



Fundação Educacional do Município de Assis  
Instituto Municipal de Ensino Superior de Assis  
Campus "José Santilli Sobrinho"

**GABRIEL SOUZA DA SILVA**

**REDES NEURAIS ARTIFICIAIS PARA A IDENTIFICAÇÃO DE  
OBJETOS CONTIDOS EM IMAGENS**

Assis  
2015

**GABRIEL SOUZA DA SILVA**

**REDES NEURAIS ARTIFICIAIS PARA A IDENTIFICAÇÃO DE  
OBJETOS CONTIDOS EM IMAGENS**

Trabalho de Conclusão de Curso  
apresentado ao Instituto Municipal de Ensino  
Superior de Assis, como requisito do Curso  
de Graduação em Ciência da Computação.

**Orientador:** Prof. Dr. Luiz Carlos Begosso

**Área de Concentração:** Informática

Assis  
2015

## FICHA CATALOGRÁFICA

SILVA, Gabriel Souza

Redes Neurais Artificiais para a identificação de objetos contidos em imagens/  
Gabriel Souza da Silva. Fundação Educacional do Município de Assis – FEMA  
– Assis, 2015.

Numero de paginas: 84

Orientador: Luiz Carlos Begosso.

Trabalho de Conclusão de Curso – Instituto Municipal de Ensino Superior de  
Assis – IMESA.

1. Redes Neurais. 2. Neurônio. 3.Perceptron.

CCD: 001.6

# **REDES NEURAIS ARTIFICIAIS PARA A IDENTIFICAÇÃO DE OBJETOS CONTIDOS EM IMAGENS**

GABRIEL SOUZA DA SILVA

Trabalho de Conclusão de Curso  
apresentado ao Instituto Municipal de Ensino  
Superior de Assis, como requisito do Curso  
de Graduação em Ciência da Computação.

Orientador: Prof. Dr. Luiz Carlos Begosso

Analizador: Prof. Me. Felipe Alexandre Cardoso Pazinatto

Assis  
2015

## DEDICATÓRIA

Dedico este trabalho a minha mãe, que sempre me apoiou em todos os momentos.

## AGRADECIMENTOS

Agradeço ao Prof. Dr. Luiz Carlos Begosso pela oportunidade e apoio na elaboração deste trabalho.

Agradeço a todos os professores por me proporcionar o conhecimento não apenas racional, mas a manifestação do caráter e afetividade da educação no processo de formação profissional.

À instituição, pela oportunidade de fazer o curso.

A todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

“Quanto mais aumenta nosso conhecimento, mais evidente fica nossa ignorância”.

(John F. Kennedy)

## **RESUMO**

Os trabalhos relacionados à Inteligência artificial estão em constante crescimento, e, o seu principal objeto de estudo é o neurônio artificial, que compõe as redes neurais artificiais. Estas redes são cópias matemáticas do cérebro humano, logo, são capazes de aprender e tomar decisões a cerca de um determinado ambiente.

Este trabalho tem como objetivo desenvolver um estudo sobre as redes neurais artificiais e, a partir desse estudo, implementar uma rede neural artificial que reconheça um determinado objeto em uma imagem.

Palavras Chave: Redes Neurais; Inteligência Artificial; Neurônio Artificial.



## **ABSTRACT**

The works related to artificial intelligence are constantly growing, and its main object of study is the artificial neuron, which are part of up the artificial neural networks. These networks are mathematical copies of the human brain, so are able to learn and take decisions about a particular environment.

This work aims to develop a software implemented as an artificial neural network to recognize a particular object in an image. Studying and describing in detail the neural network along its development.

Keywords: Neural Networks; Artificial Intelligence; Artificial Neuron.

## LISTA DE ILUSTRAÇÕES

Figura 1 - Neurônio Biológico .....	17
Figura 2 - Convenção dos eixos para representação de imagens digitais .....	19
Figura 3 - Ilustração simplificada de um neurônio biológico .....	25
Figura 4 - Comunicação entre dois neurônios.....	26
Figura 5 - Modelo artificial de um neurônio .....	27
Figura 6 - Definição de um neurônio em equações.....	27
Figura 7 - Mudança proposital causada pelo bias .....	28
Figura 8 - Equação da função de limiar.....	29
Figura 9 - Função de Limiar .....	29
Figura 10 - Representação matemática da função linear por partes.....	30
Figura 11 - Função Linear por Partes.....	30
Figura 12 - Equação que representa a função rampa .....	31
Figura 13 - Representação gráfica da função rampa .....	31
Figura 14 - Função Logística.....	32
Figura 15 - Função Sigmoidal .....	32
Figura 16 - Exemplo de arquitetura com recursão entre a camada de saída e a camada intermediária.....	33
Figura 17 - Representação simples do aprendizado não-supervisionado.....	35
Figura 18 - <i>Perceptron</i> .....	36
Figura 19 - Expressão <i>perceptron</i> .....	36
Figura 20 - Rede <i>feedforward</i> de camada simples.....	37
Figura 21 - Rede <i>feedforward</i> de múltiplas camadas .....	37
Figura 22 - Tangente hiperbólica.....	40
Figura 23 - Conceito de processamento de imagem .....	41

Figura 24 - Demonstração dos pixels que formam uma imagem .....	41
Figura 25 - Fluxo de processamento de uma imagem .....	42
Figura 26 - Etapas desenvolvidas no Matlab .....	43
Figura 27 - Imagens escolhidas para o desenvolvimento dos testes .....	44
Figura 28 - Função de processamento das imagens .....	46
Figura 29 - Imagens com a aplicação do filtro de Sobel.....	47
Figura 30 – Laranja 1 .....	47
Figura 31 - Laranja2 .....	48
Figura 32 - Cenoura1 .....	48
Figura 33 - Cenoura 2 .....	49
Figura 34 – Imagem nula.....	50
Figura 35 - Estrutura da rede neural implementada .....	51
Figura 36 - Tabela de progresso .....	52
Figura 37 - <i>Performance</i> do treinamento da rede.....	53
Figura 38 - Gráficos <i>Output/Target</i> .....	54
Figura 39 - Gráficos de gradiente, Mu e validação.....	55
Figura 40 - Estrutura da segunda rede.....	56
Figura 41 - Tabela de progresso segunda rede .....	56
Figura 42 - Laranja entre outras frutas – Laranja 3 .....	57
Figura 43 - Imagem após a aplicação do algoritmo de detecção com fundo .....	59
Figura 44 - Laranja 3 tratada .....	60
Figura 45 - Vetor Laranja 3.....	63
Figura 46 - Resultado exibido no console da IDE .....	63
Figura 47 - Vetor Cenoura.....	64
Figura 48 - Resultado da simulação com a amostra da imagem Cenoura 2.....	64
Figura 49 - Vetor Imagem nula.....	64
Figura 50 - Resultado da simulação com a Imagem nula .....	65

## LISTA DE TABELAS

Tabela 1 - Tabela de resultados.....	55
Tabela 2 - Tabela de resultados da segunda rede .....	57
Tabela 3 - Tabela de resultados da simulação com a nova imagem.....	60

## LISTA DE ABREVIATURAS E SIGLAS

RNA – Rede Neural Artificial

## SUMÁRIO

<b>1. INTRODUÇÃO .....</b>	<b>16</b>
1.1. JUSTIFICATIVA .....	19
1.2. OBJETIVOS .....	20
1.3. MOTIVAÇÃO .....	20
1.4. ESTRUTURA DO TRABALHO .....	21
<b>2. REDES NEURAIS .....</b>	<b>22</b>
2.1. CARACTERIZAÇÃO DAS RNAS .....	22
2.2. HISTÓRICO.....	23
2.3. NEURÔNIO BIOLÓGICO .....	24
2.4. NEURÔNIO ARTIFICIAL .....	26
2.4.1. Funções de Ativação.....	28
2.5. ARQUITETURA DAS REDES NEURAIS.....	32
2.6. APRENDIZADO.....	33
2.6.1. Aprendizado Supervisionado .....	34
2.6.2. Aprendizado Não-Supervisionado .....	34
2.7. PERCEPTRON.....	35
2.7.1. Rede <i>Feedforward Backpropagation</i> .....	36
2.7.2. Algoritmo de <i>Levenberg-Marquardt</i> .....	38
2.7.3. Função Tangente Hiperbólica .....	39
2.8. PROCESSAMENTO DE IMAGENS.....	40
2.8.1. Reconhecimento de Padrões em imagens.....	42
<b>3. PROJETO .....</b>	<b>43</b>
3.1. PROBLEMATIZAÇÃO .....	43
3.2. AMBIENTE DE TESTES .....	44
3.2.1. Processamento das imagens .....	44
3.2.2. Criação da Rede Neural .....	50
3.2.3. Resultados .....	55

3.3. EXTRAPOLANDO O CENÁRIO LARANJA .....	57
3.3.1. Algoritmo de detecção com fundo.....	58
3.3.2. Finalização do tratamento da imagem e entrada na rede .....	59
3.4. IMPLEMENTAÇÃO DO <i>PERCEPTRON</i> EM JAVA.....	60
3.4.1. Algoritmo <i>Perceptron</i> em Java - Treinamento .....	61
3.4.2. Algoritmo <i>Perceptron</i> em Java – Simulação .....	62
3.4.3. Resultados do treinamento e da simulação do <i>Perceptron</i> em Java .....	62
4. CONCLUSÃO .....	66
REFERÊNCIAS.....	68
APÊNDICE A - Algoritmo de Aquisição e Processamento de Imagem .....	72
APÊNDICE B - Algoritmo de Detecção de Imagem com Fundo .....	74
APÊNDICE C - Algoritmo de Aquisição e Processamento de Imagem com Fundo .....	75
APÊNDICE D - Algoritmo <i>Perceptron</i> em Java Treinamento.....	76
APÊNDICE E - Algoritmo <i>Perceptron</i> em Java Simulação .....	80
APÊNDICE F - Vetor de pesos treinados .....	83
APÊNDICE G - Algoritmo para determinar amostras no Matlab .....	84

## 1. INTRODUÇÃO

Costuma-se dizer que o funcionamento de um computador está baseado no ambiente em que vivemos, ou seja, é a abstração do comportamento dos elementos que compõem este ambiente. Como exemplo, imagine-se uma esfera, que está em estado de inércia, até que outra esfera venha na sua direção e a coloca em movimento. Para que tudo isso aconteça, algum indivíduo precisou ser instruído a instanciar estas duas esferas, posicioná-las exatamente no local em que deveriam estar e realizar a interação entre as duas, como descrito. Um computador faz exatamente o mesmo, segue instruções pré-definidas por uma entidade, para movimentar os objetos de seu mundo.

De acordo com Delgado (2006), o nosso ambiente está em constante evolução, diferente dos computadores, que fazem apenas o que são programados para fazer. Assim, fica claro compreender que, os computadores precisam possuir a estrutura do nosso ambiente e não apenas a maneira como ele se comporta.

Estudos que visavam ampliar as estratégias de redução de problemas de controle de sistemas com características não lineares, se tornaram cada vez mais frequentes, assim como a busca por estruturas de máquinas inteligentes que fossem capazes de substituir o homem em determinadas tarefas (LUDWIG JR.; COSTA, 2007).

Dessa forma, muitos destes objetivos foram atingidos devido ao desenvolvimento das técnicas que copiam o comportamento do cérebro humano (LUDWIG JR.; COSTA, 2007).

O cérebro humano é formado por bilhões de neurônios, assim, muitas pesquisas para encontrar um novo tipo de máquina, basearam-se no funcionamento e na estruturação destes neurônios.

Um neurônio é composto pelos “dendritos”, que recebem os estímulos transmitidos por outros neurônios, pelo corpo de neurônio, também denominado de “soma”, que coleta e combina as informações que recebe e pelo “axônio”, que é responsável por transmitir os estímulos para outras células (TATIBANA; KAETSU, 2009). A figura 1 ilustra um neurônio natural com as características aqui apresentadas.



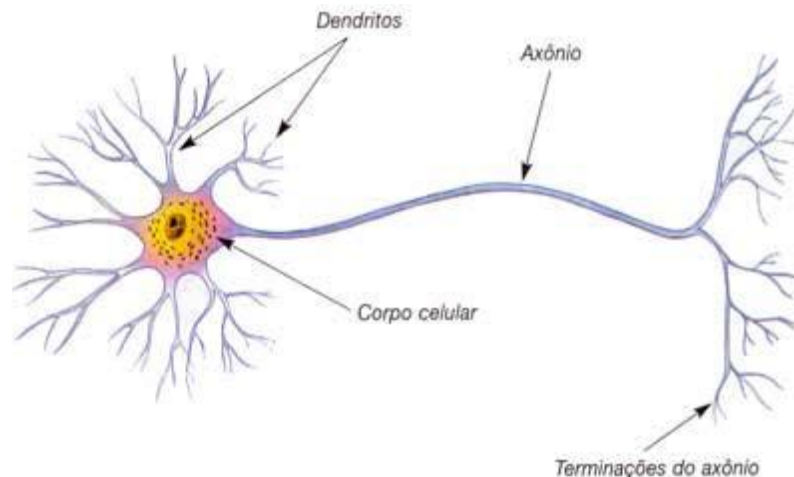


Figura 1 - Neurônio Biológico (In: BARCA et al., 2005, p. 46)

Estes pequenos processadores são distribuídos em camadas e ligados pelas conexões sinápticas. Também são chamados de células nervosas, sua principal especialização é a comunicação intercelular. A transmissão sináptica é um dos métodos que possibilitam essa comunicação neural, por meio de processos químicos e elétricos. A informação codificada por potenciais de ação é transmitida sequencialmente nos contatos sinápticos para a célula seguinte (PURVES et al., 2010, p. 5-9).

Estudos da organização celular e dos componentes moleculares de neurônios revelaram muitas de suas funções, provendo a base para a compreensão da estruturação em circuitos destes neurônios. A partir destas pesquisas, um aglomerado de possibilidades estava aberto (PURVES et al., 2010, p. 22).

Barreto (2002) destaca que na década de 40 surgiu a ideia de criar uma rede neural artificial, com um neurônio de entrada, que possui a função de receber dados externos, um neurônio de saída, que transmite os dados processados para o mundo externo ou para outro neurônio e nas camadas internas os neurônios ocultos, que auxiliam o funcionamento da rede.

Segundo Haykin (2001), redes neurais artificiais são conjuntos de processadores maciçamente paralelos e distribuídos, que possuem tendência natural de gravar conhecimento experimental e disponibilizá-lo para utilização posterior ou imediata.

Já, de acordo com a visão de Pádua (2000), uma Rede Neural Artificial (RNA) é um sistema formado por unidades de processamento simples distribuídos em paralelo, que modelam um neurônio biológico, possuindo a capacidade de mapear funções matemáticas simples sobre dados recebidos em sua entrada. Este modelo foi proposto por McCulloch e Pitts, em 1943, e é chamado pelas iniciais dos autores, MCP.

Uma RNA é parecida com o cérebro no aspecto de que o conhecimento é adquirido a partir de seu ambiente, por métodos de aprendizagem e por ajustes dos pesos de conexão entre os neurônios, os pesos sinápticos, que armazenam informações adquiridas.

Dentre as redes neurais mais conhecidas, destacam-se o *Perceptron* e a *Adaline* que se baseiam em executar operações a partir da ponderação de suas entradas (MAIA, 2012).

As RNAs são utilizadas nos dias de hoje para diversas tarefas, como na análise de aplicações financeiras, sistemas de controle e otimização, classificação, reconhecimento de voz, controle de manipuladores robóticos, reconhecimento de padrões em imagens, visão artificial, entre outras (DEMUTH & BEALE, 1996).

Como citado acima, uma das tarefas que podem ser implementadas com uma rede neural artificial, é o reconhecimento de padrões em imagens, ou, como o título deste trabalho sugere, a identificação de um objeto em uma imagem.

Muitos trabalhos já foram desenvolvidos visando esta problematização, dentre eles está o trabalho de Moreira e Costa (1996), que demonstraram uma maneira de fracionar imagens em cores, agrupando pixels em regiões utilizando a classificação. Para a implementação do projeto, Moreira e Costa (1996) utilizaram uma RNA do tipo *Self-Organizing Map*, que efetua a captura das principais cromaticidades da imagem, assim, de acordo com as classes identificadas, os pixels são classificados.

Seguindo esta linha de raciocínio, é possível concluir que a imagem se torna a entrada para a rede neural trabalhar, logo, existem diversas maneiras de representar e processar esta imagem, para que ela se torne uma entrada viável para uma rede algoritmizada.

Segundo Gonzalez e Woods (2000), “uma imagem digital é uma imagem  $f(x,y)$ ” dividida em partes menos complexas em relação a coordenadas espaciais e em brilho. Pode ser observada também como uma matriz, onde, seus elementos são chamados de pixels.

Através destes elementos uma imagem pode ser processada e interpretada, como na figura 2.

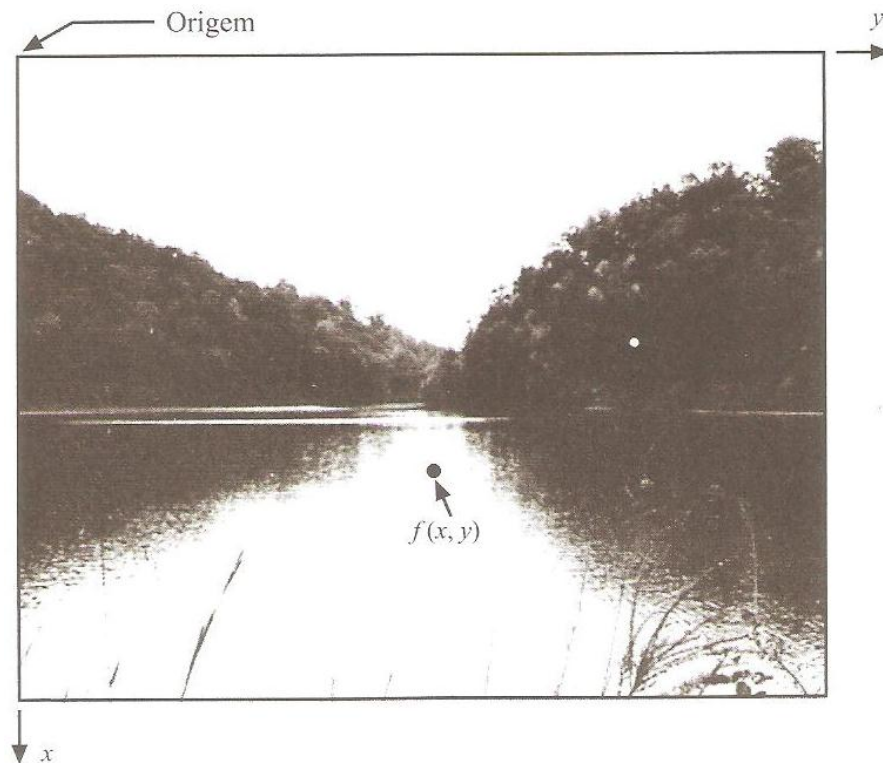


Figura 2 - Convenção dos eixos para representação de imagens digitais (In: GONZALEZ & WOODS, 2000, p. 4)

### 1.1. JUSTIFICATIVA

Existem diversas aplicações que utilizam o reconhecimento de objetos, como a visão computacional, o diagnóstico de doenças, o controle de manipuladores, a análise de situações (SILVA, 2005). Por exemplo, um semáforo que reconhecesse a estrutura de um carro, poderia avaliar uma situação e saber quando ficar verde, vermelho ou amarelo. O mesmo vale para a travessia de pedestres, que, implantando uma rede neural para reconhecer quando existe uma pessoa esperando para fazer a travessia

para o outro lado da rua, esta analisaria a situação, fechando a passagem dos carros e liberando a pessoa para atravessar a rua de acordo com a necessidade. Este trabalho se justifica pelo fato de contribuir nas pesquisas e no desenvolvimento relacionado à área de Inteligência Artificial, pois o software a ser desenvolvido será uma base para trabalhos futuros que poderão ser implementados no mundo real e trazer benefícios significantes ao ser humano.

## 1.2. OBJETIVOS

Este trabalho tem por objetivo, desenvolver uma aplicação, implementando-a como uma rede neural artificial capaz de reconhecer determinado objeto apresentado em uma imagem.

## 1.3. MOTIVAÇÃO

Tudo que o ser humano desenvolveu ao longo de sua existência, foi graças à sua capacidade de raciocínio, ou seja, sua inteligência, proporcionada pelo cérebro. Uma rede neural é uma tentativa de replicar o funcionamento deste cérebro, assim, obtém-se a capacidade de raciocínio de um cérebro, artificialmente. Esta é a base motivacional para a pesquisa das redes neurais deste projeto, pois com esta capacidade, é possível retirar o intermediário humano na tomada de decisões de qualquer aplicação, assim, existe uma grande chance de obter benefícios como, maior velocidade na apuração de dados, menor sucessão a erros, capacidade de aprendizagem superior, entre outros, que, apenas uma máquina pode alcançar.

#### 1.4. ESTRUTURA DO TRABALHO

O trabalho está dividido em quatro capítulos.

O primeiro capítulo contextualiza a área de pesquisa e apresenta os objetivos, a motivação e as justificativas para o desenvolvimento do projeto.

O segundo capítulo apresenta a fundamentação teórica sobre Redes Neurais, o qual dá sustentação para a proposta.

Para atingir o objetivo estabelecido, no terceiro capítulo, é apresentada a proposta desse trabalho.

Finalmente, no quarto capítulo, é apresentada a conclusão a respeito dos resultados obtidos após a implementação do sistema de reconhecimento, as considerações finais e o direcionamento para futuras pesquisas a partir desse trabalho.

## 2. REDES NEURAIS

Este capítulo tem por objetivo descrever o principal objeto de estudo deste trabalho, as redes neurais. Primeiramente, uma introdução apresenta as RNAs de forma objetiva e clara. Para mostrar o surgimento e o caminho que as redes neurais artificiais tiveram ante a esta pesquisa, será escrito um breve histórico. Em seguida, o neurônio biológico e o neurônio artificial serão descritos, para se compreender os objetos que constituem uma rede neural biológica e uma rede neural artificial. Na seção subsequente será demonstrada a arquitetura das redes neurais. Posteriormente, chegando ao aprendizado, será explicado o aprendizado supervisionado e o aprendizado não supervisionado. Especificamente, a seção seguinte terá por função abordar o *perceptron* de uma e de múltiplas camadas.

### 2.1. CARACTERIZAÇÃO DAS RNAS

Redes Neurais Artificiais são mecanismos computacionais que simulam o funcionamento do sistema nervoso de seres vivos (SILVA et al., 2010). A computação neural é a alternativa para fugir dos sistemas computacionais baseados em regras ou programas, pois, é uma forma não-algorítmica de computação (BRAGA et al., 1998).

Segundo Jesus (2013), uma RNA tem a capacidade de utilizar o conhecimento obtido em um procedimento chamado de aprendizagem como uma maneira de tratar a informação que lhe é fornecida.

Silva et al. (2010), cita diversas aplicabilidades deste sistema que simula a inteligência, como a classificação de padrões de escrita e fala, o reconhecimento de faces em visão computacional, o controle de trens de grande velocidade, a previsão de ações no mercado financeiro, a identificação de anomalias em imagens médicas, a avaliação de imagens captadas por satélite, o controle de aparelhos eletrônicos e eletrodomésticos, entre outras.

Uma Rede Neural trabalha da seguinte maneira: primeiramente efetua-se uma etapa chamada de aprendizagem, onde, um aglomerado de amostras é inserido na rede, e, a partir destas amostras, a rede retira as características que representam a

informação fornecida; assim, estas características extraídas serão utilizadas para trabalhar com as entradas reais da rede neural para obter o resultado do problema proposto (BRAGA et al., 1998).

## 2.2. HISTÓRICO

Foi no ano de 1943 que McCulloch e Pitts desenvolveram a primeira abstração de um neurônio biológico (EBERHART, R. & DOBBINS, R, 1990). Chamado de MCP, este trabalho visava demonstrar o modelo matemático de um neurônio, criando assim, um neurônio artificial.

O segundo trabalho foi apresentado por Donald O. Hebb no ano de 1949 levava o título de *The organization of behavior* (HEBB, 1949), de acordo com Braga et al. (BRAGA et al., 2000), Hebb afirmou que modificando os pesos de entrada dos nodos de um neurônio, alcançava-se a plasticidade de aprendizagem que poderia ser explorada em redes neurais. Após apresentar uma teoria concreta, Hebb foi reconhecido e seu trabalho ganhou o nome de Regra de Hebb, que até nos dias atuais ainda é utilizada em diversos algoritmos de aprendizagem. Widrow e Holf (1960) apresentaram outra regra de aprendizagem alguns anos depois, esta que consistia “no método do gradiente para minimização do erro na saída do neurônio com resposta linear” (BRAGA et al., 2000).

Um novo trabalho surgiu no ano de 1958, dirigido por Frank Rosenblatt, em Braga et al. (BRAGA et al., 2000) projeto este descrito como um novo modelo de rede neural, chamado de *perceptron*. Segundo Cardon e Muller (1994), este foi o primeiro modelo de rede neural a ser implementado, sendo formado por uma camada de entrada e uma de saída, para cada entrada é estabelecido um peso. Em seguida, a soma do produto entre as entradas e seus respectivos pesos passará por uma função limiar que possa definir uma saída de excitação ou inibição.

O *perceptron* funciona como um classificador de padrões linearmente separáveis (BRAGA; LUDEMIR; CARVALHO, 2000).

No ano de 1969, Minsky e Papert demonstraram os limites dos cálculos fundamentais dos *perceptrons*, sendo essa demonstração inteiramente matemática (HAYKIN, 2001).

Um exemplo citado no livro de Minsky e Papert é que um *perceptron* de camada única não tinha a capacidade de trabalhar o método da função XOR, o ou-exclusivo (WASSERMAN, 1989). Braga et al. (1998) destacam que o *perceptron* não era capaz de realizar problemas não-linearmente separáveis, como, por exemplo, a detecção de paridade.

Os anos seguintes são chamados de anos difíceis das redes neurais por Cardon e Muller (1994), devido ao decair de pesquisas relacionadas a esta área.

Cowan (1990) atribui alguns motivos para o atraso que estava por vir de mais de dez anos: tecnológico, pois não existiam locais de trabalho para experimentos e muito menos máquinas pessoais para tal e psicológico/financeiro, pois o trabalho de Minsky e Papert em 1969 não teve uma boa influência nas pesquisas relacionadas às RNAs. Nesta era, poucos trabalhos relacionados às redes neurais apareceram, um deles foi o de Teuvo Kohonem, que tratava sobre memórias associativas (KOHONEN, 1974).

Em 1982, Hopfield apresentou um trabalho que viria a encerrar o período de estagnação das pesquisas relacionadas às redes neurais. Demonstrou em seu artigo a relação entre sistemas físicos e redes recorrentes auto associativas (BRAGA et al., 1998). Estas redes propunham que a saída de um neurônio poderia servir de entrada para neurônios da mesma camada ou de camadas anteriores (ELMAN, 1990).

De acordo com Braga et al. (2000) e Silva et al. (2010), o avanço tecnológico, o desenvolvimento de algoritmos de otimização mais eficazes e robustos, a melhoria na compreensão do sistema nervoso biológico e a falta de avanços na resolução de problemas simples para um ser humano por parte das pesquisas simbolistas, acarretaram o crescimento dos trabalhos voltados às redes neurais.

## 2.3. NEURÔNIO BIOLÓGICO

Haykin (2001), descreve o cérebro como um sistema de processamento de informação único, extremamente complexo, não-linear e paralelo, que é constituído estruturalmente pelos neurônios, podendo organizá-los de acordo com a necessidade das tarefas a serem executadas.



O cérebro humano possui aproximadamente 86 bilhões de neurônios (AZEVEDO et al., 2009). Cada um destes neurônios é dividido em três partes, o corpo da célula (soma), os dendritos e o axônio (RUSSELL; NORVIG, 2010).

A figura 3 ilustra um neurônio biológico.

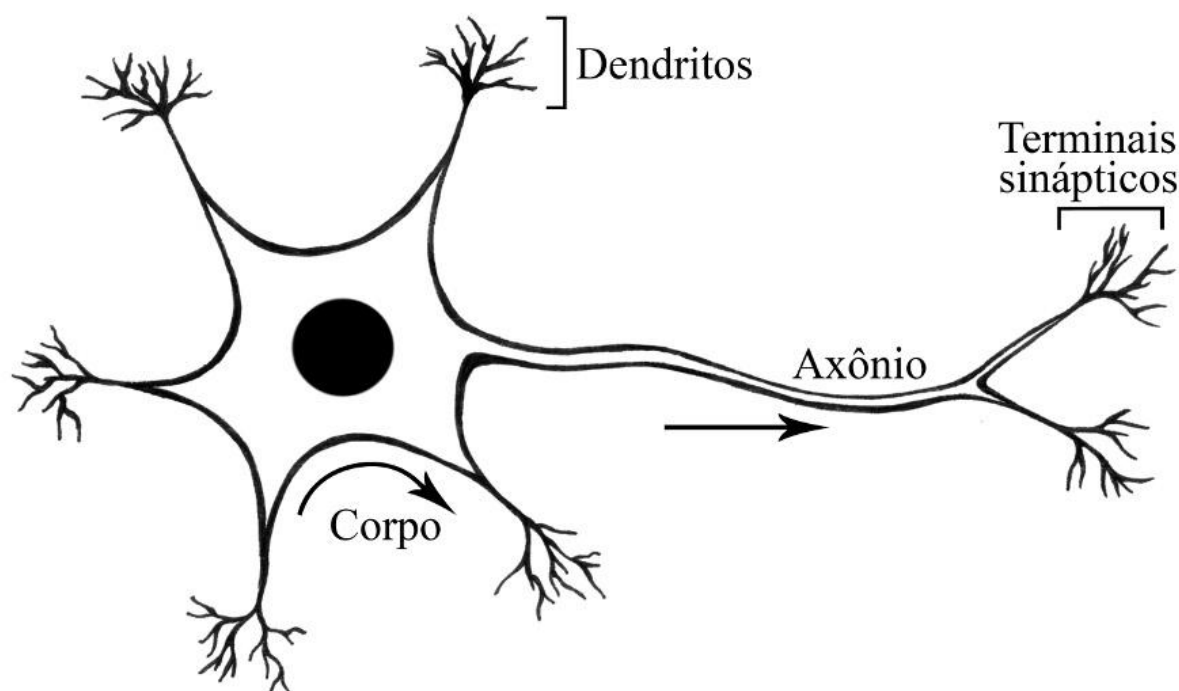


Figura 3 - Ilustração simplificada de um neurônio biológico (In: PIAZENTIN, 2011, p. 32)

Em Braga, Carvalho e Ludemir (2007), é descrito que os dendritos têm a funcionalidade de receber os dados, (impulsos nervosos), vindos de outros neurônios e transmiti-los até o corpo da célula. Os axônios são responsáveis por encaminhar estes impulsos para outros neurônios. A ligação existente entre o dendrito e o axônio é chamada de sinapse, sendo através dela que os neurônios se conectam e formam as redes neurais.

Estas sinapses operam de maneira a liberar uma substância transmissora, carregando-a a outros neurônios através da junção neural, em um processo chamado de pré-sináptico, agindo então sobre um processo pós-sináptico. Assim, segundo Shepherd e Koch (1990), um sinal elétrico pré-sináptico é convertido em um sinal

químico, e então, de volta a um sinal elétrico pós-sináptico. Simplificando, uma sinapse é uma conexão que impõe excitação ou inibição a um neurônio receptor (HAYKIN, 2001).

De uma maneira mais clara e objetiva, Braga, Carvalho e Ludemir (2007), dizem que um neurônio biológico é ativado quando recebe impulsos cuja somatória é maior do que seu limiar de excitação, também chamado de *threshold*.

Na Figura 4 é demonstrada a interação entre dois neurônios utilizando os conceitos até aqui apresentados.



Figura 4 - Comunicação entre dois neurônios (In: BALLONE, 2008)

## 2.4. NEURÔNIO ARTIFICIAL

Um neurônio artificial é uma abstração do neurônio natural e pode ser definido da seguinte forma, conforme ilustrado na figura 5:

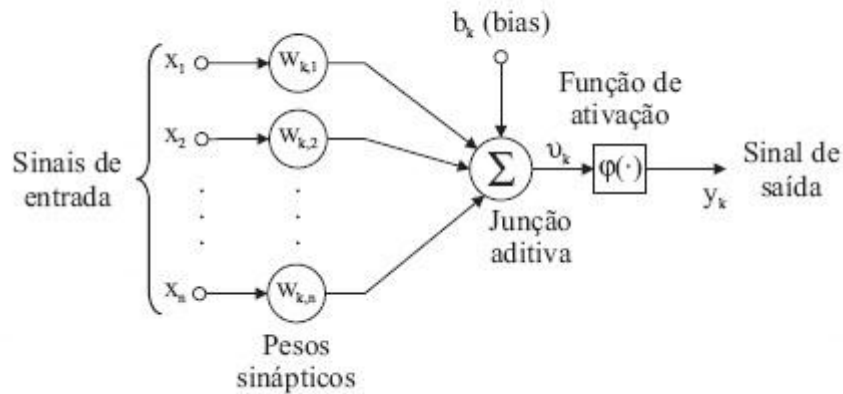


Figura 5 - Modelo artificial de um neurônio (In: HAYKIN, 2001, p. 36)

A partir da figura 5, observa-se que os dendritos são representados pelos sinais de entrada, as sinapses pelos pesos sinápticos, o corpo do neurônio pela junção aditiva e o axônio pela função de ativação (McCULLOCH; PITTS, 1943). Na Figura 5, é visualizado também um bias, descrito por  $b_k$ , que possui a finalidade de aumentar ou diminuir a entrada da função de ativação, sendo este positivo ou negativo, respectivamente (HAYKIN, 2001).

Segundo Haykin (2001), pode-se descrever um determinado neurônio K com as equações apresentadas na Figura 6.

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

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

Figura 6 - Definição de um neurônio em equações (In: HAYKIN, 2001, p.37)

Onde  $x_j$  ( $x_1, x_2, \dots, x_m$ ) identifica os sinais de entrada;  $w_{kj}$  ( $w_{k1}, w_{k2}, \dots, w_{km}$ ) descreve os pesos sinápticos do neurônio K;  $u_k$  é o resultado do combinador linear devido aos

sinais de entrada, simplificando, é a somatória da multiplicação dos pesos pelos sinais de entrada;  $b_k$  é o bias, utilizado para aumentar os graus de liberdade, tornando a rede neural adaptativa ao conhecimento fornecido a ela;  $\varphi()$  é a função de ativação; e  $y_k$  é a saída do neurônio (HAYKIN, 2001) (TATIBANA e KAETSU, 2009).

De acordo com o bias, é possível fazer modificações no campo local induzido, demonstrado no plano cartesiano da Figura 7. O bias possui um valor que é somado à saída do combinador linear do neurônio, obviamente, dependendo deste valor adicionado, o campo local induzido é modificado.

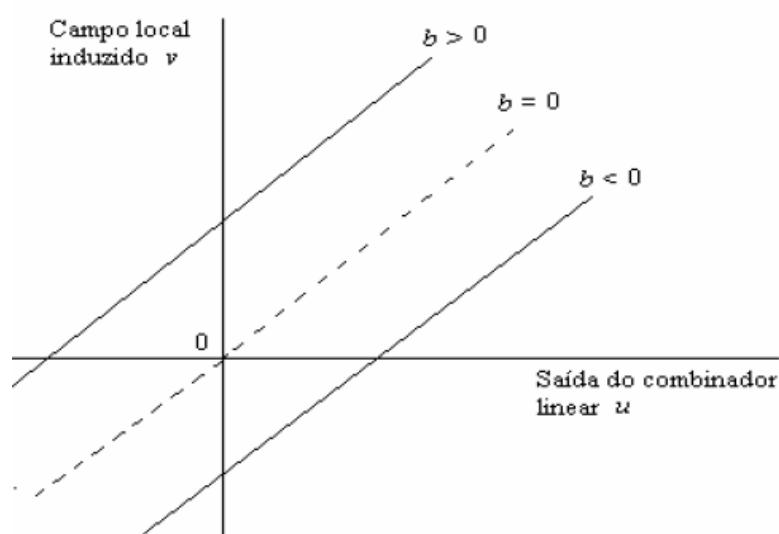


Figura 7 - Mudança proposital causada pelo bias (In: ARAÚJO, 2005, p. 30) (In: HAYKIN, 2001, p. 37)

#### 2.4.1. Funções de Ativação

De acordo com Kriesel (2005), podemos encontrar três tipos de funções que fazem parte do processamento de dados de um neurônio, a função de propagação, que efetua a soma ponderada das entradas do neurônio; a função de ativação, que define a saída de um neurônio em termos do campo local induzido e a função de saída, que transforma a ativação em saída para outros neurônios (HAYKIN, 2001).

Já, segundo Cardon e Muller (1994), é possível separar as funções que operam numa RNA em apenas dois tipos: funções para transferência de sinais entre neurônios; e funções para aprendizado de padrões.

Haykin (2001) relata que é possível encontrarmos três tipos básicos de funções de ativação:

### *Função de Limiar*

$$\varphi(v) = \begin{cases} 1, & \text{se } v \geq 0 \\ 0, & \text{se } v < 0 \end{cases}$$

Figura 8 - Equação da função de limiar (In: HAYKIN, 2001, p.38)

A figura 9 apresenta a função de limiar estendida no plano cartesiano:

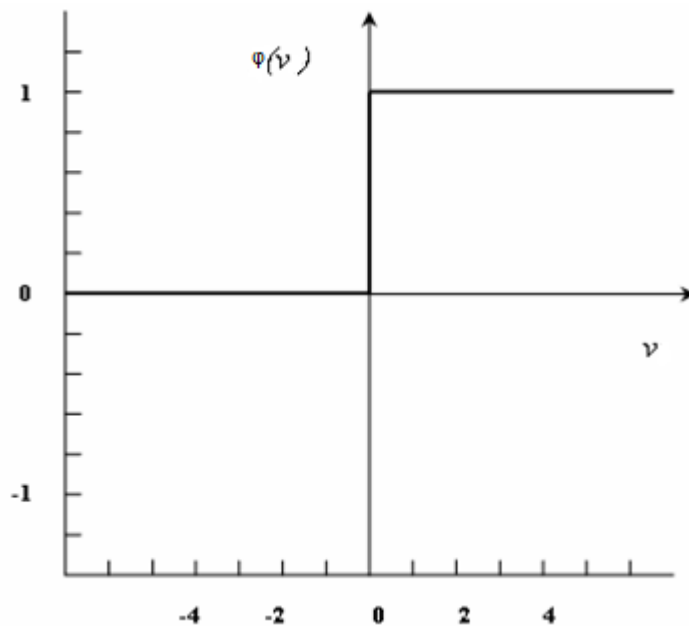


Figura 9 - Função de Limiar (In: HAYKIN, 2001, p.41)

Nesta função, a saída de um neurônio pode assumir apenas dois valores, ou seja, se o campo local induzido deste neurônio for maior ou igual à zero, então a sua saída assumirá o valor 1, caso contrário terá o valor 0 (BABINI, 2006) (HAYKIN, 2001).

### Função Linear por Partes

Segundo Haykin (2001) a Função Linear por Partes pode ser descrita como na figura 10:

$$\varphi(v) = \begin{cases} 1 & v \geq +\frac{1}{2} \\ v & +\frac{1}{2} > v > -\frac{1}{2} \\ 0 & v \leq -\frac{1}{2} \end{cases}$$

Figura 10 - Representação matemática da função linear por partes (In: HAYKIN, 2001, p. 40)

É definido que o fator de amplificação na região linear de operação é a unidade. A seguir, na Figura 11, é demonstrada a Função Linear por Partes em um plano cartesiano.

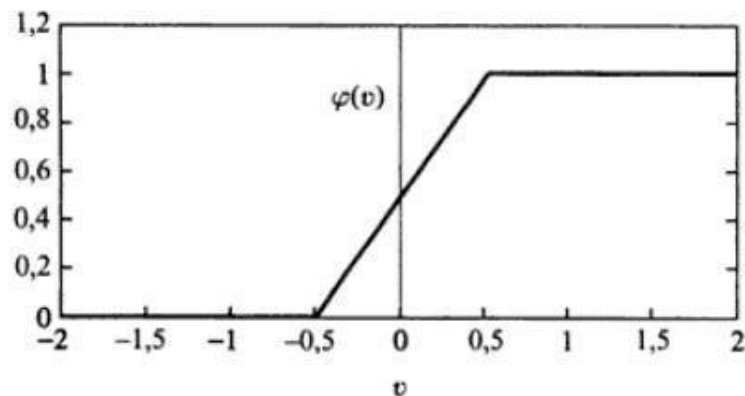


Figura 11 - Função Linear por Partes (In: HAYKIN, 2001, p. 39)

Já, de acordo com Babini (2006), esta função de ativação é definida pela equação:

$$y = \varphi(v) = av$$

Onde,  $a$  é um número real que representa a inclinação da reta, tem por finalidade definir a saída demonstrada por  $y$  para os valores de entrada  $v$ , que é o potencial de ação do neurônio.

Em Braga et al (2000), a função linear pode se tornar outra função chamada de Rampa restringindo-a a produzir valores constantes em um intervalo  $[-1, +1]$ . Representada na figura 12:

$$\varphi(v) = \begin{cases} 1 & v \geq +1 \\ v & -1 < v < +1 \\ -1 & v \leq -1 \end{cases}$$

Figura 12 - Equação que representa a função rampa (In: BABINI, 2006, p.31)

A Figura 13 representa graficamente a Função Rampa.

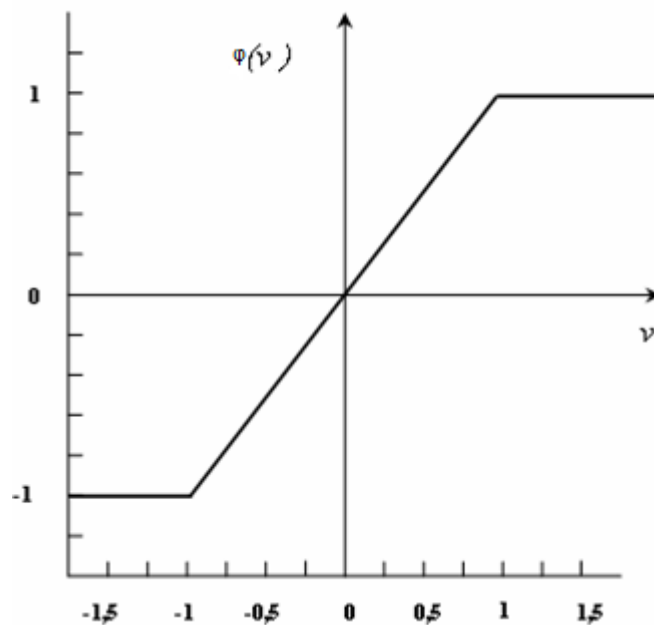


Figura 13 - Representação gráfica da função rampa (BABINI, 2006, p.31)

### *Função Sigmoides*

É a função de ativação mais comum a ser utilizada em uma arquitetura de redes neurais. Considerada uma função crescente, possui um balanceamento que se encaixa entre a conduta linear e não-linear (HAYKIN, 2001). Na figura 14 é demonstrado um exemplo de função sigmoide encontrada em Haykin (2001).

$$\varphi(v) = \frac{1}{1 + \exp(-av)}$$

Figura 14 - Função Logística (In: HAYKIN, 2001, p. 40)

Na Figura 15 é ilustrado o gráfico que representa a função sigmoide para o argumento de inclinação  $a$ .

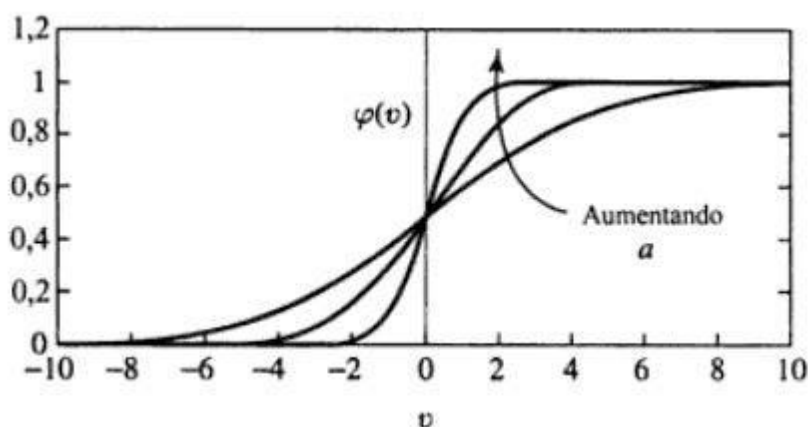


Figura 15 - Função Sigmoide (In: HAYKIN, 2001, p. 39)

## 2.5. ARQUITETURA DAS REDES NEURAIS

Silva, Spatti e Flauzino (2010), dizem que é necessária a definição de uma arquitetura na hora de dispor os neurônios uns em relação aos outros em uma determinada rede.

Esta arquitetura que necessita da topologia da rede, do número de camadas da rede, do tipo de conexão e do número dos nodos em cada camada, para que possa ser implementada (BRAGA; CARVALHO; LUDEMIR, 2000).

A figura 16 retrata um exemplo de arquitetura de redes neurais.



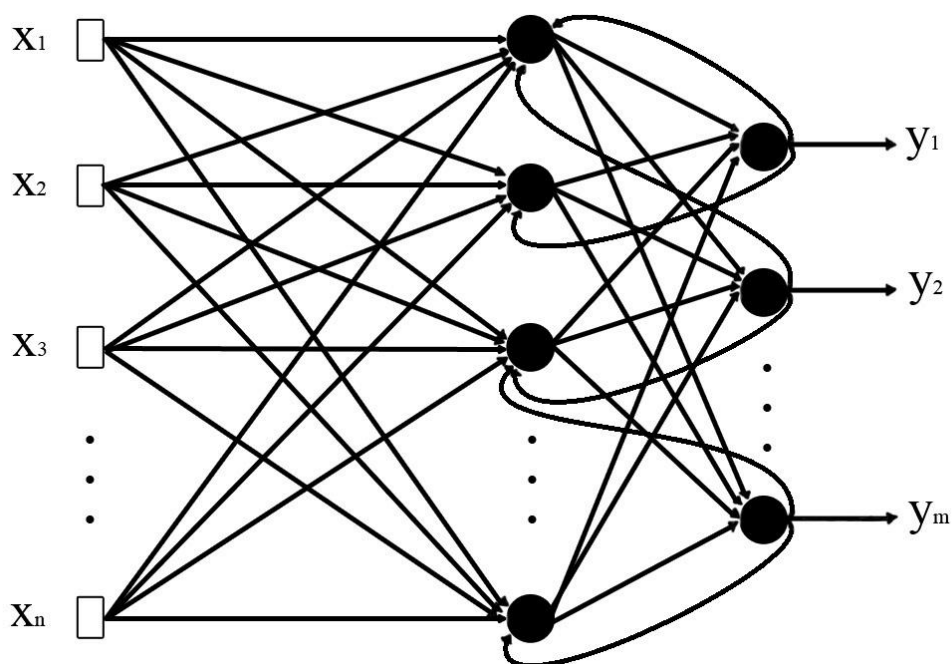


Figura 16 - Exemplo de arquitetura com recursão entre a camada de saída e a camada intermediária (In: PIAZENTIN, 2011, p. 35)

## 2.6. APRENDIZADO

Esta seção visa apresentar ao leitor uma das tarefas mais importantes a ser implementada em uma rede neural artificial, a aprendizagem. Demonstrando com uma breve introdução o que é e como é feito este procedimento.

De acordo com Braga, Carvalho e Ludemir (2007), o processo de aprendizagem é um aglomerado de procedimentos que possuem a função de adaptar os parâmetros de uma rede neural artificial, para que esta aprenda uma determinada tarefa, este processo é modelado em um algoritmo de aprendizagem.

Considerado por Haykin (2001), de extrema importância para uma rede neural, a habilidade de aprender é definida a partir de uma sequência de eventos. Primeiramente, a rede é estimulada por um ambiente, consequentemente, a partir desta estimulação, a rede sofre modificações em seus parâmetros, e por fim, uma resposta nova é gerada ao ambiente pela rede neural, graças às mudanças efetuadas em sua estrutura.

Na área de aprendizado, encontramos o associativo, onde a rede aprende a partir da relação entre os pares de estímulos, e o não-associativo, que não impõe estímulos a

serem associados, é capaz de aprender sobre suas próprias propriedades de estímulo (AZEVEDO; BRASIL; OLIVEIRA, 2000). Estes tipos serão demonstrados mais adiante como aprendizado supervisionado e aprendizado não-supervisionado, respectivamente.

### **2.6.1. Aprendizado Supervisionado**

Considerado o mais comum por Braga, Ludemir e Carvalho (2000), seja em neurônios ponderados ou não ponderados. Tem esse nome porque precisa de uma entidade para ensinar a rede e depois avaliá-la. O intuito é arranjar os parâmetros da rede, encontrando uma conexão entre as entradas e as saídas desejadas.

Este tipo de aprendizagem pode ser implementado de maneira que os parâmetros de treinamento sejam fixos, e continuem assim depois da rede ser treinada, sendo esta chamada de aprendizagem off-line. Já no método on-line, a rede precisa se adaptar frequentemente, sempre mudando seus dados (BRAGA; CARVALHO; LUDERMIR, 2007).

### **2.6.2. Aprendizado Não-Supervisionado**

Neste método não existe um supervisor para coordenar a aprendizagem da rede. Apenas serão fornecidos os sinais de entrada para o treinamento, sem especificar sinais de saída desejados (BRAGA, LUDEMIR, CARVALHO, 2000).

Segundo Silva, Spatti e Flauzino (2010), a rede neural precisa se auto organizar de acordo com as propriedades dos dados de entrada, sendo capaz de reconhecer e separar conjuntos de acordo com suas igualdades ou desigualdades.

De forma global, o aprendizado não-supervisionado pode ser representado como ilustra a Figura17.

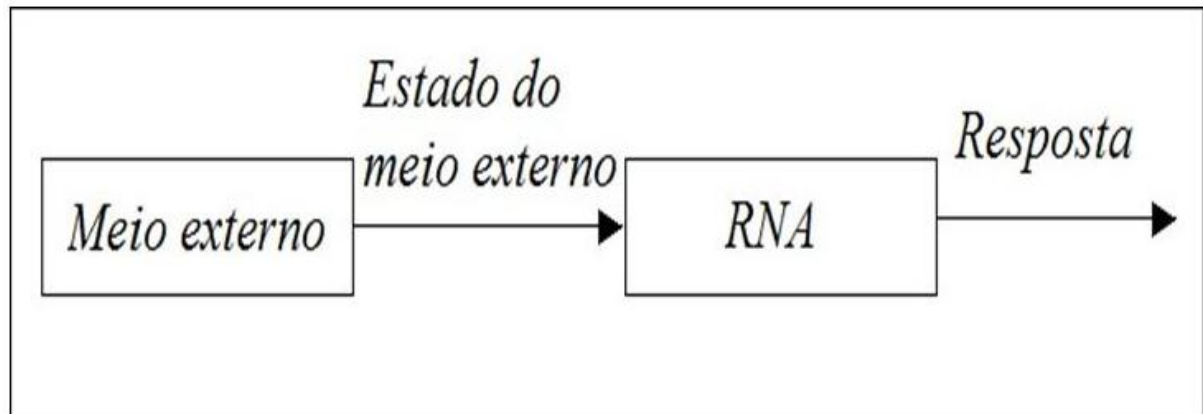


Figura 17 - Representação simples do aprendizado não-supervisionado (In: BRAGA, LUDEMIR, CARVALHO, 2000, p. 19).

## 2.7. PERCEPTRON

O *perceptron* é considerado o primeiro e mais simples modelo de uma rede neural artificial, foi criado pelo psicólogo americano Frank Rosenblatt em 1958, seu principal propósito de aplicação é a classificação de padrões linearmente separáveis. É formado por um neurônio com pesos ajustáveis e bias (HAYKIN, 2001) (SILVA et al., 2010).

O *perceptron* construído com apenas um neurônio tem a capacidade de reconhecer apenas dois padrões, mas, adicionando mais neurônios na camada de saída do *perceptron* é possível trabalhar com mais de duas classes de padrões, desde que essas classes sejam linearmente separáveis (HAYKIN, 2001).

Pertence à arquitetura *feedforward* de camada única, ou seja, o fluxo de alimentação segue sempre da camada inicial para a camada final (SILVA et al., 2010).

Na figura 18 é ilustrado um exemplo de rede *perceptron*.

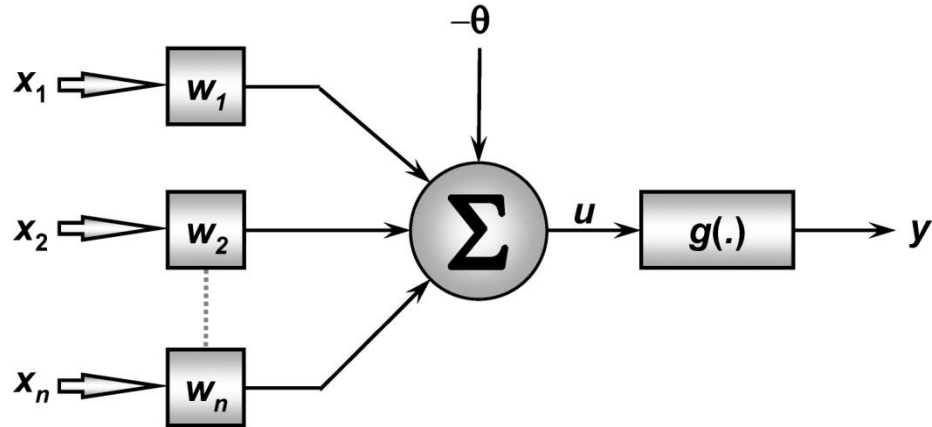


Figura 18 - *Perceptron* (In: SILVA et al., 2010, p. 58)

Matematicamente, o funcionamento de uma rede *perceptron* é dado pela seguinte expressão da figura 19:

$$\begin{cases} u = \sum_{i=1}^n w_i \cdot x_i - \theta \\ y = g(u) \end{cases}$$

Figura 19 – Expressão *perceptron* (In: SILVA et al., 2010, p. 59)

Onde  $x_i$  são as entradas da rede,  $w_i$  é o peso respectivo á entrada em calculo (i-ésima),  $\theta$  é o limiar de ativação,  $g(.)$  é a função de ativação e  $u$  é o potencial de ativação (Silva et al., 2010).

A grande vantagem do *perceptron* é a sua simplicidade de implementação, pois, dependendo da aplicação, os padrões de entrada não precisam de um pré-processamento demasiadamente elaborado, além de possuir poucos parâmetros a ajustar (CARDON & MULLER, 1994).

### 2.7.1. Rede *Feedforward Backpropagation*

Quando uma rede é considerada *feedforward*, quer dizer que o fluxo de dados é transmitido em apenas uma direção, da camada inicial para a camada final (entrada/saída). Este tipo de rede é utilizado em reconhecimento de padrões e filtragem linear, logo, os principais tipos de redes que fazem uso desta arquitetura

são o *Perceptron* e o *Adaline* (SILVA et al., 2010). Nas figuras 20 e 21 podemos ver exemplos de como essa arquitetura funciona, onde as linhas com setas indicam a direção do fluxo de informações.

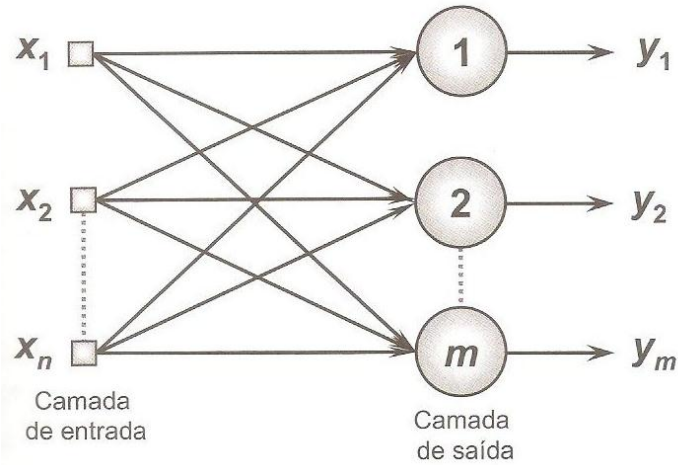


Figura 20 - Rede *feedforward* de camada simples (In: SILVA et al., 2010, p. 47)

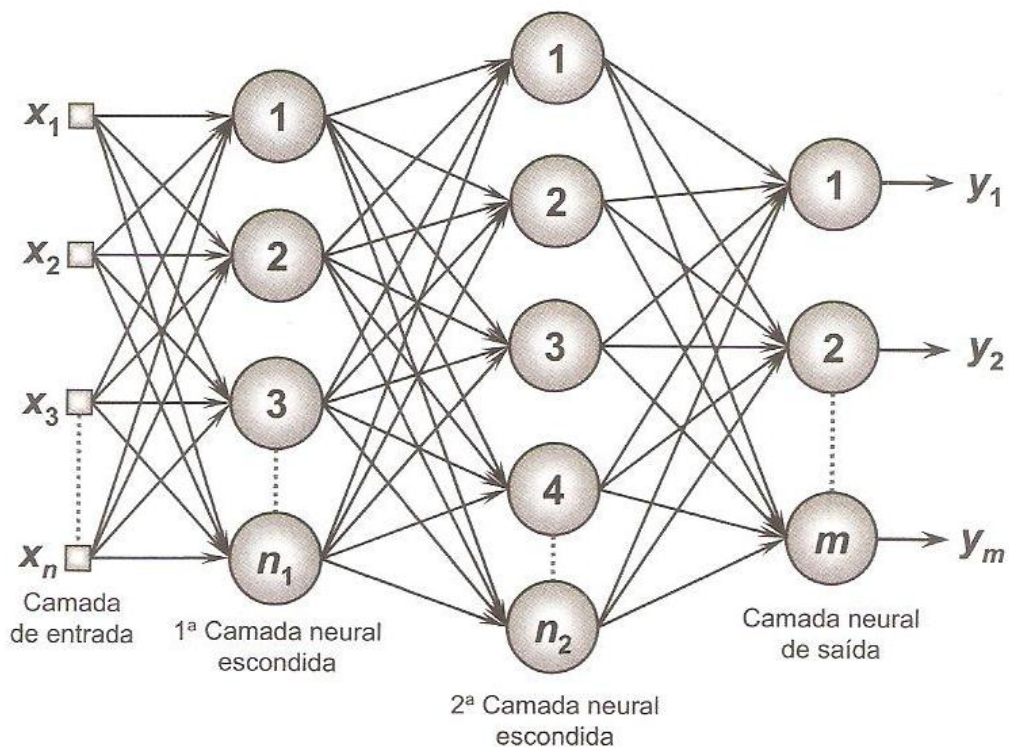


Figura 21 - Rede *feedforward* de múltiplas camadas (In: SILVA et al., 2010, p. 48)

É possível observar que na primeira figura está ilustrada uma rede neural de camada única, ou seja, com apenas uma camada com um conjunto de neurônios, já na segunda figura está representada uma rede que possui diversas camadas, cada

uma com um conjunto de neurônios. De acordo com Silva et al. (2010), redes de múltiplas camadas são utilizadas para solucionar diversas classes de problemas, como as ligadas a classificação de padrões, robótica, otimização, aproximação de funções, entre outras.

O termo *backpropagation* refere-se ao algoritmo de treinamento que implementa a rede, é comum utilizá-lo em redes de múltiplas camadas e funções de transferência não lineares (KHATCHATOURIAN & PADILHA, 2008).

De acordo com Rosa (2011) consiste em duas etapas, propagação da ativação e retro propagação do erro. A propagação da ativação é feita da camada de entrada para a camada oculta e da camada oculta para a camada de saída, enquanto que, na retro propagação do erro é feito o calculo do erro para as unidades de saída, e esses erros são retro propagados para as unidades intermediárias e de entrada, assim, estas duas etapas concluem um ciclo de ativação.

Por padrão, o algoritmo *backpropagation* utiliza o método de gradiente descendente como função de aproximação do mínimo da função erro, diferente do algoritmo de *Levenberg-Marquardt*, que será utilizado neste trabalho, o que usa uma aproximação pelo método de Newton (HAGAN; MENHAJ, 1994).

### 2.7.2. Algoritmo de *Levenberg-Marquardt*

O algoritmo de *Levenberg-Marquardt* foi desenvolvido para aproximar velocidade de treinamento de segunda-ordem sem ter que calcular a matriz de Hessian. Quando a função de desempenho tem a forma de uma soma de quadrados, tipicamente usado em redes *feedforward*, assim, a matriz hessiana pode ser aproximada como:

$$\mathbf{H} = \mathbf{J}^T \mathbf{J}$$

E o gradiente pode ser calculado como:

$$\mathbf{g} = \mathbf{J}^T \mathbf{e}$$

Onde  $\mathbf{J}$  é a matriz Jacobiana que contém as primeiras derivadas dos erros da rede relacionados com os pesos e desvios, e  $\mathbf{e}$  é um vetor de erros da rede.

Este algoritmo parece ser o método mais rápido de treinamento de redes neurais *feedforward* de tamanho moderado. Possui também uma implementação muito eficiente no Matlab, uma vez que a solução da equação de matriz é uma função *built-in* (DEMUTH & BEALE, 1998).

### 2.7.3. Função Tangente Hiperbólica

A função tangente hiperbólica foi a função de ativação escolhida para a rede neural implementada neste trabalho, que segundo Silva (1998) pode ser definida pela seguinte expressão:

$$f(x) = \frac{e^{px} - e^{-px}}{e^{px} + e^{-px}} = \tanh(px)$$

E sua derivada é dada por:

$$f'(x) = p(1 - f(x)^2)$$

Simplificando, a função tangente hiperbólica é a razão entre o seno hiperbólico e o cosseno hiperbólico, como pode ser visto na expressão abaixo:

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)}$$

Na figura 22 está ilustrado o gráfico desta função.

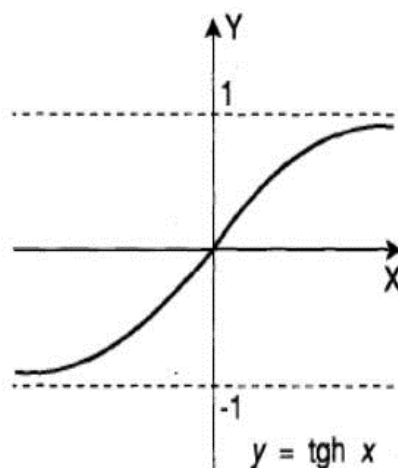


Figura 22 - Tangente hiperbólica (In: FLEMMING; GONÇALVES, 2007, p. 39)

## 2.8. PROCESSAMENTO DE IMAGENS

O processamento de imagens é um método desenvolvido para o tratamento de dados multidimensionais, tem como principal objetivo obter qualquer tipo de informação a respeito de uma imagem. Entretanto, essa metodologia é considerada complexa e trabalhosa, o que resulta quase sempre em um alto custo computacional (ALVES, 2013).

Diversas tarefas são realizadas com o auxílio do processamento de imagens, como, o tratamento de imagens digitais com filtros e compressão, e a extração de parâmetros (OSÓRIO et al., 2000).

Através de funções matemáticas implementadas em algoritmos é possível fazer a análise e realizar transformações em imagens (OSÓRIO et al., 2000). Desta maneira, uma função pode ser utilizada no processamento de uma imagem a fim de obter um determinado tipo de informação ou transformação da mesma. A figura 23 faz uma ilustração deste conceito.



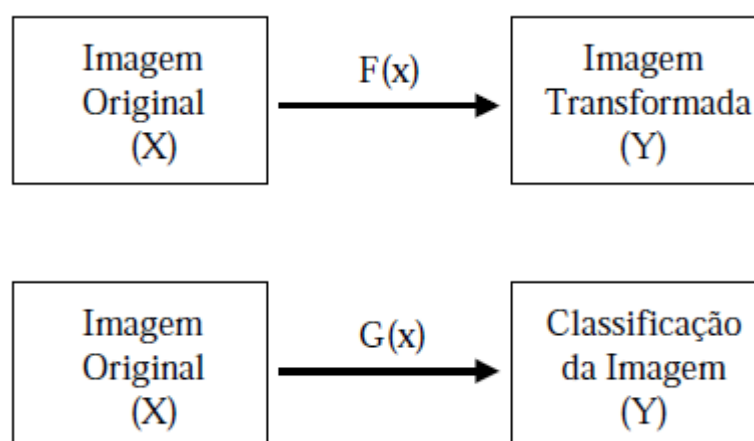


Figura 23 - Conceito de processamento de imagem (In: OSÓRIO et al., 2000, p. 19)

Um computador consegue compreender e projetar uma imagem de diversas maneiras, mas a base para toda essa interpretação são os pixels (*Picture Elements*), que, segundo Silva (2005), é a menor unidade de uma imagem digital, e é a representação numérica de um ponto dentro desta imagem. Na figura 24 podemos observar como os *pixels* representam uma imagem.

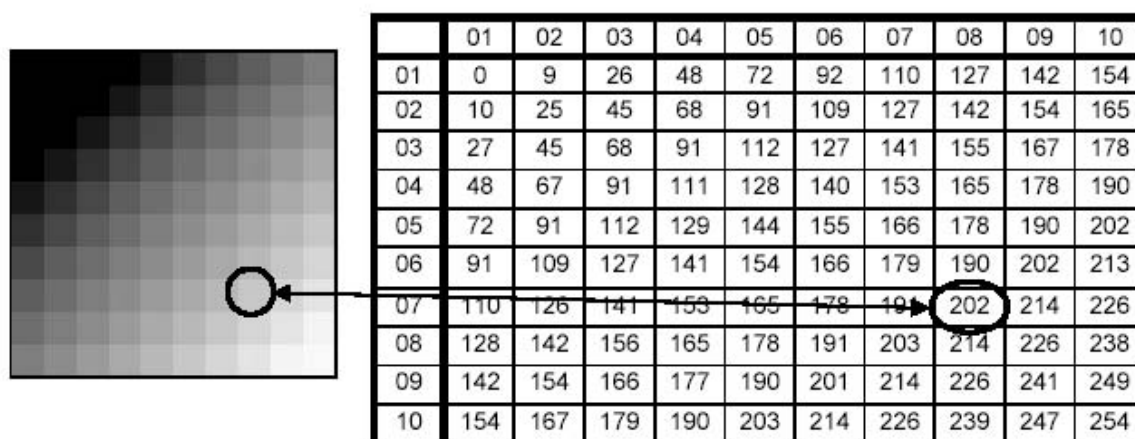


Figura 24 - Demonstração dos pixels que formam uma imagem (In: SILVA, 2005, p. 6)

Na figura 25 está ilustrado o fluxo dos elementos do processamento de uma imagem.

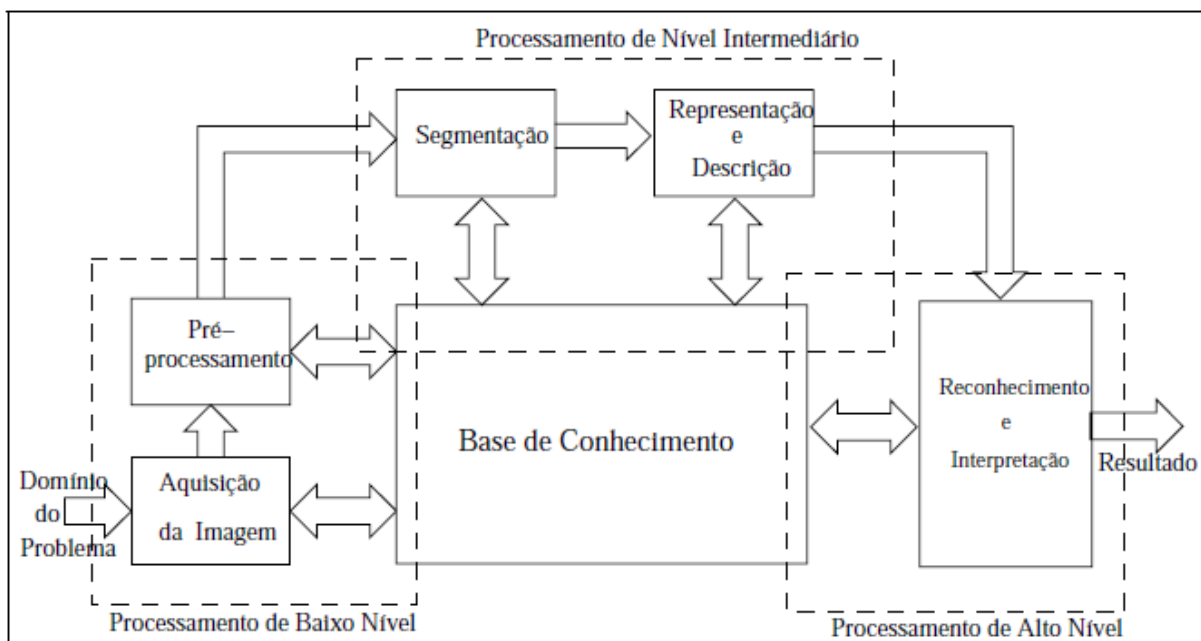


Figura 25 - Fluxo de processamento de uma imagem (In: AUGUSTO, 2002, p. 3)

### 2.8.1. Reconhecimento de Padrões em imagens

Por meio de um algoritmo de aprendizado supervisionado é possível implementar a metodologia de reconhecimento de padrões, onde as imagens precisam apenas ser separadas em classes para a aplicação do algoritmo de treinamento (OSÓRIO et al., 2000).

De acordo com Silva (2005) hoje em dia existem diversas técnicas de reconhecimento de padrões, que estão divididas em: Estatística: através de métodos estatísticos são separadas as classes das imagens previamente transformadas em um conjunto de medidas de características (exemplos: métodos probabilísticos, classificadores bayesianos e regras de decisão); Estrutural: o reconhecimento é feito com a interligação de símbolos que representam os padrões ou, a partir de uma linguagem artificial, onde os padrões de símbolos são tratados como sentenças; e Neural: este tipo de reconhecimento utiliza as Redes Neurais Artificiais, que é o foco deste trabalho.

### 3. PROJETO

Este capítulo tem como principal objetivo apresentar a implementação do trabalho, dividindo em seções para cada etapa de seu desenvolvimento.

#### 3.1. PROBLEMATIZAÇÃO

Neste trabalho foram desenvolvidos algoritmos capazes de tratar um conjunto de imagens a fim de transformá-las em entradas viáveis para uma RNA do tipo *Feedforward Backpropagation*. Para isso foi utilizado o software Matlab, que é voltado para cálculos numéricos e possui uma linguagem de programação própria. Na figura 26 é apresentada uma ilustração das etapas que foram implementadas no Matlab.

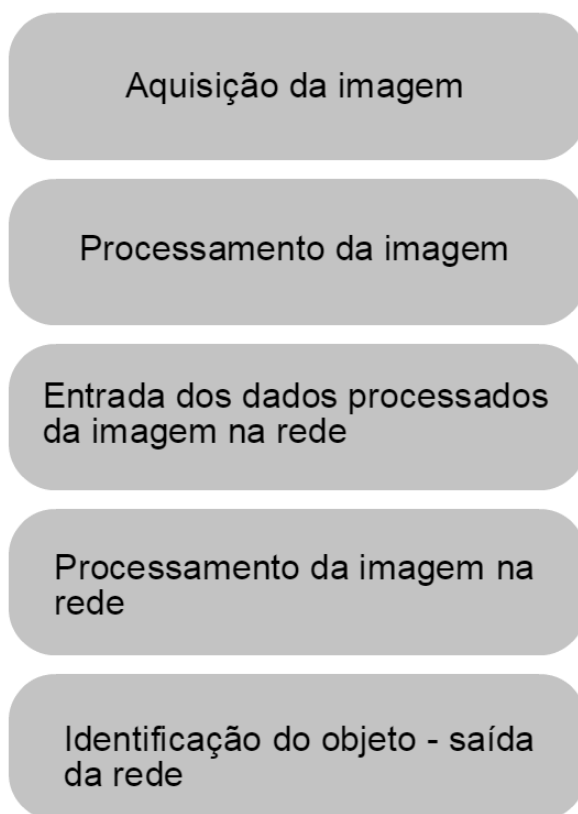


Figura 26 - Etapas desenvolvidas no Matlab

Também foi desenvolvido um algoritmo como uma rede neural capaz de identificar os padrões de um objeto dentro de uma imagem utilizando a linguagem de programação Orientada a Objetos Java, na interface de desenvolvimento Eclipse. As imagens tratadas no Matlab serviram como entrada para o treinamento e simulação desta rede.

### 3.2. AMBIENTE DE TESTES

Nesta etapa do projeto foi utilizado o software Matlab, que é um ambiente computacional técnico-científico projetado para o desenvolvimento de aplicações sofisticadas (MATSUMOTO, 2002).

Seu nome vem de *Matrix Laboratory*, pois possui uma linguagem de programação própria baseada em uma matriz. Dentre os seus principais atributos estão as *toolboxes*, que são bibliotecas ou ferramentas para um determinado tipo de aplicabilidade, por exemplo, Otimização de Sistemas, Lógica *Fuzzy*, Processamento de Sinais, *Wavelets*, Cálculo Simbólico, entre outras (MATSUNAGA, 2012). Neste trabalho foram utilizadas duas destas ferramentas, Redes Neurais (*Neural Network Toolbox*) e Processamento de Imagens (*Image Processing Toolbox*).

#### 3.2.1. Processamento das imagens

Primeiramente, uma busca de imagens foi realizada via internet, estas precisavam ser padronizadas e livres de ruídos, assim, quatro imagens foram selecionadas e inseridas no *workspace* do Matlab. Na figura 27 estão as imagens escolhidas.

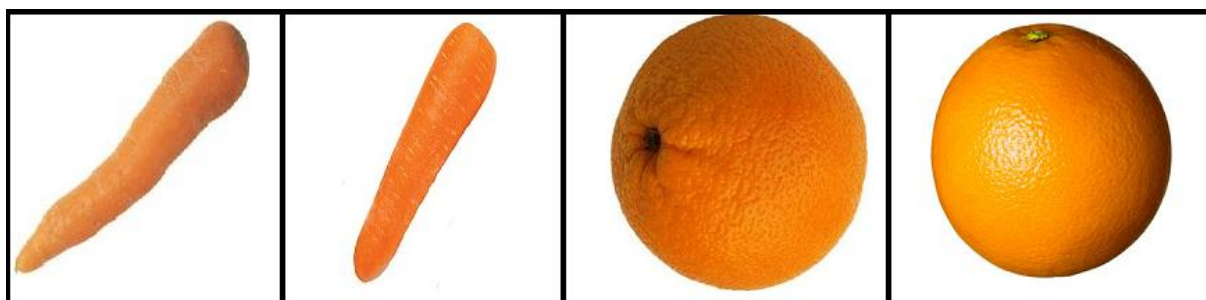


Figura 27 - Imagens escolhidas para o desenvolvimento dos testes

As imagens apresentam dois padrões que podem ser notados facilmente por qualquer pessoa, desta maneira, após a extração de dados das imagens, a rede neural pôde identificar e separar com mais velocidade e precisão esses padrões.

Uma imagem é um conjunto de pixels distribuídos especificamente em uma matriz, e, cada um destes pixels tem um valor numérico que representa uma cor, assim um computador pode interpretar e trabalhar com uma imagem digital. Este projeto fez uso destes conhecimentos para desenvolver um sistema de aquisição e tratamento de imagens em conjunto com o software Matlab.

A função apresentada na figura 28 foi criada para leitura e processamento das imagens e pode ser encontrada fazendo parte do algoritmo localizado no Apêndice A deste trabalho.

```

% Aquisição e processamento das imagens

% Leitura das imagens
im1 = imread('laranja3.png')
im2 = imread('laranja4.png')
im3 = imread('cenoura4.png')
im4 = imread('cenoura5.png')

% Binarização das imagens
im1p = im2bw(im1, 0.9);
im2p = im2bw(im2, 0.9);
im3p = im2bw(im3, 0.9);
im4p = im2bw(im4, 0.9);

% Transformação das imagens em vetores coluna
im1v = im1p(:);
im2v = im2p(:);
im3v = im3p(:);
im4v = im4p(:);

% Criação da matriz da Imagem nula
Im6p = im1p;
Im6p = ones(12);
Im6v = im6p(:);

% Criação da matriz de entrada da rede
M = [im1v,im2v,im6v,im6v];

% Criação do vetor de saída desejada da rede
Result = [1,1,-1,-1];

```

Figura 28 - Função de processamento das imagens

No momento da leitura de uma imagem, o Matlab cria uma matriz correspondente a ela em sua área de trabalho. Assim, foram criadas quatro matrizes, uma para cada imagem.

Em seguida, foi utilizado o filtro de Sobel para a detecção de bordas, o resultado pode ser visto na figura 29.

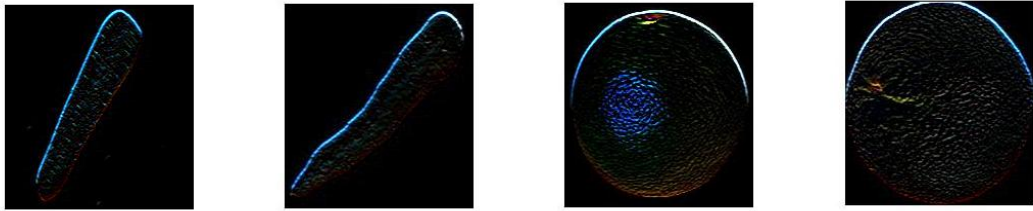


Figura 29 - Imagens com a aplicação do filtro de Sobel

As matrizes obtidas após a aplicação do filtro de Sobel ainda não estavam adequadas para trabalhar com uma rede neural, então surgiu a proposta de aplicar a binarização nas imagens, que é a transformação das imagens em um arranjo lógico. Feita a binarização, foram obtidas as matrizes que podem ser vistas nas figuras 30, 31, 32 e 33, onde ainda é possível observar a olho nu o padrão de cada imagem.

```

true true true true false false false false true true true true
true true false false false false false false false false true true
true false false false false false false false false false false true
true false false false false false false false false false false true
false false false false false false false false false false false false
false false false false false false false false false false false false
false false false false false false false false false false false false
false false false false false false false false false false false false
true false false false false false false false false false false true
true false false false false false false false false false false true
true true false false false false false false false false true true
true true true true false false false false true true true true

```

Figura 30 – Laranja 1

true true true false false false false false false true true true  
 true true false false false false false false false false true true  
 true false false false false false false false false false false true  
 true false false false false false false false false false false true  
 false false false false false false false false false false false  
 false false false false false false false false false false false  
 false false false false false false false false false false false  
 false false false false false false false false false false false  
 false false false false false false false false false false false  
 true false false false false false false false false false false true  
 true true false false false false false false false false true true  
 true true true false false false false false false true true true

Figura 31 - Laranja 2

true true true true true true true false true true true true  
 true true true true true true false false false true true true  
 true true true true true false false false false true true true  
 true true true true true false false false false true true true true  
 true true true true false false false true true true true true true  
 true true true false false false true true true true true true true  
 true true true false false false true true true true true true true  
 true true true false false true true true true true true true true  
 true true false false false true true true true true true true true  
 true true false false true true true true true true true true true  
 true true false true true true true true true true true true true

Figura 32 - Cenoura1





Figura 33 - Cenoura 2

Finalmente obtive se os dados matemáticos que representam com singularidade cada padrão apresentado nas imagens.

Assim, uma última matriz foi adicionada ao *workspace* do Matlab, seu objetivo era apresentar uma amostra completamente nula em relação aos dois padrões trabalhados até aqui. Esta matriz pode ser vista logo abaixo, na figura 34.



Figura 34 – Imagem nula

Dando sequência ao trabalho, as matrizes foram transformadas em vetores, onde, os correspondentes às laranjas e a imagem nula foram atribuídos cada um a uma coluna da matriz principal que servirá de entrada para a rede neural.

### 3.2.2. Criação da Rede Neural

Com o auxílio do *Neural Network Toolbox* foi criada a Rede Neural no Matlab. O tipo da rede escolhido foi *Feed-forward backpropagation*, com a função de treinamento *Levenberg-Marquardt* (`trainlm`) e a função de performance *Mean Squared Error* (`mse`). A função de transferência escolhida foi a tangente hiperbólica, pois para esta função a saída do neurônio pode variar de  $-1$  a  $1$ , permitindo maior eficiência da rede durante o treinamento do que no caso de saídas variando de  $0$  a  $1$ . Esta função foi usada em todos os neurônios da rede, tanto os da camada intermediária como os da camada de saída. A seguir está descrita a sintaxe para criar a rede no Matlab e o algoritmo pode ser encontrado no Apêndice A do trabalho:

***redeneural = newff (A , [ B ] , { C } , D , E , F )***

Onde, **A** são os valores mínimos e máximos dos padrões de entrada, **B** são os números de neurônios de cada camada, **C** são as funções de transferência das camadas, **D** é a função de treinamento da rede, **E** é a função de aprendizado dos pesos, **F** é a função de performance.

Na figura 34 é possível observar como ficou a estrutura da rede neural.

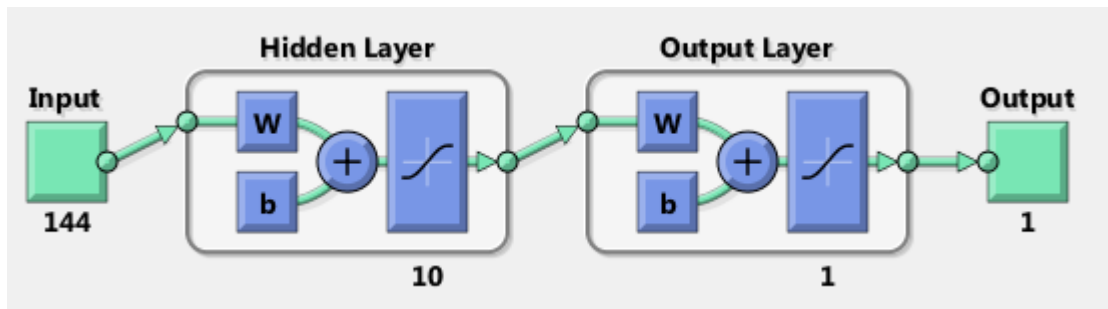


Figura 35 - Estrutura da rede neural implementada

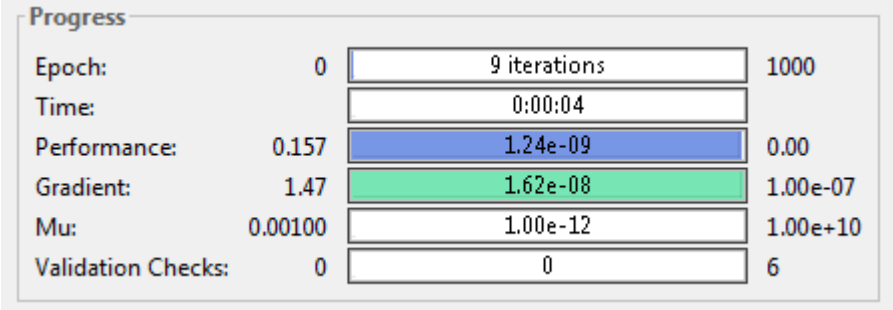
A matriz principal obtida após a binarização das imagens foi utilizada como dado de entrada para o treinamento da rede, sendo que, cada coluna desta matriz identificava uma imagem. Um vetor que deveria corresponder a cada resultado esperado por cada coluna foi criado, assim, a rede deveria trabalhar para reconhecer os padrões apresentados pelas laranjas, treinando e adaptando seus pesos para que o número 1 deste vetor as identificasse.

Para o treinamento da rede com a função *trainlm (Levenberg-Marquardt)*, foi utilizado o seguinte comando:

***[ novarede, pr ] = train (novarederedeneural , M , Alvo )***

Onde, ***novarede*** é uma nova rede neural que salvará os dados da rede que será treinada, ***pr*** é onde serão gravados os parâmetros de treinamento (número de épocas e performance), ***train*** é o comando para treinar a rede, ***novarederedeneural*** é a rede que será treinada, ***M*** é a matriz de entrada (vetores coluna) e ***Alvo*** é o resultado esperado da rede.

Durante a execução do treinamento da rede, o Matlab oferece uma interface de análise de progresso que pode ser vista na figura 35.



The image shows a 'Progress' window from MATLAB. It contains a table with training metrics. The 'Performance' row is highlighted in blue, and the 'Gradient' row is highlighted in green. The 'Mu' row is highlighted in light green. The 'Validation Checks' row is highlighted in light yellow. The 'Epoch' row is highlighted in light blue. The 'Time' row is highlighted in light green. The 'Performance' row is highlighted in blue. The 'Gradient' row is highlighted in green. The 'Mu' row is highlighted in light green. The 'Validation Checks' row is highlighted in light yellow.

Progress			
Epoch:	0	9 iterations	1000
Time:		0:00:04	
Performance:	0.157	1.24e-09	0.00
Gradient:	1.47	1.62e-08	1.00e-07
Mu:	0.00100	1.00e-12	1.00e+10
Validation Checks:	0	0	6

Figura 36 - Tabela de progresso

Podemos observar o número de épocas, que é um tipo de contador de ciclos que possui o objetivo de contar e limitar a quantidade de tentativas da rede de adequar seus pesos para o reconhecimento do objeto em questão. Neste caso, em particular, foram realizadas nove iterações, o tempo de duração da execução da rede foi de quatro segundos, com uma *performance* de 1.24e-09, um gradiente de 1.62e-08 e Mu de 1.00e-12. O Mu é um parâmetro de controle para o algoritmo de treinamento, onde é estipulado um valor inicial e ao longo da execução da RNA este valor aumenta gradativamente junto com a rede, parando apenas quando a rede alcança o resultado que busca.

A figura 36 representa o desempenho da rede ao longo de sua execução.

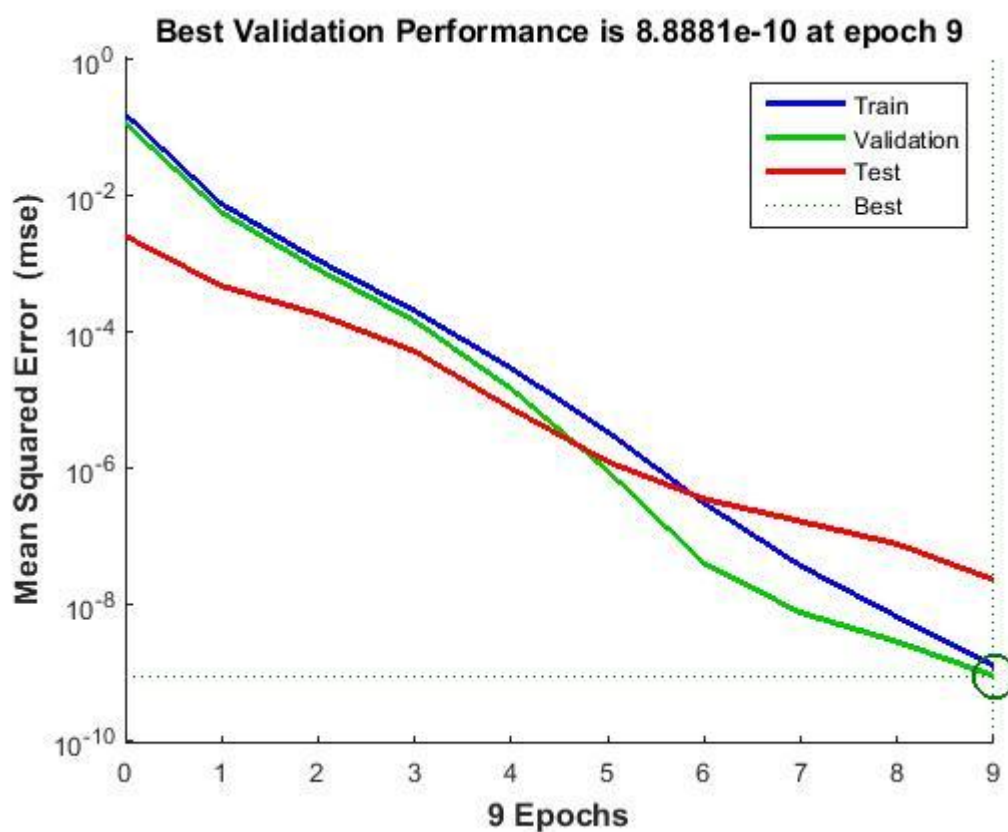


Figura 37 - *Performance* do treinamento da rede

Na figura 37 é possível observar os gráficos de treinamento, validação e teste demonstrando a adequação do alvo que a rede atingiu com a saída desejada.

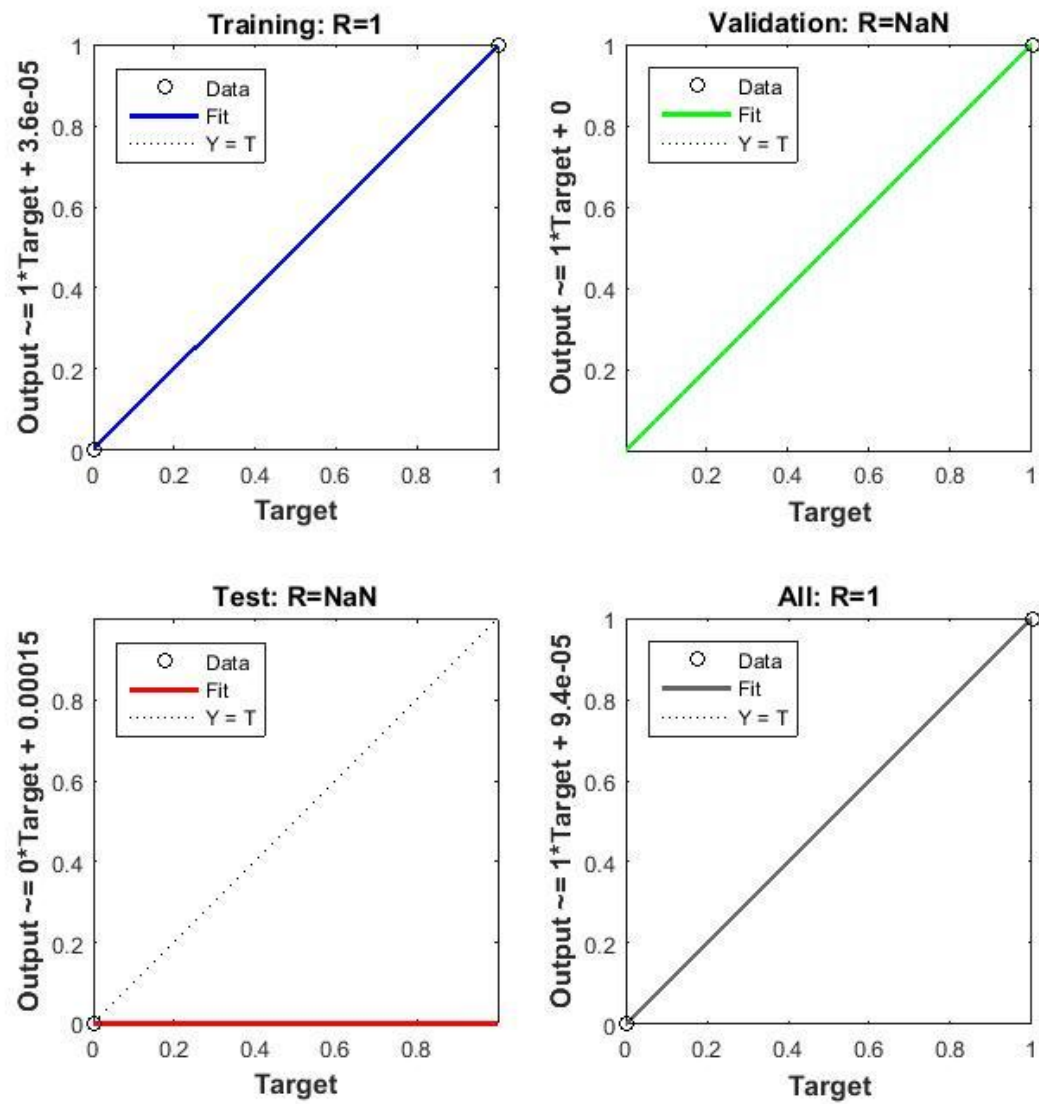


Figura 38 - Gráficos *Output/Target*

Na figura 38 é demonstrado o gradiente, o Mu e a checagem de validação.

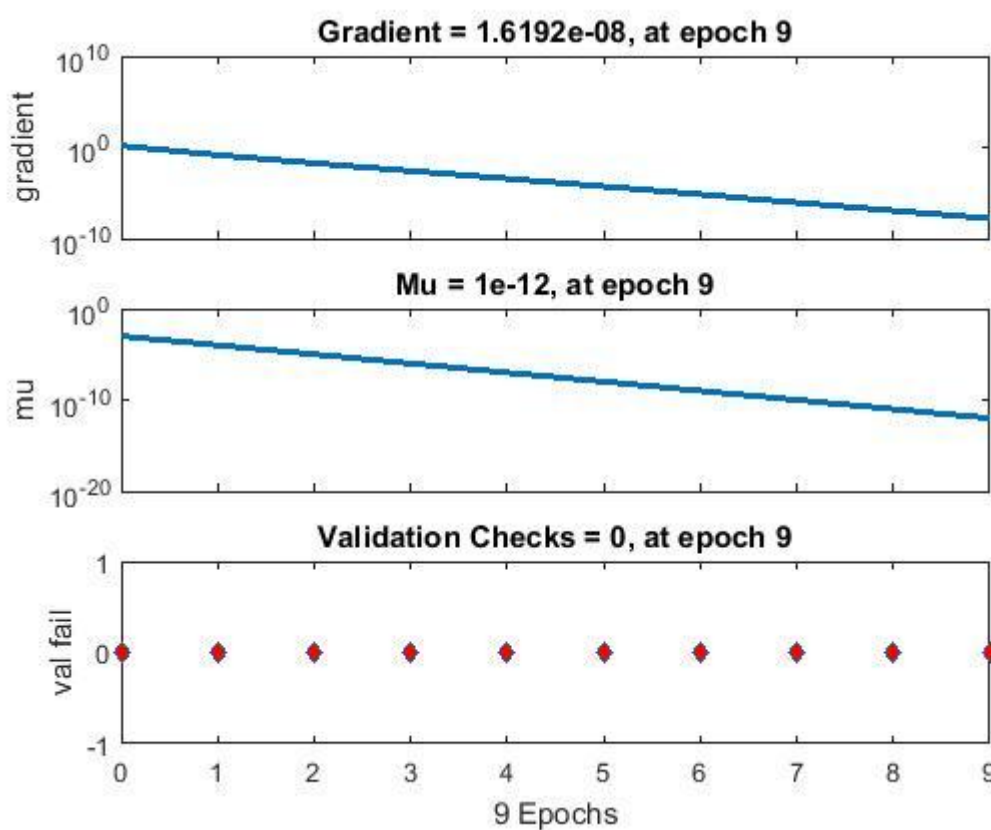


Figura 39 - Gráficos de gradiente, Mu e validação

### 3.2.3. Resultados

Após o treinamento da rede, foi obtido o vetor de saída chamado *network\_outputs*, que está identificado na tabela 1 como Resultado.

Imagem	Classe	Resultado	Precisão
Cenoura 1	2	-0,9976	99,76%
Cenoura 2	2	-1,0000	100%
Laranja 1	1	0,9999	99,9%
Laranja 2	1	1,0000	100%
Imagem nula	0	-1,0132	100%

Tabela 1 - Tabela de resultados

Observa-se que a rede neural conseguiu identificar e classificar as classes quase perfeitamente, com as laranjas apresentando a saída no valor 1, o reconhecimento, e as demais imagens com a saída no valor próximo a -1, o não reconhecimento. Na tentativa de corrigir essa pequena margem de erro, novos testes foram feitos.

Após um determinado número de testes, foi encontrada uma estrutura para a rede que obteve o êxito procurado pelo trabalho.

Esta segunda rede assemelha-se a antiga rede em relação a todos os algoritmos e funções utilizadas, porém, esta possui quatro camadas, com quinze neurônios nas três primeiras camadas escondidas e um para a última camada. Na figura 39 pode-se observar a ilustração da sua estrutura.

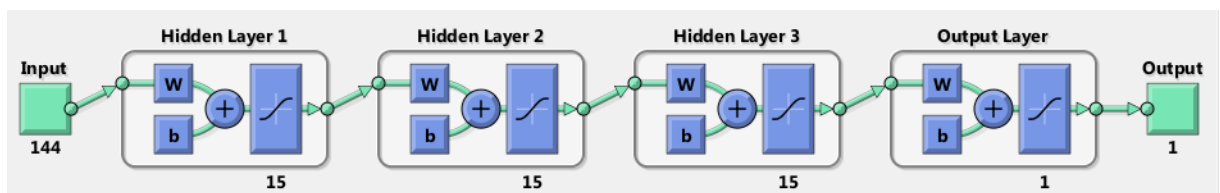


Figura 40 - Estrutura da segunda rede

Na figura 40 estão os parâmetros obtidos após o seu treinamento.

Progress				
Epoch:	0	10 iterations		1000
Time:		0:00:12		
Performance:	1.36	1.02e-09		0.00
Gradient:	13.4	1.55e-08		1.00e-07
Mu:	0.00100	1.00e-13		1.00e+10
Validation Checks:	0	0		6

Figura 41 - Tabela de progresso segunda rede

Obtendo então os seguintes resultados na tabela 2:



<b>Imagem</b>	<b>Classe</b>	<b>Resultado</b>	<b>Precisão</b>
Cenoura 1	2	-1,0000	100%
Cenoura 2	2	-1,0000	100%
Laranja 1	1	1,0000	100%
Laranja 2	1	1,0000	100%
Imagem nula	0	-1,0132	100%

Tabela 2 - Tabela de resultados da segunda rede

No Apêndice G deste trabalho pode ser encontrado um algoritmo desenvolvido no Matlab, cujo objetivo é determinar se a amostra foi reconhecida ou não pela RNA a partir dos resultados exibidos na Tabela 2.

### 3.3. EXTRAPOLANDO O CENÁRIO LARANJA

Nesta etapa, uma nova imagem foi atribuída ao trabalho com o intuito de testar a eficiência da rede sobre uma amostra que ainda não reconhece. Esta situação se caracteriza pela extrapolação da imagem com uma única laranja, para um cenário com outras frutas, além da própria laranja.

A figura 41 ilustra esse novo cenário.



Figura 42 - Laranja entre outras frutas – Laranja 3

Esta imagem foi escolhida por apresentar um cenário diferente do anterior. Aqui, uma laranja foi fotografada em meio a outras frutas, que traz a aplicabilidade da rede neural para um tipo de amostragem mais realista e sofisticado.

Após a aquisição desse novo modelo de imagem foi necessária a criação de um algoritmo que eliminasse as outras frutas e mantivesse apenas a laranja. A próxima seção trata exclusivamente deste algoritmo.

### **3.3.1. Algoritmo de detecção com fundo**

A rede neural está treinada para reconhecer um padrão, onde, o objeto a ser reconhecido forma uma matriz de zeros e uns que denota exatamente o seu formato, assim, a necessidade deste algoritmo dá-se ao fato de que esta nova imagem possui outros objetos ao seu redor que não se adequam ao padrão que foi apresentado no treinamento da rede aqui proposta. O objetivo deste algoritmo é detectar a laranja e eliminar qualquer outro tipo de objeto. O algoritmo que faz a detecção do objeto encontra-se no Apêndice B.

O algoritmo é composto por uma variável chamada imagem que tem a função de armazenar a matriz de pixels da imagem a ser tratada, no caso, a figura apresentada anteriormente com o nome de Laranja 3.

O método para detectar a laranja foi baseado nos pixels que a compõem, assim, um laço de repetição *for* foi instanciado para percorrer a imagem em busca dos pixels que se encontrassem dentro das seguintes condições:

Se o pixel vermelho na linha e na coluna *x* da imagem for maior que 210, o pixel verde na linha e na coluna *x* da imagem for maior que 31, o pixel azul na linha e na coluna *x* da imagem for maior que 0 e se o pixel vermelho na linha e na coluna *x* da imagem for menor que 255, o pixel verde na linha e na coluna *x* da imagem for menor que 221 e o pixel azul na linha e na coluna *x* da imagem for menor que 110.

O pixel que se encaixe nas requisições é imediatamente modificado, trocando seus valores de RGB para 0, ou seja, a cor preta. Já o pixel que for encontrado fora das condições é pintado de branco, tendo seus valores de RGB modificados para 255.

O resultado da aplicação do algoritmo na imagem está ilustrado na figura 42.

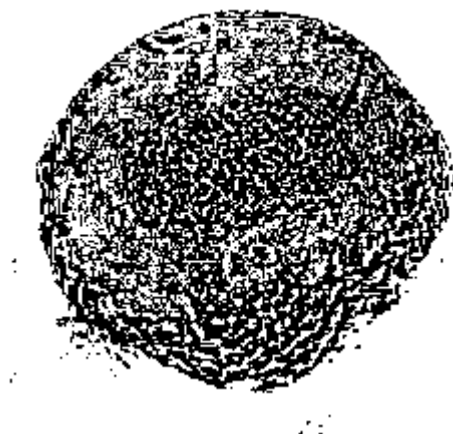


Figura 43 - Imagem após a aplicação do algoritmo de detecção com fundo

É possível observar que o fundo da imagem foi totalmente eliminado, deixando apenas o objeto no formato da laranja, assim, conclui-se que o algoritmo cumpriu com o seu principal objetivo e passamos para as próximas etapas.

### **3.3.2. Finalização do tratamento da imagem e entrada na rede**

Para a imagem servir de entrada para a RNA ainda foram necessários alguns ajustes.

A imagem foi redimensionada para  $12 \times 12$  e recortada para eliminar o espaço em branco desnecessário, deixando a imagem mais robusta e adequando as entradas anteriores da rede.

A partir daqui, o Algoritmo de Aquisição e Processamento de Imagem com Fundo, encontrado no Apêndice C do trabalho, foi executado e utilizou-se a imagem resultante do Algoritmo de Detecção com Fundo, que inicialmente é binarizada, e o resultado pode ser visto na figura 43.

```

true true true true false false false false true true true true
true true false false false false false false false false true true
true true false false false false false false false false true true
false false false false false false false false false false false
false false false false false false false false false false false
false false false false false false false false false false false
false false false false false false false false false false false
false false false false false false false false false false false
false false false false false false false false false false false
false false false false false false false false false false false
false false false false false false false false false false false
true true true true false false false false true true true true
true true true true false false false false true true true true

```

Figura 44 - Laranja 3 tratada

Neste ponto a matriz da imagem está pronta e adequada ao padrão ensinado para a rede neural, restando apenas transformá-la em um vetor linha. O algoritmo faz esta transformação e em seguida executa a simulação da RNA com o vetor resultante. O resultado foi plenamente satisfatório, a RNA identificou o padrão de uma laranja com uma precisão de 99,9 % como pode ser visto na tabela 3:

Imagem	Classe	Resultado	Precisão
Laranja 3	1	0,9999	99,9%

Tabela 3 – Tabela de resultados da simulação com a nova imagem

### 3.4. IMPLEMENTAÇÃO DO *PERCEPTRON* EM JAVA

As RNA's também estão presentes nas linguagens de programação de alto nível, onde é possível enxergar a lógica dando vida a essas complexas e fascinantes abstrações do cérebro humano.

Construiu-se uma implementação com a linguagem Java, a partir do algoritmo apresentado por SILVA, SPATTI e FLAUZINO (2010).

#### **3.4.1. Algoritmo *Perceptron* em Java - Treinamento**

O algoritmo estudado pode ser visualizado por completo no Apêndice D do trabalho. Ele possui como principal objetivo treinar a RNA com as amostras obtidas na seção 3.2.1, que são os vetores resultantes do tratamento das imagens Laranja 1, Laranja 2, Cenoura 1, Cenoura 2 e a matriz Imagem nula.

Os vetores das laranjas e da Imagem nula foram colocados em uma matriz para servir de entrada para a rede, onde, cada vetor representa uma coluna desta matriz, esta mesma modelagem também já foi realizada anteriormente no Matlab.

Um vetor que possui as saídas desejadas para cada uma das amostras foi criado, e, para armazenar os pesos, outro arranjo unidimensional também foi criado. O vetor de pesos foi inicializado com valores randômicos da divisão de 1.0 por números de 1 a 100, assim os pesos sempre alcançariam um número baixo, mas nunca o zero.

Em um arco de repetição é feita a somatória da multiplicação de cada peso pela sua respectiva amostra, esta somatória passa por uma função de transição, que testa se a mesma é maior ou igual a zero, caso a resposta seja verdadeira, o retorno da função será no valor 1, caso contrário o retorno será -1.

O resultado é conferido com o vetor de saída desejada, se os dois se igualarem o treinamento para a amostra é finalizado, caso contrário, uma nova linha de aprendizado é definida, multiplicando a anterior pela saída desejada para a amostra, menos o retorno da função de transição.

Os pesos são atualizados somando o seu valor anterior ao novo aprendizado e multiplicando pelas suas respectivas amostras.

Após obter o completo treinamento da rede, passa-se para a etapa de simulação, que é apresentada na próxima seção.

### 3.4.2. Algoritmo *Perceptron* em Java – Simulação

Este algoritmo visa simular a rede neural já treinada no algoritmo apresentado na seção anterior. Pode ser encontrado no Apêndice E do projeto.

A amostra utilizada é derivada do resultado do tratamento da imagem da laranja com fundo encontrada na seção 3.3.2.

O vetor binário da imagem foi submetido a um método que realizou a somatória da multiplicação dos pesos pelas amostras. Estes pesos vieram do algoritmo de treinamento da rede e já estão adequados para reconhecer os padrões das amostras.

A soma é dada como entrada para a função de transição, que verifica se o resultado é 1 (Laranja) ou -1 (amostra não identificada).

Na próxima seção estão retratados os resultados do treinamento e da simulação da RNA.

### 3.4.3. Resultados do treinamento e da simulação do *Perceptron* em Java

O número de épocas necessárias para atingir os pesos ideais no algoritmo de treinamento foi quatro, o vetor dos pesos pode ser visto no Apêndice F do trabalho.

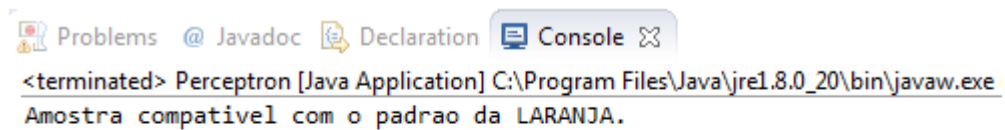
A simulação da rede obteve sucesso, pois foi capaz de identificar a amostra e classificá-la como uma laranja. Pode-se observar a matriz que passará pela simulação na figura 44 e em seguida, na figura 45 está a demonstração do resultado exibido no console da IDE de desenvolvimento do algoritmo, após a sua execução.

```

...
double[] vetorAmostra = new double[] {
    1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0,
    1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0,
    1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0,
    1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0,
    0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
    0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
    0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
    0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
    1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0,
    1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0,
    1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0,
    1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0
}; //*/// LARANJA.

```

Figura 45 - Vetor Laranja 3



```

<terminated> Perceptron [Java Application] C:\Program Files\Java\jre1.8.0_20\bin\javaw.exe
Amostra compativel com o padrao da LARANJA.

```

Figura 46 - Resultado exibido no console da IDE

Para consolidar a confiança na RNA foram realizadas mais duas simulações, a primeira com uma imagem representando uma cenoura e a segunda com a imagem nula.

O vetor utilizado para a simulação com a cenoura está na figura 47 e o resultado pode ser visto na figura 48.

```
double[] vetorAmostra = new double[] {
    1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0,
    1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0,
    1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 1.0,
    1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0,
    1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0,
    1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0,
    1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0,
    1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0,
    1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
    1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
    0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
    1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0
}; //*/// CENOURA
```

Figura 47 - Vetor Cenoura

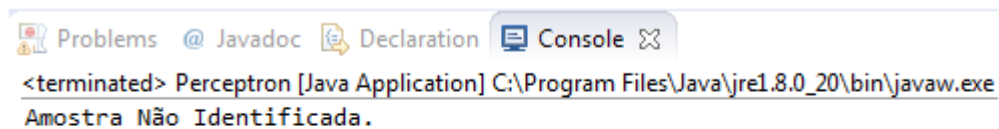


Figura 48 - Resultado da simulação com a amostra da imagem Cenoura 2

Após a simulação com a imagem nula foi obtido um resultado que pode ser visto na figura 50 e o vetor utilizado está localizado na figura 49.

```
double[] vetorAmostra = new double[] {
    1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0,
    1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
    1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
    1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
    1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
    1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
    1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
    1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
    1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
    1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
    1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
    1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
    1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
    1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0
}; // NULO
```

Figura 49 - Vetor Imagem nula



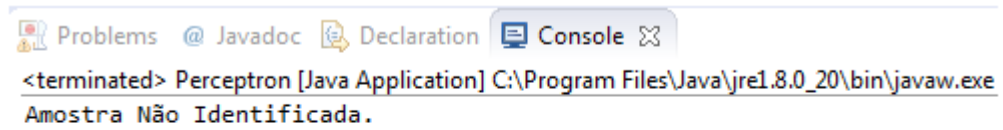


Figura 50 - Resultado da simulação com a Imagem nula

## 4. CONCLUSÃO

Este trabalho propôs o desenvolvimento de uma aplicação capaz de reconhecer um objeto dentro de uma imagem utilizando os conceitos de Redes Neurais Artificiais. Com a utilização do ambiente computacional técnico-científico Matlab foi possível alcançar este objetivo.

A rede neural do tipo *Perceptron feedforward Backpropagation*, também desenvolvida neste trabalho, provou sua eficiência após a segunda estrutura, que obteve 100% de precisão no reconhecimento dos padrões das amostras.

Desta forma o projeto pôde evoluir e trabalhar com um tipo de amostragem mais complexa, onde era apresentada uma imagem com o objeto a ser reconhecido sobre um fundo contendo outros objetos. Para esta nova imagem foi necessário o desenvolvimento de um algoritmo que extraísse apenas o que era essencial para a entrada da RNA, retirando qualquer tipo de ruído da amostra.

Após obter os resultados do tratamento, foi efetuada a simulação da rede sobre a nova amostra, que alcançou uma precisão de 99% no reconhecimento do objeto apresentado.

Como as amostras se mostraram uma excelente base de dados para trabalhos com uma RNA, dois algoritmos que implementam uma rede neural em Java foram utilizados para processá-las e reconhecê-las. Os algoritmos apresentaram alguma simplicidade estrutural, além de extrema eficiência e rapidez, alcançando um treinamento em apenas quatro ciclos e identificando as amostras perfeitamente em questão de milissegundos.

É possível concluir que os algoritmos desenvolvidos cumpriram com seus propósitos, proporcionando diversos métodos e conceitos envolvendo as RNAs e disponibilizando um material claro e objetivo para os interessados.

Como trabalho futuro pretende-se ampliar as bases de treinamento, com padrões mais complexos e maior variedade. Pretende-se extrair o máximo de características das amostras para evitar falhas, já que o próximo passo é aplicar esta metodologia em componentes do mundo real, por exemplo, uma máquina localizada em uma linha de produção que reconheça e separe o produto bom do produto com defeito aparente. É de interesse, também, a criação de uma interface para os algoritmos

apresentados neste projeto, e trabalhando com uma câmera acoplada, esta tarefa já seria possível.

## REFERÊNCIAS

- ALVES, Natália M. E. **A equação do calor aplicada ao processamento de imagens**. 2013. Monografia de Graduação – Instituto de Ciências Exatas - Departamento de Matemática – Universidade Federal de Minas Gerais, 2013.
- AUGUSTO, F. R. **Localização e Reconhecimento de Placas de Sinalização Utilizando um Mecanismo de Atenção Visual e Redes Neurais Artificiais**. Universidade Federal de Campina Grande, 2002.
- ARAÚJO, MILTON ALUÍSIO G.. **Previsão de demanda de energia elétrica por meio de redes neurais artificiais**. 2005. Dissertação de Mestrado – Programa de Pós-Graduação em Administração – Universidade Federal do Rio Grande do Sul, Rio Grande do Sul, Porto Alegre, 2005.
- AZEVEDO, F. M.; BRASIL, L. M.; OLIVEIRA, R. C. L. **Redes Neurais com Aplicações em Controle e em Sistemas Especialistas**. [S.I.]: VISUAL BOOKS, 2000. ISBN 8575020056.
- BABINI, Maurizio. **Reconhecimento de Padrões Lexicais por meio de Redes Neurais**. 2006. Dissertação de Mestrado - Engenharia Elétrica (Área de Concentração: Automação) - Faculdade de Engenharia de Ilha Solteira, UNESP – Câmpus de Ilha Solteira, Ilha Solteira, 2006.
- BALLONE, G. **Neurônios e Neurotransmissores**. Psiquweb, 2008. Disponível em: <<http://www.psiqweb.med.br/site/DefaultLimp.aspx?area=NO/LerNoticia&idNoticia=290>>. Acesso em: 19 fev. 2015.
- BARCAL, Maria Carolina Stockler; SILVEIRA, Tiago Redondo de Siqueira; MAGINI, Marcio. Treinamento de Redes Neurais Artificiais: O algoritmo *Backpropagation*. In: INIC 2005 - IX ENCONTRO LATINO AMERICANO DE INICIAÇÃO CIENTÍFICA, 9, 2005, Jacareí, Brasil. **Anais do IX Encontro Latino Americano de Iniciação Científica**, 2005, p. 46-49.
- BARRETO, J. B. **Introdução às Redes Neurais Artificiais**. 2002. 57p. Dissertação - Departamento de Informática e de Estatística - Laboratório de Conexão e Ciências Cognitivas UFSC, Santa Catarina, Florianópolis, 2002.
- BRAGA, Antônio P., LUDERMIR, Teresa, CARVALHO, André C.P.L.F. **Redes Neurais Artificiais: teoria e aplicações**. Rio de Janeiro: LTC, 2000.
- BRAGA, A., CARVALHO, A., e LUDERMIR, T.. **Redes Neurais Artificiais: Teoria e Aplicações**. 2ª edição. Rio de Janeiro: LTC, 2007.
- CARDON, André; MÜLLER, Daniel Nehme. **Introdução Às Redes Neurais Artificiais**. 1994. 31p. Curso de Pós-Graduação em Ciência da Computação - Instituto de Informática - Universidade Federal do Rio Grande do Sul, Rio Grande do Sul, Porto Alegre, 1994.

COWAN, J.D., 1990. **Neural networks: The early days**, *Advances in Neural Information Processing Systems*, vol.2, pp. 828-842, San Mateo, CA: Morgan Kaufman.

DELGADO, Cynthia. **Finding the Evolution in Medicine**. Artigo em NIH Record (National Institute of Health). Disponível em: <[http://nihrecord.nih.gov/newsletters/2006/07\\_28\\_2006/story03.htm](http://nihrecord.nih.gov/newsletters/2006/07_28_2006/story03.htm)>. Acesso em: 10 out. 2014.

DEMUTH, H.; BEALE, M. **Neural Network Toolbox User's Guide**. Versão 4.0. The Mathworks Inc, 1998.

ELMAN, Jeffrey L.. **Finding structure in time**. Cognitive Science Volume 14, Issue 2, Pages 179-211, 1990.

F.A.C. AZEVEDO e COLABORADORES (2009). **Equal numbers of neuronal and non-neuronal cells make the human brain an isometrically scaled-up primate brain**. *Journal of Comparative Neurology* vol. 513: pp. 532-541.

FLEMMING, Diva Marília; GONÇALVES, Mírian Buss. **Cálculo A: funções, limite, derivação, integração**. Makron Books, 2007.

GONZALEZ, R.C. & WOODS, R.E. **Processamento de Imagens Digitais**. São Paulo: Edgard Blücher Ltda., 2000.

HAGAN, M. T.; MENHAJ, M. B. **Training feedforward networks with the Marquardt algorithm**. *IEEE Transactions on Neural Networks*, 5 (6): 989-993, 1994.

HAYKIN, S. **Redes Neurais: Princípios e prática**. 2. ed. Tradução de Paulo Martins Angel. Porto Alegre: Editora Bookman, 2001.

JESUS, Willian Moldenhauer. **Uso de redes neurais artificiais auto regressivas para estimar a capacidade de refrigeração de compressores através de dados de regime transiente**. 2013. 58 p.. Trabalho de Conclusão de Curso (Graduação em Engenharia da Mobilidade) - Universidade Federal de Santa Catarina, Campus Joinville. Florianópolis, SC, 2013.

KHATCHATOURIAN, Oleg; PADILHA, Fábio RR. **Reconhecimento de variedades de soja por meio do processamento de imagens digitais usando redes neurais artificiais**. *Eng Agric Jaboticabal*, v. 28, n. 4, p. 759-69, 2008.

KRIESEL, D.. **A Brief Introduction to Neural Networks (ZETA2-EN)**. 2005. Disponível em: <[http://www.dkriesel.com/en/science/neural\\_networks](http://www.dkriesel.com/en/science/neural_networks)>. Acesso em: 10 de fev. 2015

KOHONEN, Teuvo. An adaptive associative memory principle. *IEEE Transactions on Computers*, n. 4, p. 444-445, 1974.

LUDWIG JR., O; COSTA, Eduard Montgomery M. **Redes Neurais: Fundamentos e Aplicações com Programas em C**. Rio de Janeiro: Editora Ciência Moderna Ltda. 2007.

MAIA, Renato Dourado. **Inteligência Computacional – Redes Neurais – Adaline**. 2012. 36p. Faculdade de Ciências e Tecnologia de Montes Claros - Fundação Educacional Montes Claros, Minas Gerais, Montes Claros, 2012.

MATSUMOTO, E. Y. **MATLAB 6.5: Fundamentos de programação**. São Paulo: Erica. 2002.

MATSUNAGA, Victoria Yukie. **Curso de redes neurais utilizando o Matlab**. Belém do Pará, 2012.

MCCULLOCH, W. S., PITTS, W. **A logical calculus of the ideas immanent in nervous activity**. In: Bulletin of Mathematical Biophysics, 1943.

MOREIRA, J., COSTA, L. F. **Neural-based Color Image Segmentation and Classification using Self-organizing Maps**. Anais do IX SIBGRAPI, pp. 47-54, 1996.

OSÓRIO, Fernando S.; BITTENCOURT, João Ricardo; OSÓRIO, Fernando Santos. **Sistemas Inteligentes baseados em redes neurais artificiais aplicados ao processamento de imagens**. In: Workshop de Inteligência Artificial. Vol. 1. 2000.

PADUA, A. et al. **Redes Neurais Artificiais-Teoria e Aplicações**. Rio de Janeiro: Editora LTC, 2000.

PIAZENTIN, Denis Renato de Moraes. **Troca de Chaves Criptográficas utilizando Criptografia Neural**. Trabalho de conclusão de curso (Graduação em Ciência da Computação)– Curso de Ciência da Computação, Fundação de Ensino “Eurípides Soares da Rocha”, mantenedora do Centro Universitário Eurípides de Marília – UNIVEM, Marília, 2011.

PURVES, Daleet al. **Neurociências**. 4. ed. Porto Alegre: Editora Artmed, 2010.

ROSA, João Luís Garcia. **SCC-5809 Redes Neurais. Slides e listas de exercícios. Capítulo 5 - Perceptron Multicamadas**. Programa de Pós-Graduação em Ciência de Computação e Matemática Computacional – Departamento de Ciências de Computação – Universidade de São Paulo, São Carlos, 2011.

RUSSELL, S., NORVIG, P. **Artificial intelligence: a modern approach**. [S.l.]: Prentice Hall, 2010. (Prentice Hall series in artificial intelligence). ISBN 9780136042594.

SILVA, Ivan Nunes da; SPATTI, Danilo Hernane; FLAUZINO, Rogério Andrade. **Redes Neurais Artificiais para engenharia e ciências aplicadas** – São Paulo: Artliber, 2010.

SILVA, Leandro Nunes de Castro. **Análise e síntese de estratégias de aprendizado para Redes Neurais Artificiais**. Monografia de Graduação – Departamento de Engenharia de Computação e Automação Industrial – Universidade Estadual de Campinas – Faculdade de Engenharia Elétrica e de Computação, São Paulo, Campinas, 1998.

SILVA, Renato Ramos. **Reconhecimento de imagens digitais utilizando redes neurais artificiais**. 2005. Monografia de Graduação – Departamento de Ciência da Computação –Universidade Federal de Lavras, Minas Gerais, Lavras, 2005.

SHEPHERD, G. M., KOCH. 1990. “Introduction to synaptic circuits,” in **The Synaptic Organization of the Brain**, G.M. Shepherd , ed., pp.3-31. New York: Oxford University Press.

TATIBANA, Cassia Yuri; KAETSU, Deisi Yuri. **Uma introdução às redes neurais**. Departamento de Informática (DIN) da Universidade Estadual de Maringá (UEM). Disponível em: <http://www.din.uem.br/ia/neurais/#links>. Acesso em: 12 out. 2014.

WASSERMAN, P. **Neural Computing Theory and Practice**. Van Nostrand Rheinhold, New York, 1989.

## APÊNDICE A - Algoritmo de Aquisição e Processamento de Imagem

```
% Redes Neurais Artificiais para a identificação de objetos contidos em
% imagens

% Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de
% Ensino Superior de Assis, como requisito do Curso de Graduação

% Gabriel Souza da Silva

% Algoritmo Principal

% Aquisição e processamento das imagens

% Leitura das imagens
im1 = imread('laranja3.png')
im2 = imread('laranja4.png')
im3 = imread('cenoura4.png')
im4 = imread('cenoura5.png')

% Binarização das imagens
im1p = im2bw(im1, 0.9);
im2p = im2bw(im2, 0.9);
im3p = im2bw(im3, 0.9);
im4p = im2bw(im4, 0.9);

% Transformação das imagens em vetores coluna
im1v = im1p(:)
im2v = im2p(:)
im3v = im3p(:)
im4v = im4p(:)

% Criação da matriz da Imagem nula
Im6p = im1p;
Im6p = ones(12);
Im6v = im6p(:)

M = [im1v,im2v,im6v,im6v];% Criação da matriz de entrada da rede
Result = [1,1,-1,-1];% Criação do vetor de saída desejada da rede

% Arquitetura da Rede

numero_entradas = 1; % numero de entradas
numero_intermediaria = 10; % numero de neuronios na camada intermediária
numero_saida = 1; % numero de neuronios na camada de saída
tipo_intermediaria = 'tansig'; % tipo de neuronios da camada intermediária
tipo_saida = 'tansig'; % tipo de neuronios da camada de saída
tipo_treinamento = 'learngdm'; % tipo de treinamento da rede

% Criação da Rede Neural

redeneural = newff(M,
Result,[numero_intermediaria,numero_saida],[tipo_intermediaria,
tipo_saida],'trainlm', tipo_treinamento);
```



```
% A função newff cria uma rede do tipo Backpropagation

% Configuração dos parâmetros para o treinamento da Rede Neural

redeneural.trainParam.show = 25; % Atualização da tela
redeneural.trainParam.lr = 0.2; % Taxa de aprendizado
redeneural.trainParam.mc = 0.9; % Taxa de momentum
redeneural.trainParam.goal = 0; % Erro final desejado
redeneural.trainParam.epochs = 1000; % Número de épocas

% Treinamento da Rede Neural

redeneural = train(redeneural, M, Result);

% Simulação da Rede Neural

saida = sim(redeneural, im1v);
```

## APÊNDICE B - Algoritmo de Detecção de Imagem com Fundo

```
% Redes Neurais Artificiais para a identificação de objetos contidos em
% imagens

% Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de
% Ensino Superior de Assis, como requisito do Curso de Graduação

% Gabriel Souza da Silva

% Algoritmo de detecção de Imagem com fundo

% Leitura da imagem
imagem = imread('laranjafr4.png');

% Armazenamento do numero de linhas e colunas da imagem
linhasTotaisImagem = size(imagem, 1);
colunasTotaisImagem = size(imagem, 2);

% Loop que detecta a laranja na imagem a partir de um
% classificador de pixels em RGB, tornando preto(0) onde
% existe a laranja e branco(255) onde não existe a laranja

for i=1:linhasTotaisImagem
    for j=1:colunasTotaisImagem
        laranja = false;
        if (imagem(i,j,1) > 210 && imagem(i,j,2) > 31 && imagem(i,j,3)
            > 0)
            if (imagem(i,j,1) < 255 && imagem(i,j,2) < 221 &&
                imagem(i,j,3) < 110)
                laranja = true;
                imagem(i,j,1) = 0;
                imagem(i,j,2) = 0;
                imagem(i,j,3) = 0;
            end
        end
        if laranja == false
            imagem(i,j,1) = 255;
            imagem(i,j,2) = 255;
            imagem(i,j,3) = 255;
        end
    end
end
```

## APÊNDICE C - Algoritmo de Aquisição e Processamento de Imagem com Fundo

```
% Redes Neurais Artificiais para a identificação de objetos contidos em  
% imagens  
  
% Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de  
% Ensino Superior de Assis, como requisito do Curso de Graduação  
  
% Gabriel Souza da Silva  
  
% Programa de Aquisição e Processamento de Imagem com Fundo  
  
% Neste programa será utilizada a imagem resultante do processamento do  
% programa de detecção e salva no workspace do projeto com o nome de  
% imagem  
  
% Binarização da imagem  
im5p = im2bw(imagem, 0.9);  
  
% Transformação da imagem em um vetor coluna  
im5v = im5p(:)  
  
% Simulação da Rede Neural com a nova imagem  
  
saida = sim(redeneural, im5p);
```





```

        {1.0,    0.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {1.0,    0.0,    1.0,    1.0},
        {1.0,    1.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {1.0,    1.0,    1.0,    1.0},
        {1.0,    1.0,    1.0,    1.0},
        {1.0,    1.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {1.0,    1.0,    1.0,    1.0},
        {1.0,    1.0,    1.0,    1.0},
        {1.0,    1.0,    1.0,    1.0},
        {1.0,    1.0,    1.0,    1.0},
        {1.0,    1.0,    1.0,    1.0},
        {1.0,    1.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {0.0,    0.0,    1.0,    1.0},
        {1.0,    0.0,    1.0,    1.0},
        {1.0,    1.0,    1.0,    1.0},
        {1.0,    1.0,    1.0,    1.0},
        {1.0,    1.0,    1.0,    1.0}
    };

    double[] saidasdesejadas = new double[] { 1.0, 1.0, -1.0, -1.0 };
    double[] w = new double[144];

    // Definição de um valor inicial aleatório para os pesos (w)
    for (int i = 0; i < w.length; i++) {
        w[i] = 1.0 / (random.nextInt(100) + 1.0);
    }

```

```

double linhaAprendizado = 0.05;
int epocas = 0;
String erro;

do {
    erro = "inexiste";
    for (int k = 0; k < 4; k++) {
        double wsoma = 0.0;
        for (int i = 0; i < 144; i++) {
            wsoma += w[i] * amostrasTreinamento[i][k];
        }

        int x = funcaoTransicao(wsoma);

        // Teste para saber se o resultado do treino condiz com
        // o resultado desejado
        if (x != saidasDesejadas[k]) {

            double novoAprendizado = linhaAprendizado *
(saidasDesejadas[k] - x);
            for (int i = 0; i < w.length; i++) {
                w[i] = w[i] + novoAprendizado *
amostrasTreinamento[i][k];
            }

            erro = "existe";
        }
    }
    epocas++;
} while (erro.equals("existe"));

for (int i = 0; i < w.length; i++) {
    System.out.println("Peso treinado = " + i + w[i]);
    VetorPesos += w[i] + (((i + 1) != w.length) ? ", " : "");
}
System.out.println(VetorPesos);
System.out.println("Epocas - " + epocas);
}

private static int funcaoTransicao(double wsoma) {
    return (wsoma >= 0.0) ? 1 : -1;
}
}

```

## APÊNDICE E - Algoritmo *Perceptron* em Java Simulação

[illegible]



```

double[] w = new double[] {
    0.2333333333333333, -0.06774193548387097, 0.024999999999999994,
    -0.08717948717948719, -0.0868421052631579, -0.08989898989898991, -
    0.08936170212765958, -0.08913043478260871, -0.07297297297297298, -
    0.0878048780487805, -0.04736842105263159, 0.9, -0.08969072164948454,
    0.0, -0.08750000000000001, -0.0878048780487805, -0.08913043478260871,
    -0.08823529411764706, -0.04736842105263159, -0.0642857142857143,
    0.06666666666666665, -0.07142857142857144, 0.9, -0.07916666666666668,
    -0.08, -0.08305084745762713, -0.08717948717948719, 0.4, -
    0.06774193548387097, -0.08969072164948454, -0.009090909090909094, -
    0.08181818181818182, -0.08876404494382023, -0.07872340425531915, -
    0.08529411764705883, -0.037500000000000006, -0.08333333333333334, -
    0.05652173913043479, -0.0878048780487805, -0.08412698412698413, -
    0.04736842105263159, -0.08648648648648649, -0.08969072164948454, -
    0.08701298701298701, 0.01111111111111111, -0.08305084745762713, 0.4, -
    0.06969696969696967, -0.07916666666666668, -0.08717948717948719, -
    0.08571428571428572, -0.07560975609756099, -0.08529411764705883, -
    0.08850574712643679, 0.4, -0.08181818181818182, -0.07959183673469389,
    -0.08484848484848485, -0.07727272727272727, -0.05652173913043479,
    0.01111111111111111, -0.08648648648648649, -0.07727272727272727, -
    0.08611111111111111, -0.06296296296296297, -0.08305084745762713, -
    0.06875, -0.07500000000000001, -0.07297297297297298, -
    0.08360655737704918, -0.0736842105263158, -0.08979591836734695, -
    0.08275862068965517, -0.08901098901098901, -0.04736842105263159,
    0.2333333333333333, -0.08275862068965517, -0.07500000000000001, 0.1,
    0.9, -0.06666666666666668, -0.07297297297297298, -0.0782608695652174,
    0.9, -0.08529411764705883, -0.08750000000000001, -0.0696969696969697,
    -0.08412698412698413, -0.084375, 0.9, 0.2333333333333333, -
    0.08958333333333333, -0.04736842105263159, -0.08913043478260871, -
    0.06296296296296297, -0.08214285714285716, -0.07297297297297298, -
    0.08888888888888889, 0.4, -0.06666666666666668, -0.02857142857142858,
    0.01111111111111111, -0.08113207547169812, -0.08360655737704918, -
    0.0855072463768116, -0.07727272727272727, -0.0411764705882353, -
    0.01666666666666667, 0.1, -0.03333333333333334, -
    0.07959183673469389, 0.4, -0.08360655737704918, -0.08876404494382023,
    -0.01666666666666667, -0.08876404494382023, -0.0803921568627451, -
    0.08611111111111111, -0.0736842105263158, -0.0878048780487805, -
    0.04444444444444445, -0.08648648648648649, 0.9, -0.08571428571428572,
    -0.08913043478260871, -0.0782608695652174, -0.07560975609756099, -
    0.08412698412698413, 0.2333333333333333, -0.0863013698630137, -
    0.08214285714285716, -0.07297297297297298, -0.09000000000000001, -
    0.08412698412698413, -0.08611111111111111, -0.05, -
    0.07058823529411765, -0.08823529411764706, -0.08850574712643679, -
    0.09000000000000001, -0.08924731182795699, -0.0782608695652174, -
    0.08979591836734695, -0.07222222222222223};

```

```

double wsoma = 0.0;
for (int k = 0; k < vetorAmostra.length; k++) {
    wsoma += w[k] * vetorAmostra[k];
}

```

```

// Função de verificação
int x = funcaotransicao(wsoma);

```

```

if (x == -1) {

```

```
        System.out.println("Amostra Não Identificada.");
    } else if (x == 1) {
        System.out.println("Amostra compativel com o padrao da
LARANJA.");
    }
}

private static int funcaotransicao(double wsoma) {
    return (wsoma >= 0.0) ? 1 : -1;
}
}
```

## APÊNDICE F - Vetor de pesos treinados

```
double[] w = new double[] {
    0.03225806451612903, 0.011235955056179775, 0.009999999999999995,
    0.016949152542372878, -0.06551724137931035, -0.08734177215189874,
    -0.08305084745762713, -0.07142857142857144, 0.11111111111111111,
    0.1, 0.010204081632653059, 0.014084507042253516,
    0.018867924528301883, 0.010989010989010992, -0.05238095238095239,
    -0.02857142857142858, -0.08305084745762713, -0.08275862068965517,
    -0.08979591836734695, -0.08936170212765958, -0.06875,
    -0.0868421052631579, 0.07692307692307693, 0.018518518518518517,
    0.018181818181818188, -0.009090909090909094, -0.0764418604651163,
    -0.07777777777777778, -0.08888888888888889, -0.08979591836734695,
    -0.07142857142857144, -0.0736842105263158, -0.08989898989898991,
    0.047619047619047616, 1.0, 0.11333333333333334,
    0.045454545454545456, -0.08823529411764706, -0.060000000000000005,
    0.23333333333333333, -0.07872340425531915, -0.07222222222222223,
    0.03225806451612903, 0.021739130434782608, 0.014705882352941176,
    0.09090909090909091, 0.010752688172043012, 0.021276595744680854,
    0.04444444444444445, -0.0855072463768116, -0.05,
    -0.0411764705882353, 0.02702702702702703, 0.012987012987012988,
    0.030303030303030304, 0.0136986301369863, 0.11111111111111111,
    0.02040816326530612, -0.01666666666666667, -0.08823529411764706,
    -0.08666666666666667, -0.08969072164948454, 0.011904761904761904,
    0.011111111111111112, 0.047619047619047616, 0.012048192771084338,
    0.043478260869565216, 0.02857142857142857, -0.08734177215189874,
    -0.07959183673469389, -0.08507462686567165, -0.05238095238095239,
    -0.0803921568627451, 0.016129032258064516, 0.015151515151515152,
    0.025, 0.041666666666666664, 0.010638297872340425,
    -0.05652173913043479, -0.07142857142857144, -0.08412698412698413,
    -0.037500000000000006, -0.08571428571428572, -0.08611111111111111,
    0.03225806451612903, 0.011764705882352941, 0.013333333333333334,
    0.022727272727272728, 0.02564102564102564, -0.04736842105263159,
    -0.05454545454545455, -0.08979591836734695, -0.08,
    -0.0761904761904762, -0.08305084745762713, -0.0803921568627451,
    0.0196078431372549, 0.022727272727272728, 0.029411764705882353,
    -0.08701298701298701, -0.08795180722891567, -0.07872340425531915,
    -0.08275862068965517, -0.06551724137931035, -0.07560975609756099,
    -0.08936170212765958, -0.0642857142857143, 0.013157894736842105,
    0.010416666666666671, -0.08876404494382023, -0.023076923076923078,
    -0.0782608695652174, -0.08958333333333333, -0.08947368421052632,
    -0.05, -0.08888888888888889, -0.08901098901098901,
    -0.06774193548387097, -0.08181818181818182, 0.1,
    0.015151515151515152, 0.03225806451612903, -0.08876404494382023,
    -0.08979591836734695, -0.0696969696969697, -0.037500000000000006,
    -0.08571428571428572, -0.06551724137931035, -0.05652173913043479,
    -0.08936170212765958, 0.045454545454545456, 0.0144927536231884,
    0.02857142857142857, 0.2, 0.016666666666666663,
    0.06666666666666667, 0.9, -0.0736842105263158,
    -0.0868421052631579, -0.04444444444444445, 0.02439024390243902,
    0.012048192771084335, 0.019230769230769232, 0.0625 };
```

## APÊNDICE G – Algoritmo para determinar amostras no Matlab

```
% Redes Neurais Artificiais para a identificação de objetos contidos em
% imagens

% Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de
% Ensino Superior de Assis, como requisito do Curso de Graduação

% Gabriel Souza da Silva

% Programa para determinar amostras

saidanova = sim(novaredeneural, im6v);
saidanova2 = sim(novaredeneural, im1v);
if (saidanova >= 0)
    fprintf('Amostra reconhecida\n');
    fprintf('Resultado: %d', saidanova);
end
if (saidanova < 0)
    fprintf('Amostra não reconhecida\n');
    fprintf('Resultado: %d\n', saidanova);
end

if (saidanova2 >= 0)
    fprintf('Amostra reconhecida\n');
    fprintf('Resultado: %d', saidanova2);
end
if (saidanova2 < 0)
    fprintf('Amostra não reconhecida\n');
    fprintf('Resultado: %d\n', saidanova2);
end
```