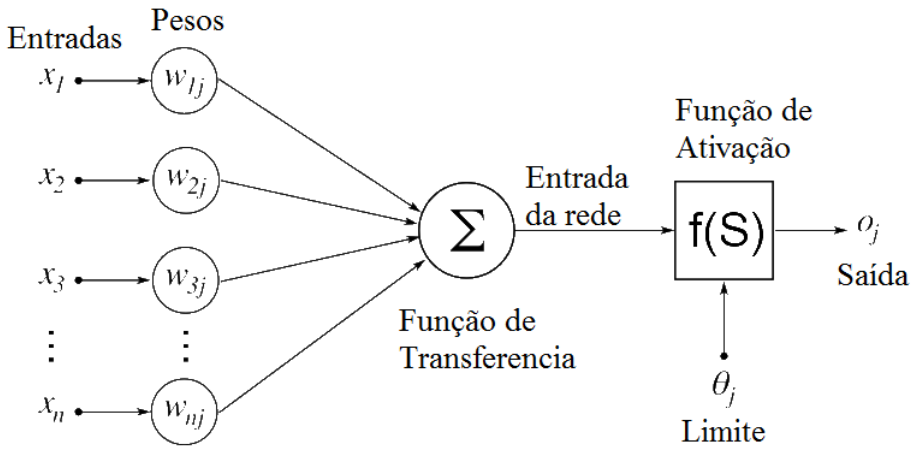

Figura 2 – Estrutura do Elemento de Processamento: Adaptado de https://commons.wikimedia.org/wiki/File:ArtificialNeuronModel_english.png

A avaliação dos valores pelas funções de transferência e ativação ocorre em cada EP e é repetido em todos eles até o final da RNA, que é na maioria dos casos implementada em camadas (conjuntos de EPs) amplamente conectadas. Isto é, a saída de todos os EPs de uma camada serve como informação de entrada para todos os EPs da camada seguinte. Como padronização, é utilizada a nomenclatura de Camada de Entrada e Camada de Saída para a primeira e última camada da RNA, respectivamente. Isso é exemplificado na figura 3.

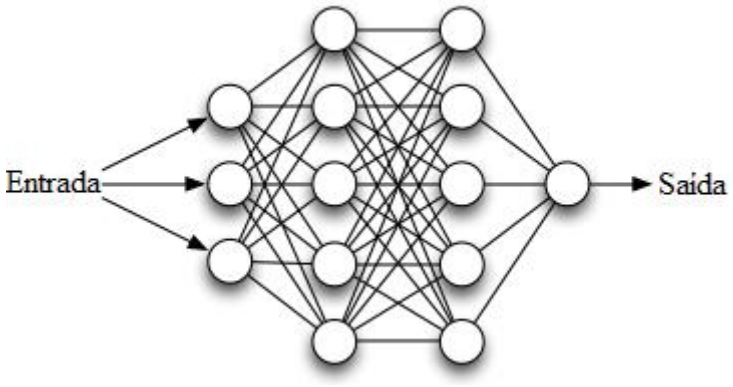

Figura 3 – Modelo de RNA amplamente conectada: Adaptado de http://static1.squarespace.com/static/51d342a0e4b0290bcc56387d/t/54497f24e4b0c5079a9976ef/1414102821758/

A RNA possui dois processos importantes: o Treinamento e o Reconhecimento. O treinamento consiste em apresentar uma quantidade volumosa de padrões para a RNA, para que se adapte aos possíveis cenários que podem vir a ser apresentados durante a sua execução. Existem dois estilos de treinamento, o supervisionado e o não supervisionado.

No treinamento supervisionado, a RNA recebe os padrões em pares, isto é, o padrão a ser reconhecido e o resultado esperado desse reconhecimento. Assim, se o resultado apresentado pela rede não é o recebido como parâmetro para treinamento, devem ser ajustados os pesos das conexões dos neurônios para que se obtenha o resultado desejado.

Em contrapartida, o não supervisionado não recebe a saída desejada para o padrão de entrada apresentado, ela deve ajustar os padrões em grupos semelhantes por si só. Este processo geralmente envolve processos de competição e cooperação entre os neurônios.

O reconhecimento é resultado do treinamento posto em prática. Para (RABUÑAL, 2005) "É a tarefa realizada por uma RNA treinada para responder quando um vetor apresentado é similar a um vetor aprendido. A rede 'reconhece' a entrada como um dos vetores originais, mesmo que a informação contenha ruído ou falta de dados".

2.1.1 MODELOS DE REDES NEURAIS ARTIFICIAIS

Ao longo dos anos de desenvolvimento e pesquisa sobre RNAs, diferentes formas de implementação surgiram com diferentes modelos de RNAs (DE WILDE, 1997) (FAUSETT, 1994) (SIMON, 1994). Os Modelos de RNAs são toda a arquitetura de como uma RNA é definida e como ela funciona. O estudo de diversos modelos de RNAs está fortemente ligado a este trabalho. Com base nos modelos existentes, alguns podem ser descartados por não se adequarem de forma satisfatória com o reconhecimento de imagens, assim como outros já indicam que devem receber uma atenção maior por melhor se adequarem com a implementação proposta. Alguns modelos de RNAs bastante estabelecidos são:

Perceptron: O modelo *Perceptron* não foi criado com a intenção de servir como cópia detalhada de um sistema nervoso. Ele é uma rede simples desenhado para permitir o estudo dos relacionamentos e organização de uma rede de neurônios, organização do ambiente e o desempenho da qual é capaz (PARKS, LEVINE, LONG, 1998). *Perceptron* é um exemplo de treinamento supervisionado, sendo que sempre com uma entrada já é conhecida a saída esperada.

Kohonen: Este modelo (também chamado de mapa auto-organizável) é um exemplo de treinamento não supervisionado. Segundo (PARKS, LEVINE, LONG, 1998) "Estes tipos de redes (auto-organizáveis) são capazes de modelar a função de distribuição probabilística dos

dados de entrada". Isto é, a função de avaliação sofre alterações à medida da execução do treinamento e, com isso, é feito um agrupamento de entradas semelhantes.

Backpropagation Net: Mais do que um modelo apenas, o *Backpropagation* pode ser utilizado como um algoritmo de aprendizado em diversos modelos. Desde sua publicação em 1986, o aprendizado pelo *backpropagation* se tornou o método de treinamento mais popular em RNAs. A razão para isso é a simplicidade e o poder relativo que o algoritmo oferece. Isso, porque, diferentemente dos aprendizados de *Perceptron* e *Widrow-Hoff,* ele pode ser aplicado em redes não-lineares com conectividade arbitrária (CHAUVIN, RUMELHART, 2013). O conceito do algoritmo é retro propagar os erros gerados durante a propagação para calcular a diferença entre os valores esperados e obtidos para, então, atualizar os pesos da rede e reiniciar o processamento.

Hopfield: Em 1982, John Hopfield introduziu um novo conceito de CAM (*Content-addressable memory*) onde os impulsos de entrada e saída são representados pela mesma coleção de neurônios (COUGHLIN, BARAN, 1995). Este modelo também utiliza treinamento supervisionado e não contém múltiplas camadas.

Bidirectional Associative Memory (BAM): Este modelo se assemelha ao modelo de Hopfield, porém, introduzindo o processamento da camada de entrada para a saída e da camada de saída para a entrada. Assim, a BAM reconhece pares de padrões nos dois sentidos: apresentando o padrão de entrada, reconhece o padrão de saída correspondente; apresentando o padrão de saída, reconhece o padrão de entrada correspondente (WU, 1993).

Adaptive Resonance Theory (ART): É um *framework* de RNAs bem estabelecido, desenvolvido no Centro para Sistemas Adaptativos da Universidade de Boston. É baseado num estudo profundo de modelos matemáticos que tornou possível a invenção de uma série de arquiteturas ART (SERRANO-GOTARREDONA, LINARES-BARRANCO, ANDREOU, 2012). Alguns modelos criados a partir deste conceito são: ART1 (GROSSBERG, 1987), ARTMAP (CARPENTER, GROSSBERG, REYNOLDS 1991), Fuzzy-ART (CARPENTER, GROSSBERG, REYNOLDS, 1991) e Fuzzy-ARTMAP (CARPENTER, 1992). Os modelos gerados podem ser tanto supervisionados quanto não-supervisionado.

Uma área que tem forte ligação com RNAs é a área de *Deep Learning*, que é uma solução que permite aprender pela experiência e entender o mundo como uma hierarquia de conceitos, que se inter-relacionam. Por aprender pela experiência, é dispensável a

necessidade de um humano para especificar todo o conhecimento necessário, e por possuir o conhecimento como uma hierarquia de conceitos, é possível para o sistema computacional construir conceitos complexos a partir de conceitos simples (GOODFELLOW, BENGIO, COURVILLE, 2016).

Neste contexto, as RNAs podem conter múltiplas camadas, onde cada camada é responsável por um dos conceitos, e por fim, juntando todos os conceitos simples, ou, camadas intermediárias, resultaria em um conceito complexo, a saída da RNA (NIELSEN, 2016). Por exemplo, em reconhecimento de imagens, é possível modelar as camadas intermediárias para reconhecer padrões específicos da imagem, como "A primeira camada pode reconhecer as bordas, os neurônios da segunda camada podem aprender a reconhecer padrões mais complexos, como triângulos ou retângulos, construídos pelas bordas" (NIELSEN, 2016).

2.2. JOGOS COMPUTACIONAIS

"Um jogo é um tipo de atividade, conduzida no contexto de uma realidade simulada, na qual o (s) participante (s) tenta (m) conquistar pelo menos um objetivo arbitrário não trivial atuando de acordo com regras" (ADAMS, 2010).

A seguir, uma breve descrição dos elementos considerados essenciais para os jogos, segundo o livro de (ADAMS, 2013):

- *Play*: O jogar é uma forma participativa de entretimento, diferente de livros, filmes e peças. A ideia deste ponto é o modo com que o jogador molda o jogo em sua própria experiência, sendo diferente e dinâmica para cada um.
- *Pretending*: "O fingir é o ato de criar uma realidade imaginária na mente...". Isto refere-se à capacidade do jogo de imergir o jogador em um mundo próprio, com seu próprio contexto e regras a serem seguidas, na maioria das vezes, longe das limitações do mundo real.
- *Goal*: O jogo precisa ter um objetivo ou objetivos. Mesmo jogos cujo foco é a criatividade possuem desafios, os de exercerem essa criatividade. Um ponto fortemente relacionado com o objetivo são as regras do jogo, pois elas normalmente caracterizam o objetivo central, e por consequência, a condição de vitória do jogo.

Os jogos ao longo dos anos foram sendo separados em tipos, mas o termo mais popular para definir tipos quando referente a jogos é gênero. Os gêneros podem se dividir em uma quantidade volumosa de subgêneros. Assim, a seguir apresenta-se uma lista com os principais gêneros de jogos eletrônicos, seguido de exemplos (ADAMS, 2013), e também, é importante ressaltar que os jogos em sua grande maioridade pertencem a mais de um gênero ao mesmo tempo.

- *Shooters:* Podendo ser tanto 2D como 3D, os jogos de tiro vêm se tornando incrivelmente populares nos últimos anos. Os jogos consistem em o jogador controlando um *avatar* tendo que desviar ou se proteger de inimigos que tentam acertá-lo com tiros. Exemplos são: Metal Slug; Battlefield; Call of Duty.
- *Platform:* Jogos de plataforma são tipicamente bidimensionais, onde o jogador controla um *avatar* que se move em um cenário verticalmente exagerado, tentando evitar obstáculos e batalhando com inimigos. Exemplos são: Super Mario Bros; Castlevania; Super Meat Boy.
- *Fighting:* Os jogos de luta demandam da habilidade física, reflexos e *timing* do jogador. Esses jogos simulam o combate corpo a corpo, geralmente utilizando de movimentos exagerados inspirados em artes marciais. Um subgênero comum de jogos de luta, é o que os jogadores podem se juntar e lutar contra um grande número de inimigos em conjunto, geralmente sendo chamado de *beat-em-ups* ou *brawlers*. Exemplos são: Street Fighter; Mortal Kombat; Castle Crashers.
- *Strategy:* Desafios estratégicos, táticos e, às vezes, lógicos, são os principais elementos dos jogos de estratégia. Geralmente, contém elementos de exploração e economia para deixar o jogo mais duradouro e variado. Exemplos são: Age of Empires; Command and Conquer: Red Alert; Civilization.
- *Role-Playing Games (RPG):* RPGs permitem ao jogador interagir com o ambiente a sua volta em uma variedade maior que dos outros gêneros e desempenhar um papel mais rico do que a maioria dos jogos proporciona. Uma grande característica dos RPGs é o senso de crescimento de personagem, onde muitas vezes ele começa como uma pessoa comum, e termina com grandes poderes adquiridos ao longo do jogo. Exemplos são: Final Fantasy; Pokemon; The Witcher.

- *Sports:* Jogos de esporte tem como objetivo simular um esporte real. Muitos destes jogos oferecem a administração como parte da experiência de jogo. Exemplos são: FIFA; NBA; Madden.
- *Vehicle Simulator:* Estes simuladores criam a experiência de pilotar um veículo, real ou imaginário. Quando a simulação é de um veículo real, o grande objetivo é simular realisticamente o comportamento do mesmo, levando em conta que o jogador com interesse neste gênero provavelmente já vai ter um conhecimento prévio sobre o veículo simulado. Exemplos são: Forza; Flight Simulator.
- *Construction and Simulation:* São jogos de simulação e administração. Eles oferecem aos jogadores a chance de construir coisas, como cidades, impérios e estruturas, operando dentro de um sistema de economia. O objetivo do jogador normalmente é não derrotar um inimigo, mas sim, criar algo dentro de um contexto em progresso. Exemplos são: FarmVille; Sim City; Roller Coaster Tycoon.
- *Adventure:* Um jogo de aventura se assemelha bastante com um jogo de RPG, entretanto, enquanto o RPG foca no crescimento do personagem em um sentido numérico, um jogo de aventura foca no crescimento dramático. Neste sentido, o protagonista deve ser bem desenvolvido para que o jogador de alguma forma se importe com os eventos que acontecem em torno deste personagem. Exemplos são: Uncharted; The Legend of Zelda; Heavy Rain.
- *Puzzle:* Nestes jogos, o objetivo principal é a resolução de quebra cabeças, sendo eles pequenos e separados ou se alinhando por um objetivo maior. Geralmente, os desafios seguem desafios relacionados de mesmo tema ou estilo. Exemplos são: The Witness; Portal; Cut the Rope.

2.3. TRABALHOS RELACIONADOS

Nesta seção, são apresentados alguns trabalhos com relação aos temas de Jogos ou Redes Neurais Artificiais.

No trabalho de (BOYAN, 1992), foi estudado um método de *Temporal Differences* (TD), uma técnica de aprendizado que dita o provável resultado de uma sequência a partir do tempo. Neste trabalho, o autor definiu uma RNA do modelo *Multilayer Perceptron* para analisar os padrões dos jogos de Gamão e Jogo da Velha, com seu algoritmo de Gamão competindo nas Olimpíadas de Jogos Computacionais, em Londres em 1992, onde terminou

em segundo lugar. O artigo não relatou com detalhes a arquitetura e parâmetros da rede utilizada.

O trabalho (LUBBERTS, MIIKKULAINEN, 2001) já tem uma abordagem diferente na utilização de RNAs. A proposta foi desenvolver uma rede para jogar o jogo Go (também conhecido como *Weiqi* ou *Baduk***)**, um jogo de origem chinesa que fez muito sucesso no Japão. O treinamento para a RNA se tornou um problema, já que não havia programas para jogar contra a RNA (para gerar o aprendizado) em um nível significativo. Assim, a solução encontrada foi a utilização de duas RNAs que competissem uma com a outra. Enquanto a rede principal, denominada de *hosts*, mantinha o foco em aprender novas estratégias para vencer o jogo, a segunda rede, denominada de *parasites*, tentava explorar as fraquezas da rede principal. A camada de entrada desta rede consiste em dois dados de entrada para cada intersecção do tabuleiro, um informando se existe uma peça preta ocupando esta parte do tabuleiro e um indicando o mesmo para uma peça branca. A camada de saída consiste em um dado para cada espaço do tabuleiro, com valores que podem variar de 0 até 1, indicando o quão bom é o movimento de adicionar uma peça neste espaço. Quando o número retornado pela rede para todos os espaços testados é inferior que 0,5 a rede passa a vez. A utilização desse sistema de co-evolução trouxe resultados muito melhores que os limitados pelos oponentes disponíveis, e o autor acredita que estes sistemas de cooperação provavelmente serão parte das futuras soluções mais robustas do mesmo tipo de problema.

O trabalho (KRIZHEVSKY, 2010) apresenta um conceito que é denominado de *Convolutional Networks* para classificação de imagens de alta resolução. O autor desenvolveu uma arquitetura de RNA que, essencialmente, possui dois processos separados realizando o processamento dos dados de entrada. A RNA possui cinco camadas co-evolutivas, três camadas amplamente conectadas e possui treinamento supervisionado. O número de neurônios varia de camada para camada, mas no total foram utilizados 650.000 neurônios. Durante o processo de reconhecimento, as camadas co-evolucionais processam suas entradas separadamente e, no final, os dois processos "comunicam-se" entre si pelas camadas amplamente conectadas, conforme a figura 4.

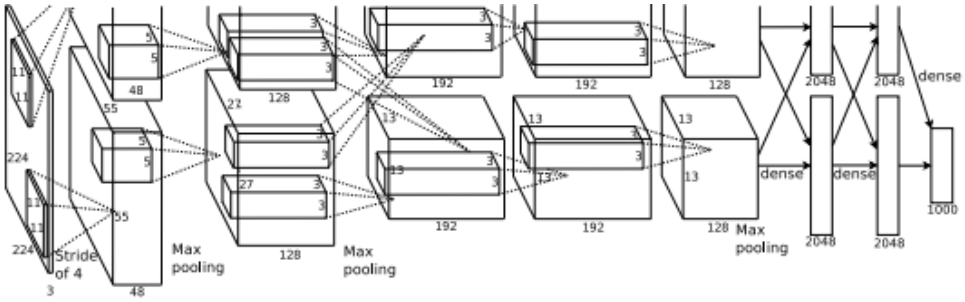

Figura 4 – Arquitetura da RNA de Krizhevsky: (KRIZHEVSKY, 2010)

Os resultados gerados por essa rede foram excelentes, visto que o trabalho foi submetido para a competição anual *ImageNet Large-Scale Visual Recognition Challenge* (ILSVRC), e conseguiu a melhor colocação em dois tipos de testes, o *top-1* e *top-5*. Sendo que o primeiro testa se a resposta da rede é a esperada, e o último testa se um dos 5 primeiros resultados da rede é o esperado.

A Tabela 1 apresenta um quadro comparativo dos trabalhos relacionados estudados e apresentados nesta seção.

Tabela 1 – Quadro comparativo

Autores	**Objetivos**	**Modelo de RNA utilizado**	**Resultados**
Boyan (2012)	Desenvolver uma RNA para analisar os padrões dos jogos de Gamão e Jogo da Velha	*Multilayer Perceptron*	O algoritmo de Gamão competiu nas Olimpíadas de Jogos Computacionais em Londres de 1992, conquistando segundo lugar
Lubberts e Miikurulainen (2001)	Desenvolver uma rede para jogar o jogo Go, utilizando duas RNAs que competissem uma com a outra para realizar o treinamento. Uma rede focava em descobrir novas estratégias para vencer o jogo enquanto a outra tentava explorar as fraquezas da primeira.	Não informado	A utilização desse sistema de co-evolução trouxe resultados muito melhores que os limitados pelos oponentes disponíveis, e o autor acredita que estes sistemas de cooperação provavelmente serão parte das futuras soluções mais robustas do mesmo tipo de problema.
Krizhevsky (2010)	Desenvolver uma arquitetura de RNA que, essencialmente, possui dois processos separados realizando o processamento dos dados de entrada. Durante o processo de reconhecimento, as camadas co-evolucionais processam suas entradas separadamente e, no final, os dois processos "comunicam-se" entre si pelas camadas amplamente conectadas.	*Convolutional Networks*	Os resultados gerados por essa rede foram excelentes, visto que o trabalho foi submetido para a competição anual *ImageNet Large-Scale Visual Recognition Challenge* (ILSVRC), e conseguiu a melhor colocação em dois tipos de testes, o top-1 e top-5. Sendo que o primeiro testa se a resposta da rede é a esperada, e o último testa se um dos 5 primeiros resultados da rede é o esperado.

A partir dos trabalhos estudados, pode-se observar que RNAs são uma área em constante evolução, sendo que já em 1992 haviam implementações feitas especificamente para jogos computacionais. Além disso, a implementação da RNA conquistou o segundo lugar nas Olimpíadas de Jogos Computacionais no trabalho de (BOYAN, 2012) e a melhor colocação no ILSVRC segundo o trabalho de (KRIZHEVSKY, 2010). O que comprova que não apenas é possível utilizar RNAs como método para resolução de problemas, mas que também é uma forma muito eficiente para isso. Também, existem várias maneiras de utilizar as RNAs para chegar aos fins desejados, como demonstra o trabalho de (LUBBERTS, MIIKURULAINEN, 2001), onde o processo de treinamento não precisou ser completamente modelado por uma pessoa, mas sim realizado pelas próprias RNAs desenvolvidas no trabalho.

Também foram encontrados diversos trabalhos que utilizaram RNAs para reconhecer imagens, porém o foco destes estava no pré-processamento das características da imagem para deixá-las aptas a ser processada pela RNA. Alguns exemplos são: (PERELMUTER, 2006) com Redes Neurais Aplicadas ao Reconhecimento de Imagens Bi-dimensionais; (MARCOMINI, 2013) com Aplicação de modelos de Redes Neurais Artificiais na segmentação e classificação de nódulos em imagens de ultrassonografia de mama; (KHATCHATOURIAN, PADILHA, 2008) com Reconhecimento de variedades de soja por meio do processamento de imagens digitais usando Redes Neurais Artificiais; (GIMENEZ, 2011) com Identificação de bovinos através de reconhecimento de padrões do espelho nasal utilizando Redes Neurais Artificiais.

3. METODOLOGIA

A pesquisa foi de natureza qualitativa, cujos objetivos são exploratórios, por envolver a explicação de conceitos relacionados às RNAs, assim como descritivos por englobar uma coleta e análise de dados relacionados a essas RNAs. Para tal, foram feitas pesquisas em artigos, livros e periódicos com o objetivo de definir as características da RNA a ser utilizada na aplicação.

Algumas das características específicas analisadas nas redes foram: modelo utilizado, número de neurônios na camada de entrada, número de neurônios na camada intermediária, funções de processamento e parâmetros nos algoritmos.

A Figura 5 apresenta o fluxo de desenvolvimento da pesquisa.

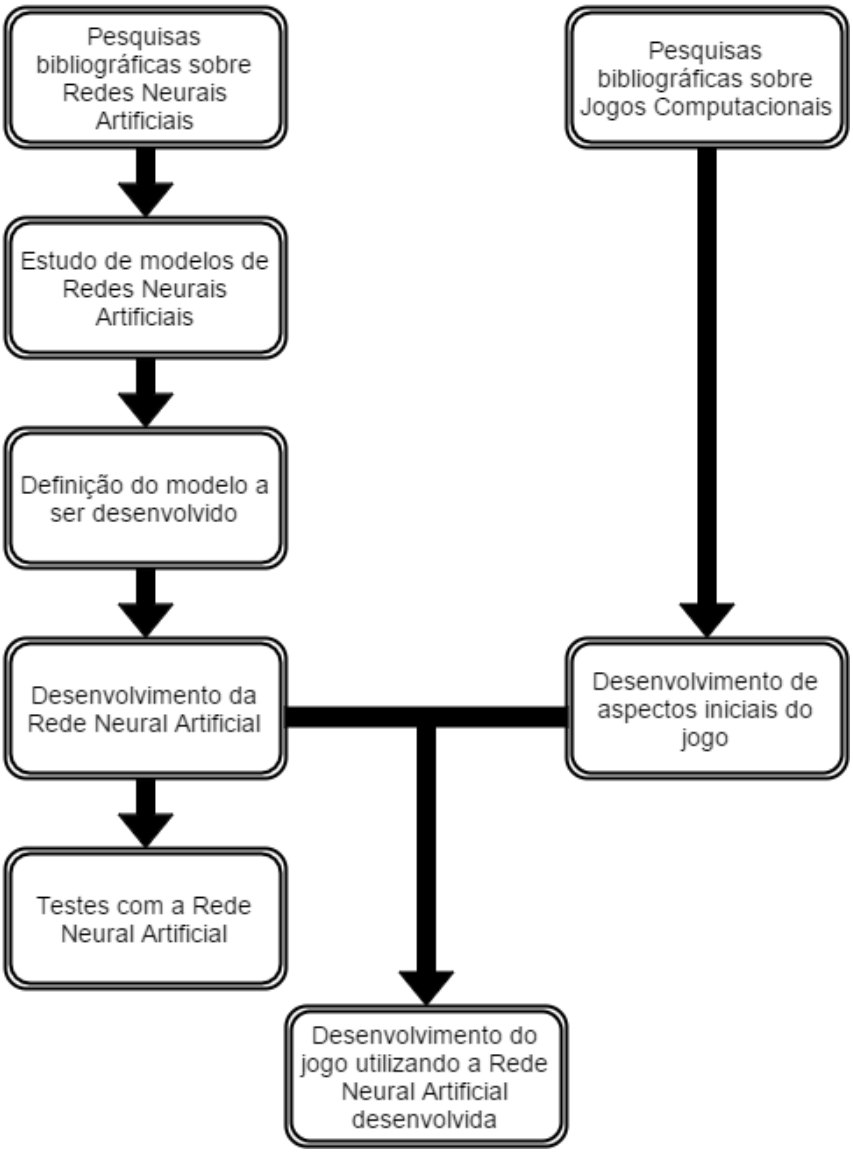

Figura 5 – Fluxo do desenvolvimento do trabalho

Inicialmente, foi realizada uma pesquisa bibliográfica sobre os temas principais do trabalho, Redes Neurais Artificiais e Jogos computacionais. Durante a pesquisa, o modelo *Multilayer Percepron* foi bastante utilizado em aplicações (BOYAN, 2012) (BOW, 2002) (ALPAYDIN, 2009) (KOLLIAS, 2006) com foco em jogos, e outras não. Sendo assim, este foi o modelo de RNA escolhido para o desenvolvimento do trabalho.

Assim deu-se início ao desenvolvimento da RNA, criando as estruturas de dados essenciais para o armazenamento dos neurônios e funções de inicialização da rede, estas etapas estão desenvolvidas em detalhes no capítulo 4. Juntamente com a RNA, também foi dado início ao desenvolvimento do sistema de entradas do jogo, especificamente, foram desenvolvidos o quadro e a interação com o mouse, que são o mecanismo de entrada utilizado pelo jogador no jogo.

Com o desenvolvimento da RNA evoluindo, constantemente foram feitos testes de desempenho e avaliação para ver se ela estava se comportando de acordo com o esperado. Quando o desenvolvimento dela foi concluído, o desenvolvimento de aspectos de jogos tomou prioridade durante o resto do tempo hábil do trabalho. E por fim, a RNA em sua versão final foi submetida a vários testes de força bruta, onde seus parâmetros foram variados em laços de repetição, seguido do treinamento para cada uma das combinações atingidas.

O jogo desenvolvido consta em uma fase de demonstração da aplicação da RNA no reconhecimento de padrões desenhados. Inicialmente, o jogador deve escolher os padrões de imagem que deseja utilizar durante a fase, isso irá ativar o treinamento da RNA para os padrões escolhidos. Depois disso o jogador pode iniciar a fase de teste para jogar, caso ele não tenha escolhido os padrões, os 3 padrões iniciais serão automaticamente escolhidos e treinados.

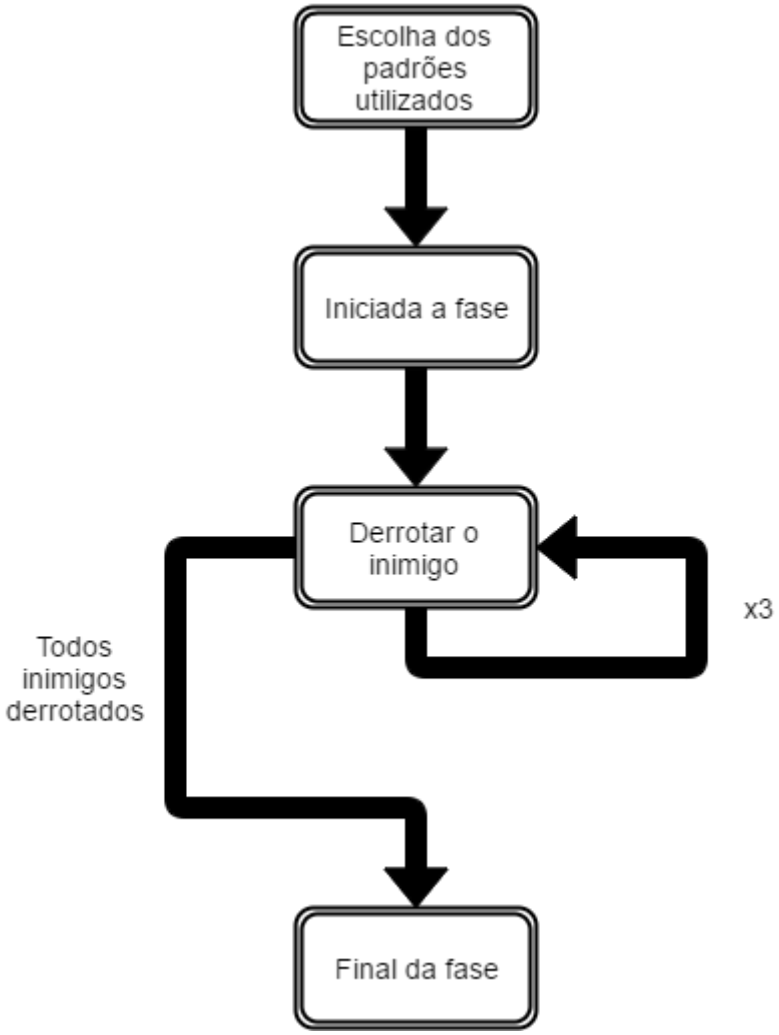

Figura 6 – Fluxo do jogo

Ao começar a fase, o jogador vai se encontrar com um inimigo, onde os dois irão lutar. Os padrões escolhidos anteriormente determinam quais ações o jogador vai ter contra o oponente, sendo que ele deve desenhar o padrão escolhido numa região reservada da tela em um tempo pré-determinado para executar esta ação. Quando o jogador derrotar 3 inimigos, ou for derrotado, ele vai concluir a fase e ser levado de volta para o menu principal do sistema.

3.1. UNITY 3D

A *engine* de jogos utilizada para o desenvolvimento do trabalho foi a Unity3D. Esta *engine* se tornou muito popular nos últimos anos devido ao fato de possuir uma interface que permite aos usuários desenvolverem jogos simples possuindo pouco conhecimento em programação, além de não possuir custos para pequenos desenvolvedores utilizarem a ferramenta comercialmente (UNITY3D, 2016).

A ferramenta trata de maneira automática, porém personalizável, vários aspectos que são essenciais no desenvolvimento de jogos, como por exemplo: gravidade, colisões, iluminação e câmeras. De maneira geral, os projetos são tratados da seguinte maneira: Cada segmento do jogo é dividido em cenas, e essas cenas contêm os objetos que realizam todas as interações com o usuário. Um objeto de jogo pertence a uma classe mestre denominada *GameObject*, e nele podem ser agregados múltiplas outras classes de acordo com a necessidade, como por exemplo, uma emissão de luz, ou emissão de áudio.

Juntamente com estes componentes já desenvolvidos, também é possível a ligação de *scripts* a cada *GameObject*, que pode manipular todos os valores deste objeto, e acessar suas funções e propriedades. Desta forma, é descrito o comportamento que cada objeto deve ter de acordo com a necessidade de cada jogo. São aceitos *scripts* desenvolvidos nas linguagens de programação *C#* e *Javascript*. A RNA e o jogo foram desenvolvidos dentro do ambiente da *Unity*.

4. SISTEMA DESENVOLVIDO

A ideia principal deste trabalho foi o desenvolvimento de um sistema de entradas para ser utilizada em um jogo cujas imagens são reconhecidas pelo uso de uma RNA *Perceptron* Multicamadas, com o algoritmo de aprendizagem *Backpropagation*. O jogo se enquadra nos seguintes gêneros citados na seção do capítulo 2: plataforma e aventura. No decorrer do jogo, vão ocorrer combates, onde o jogador deve resolvê-los desenhando, em uma área específica da tela, padrões que representam os movimentos possíveis do personagem naquele momento. Utilizando o reconhecimento de imagens, o sistema irá reconhecer a perfeição com que o jogador desenhou um padrão, e executará o movimento desejado com eficiência proporcional à porcentagem de acerto do desenho.

Ao iniciar a batalha, o jogador terá acesso visual a padrões de desenho previamente escolhidos, junto com informações relacionadas com a batalha (pontos de vida, temporizadores, entre outros), e uma área da tela reservada para desenhar. Nesta área, o jogador deve desenhar um dos padrões indicados com um limite de tempo. Assim que esse limite terminar, a RNA vai reconhecer o desenho, e caso for reconhecido, o personagem vai executar a ação correspondente ao padrão desenhado.

O quadro onde serão feitos os desenhos vai ser traduzido em uma matriz binária, onde os espaços pretos (pintados) representam o estado 1 e os em branco representam o estado 0. Esta coleção de dados vai ser a entrada da RNA, que vai devolver a indicação de qual padrão o jogador desenhou.

4.1 REDE NEURAL ARTIFICIAL DESENVOLVIDA

A RNA desenvolvida foi utilizando o modelo *Perceptron Multilayer* com o algoritmo de aprendizado *Backpropagation*. Os parâmetros de entrada são dinâmicos, e podem ser definidos a qualquer momento pela interface da *Unity*, o que se mostrou muito útil na realização de testes. Depois de testados diversos valores, os parâmetros escolhidos foram:

- 900 Neurônios de entrada, que correspondem a uma matriz de 30x30 onde é realizado o desenho do padrão para o reconhecimento.
- 4 Neurônios de saída, gerando um total de 16 padrões de saída distintos.

Quanto à estrutura dos neurônios, foi desenvolvida uma lista única contendo todos os neurônios da rede e uma lista contendo as conexões dos neurônios (Figura 7). As etapas da

rede foram divididas em funções separadas para melhor organização do código, sendo elas apresentadas na Figura 8.

```
public class Neurons
{
    public int index;
    public float neuronValue;
    public float neuronError;
    public List<neuronOutput> outputList= new List<neuronOutput>();
}

public class neuronOutput
{
    public int outNeuron;
    public float outWeight;
}
```

Figura 7 – Estrutura de dados da RNA

```
//Cria lista que vai conter todos neuronios
List<Neurons> neurons = new List<Neurons>();

//Inicializa os valores da rede
initializeNet(neurons);

//Busca padroes de entrada para o treinamento
getTrainingList(entriesList);

//Busca as saidas para o treinamento
getOutputList(outputList);

while (endTraining == false)
{

    //Se chegou ao fim das lista de entrada e saida, volta para o inicio
    if (i == entriesList.Count)
    {
        i =0;
        finalErrorRate = 0;
    }


    //Seta proximo padrao de entrada para o inicio da RNA
    setNextEntry(entriesList[i], neurons);

    //Calcula os valores dos neuronios
    setNetValues(neurons);

    //Verifica se a entrada esta com a saida correspondente, e ajusta os pesos das ligacoes caso necessario
    calculateError(entriesList[i], outputList[i], neurons);

    i++;
    controle++;

    // Se o erro quadratico for menor ou igual ao erro aceitavel e estiver no ultimo elemento da lista de entradas
    if (finalErrorRate <= tolerableError && i+1 == entriesList.Count)
    {
        Debug.Log("aprendeu!!!");
        endTraining = true;
    }
    // Controla o numero total de iteracoes possivel
    if (controle == 10000)
    {
        Debug.Log("caiu no break");
        endTraining = true;
    }
}
```

Figura 8 – Divisão de funções da RNA

Durante o desenvolvimento da rede foi adotado o padrão de que, em todo ponto onde é necessário acessar um elemento da lista de neurônios, o índice deste elemento é calculado através de funções matemáticas. Isso foi feito para evitar comandos de localização (*Find, FindIndex*, outros) com a intenção de tornar o acesso aos elementos mais rápido.

Na figura 9 está apresentado um exemplo da inicialização dos índices dos elementos. Observa-se que todos os comandos de repetição "*for*" são baseados nas variáveis de números de neurônios e números de camadas, assim, independentemente de quais valores forem utilizados para a criação da rede, a função será executada corretamente e sem comandos de localização.

```
public void initializeNet(List<Neurons> neurons)
{
    // Inicializa os neuronios da camada de entrada
    for (int i = 0; i < entryPatter; i++)
    {
        neurons.Add(new Neurons());
        neurons[i].index = i;
    }

    //Inicializa os neuronios das camadas intermediarias
    for (int i = 1; i <= hiddenLayers; i++)
    {
        for (int x = 0; x < hiddenNeurons; x++)
        {
            neurons.Add(new Neurons());
            neurons[neurons.Count - 1].index = neurons.Count - 1;
        }
    }

    // Inicializa os neuronios da camada de saida
    for (int i = 0; i < outputNeurons; i++)
    {
        neurons.Add(new Neurons());
        neurons[neurons.Count - 1].index = neurons.Count - 1;
    }

    //Inicializa os pesos dos neuronios da camada de entrada
    for (int i = 0; i < entryPatter; i++)
    {
        for (int x = 0; x < hiddenNeurons; x++)
        {
            neuronOutput auxNeuronOut = new neuronOutput();
            auxNeuronOut.outNeuron = entryPatter + x;
            auxNeuronOut.outWeight = Random.Range(-0.5f, 0.5f);
            neurons[i].outputList.Add(auxNeuronOut);
        }
    }
    //Inicializa os pesos dos neuronios das camadas intermediarias
    for (int i = 1; i < hiddenLayers; i++)
    {
        for (int x = entryPatter + ((i - 1) * hiddenNeurons); x < entryPatter + (i * hiddenNeurons); x++)
        {
            for (int y = 0; y < hiddenNeurons; y++)
            {
                neuronOutput auxNeuronOut = new neuronOutput();
                auxNeuronOut.outNeuron = entryPatter + ((i - 1) * hiddenNeurons) + y + hiddenNeurons;
                auxNeuronOut.outWeight = Random.Range(-0.5f, 0.5f);
                neurons[x].outputList.Add(auxNeuronOut);
            }
        }
    }

    //Inicializa os pesos da ultima camada intermediaria
    for (int i = entryPatter + hiddenNeurons * (hiddenLayers - 1); i < entryPatter + hiddenNeurons * hiddenLayers; i++)
    {
        for (int x = 0; x < outputNeurons; x++)
        {
            neuronOutput auxNeuronOut = new neuronOutput();
            auxNeuronOut.outNeuron = (entryPatter + hiddenNeurons * hiddenLayers) + x;
            auxNeuronOut.outWeight = Random.Range(-0.5f, 0.5f);
            neurons[i].outputList.Add(auxNeuronOut);
        }
    }
}
```

Figura 9 – Função initializeNet implementada

O fluxo do sistema ocorre nas seguintes etapas:

- Desenvolvida uma lista da estrutura "*Neurons*", onde são armazenados os elementos da rede.
- Inicializados os valores de índice dos neurônios e atribuído valor aleatório (entre -0,5 e 0,5) para os pesos.
- Carregados padrões de entrada para o treinamento.
- Carregados padrões de saída para o treinamento.
- Enquanto o erro quadrático for maior que o limite estabelecido ou não atingiu número de iterações definido.
    - Testa se o laço passou da última entrada apresentada, e reseta o erro final e o índice do laço em caso positivo.
    - Atribui o padrão de entrada atual para os neurônios de entrada da rede.
    - Calcula todos os valores dos neurônios da rede.
    - Acumula o erro da camada de saída
    - Faz o reajuste de pesos da rede

4.1.1. CÁLCULO DOS VALORES DOS NEURÔNIOS

O cálculo para definir os valores dos neurônios envolve duas funções, a função de soma e a função de transferência. A função de soma é dada pela seguinte fórmula:

$$\sum_{i=1}^{n} X_i \cdot W_{ij}$$

Sendo assim, é obtido o somatório dos valores dos neurônios de entrada (X) multiplicados pelo valor da conexão - peso (W - *weigth*), de todos os neurônios que se conectam com o neurônio alvo. Este valor obtido é então utilizado na função de transferência:

$$\frac{1}{1 + \exp(-Y)}$$

É a exponencial logarítmica do valor resultante da função de soma.

4.1.2. CÁLCULO DOS PESOS DOS NEURÔNIOS

Após o cálculo dos valores dos neurônios, é que o algoritmo *Backpropagation* entra em ação. Existe um valor de erro atribuído a cada neurônio da rede, e é utilizado para atualizar os pesos. Inicialmente, é definido esse valor de erro obtendo a diferença entre o valor desejado (X_d) e o valor obtido (X_o) e multiplicando pela diferença de 1 com o valor obtido elevado ao quadrado.

$$(X_d - X_o) \cdot (1 - X_o^2)$$

Depois de definido valor de erro, buscam-se os valores de taxa de aprendizado, valor do peso anterior e valor do neurônio anterior. Com isso, é possível calcular o peso novo da seguinte forma:

$$W_a - (ETA \cdot E \cdot X_a)$$

Multiplicando a taxa de aprendizado (ETA) com o erro do neurônio (E) e com o valor do neurônio anterior (X_a). E então esse valor obtido é somado ao valor do peso anterior (W_a), obtendo assim, o novo valor de peso.

4.1.3. CÁLCULO DO ERRO QUADRÁTICO

Para definir o erro quadrático, que é utilizado para verificar se a rede já chegou a um estado aceitável de aprendizado, utilizou-se a seguinte fórmula:

$$\frac{\sum(X_d - X_o^2)}{2}$$

É calculada a diferença entre o valor desejado (X_d) e o valor obtido (X_o), e então valor é elevado ao quadrado. Feito o somatório do resultado deste cálculo aplicado em cada um dos neurônios de saída, é então dividido por 2.

Este processo deve ser feito tantas vezes quanto há padrões distintos de entradas para serem apresentados, uma vez que, quando todos os padrões forem apresentados, é feito mais uma vez um somatório de todos os erros calculados para então comparar com a taxa de erro aceitável. Quando esse valor de erro final for menor que a taxa de erro aceitável, é considerado que a rede está treinada.

O reconhecimento da rede é simplesmente o cálculo de pesos dos neurônios, sem a atualização de pesos, então, quando é necessário que seja feito um reconhecimento, é apenas

atribuído aos neurônios de entrada o padrão a ser reconhecido e invocado o método responsável por calcular os pesos da rede, *setNetValues*. Calculados os valores dos neurônios, a camada de saída determina qual o padrão reconhecido, mas estes valores sempre serão entre 0 e 1, não sendo exatos. Sendo assim, foi considerado que, quando o valor é maior ou igual a 0.5, este é tratado como 1, e caso contrário, ele será 0.

4.2 CARACTERÍSTICAS DO JOGO

O jogo desenvolvido consiste em uma *demo* (Demonstração) de como poderia ser desenvolvido um jogo completo e com todos os recursos. O jogo é denominado *Paresy (Patterns Recognition System)*. Neste trabalho, as imagens utilizadas foram buscadas na *internet*, desenvolvidas pelo autor ou por terceiros, por não possuir fins lucrativos. A *demo* consiste em uma fase contendo batalhas ao longo do caminho utilizando os padrões de desenho disponíveis no sistema. Ao iniciar o jogo, o jogador será apresentado à cena de título, onde poderá escolher entre jogar uma fase de testes, escolher os padrões para a próxima aventura, rodar o módulo de testes da RNA ou sair.

A Figura 10 apresenta a tela inicial apresentada no jogo.

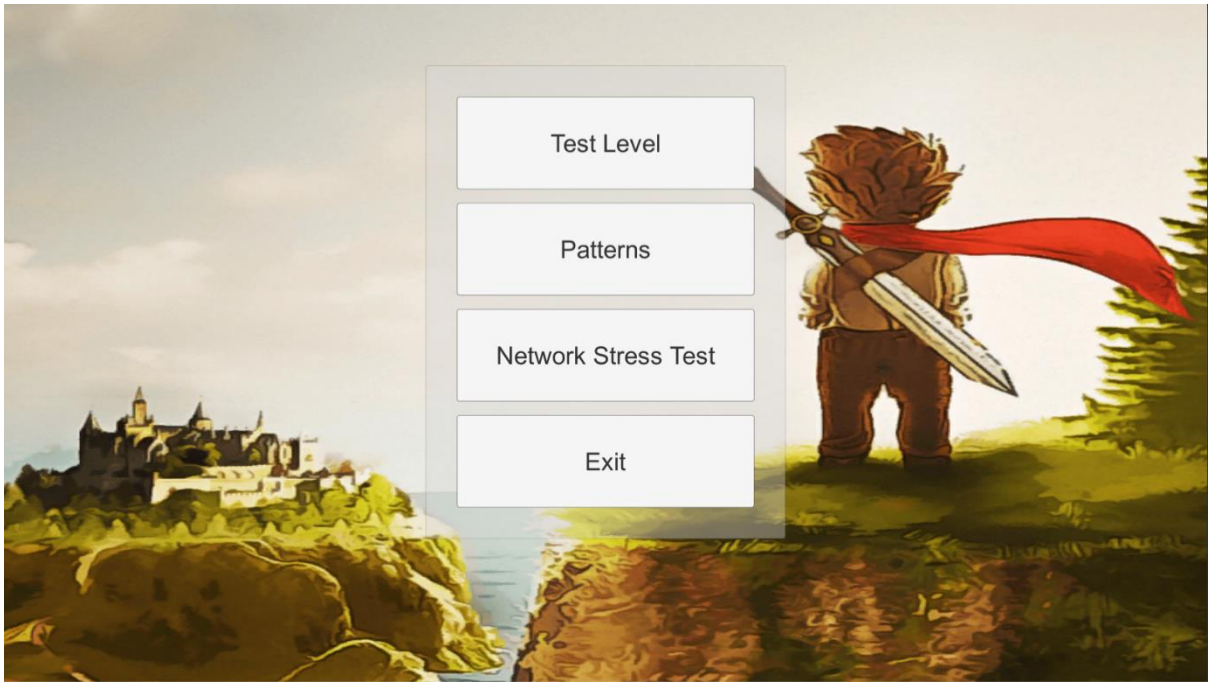

Figura 10 - Tela de título

O jogo estará enquadrado em dois grandes sistemas, um para as fases do jogo e outro para gerenciamento e decisão da próxima ação da jornada. O sistema responsável pelas fases será linear, o personagem se moverá automaticamente em um cenário bidimensional, bastante comum em jogos de plataforma, até encontrar inimigos ou pontos críticos para a continuação da história. Quando houver um encontro com inimigos, entrará em contexto o sistema de batalha do jogo.

A Figura 11 apresenta o fluxograma seguido durando uma batalha e a parte do jogo onde a RNA é utilizada.

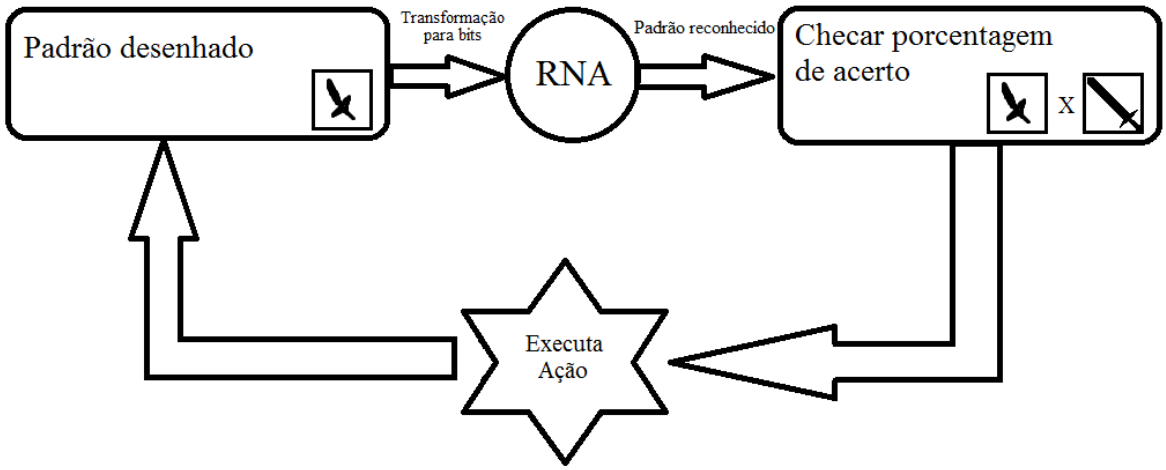

Figura 11 – Fluxograma de uma Batalha

Quando o tempo para desenhar termina, o padrão da tela é enviado para a RNA para determinar qual padrão foi desenhado. Depois disso o jogo compara todos os elementos do desenho com o padrão encontrado, sendo que, como existem 900 elementos num total, se o desenho contiver, por exemplo, 450 acertos, isso resulta em 50% do desenho correto. Para determinar o valor final das ações, foram utilizadas as seguintes regras:

- Porcentagem de acerto menor que 30%, valor final = 0.
- Porcentagem de acerto entre 31% e 70%, valor final = valor da ação mais valor da ação multiplicado pela porcentagem de acerto / 100.
- Porcentagem de acerto entre 71% e 100%, valor final = valor da ação mais valor da ação multiplicado pela porcentagem de acerto / 50.

O sistema responsável pelo gerenciamento entra quando o jogo se encontra fora das fases. Nele, o jogador vai administrar os padrões de desenho. Estes elementos vão prover diferentes vantagens para o personagem, reduzindo o nível de dificuldade das batalhas. Assim que satisfeito com o gerenciamento, o jogador deve então selecionar uma fase para jogar. A Figura 12 apresenta os padrões de desenho disponíveis para o jogador escolher antes de começar uma nova aventura, juntamente com o efeito de cada um.

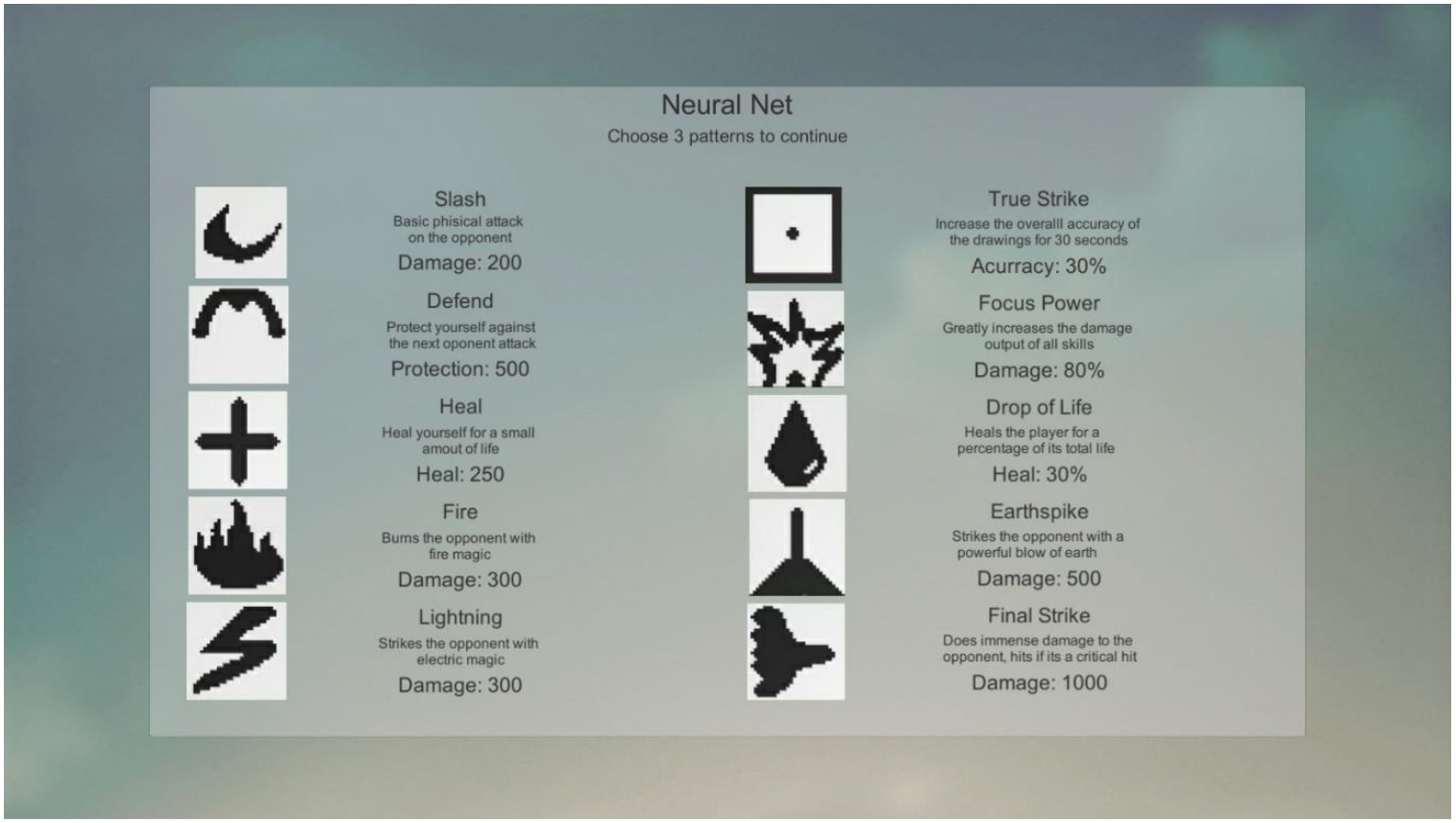

Figura 12 - Tela de seleção de padrões de desenho

Os conflitos são realizados da seguinte maneira: Tanto o jogador quanto o oponente possuem um valor de pontos de vida, quando este valor é zerado, seu respectivo dono é derrotado. Existem também temporizadores tanto para o jogador quanto para o oponente, todos estes dados estão sempre visíveis na tela. Quando o temporizador do oponente chega a seu fim, ele realiza um ataque, removendo uma parcela dos pontos de vida do jogador. Quanto ao jogador, isso também se aplica, porém, ele tem diversas ações as quais ele pode escolher (Figura 12), sendo umas mais eficientes que outras em determinadas situações do combate.

A Figura 13 apresenta um exemplo do meio de um combate entre o jogador e um inimigo.

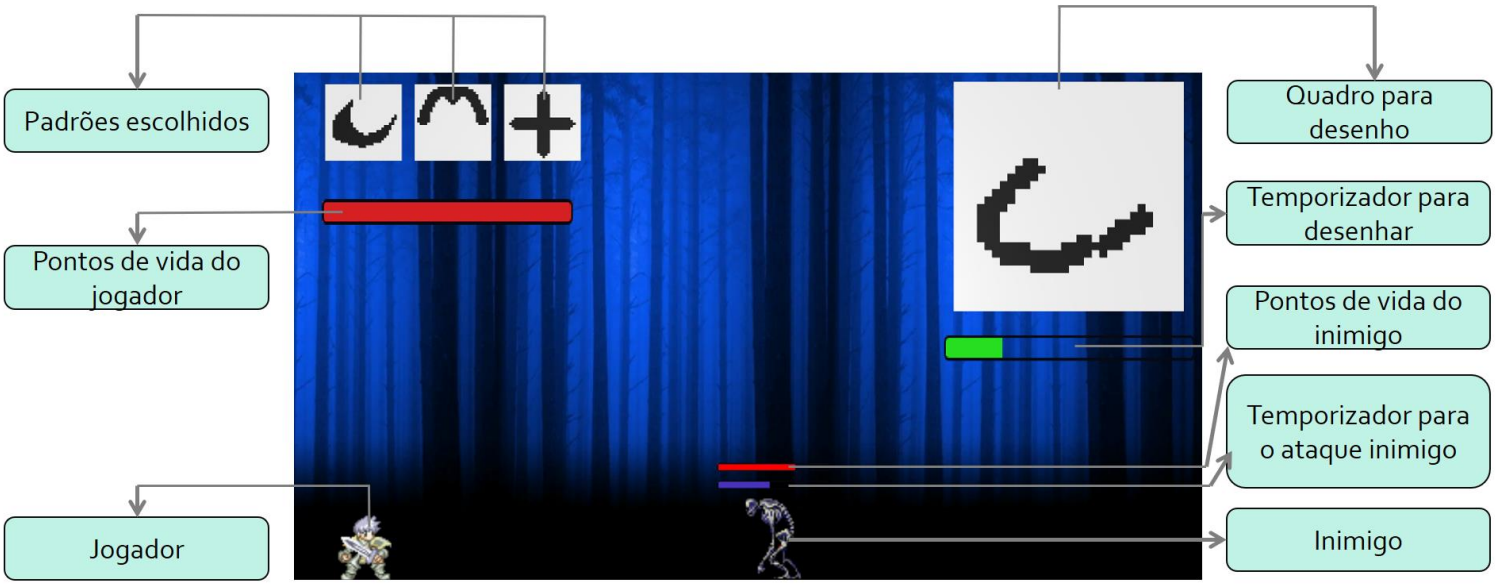

Figura 13 - Exemplo de combate

Na fase de testes desenvolvida, o jogador deve derrotar três inimigos, que ficam mais forte ao longo do progresso pela fase, para ganhar o jogo.

5. RESULTADOS OBTIDOS

Foi desenvolvido um módulo com a RNA eu seu estado final com a finalidade de fazer vários testes repetidos para avaliar seu desempenho. O funcionamento deste módulo se dá em variar os diferentes parâmetros que são dinâmicos na rede individualmente e mantendo os outros parâmetros com valores padrão, testando assim a alteração no treinamento relativa a cada um deles separadamente. Os parâmetros são:

- Número de entradas: Número de padrões de entrada apresentados à rede, cada entrada possui 900 bits.
- Neurônios intermediários: Quantidade de neurônios na camada intermediária da rede, o valor padrão é de 10 neurônios intermediários.
- Taxa de Aprendizado (ETA): Taxa de aprendizado utilizado nos cálculos de atualização de pesos do *Backpropagation*, seu valor padrão é de 0,3.
- Taxa de Erro aceitável: Valor de erro o qual foi comparado com o resultado do cálculo do erro quadrático para definir se a rede havia se estabilizado, o valor padrão é de 0,005.

Também, foi definido um número de iterações limite para a execução da rede de 10000 ciclos, isso foi feito para evitar a rede de se prender em um laço infinito de atualização de pesos e nunca chegar na condição de erro aceitável. Por fim, também foi medido o tempo, em minutos e segundos, que cada treinamento levou para ser completado. Na tabela 2 estão demonstrados os treinamentos feitos com os valores padrão para todos os números de entradas.

Tabela 2 - Treinamento variando o número de entradas

Número de Entradas	Neurônios Intermediários	ETA	Número de Iterações	Erro Aceitável	Tempo
3	10	0.3	323	0.005	00:12
4	10	0.3	567	0.005	00:38
5	10	0.3	2009	0.005	02:11
6	10	0.3	4001	0.005	04:11
7	10	0.3	9729	0.005	09:44
8	10	0.3	10000	0.005	09:46
9	10	0.3	10000	0.005	10:17
10	10	0.3	10000	0.005	09:50

A partir dos dados da tabela 2 pode-se concluir que o treinamento é exponencialmente mais lento quando se aumenta o número de entradas. Sendo que, já com 8 entradas o treinamento já ultrapassa o número limite de iterações delimitado. Foi com base nestes resultados que a decisão de realizar o treinamento múltiplas vezes com apenas 3 entradas foi tomada. A seguir seguem os resultados dos treinamentos variando os demais parâmetros da rede. São apresentadas as variações de neurônios intermediários, ETA e erro aceitável, respectivamente nas tabelas 3, 4 e 5.

Tabela 3 - Treinamento variando os neurônios intermediários

Número de Entradas	Neurônios Intermediários	ETA	Número de Iterações	Erro Aceitável	Tempo
3	10	0.3	269	0.005	00:18
3	20	0.3	101	0.005	00:14
3	30	0.3	56	0.005	00:11
3	40	0.3	38	0.005	00:10
3	50	0.3	20	0.005	00:07
4	10	0.3	567	0.005	00:38
4	20	0.3	223	0.005	00:30
4	30	0.3	251	0.005	00:51
4	40	0.3	323	0.005	01:29
4	50	0.3	235	0.005	01:21
5	10	0.3	2009	0.005	02:11
5	20	0.3	1079	0.005	02:24
5	30	0.3	1399	0.005	04:42
5	40	0.3	819	0.005	03:43
5	50	0.3	3304	0.005	18:39

Tabela 4 - Treinamento variando o ETA

Número de Entradas	Neurônios Intermediários	ETA	Número de Iterações	Erro Aceitável	Tempo
3	10	0.1	788	0.005	00:53
3	10	0.2	365	0.005	00:24
3	10	0.3	323	0.005	00:22
3	10	0.4	215	0.005	00:14
3	10	0.5	134	0.005	00:09
4	10	0.1	3283	0.005	03:40
4	10	0.2	971	0.005	01:05
4	10	0.3	435	0.005	00:29
4	10	0.4	507	0.005	00:34
4	10	0.5	395	0.005	00:26
5	10	0.1	6284	0.005	06:57
5	10	0.2	3729	0.005	04:05
5	10	0.3	5199	0.005	05:34
5	10	0.4	1484	0.005	01:36
5	10	0.5	9789	0.005	10:10

Tabela 5 - Treinamento variando o erro aceitável

Número de Entradas	Neurônios Intermediários	ETA	Número de Iterações	Erro Aceitável	Tempo
3	10	0.3	1043	0.001	01:11
3	10	0.3	338	0.003	00:22
3	10	0.3	212	0.005	00:14
3	10	0.3	209	0.007	00:14
3	10	0.3	143	0.009	00:09
4	10	0.3	6811	0.001	07:28
4	10	0.3	879	0.003	00:57
4	10	0.3	563	0.005	00:37
4	10	0.3	507	0.007	00:33
4	10	0.3	227	0.009	00:14
5	10	0.3	6029	0.001	06:33
5	10	0.3	2899	0.003	03:10
5	10	0.3	2304	0.005	02:25
5	10	0.3	1199	0.007	01:16
5	10	0.3	1669	0.009	01:46

A partir dos dados relatados é possível concluir que, o número de neurônios intermediários parece reduzir o tempo de treinamento quando são utilizadas apenas 3 entradas, porém, isso se inverte a medida que mais entradas são adicionadas. O ETA possui um efeito similar quando utilizado com poucas entradas, porém, como se observa nas linhas contendo 5 entradas, o tempo parece bastante aleatório em função deste atributo.

Isso ocorre porque esta constante está aplicada no cálculo de atualização de pesos e os valores de pesos são gerados aleatoriamente. Os valores aleatórios podem ser próximos ou distantes dos valores definitivos utilizados quando a RNA se estabilizar, e como nenhum padrão é resgatado a partir da variação do ETA, conclui-se que esta constante tem pouca relevância para a eficácia da RNA. Por último, quanto maior o erro aceitável, menor é o tempo de treinamento, esta constatação é bastante lógica sendo que, como explicado nas seções anteriores, o critério de parada do treinamento é que o valor do erro quadrático tem de ser menor que o erro aceitável. Um erro aceitável com valor muito alto acarreta numa rede com reconhecimento mais superficial.

Não é possível avaliar a qualidade do reconhecimento da RNA (taxas de reconhecimento assim como falsos positivos e negativos), devido ao fato do treinamento ser refeito cada vez que três padrões são selecionados, o que efetivamente gera uma nova configuração para a RNA. Sendo assim, dependendo dos padrões selecionados, o reconhecimento pode ser mais ou menos preciso, considerando que quando padrões com um nível alto de similaridade são escolhidos, o reconhecimento deles se torna menos preciso do que quando escolhidos padrões fortemente distintos.

A topologia final utilizada na RNA foi escolhida por apresentar os melhores tempos de treinamento em uma média geral. Os valores dos parâmetros foram os valores padrão seguintes:

- Número de entradas: 3
- Neurônios intermediários 10
- Taxa de Aprendizado (ETA): 0,3.
- Taxa de Erro aceitável: 0,005.

6. CONCLUSÃO

O propósito principal deste trabalho foi desenvolver uma RNA para ser utilizada no reconhecimento de padrões de imagem em um jogo computacional. O sistema proposto contribui no desenvolvimento de futuras RNAs com necessidades similares às apresentadas neste trabalho. Pode também ser considerado um benefício para a questão social, visto que os jogos estão enraizados na cultura contemporânea. Além disso, ao demonstrar que uma RNA eficiente pode ser aplicada em jogos, isto pode instigar a indústria a utilizar esta técnica para o desenvolvimento de experiências melhores e individualizadas.

Para elaboração deste trabalho foram necessárias pesquisas em diferentes áreas do conhecimento. Inicialmente foram estudadas as Redes Neurais Biológicas, para prover um conhecimento básico da estrutura e do funcionamento de uma RNA. Foram estudadas em seguida as Redes Neurais Artificiais, desenvolvendo o conhecimento sobre os modelos de neurônios, modelos de RNAs, diferentes métodos e meios de implementação dessas redes e da relação que a técnica de RNAs tem com o reconhecimento de imagens. Foram estudados jogos computacionais, consistindo na definição das etapas fundamentais do desenvolvimento de um jogo, assim como a definição de vários termos utilizados na indústria desse meio. Foi feita a leitura de trabalhos relacionados aos temas citados. E por fim, o desenvolvimento da RNA e da *demo* de uma interface de entrada utilizada em um jogo computacional.

Tabela 6 - Quadro comparativo com este trabalho incluso

Autores	**Objetivos**	**Modelo de RNA utilizado**	**Resultados**
Boyan (2012)	Desenvolver uma RNA para analisar os padrões dos jogos de Gamão e Jogo da Velha	*Multilayer Perceptron*	O algoritmo de Gamão competiu nas Olimpíadas de Jogos Computacionais em Londres de 1992, conquistando segundo lugar
Lubberts e Miikurulainen (2001)	Desenvolver uma rede para jogar o jogo Go, utilizando duas RNAs que competissem uma com a outra para realizar o treinamento. Uma rede focava em descobrir novas estratégias para vencer o jogo enquanto a outra tentava explorar as fraquezas da primeira.	Não informado	A utilização desse sistema de co-evolução trouxe resultados muito melhores que os limitados pelos oponentes disponíveis, e o autor acredita que estes sistemas de cooperação provavelmente serão parte das futuras soluções mais robustas do mesmo tipo de problema.
Krizhevsky (2010)	Desenvolver uma arquitetura de RNA que, essencialmente, possui dois processos separados realizando o processamento dos dados de entrada. Durante o processo de reconhecimento, as camadas co-evolucionais processam suas entradas separadamente e, no final, os dois processos "comunicam-se" entre si pelas camadas amplamente conectadas.	*Convolutional Networks*	Os resultados gerados por essa rede foram excelentes, visto que o trabalho foi submetido para a competição anual *ImageNet Large-Scale Visual Recognition Challenge* (ILSVRC), e conseguiu a melhor colocação em dois tipos de testes, o top-1 e top-5. Sendo que o primeiro testa se a resposta da rede é a esperada, e o último testa se um dos 5 primeiros resultados da rede é o esperado.
Lohmann (2016)	Desenvolver um sistema baseado em RNAs para reconhecer padrões desenhados por um jogador e a semelhança destes com as imagens originais apresentadas em um jogo computacional.	*Multilayer Perceptron*	O treinamento é mais lento quando se aumenta o número de entradas e neurônios intermediários. O ETA tem um efeito bastante aleatório. Quanto maior o erro aceitável, menor é o tempo de treinamento

Os resultados obtidos confirmam a teoria estudada: Aumentando o número de entradas, o tempo de treinamento também aumenta; com a redução do erro aceitável, o tempo de treinamento também cai. Enquanto os outros parâmetros não influenciam tanto, ou influenciam apenas uma fração do universo de possibilidades que a RNA pode passar quando é processada. A RNA foi desenvolvida com sucesso no jogo *Paresy,* tornando-o funcional e jogável, cumprindo com os objetivos descritos na primeira parte deste trabalho, afirmando que "O uso de RNAs é uma estratégia adequada para avaliar (aprender e reconhecer) as ações de jogadores em jogos computacionais".

Para trabalhos futuros, o grande foco poderia ficar no desafio de desenvolver um jogo completo utilizando a RNA e com as ideias desenvolvidas neste trabalho, com enredo, sistema de evolução do personagem, sonorização e expansão das fases, inimigos e padrões demonstrados neste trabalho. A implementação de outros modelos de RNAs como por exemplo *Kohonen,* também poderiam ser desenvolvidos, comparando com o modelo *Multilayer Perceptron* implementado para avaliar qual possui melhor desempenho.

REFERÊNCIAS

ADAMS, Ernest. Fundamentals of Game Design, San Francisco: New Riders. 2010, 696p.

ALPAYDIN, Ethem. Introduction to Machine Learning, MIT Press. 2009, 584p.

BOYAN, Justin. Modular Neural Networks for Learning Context-Dependent Game Strategies. 1992

BOW, Sing T. Pattern Recognition and Image Preprocessing, CRC Press. 2002, 720p.

CARPENTER, Gail. GROSSBERG, Stephen. REYNOLDS, John. Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system, Neural Networks, Vol 4, 1991, pp 759-771

CARPENTER, Gail. GROSSBERG, Stephen. REYNOLDS, John. ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network, Neural Networks, Vol 4, 1991, pp 565-588

CARPENTER, Gail. GROSSBERG, Stephen. REYNOLDS, John. Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system, Neural Networks, Vol 4, 1991, pp 759-771

CARPENTER, Gail. GROSSBERG, Stephen. REYNOLDS, John. Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps, IEEE Transactions on Neural Networks, Vol 3, 1991, pp 698-713

CARPENTER, Gail. GROSSBERG, Stephen. REYNOLDS, John. MARKUZON, Natalya. ROSEN, David. Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps, IEEE Transactions on Neural Networks, Vol 3, 1992, pp 698-713

CHAUVIN, Yves. RUMELHART, David. Backpropagation: Theory, Architectures, and Applications. 2013, 576p.

COUGHLIN, James. BARAN, Robert. Neural Computation in Hopfield Networks and Boltzmann Machines. 1995, 281p.

DE WILDE, Philippe. Neural network models: theory and projects. London: Springer, 1997. 174p.

FAUSETT, Laurane. Fundamentals of neural networks: architectures, algorithms, and applications. Englewood Cliffs: Prentice Hall, 1994. 461p.

GIMENEZ, Carolina Melleiro. Identificação de bovinos através de reconhecimento de padrões do espelho nasal utilizando Redes Neurais Artificiais. Universidade de São Paulo: São Paulo. 2011. (Dissertação de Mestrado)

GOODFELLOW, Ian. BENGIO, Yoshua. COURVILLE, Aaron. Deep Learning (Adaptive Computation and Machine Learning series). Cambridge: MIT Press. 2016. 800p.

GROSSBERG, Stephen. Competitive learning: From interactive activation to adaptive resonance, Cognitive Science, Vol 11, 1987, pp 23-63

KOLLIAS, Stefanos. Artificial Neural Networks - ICANN 2006. Springer Science & Business Media. 2006, 1008p.

KRIZHEVSKY, Alex. SUTSKEVER, Ilya. HINTON, Geoffrey. ImageNet Classification with Deep Convolutional Neural Networks. 2010

KUMAR, Satish. Neural Networks: A Classroom Approach, Noida: Tata McGraw-Hill. 2004, 736p

LODISH, Harvey. Berk, Arnold. ZIPURSKY, Lawrence. MATSUDAIRA, Paul. BALTIMORE, David. DARNELL, James. Molecular Cell Biology. New York: W. H. Freeman. 4th edition. 2000. 1184 p.

LUBBERTS, Alex; MIIKKULAINEN, Risto. Co-Evolving a Go-Playing Neural Network. Symposium on Computational Intelligence and Games. In: Coevolution: Turning Adaptive Algorithms upon Themselves, Birds-of-a-Feather Workshop, Genetic and Evolutionary Computation Conference (Gecco-2001, San Francisco). 2001. p. 1-6.

MARCOMINI, Karem Daiane. Aplicação de modelos de Redes Neurais Artificiais na segmentação e classificação de nódulos em imagens de ultrassonografia de mama. Universidade de São Carlos: São Paulo. 2013. 131 p. (Dissertação de Mestrado)

MOHAMAD, Hassoun. Fundamentals of Artificial Neural Networks, Cambridge: MIT Press. 1995, 511p

NIELSEN, Michael. Neural Networks and Deep Learning. 2016. Disponível em: http://neuralnetworksanddeeplearning.com/ Acessado em: 2016

PARKS W, Randolph. LEVINE S, Daniel. LONG L, Debra. Fundamentals of Neural Network Modeling: Neuropsychology and Cognitive Neuroscience. 1998, 428p.

PERELMUTER Guy. Redes Neurais Aplicadas ao Reconhecimento de Imagens Bi-dimensionais. Pontifícia Universidade Católica do Rio de Janeiro: Rio de Janeiro. 2006. 148 p. (Dissertação de Mestrado)

RABUÑAL, Juan. Artificial Neural Networks in Real-Life Applications, Calgary: Idea Group Inc. 2005 395p

SERRANO-GOTARREDONA, Teresa. LINARES-BARRANCO, Bernabé. ANDREOU, Andreas. Adaptive Resonance Theory Microchips: Circuit Design Techniques. 2012, 234p

SIMON, Haykin. Neural networks: a comprehensive foundation. Upper Saddle River: Prentice Hall, 1994. 696p.

UNITY3D. Disponível em: https://unity3d.com Acessado em 2016

VIDEO GAME INDUSTRY. Disponível em: http://vgsales.wikia.com/wiki/Video_game_industry#Comparison_with_other_forms_of_entertainment Acessado em: 2015

WERBOS, Paul John. The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting. John Wiley & Sons. 1994, 319p.

WU, Jian-Kang. Neural Networks and Simulation Methods. 1993, 456p