FUNDAÇÃO DE ENSINO “EURÍPIDES SOARES DA ROCHA”
CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA – UNIVEM CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

LUIS FERNANDO MARTINS CARLOS JUNIOR

RECONHECIMENTO FACIAL UTILIZANDO REDES NEURAIS

MARÍLIA
2011

LUIS FERNANDO MARTINS CARLOS JUNIOR

RECONHECIMENTO FACIAL UTILIZANDO REDES NEURAIS

Trabalho de Curso apresentado ao Curso de Bacharelado em Ciência da Computação da Fundação de Ensino "Eurípides Soares da Rocha", mantenedora do Centro Universitário Eurípides de Marília – UNIVEM, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador:
Prof. Me. Rodolfo Barros Chiaramonte

MARÍLIA
2011

Carlos Jr., Luis Fernando Martins

Reconhecimento Facial Utilizando Redes Neurais / Luis Fernando Martins Carlos Junior; orientador: Rodolfo Barros Chiaramonte. Marília, SP:[s.n], 2011 52 f.

Trabalho 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.

1. Reconhecimento Facial 2. Redes Neurais

CDD:006.32

CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO – AVALIAÇÃO FINAL

Luís Fernando Martins Carlos Júnior

RECONHECIMENTO FACIAL UTILIZANDO REDES NEURAIS

Banca examinadora da monografia apresentada ao Curso de Bacharelado em Ciência da Computação do UNIVEM/F.E.E.S.R., para obtenção do Título de Bacharel em Ciência da Computação.

Nota: 9 (nove)

Orientador: Rodolfo Barros Chiaramonte

1°. Examinador: Mauricio Duarte

2°. Examinador: Leonardo Castro Botega

Marília, 29 de novembro de 2011.

Dedico este trabalho a duas pessoas que foram de extrema importância durante minha trajetória, sem as quais eu não teria conseguido chegar até aqui: aos meus pais Luis e Maria que sempre me incentivaram e acreditaram no meus sonhos.

Agradecimentos

Agradeço ao Centro Universitário Eurípedes de Marília (UNIVEM), aos diretores, professores e funcionários pelo ótimo trabalho junto a instituição, que contribuiu positivamente para a conclusão do meu trabalho.

Agradeço ao meu professor orientador Prof. Me. Rodolfo Barros Chiaramonte, por me orientar durante o trabalho, por seus conselhos, dicas, críticas e sugestões, que vieram a enriquecer o trabalho.

Aos amigos e colegas de classe Denis, Hellen, Jonathan e Raphael, por sua ajuda, seja esta na forma de conselhos, dicas, incentivos ou apoio moral.

Agradeço a minha família, meus pais Luis e Maria, minha irmã Juliana, meu irmão João, minha avó Yolanda, que me incentivaram e apoiram, me motivando a alcançar minha meta.

Agradeço também aos amigos Cristiano e Robinho pelo apoio moral e incentivo que me motivaram a não me desviar do meu objeto e acreditar que conseguiria alcança-ló.

"Aquele que se empenha a resolver as dificuldades resolve-as antes que elas surjam. Aquele que se ultrapassa a vencer os inimigos triunfa antes que as suas ameaças se concretizem"
Sun Tzu

"NO PAINS, NO GAINS.
If little labour, little are our gains:
Man's fortunes are according to his pains."
Robert Herrick

Carlos Jr., Luis Fernando Martins **Reconhecimento Facial Utilizando Redes Neurais.** 2011 52 f. Trabalho 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.

RESUMO

O Reconhecimento Automático de faces é considerado uma tarefa bem complexa, pois é influenciado por diversos fatores como variação de iluminação, ruídos na imagem, expressão facial, óculos, mudança nos cabelos, barba ou bigode e pose. Devido as dificuldades comumente encontradas na área surgiram diversas abordagens diferentes para a resolução do problema do reconhecimento facial. Este trabalho apresenta uma abordagem para a resolução do problema de reconhecimento facial, utilizando uma rede neural do tipo MLP (Multilayer Perceptron) treinada com o algoritmo Rprop. O objetivo deste trabalho é demonstrar a eficiência de redes do tipo MLP aplicada no reconhecimento de faces. Para tal foi implementado um sistema que deve identificar indivíduos a partir de fotografias da face, fornecidas pelo banco de faces ORL(Olivetti Research Laboratory). Os testes foram realizados considerando variações na quantidade de camadas intermediárias(ocultas), no número de neurônios em cada camada intermediária e nas dimensões das imagens de entrada da rede. Onde foi avaliado o desempenho de cada variação da rede, com o intuito de identificar a estrutura que consiga o melhor desempenho.

Palavras-chave: Reconhecimento Facial, Redes Neurais Artificiais, Reconhecimento de Padrões, Biometria

Carlos Jr., Luis Fernando Martins **Reconhecimento Facial Utilizando Redes Neurais.** 2011 52 f. Trabalho 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.

ABSTRACT

The Automatic Face Recognition is considered a very complex task, as it is influenced by several factors, such as changes in lighting, noise in the image, facial expression, glasses, change the hair, beard or mustache and pose. Currently several techniques can be found for the recognition of faces. Due to the difficulties commonly found in the area there were several different approaches to solving the problem of face recognition. This work presents an approach to solve the problem of face recognition using a neural network MLP (Multilayer Perceptron)trained with the algorithm Rprop. The objective of this study is to demonstrate the effectiveness of the MLP networks applied to face recognition. A system that identify individuals from photographs of the face was implemented. The photographs were provided by ORL (OlivettiResearch Laboratory) database of faces. The tests were performed considering variations in the amount of intermediate layers (hidden), the number of neurons in each hidden layer and the size of the input images of the network. The performance of each variation of the network were measured, in order to identify the structure that achieves the best performance.

Keywords: Face Recognition, Artificial Neural Networks, Pattern Recognition, Biometry

Lista de Figuras

Lista de Tabelas

Sumário

Introdução

É frequente situações onde as pessoas necessitam assegurar sua autenticidade, como na realização de transações bancárias, identificação em empresas, aeroportos e shopping centers. Citam-se como meios de identificação mais utilizados, a aplicação de senhas e uso de cartões com chips ou dispositivos magnéticos que permitem a identificação do indivíduo. O problema com esses métodos é que qualquer indivíduo pode conseguir a senha ou o cartão de outro (HEINEN, 2002). Com a biometria esses tipos de problemas são extintos, ou pelo menos amenizados.

Existem diversas características biológicas que podem ser utilizadas em um processo de identificação, dentre as principais tem-se: impressão digital, retina, íris, geometria da mão, face, voz e assinatura. Todos os métodos anteriormente citados têm suas vantagens e desvantagens, cabendo ao utilizador optar pelo método que melhor se encaixe na aplicação desejada, sendo que cada um obterá um maior sucesso em uma determinada aplicação.

Este trabalho escolhe o Reconhecimento Facial como objeto de estudo, pois este possui algumas vantagens sobre as demais tecnologias biométricas, pois é natural, não-intrusivo e de fácil utilização (LI; JAIN, 2005). O reconhecimento facial é uma area de pesquisa bem abrangente e envolve várias disciplinas, como: processamento de imagem, reconhecimento de padrões, visão computacional e redes neurais. Desde de quando começou a ser estudada até os dias de hoje, surgiram diversas técnicas e abordagens para a resolução do problema do reconhecimento de faces.

As redes neurais artificiais são um modelo computacional inspirado no cérebro humano e possuem a capacidade de aprender e generalizar através de exemplos. As redes neurais tem contribuído muito no desenvolvimento de sistemas de reconhecimento e classificação de padrões e são utilizadas em vários trabalhos voltados ao reconhecimento de expressões faciais (XIAO et al., 1999). O presente trabalho propõe uma abordagem para a resolução do problema, utilizando redes neurais.

A abordagem proposta nesse trabalho consiste de uma rede neural do tipo MLP (Multilayer Perceptron) treinada com o algoritmo Rprop (uma variação do backpropagation). A rede será treinada a partir de fotografias da face de indivíduos, após treinada a rede será submetida à testes de reconhecimento, onde recebe uma outra imagem de um dos indivíduos com o qual a rede foi

treinada anteriormente. O objetivo deste trabalho é implementar um sistema de reconhecimento facial utilizando redes neurais, testar e analizar os resultados.

1 Reconhecimento Facial

Este capítulo faz um levantamento bibliográfico sobre o reconheciemento facial. Será abordada suas motivações, vantagens em relação a outros sistemas biométricos, dificuldades encontradas e metodologias existentes.

1.1 Considerações iniciais

O reconhecimento facial vem recebendo uma atenção significativa principalmente durante os últimos anos, pode-se citar ao menos dois motivos para isto: o primeiro é o grande número e variedades de aplicações possíveis, sejam elas comerciais, militares ou de segurança pública e o segundo é a disponibilidade de tecnologias viáveis após décadas de pesquisa (ZHAO et al., 2003).

O reconhecimento facial, seja ele a partir de imagens estáticas ou imagens de vídeos, é considerada um área de pesquisa bem promissora, abrangendo diversas disciplinas como: processamento de imagem, reconhecimento de padrões, visão computacional e redes neurais (CHELLAPPA; WILSON; SIROHEY, 1995). Existem diversas aplicações onde a identificação humana é necessária e o reconhecimento facial possui algumas vantagens sobre as demais tecnologias biométricas, pois é natural, não-intrusiva e de fácil utilização (LI; JAIN, 2005).

De maneira geral o problema pode ser especificado da seguinte forma: dado uma fotografia ou imagem de vídeo identificar uma ou mais pessoas na cena usando um banco de dados armazenado de faces. A solução do problema envolve a segmentação da face a partir de cenas cheias e/ou confusas, extração de atributos faciais, identificação e correspondência (CHELLAPPA; WILSON; SIROHEY, 1995).

Existem diversos problemas que comprometem a eficácia de um sistema de reconhecimento facial como variação de iluminação, ruídos na imagem, expressão facial, óculos, mudança nos cabelos, barba ou bigode e pose. Devidos ao problemas acima citados o desempenho dos sistemas de reconhecimento facial ainda estão distantes da percepção humana e são relativamente

pobres comparando com outros sistemas de reconhecimento biométrico como reconhecimento de impressão digital ou íris, sendo que tipicamente são apresentadas taxas de erro entre 2 – 25% (BARRETT, 1998).

Um sistema de reconhecimento facial geralmente consiste de quatro módulos: detecção facial, normalização, extração de características, e comparação (LI; JAIN, 2005). A detecção facial identifica a área referente a uma face em uma imagem e a isola do restante da imagem. A normalização visa a localização mais exata da face e compensa variações que possam existir em uma face. A extração de características analisa alguns componentes faciais como olhos, nariz e boca gerando um conjunto de atributos que será utilizado na fase de comparação. Na fase comparação é verificado a semelhança entre o conjunto de atributos obtidos com os mesmos atributos armazenados no banco de dados.

1.2 Detecção Facial e Extração de Características

A detecção facial ou segmentação é o primeiro passo a ser realizado em um sistema de reconhecimento facial, consiste em identificar e isolar a área referente a face em uma imagem. A detecção facial pode ser realizada com base em vários estímulos: cor de pele, formato do rosto ou cabeça, aparência da face, ou a combinação destes. A maior parte dos algoritmos de sucesso de detecção facial são os baseados em aparência sem usar outros estímulos (LI; JAIN, 2005).

A extração de características gera o conjunto de atributos que serão utilizados no processo de comparação com o banco de dados. Três tipos de métodos de extração de características podem ser distinguidos:

- métodos genéricos baseados em arestas linhas e curvas;
- métodos baseados no modelo de características que são usados para detectar características faciais como olhos;
- métodos estruturais de correspondência, que levam em consideração restrições geométricas sobre as características.

Inicialmente as abordagens eram centradas nas características individuais, por exemplo uma abordagem baseada em modelo foi descrito em (HALLINAN, 1991) para detectar e reconhecer o olho humano em uma face frontal. Estes métodos têm dificuldade quando a aparência de características se alteram significativamente, por exemplo, boca aberta, olhos fechados ou com

óculos. Para detectar as características de forma mais confiável, as abordagens recentes têm usado métodos estruturais de correspondência, por exemplo, o modelo Active Shape (COOTES et al., 1995). Comparando com os métodos anteriores, estes métodos recentes são muito mais robustos em termos de lidar com variações (ZHAO et al., 2003).

1.3 Metodologias de Reconhecimento Facial

De acordo com Zhao et al. (2003) os métodos de reconhecimento facial podem ser classificados de acordo com a forma que analisam a face considerando-a como um todo ou se baseando em características isoladas, assim pode-se dividí-los nos três grupos seguintes:

- Métodos baseados em características: realizam o reconhecimento de acordo com determinadas características individuais da face e suas relações geométricas, analisando características como olhos, nariz e boca, assim como medidas de distâncias e ângulos entre tais características.
- Métodos holísticos: realizam o reconhecimento analisando a face como um todo sem se preocupar em identificar características isoladas.
- Métodos híbridos: a fusão dos anteriores, inspirado na percepção humana que realiza o reconhecimento analisando tanto características locais quanto a região inteira da face, uni as melhores características dos dois métodos.

As décadas de 70 e 80 tinham como métodos tradicionais as estratégias baseadas em análise das características e Template Matching, enquanto que a partir dos anos 90 surgiram métodos baseados em transformações lineares (e também não lineares) e redes neurais, que apresentaram melhores desempenhos com relação a bases de dados grandes (superiores a 100 imagens). Portanto enfatizando as técnicas em reconhecimento de faces através de fotografias desenvolvidas nos últimos anos pode-se recorrer tabela 1.

1.3.1 Métodos Baseados em Características

Métodos baseados em características são aqueles que fazem o reconhecimento baseado em características individuais da face. Dentre estes métodos pode-se citar o método baseado em características geométricas da face. Este método utiliza características da face como olhos, boca e nariz, e parâmetros pertencentes a estas características como distâncias entre os olhos, o

Tabela 1: Técnicas de Reconhecimento de faces (fotografias) desenvolvidas nos últimos anos (ZHAO et al., 2003).

Técnicas	Trabalhos Representativos
Métodos Holísticos	
Principal Component Analysis (PCA)	Aplicações diretas do PCA (KIRBY; SIROVICH, 1990)(TURK; PENTLAND, 1991)
Eigenfaces Probabilístico	Medidas de Probabilidade (MOGHADDAM; PENTLAND, 1997)
Fisherfaces/sub-espaço LDA	LDA (BELHUMEUR et al., 1997)(SWETS; WENG, 1996) (ZHAO et al., 1998)
SVM	Problema baseado em separação de classes (PHILLIPS, 1998)
ICA	Análises de características com ICA (BARTLETT et al., 1998)
Outros Métodos	
LDA	LDA aplicado diretamente nas faces (ETEMAD; CHELLAPPA, 1997)
PDBNN	Rede neural baseada em decisão probabilística (LIN et al., 1997)
Métodos Baseados em Características	
Método Geométrico Puro	um dos primeiros métodos (KANADE, 1973), métodos recentes (COX et al., 1996)
Arquitetura de Link Dinâmico	(OKADA et al., 1998)(WISKOTT et al.,1997)
Modelo *Hidden Markov*	(NEFIAN; HAYES, 1998)(SAMARIA, 1994)
Métodos Hibrídos	
Eigenfaces Modular	*Eigenface e Eigenmódulos*(PENTLAND et al., 1994)
Hibrído LFA	método local de características (PENEV; ATICK, 1996)
Baseado em Componentes	região da face e componentes (HUANG et al., 2003)

tamanho da boca, e a distância entre a boca e o nariz, para montar um vetor característico que será utilizado para identificar um indivíduo associado ao banco de dados.

Muitas pessoas têm explorados estes métodos, Kanade (1973) apresentou um método automático de extração de características e relatou um reconhecimento entre 45 - 75% com um banco de dados de 20 pessoas, enquanto Brunelli e Poggio (1993) utilizaram um método baseado num conjunto de relações geométricas como a largura e o comprimento do nariz a posição da boca e o formato do queixo. Eles relatam uma taxa de 90% de reconhecimento sobre uma base de dados de 47 pessoas. Cox, Ghosn e Yianilos (1996) introduziram uma técnica que mistura distâncias, que atinge uma taxa de reconhecimento de 95%, utilizando um banco de dados de 95 imagens de um total de 685 indivíduos, onde cada face é representada por 30 distâncias extraídas manualmente.

1.3.2 Métodos Holísticos

Os métodos holísticos são aqueles que efetuam o reconhecimento considerando a face como um todo, utilizando de informações globais da face, sem analisar características físicas isoladas. As informações globais da face podem ser representadas por um vetor derivado diretamente da informação dos pixels das imagens da face, contudo este vetor pode ser considerado demasiadamente grande tornando o processo dispendioso, assim surgiram diversas técnicas com o objetivo de conseguir um taxa de reconhecimento considerável e ao mesmo tempo economizar recursos de processamento. Os métodos estatísticos possuem uma importante característica que é a compressão das informações contidas nas imagens (redução de dimensionalidade) e apenas após isto é realizada a classificação, dentre estes métodos os mais relevantes são Análise de Componente Principal (PCA) (TURK; PENTLAND, 1991a) (TURK; PENTLAND, 1991b), Análise de Componente Independente (ICA) (BARTLETT; LADES; SEJNOWSKI, 1998), e Análise Discriminante Linear (LDA) (BELHUMEUR; HESPANHA; KRIEGMAN, 1997) (SWETS; WENG, 1996a) (SWETS; WENG, 1996b). Os trés métodos se baseiam em análises estatísticas dos vetores da face, e em transformações lineares e não lineares dos sistemas de coordenadas onde se encontram estes vetores.

1.3.3 Redes Neurais Aplicadas no reconhecimento facial

Uma solução não linear para o problema do reconhecimento facial é dada pelas redes neurais, largamente utilizadas em problemas de reconhecimento de padrões e readaptada para o reconhecimento de faces. A idéia básica seria dimensionar a rede com um neurônio para cada pixel da imagem, contudo devido as dimensões do modelo as redes não são treinadas diretamente com as imagens como entrada, isto tornaria a rede muito complexa e difícil de treinar, ao invés disso a rede recebe como entrada dados processados por alguma técnica de redução de dimensionalidade (ABATE et al., 2007).

Uma abordagem para o problema foi proposta por Cottrell e Fleming (1990) utilizando duas redes perceptron back-propagation, onde a primeira trabalha em modo auto-associativo extraindo características para a segunda, que realiza a classificação. Embora as redes neurais possam ser usadas para problemas envolvendo muitas imagens Conttrell e Fleminig mostram que mesmo em condições favoráveis a solução proposta por eles não consegue um resultado melhor que o PCA.

Outros tipos de redes também tem sido utilizados em trabalhos na área buscando explorar propriedades particulares de cada rede. Foi apresentado por Lawrence et al. (2002) um sistema

que combinava uma amostragem local da imagem, uma rede neural SOM, e uma rede neural de convolução ou Convolution Neural Network, segundo os autores essa rede apresentou uma taxa de erro de 3,83%.

Outro trabalho importante foi uma análise com redes perceptron multicamadas realizada por Rizk e Taha (2002) utilizando o algoritmo de treinamento Backpropagation (Multilayer Perceptron Backpropagation Neural Networks - MLP/ BP NN), rede neural de Função Base Radial (Radial Basis Function Neural Networks - RBF NN), e rede neural de Cluster Multicamadas (Multilayer Cluster Neural Networks - MCNN) para aplicações no reconhecimento de faces. As redes neurais foram alimentadas com um vetor característico extraído das imagens através de vários métodos de extração e compressão de dados, eles são: Discrete Wavelet Transform (DWT), Discrete Radon Transform (DRT), Discrete Cosine Transform (DCT) e Principal Component Analysis (PCA).

2 Redes Neurais

Este capítulo apresenta os conceitos básicos sobre as redes neurais artificiais. Inicialmente é realizado um breve histórico, em seguida é apresentada uma revisão sobre redes biológicas. Na sequência é apresentado o modelo de neurônio artificial e os tipos de treinamento. E por fim são apresentados dois modelos de redes neurais, o Perceptron e o MLP (Multilayer Perceptron).

2.1 Histórico

O primeiro modelo artificial de um neurônio biológico foi concebido por McCulloch e Walter Pitts, em 1943. O trabalho de McCulloch e Pitts foi voltado em descrever um modelo artificial de um neurônio e apresentar suas capacidades computacionais e o neurônio apresentado por eles foi chamado de MCP. Somente alguns anos depois do trabalho de McCulloch e Pitts o aprendizado de redes biológicas veio a ser objeto de estudo, e primeiro trabalho com relação ao aprendizado de redes neurais foi apresentado por Donald Hebb em 1949. Em 1958 Rosenblatt apresentou um novo modelo de neurônio o perceptron, seu modelo consistia em acrescentar sinapses ajustáveis as redes com neurônios MCP, permitindo que a rede pudesse ser treinada para classificar alguns tipos padrões. Porém em 1969, Minsky e Papert demonstraram que o perceptron de uma unica camada apresentado por Rosenblatt era limitado a resolver problemas linearmente separáveis. Durante a década de 70 o estudo de redes neurais ficou adormecido, devido principalmente a repercussão do trabalho de Minsky e Papert, apesar de algumas poucas pesquisas na área. A retomada por pesquisas em RNAs aconteceu nos anos 80, motivadas principalmente por um artigo publicado em 1982 por John Hopifield, que chamou a atenção para as propriedades associativas das RNAs, mostrando a relação entre redes recorrentes auto-associativas e sistemas físicos. Alguns anos depois com o surgimento do algoritmo back-propagation(RUMELHART; HINTON; WILLIAMS, 1986), mostrou-se que a visão de Misky e Papert sobre o perceptron era bastante pessimista e que as RNAs de múltiplas camadas são capazes de resolver “problemas difíceis de aprender”(BRAGA; CARVALHO; LUDERMIR, 2007).

2.2 Neurônio Biológico

O neurônio é uma célula do cérebro, cuja principal função é coletar e disseminar sinais elétricos. Acredita-se que redes formadas por esses neurônios sejam a principal responsável pela capacidade de processamento do celébro humano (RUSSELL; NORVIG, 2003). Segundo Braga, Carvalho e Ludermir (2007) o neurônio biológico pode ser divido em três partes, corpo celular, dendritos e axônio, cada uma com sua própria função, no entanto complementares entre si. Os dendritos possuem a função de receber as informações (impulsos nervosos), proveniente de outros neurônios e conduzi-las ao corpo-celular, onde a informação é processada, e novos impulsos são gerados, esses impulsos passam pelo axônio onde são transmitidos a outros neurônios. A região onde ocorre o contato entre a saída axônica de um neurônio e o dendrito de outro é chamada de sinape. É através das sinapses que os neurônios são unidos funcionalmente, formando as redes neurais biológicas. Na figura 1 pode ser observado os componentes de um neurônio biológico.

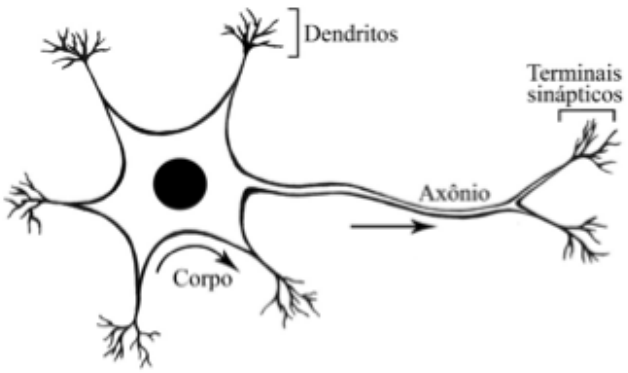


Figura 1: Modelo de neurônio biológico (PIAZENTIN; DUARTE, 2011).

2.3 Neurônio Artificial

A natureza sempre serviu de inspiração para a ciência e o neurônio artificial é um exemplo disto. O MCP, neurônio proposto por McCulloch e Pitts é uma simplificação do que se sabia na época o respeito do neurônio biológico. Este modelo de possui n entradas, que representam os dendritos, cada uma destas entradas recebem valores $x_1, x_2, x_3, ..., x_n$, representando as ativações dos neurônios anteriores e um único terminal de saída y, que representa o axônio. O comportamento das sinapes é obtido através de pesos $w_1, w_2, w_3, ..., w_n$, que são vinculados as entradas, onde estes pesos podem ter valor positivo ou negativo dependendo do tipo de sinapse. Um representação gráfica desse modelo pode ser observado na figura 2.

A ativação do neurônio artificial é representado por um mecanismo simples que faz a soma dos valores $x_i w_i$ recebidos pelo neurônio e decide se este deve ou não disparar (saída 1 ou 0), comparando o valor obtido na soma com o limiar de exitação (threshold). No caso do MCP a saída do neurônio é obtida através de uma "função de ativação". A função de ativação é responsável por emitir a saída y do neurônio através dos vetores de pesos $w = (w_1, w_2, w_3, \ldots, w_4)$ e de entrada $x = (x_1, x_2, x_3, ..., x_4)$. A função de um neurônio MCP, pode ser observada na equação 2.1.

$$f(u) = \begin{cases} 1 & \sum_{i=1}^{n} x_i w_i \geq \theta \\ 0 & \sum_{i=1}^{n} x_i w_i < \theta \end{cases} \tag{2.1}$$

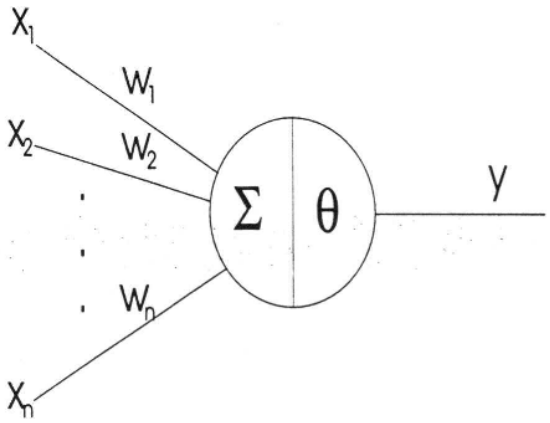


Figura 2: Neurônio MCP (BRAGA; CARVALHO; LUDERMIR, 2007).

A partir do modelo proposto por McCulloch e Pitts foram derivados vários outros modelos que permitem a produção de uma saída qualquer, não necessariamente zero ou um, e com diferentes funções de ativação.

2.4 Aprendizado

As RNAs tem como principal característica a capacidade de aprendizado através de exemplos, pois é esta característica que difere a abordagem conexionista (RNAs) da IA simbólica. Na IA simbólica o conhecimento é adquirido através de regras explicitas, enquanto que as RNAs adquirem o conhecimento através de ajustes em seus pesos (BRAGA; CARVALHO; LUDERMIR, 2007). De maneira geral o aprendizado pode ser definido da seguinte forma:

> Aprendizagem é um processo pelo qual os parâmetros livres de uma rede neural são adaptados através de um processo de estimulação pelo ambiente no qual a rede está inserida. O tipo de aprendizagem é determinada pela maneira pela qual a modificação dos parâmetros ocorre (MENDEL; MCLAREN, 1970).

Para o aprendizado de uma rede é necessário primeiramente um critério para medir o desempenho da mesma. Através desse critério é possível avaliar a rede, após esta ter sido submetida os ao conjunto de dados fornecidos pelo ambiente. E com esta avaliação é possível calcular o erro atual da rede. Um algoritmo de treinamento tem como objetivo diminuir o erro atual da rede a medida que o aprendizado vai acontecendo. Na equação 2.2, pode ser observado de forma genérica como o vetor de pesos pode ser ajustado durante o treinamento visando diminuir o erro da rede. Onde $w(t)$ e $w(t+1)$ representam os valores dos pesos nos instantes t e t+1, respectivamente, e $\Delta w(t)$ é o ajuste aplicado aos pesos. A maioria dos algoritmos de treinamento funcionam desta forma diferindo apenas na forma de como $\Delta w(t)$ é calculado.

$$w(t+1) = w(t) + \Delta w(t) \tag{2.2}$$

Segundo Azevedo, Brasil e Oliveira (2000) o aprendizado pode ser classificado em dois tipos: associativo e não associativo. No aprendizado associativo a rede aprende sobre o relacionamento entre pares de estímulos, este tipo de aprendizado é modelo para RNA supervisionadas. Enquanto que no aprendizado não-associativo, não há estímulos secundários para associar com os estímulos primários. Neste tipo de aprendizado, a repetição de um estímulo fornece a oportunidade para aprender sobre suas propriedades, este tipo de aprendizado é o modelo em RNA não-supervisionada.

- Aprendizado Supervisionado: neste tipo de treinamento, para cada entrada submetida à rede também é submetida uma saída desejada. A partir de cada entrada a rede produz uma saída como resposta, esta saída é comparada com a saída desejada. Se a saída corrente for diferente da saída desejada, a rede gera um erro, o qual é utilizado para calcular o ajuste que deve ser feito nos pesos da rede com intuito de minimizar o erro. Este processo de comparação da saída e minimização de erros é realizado por um circuito especial chamado supervisor ou professor, daí o nome aprendizado supervisionado (AZEVEDO; BRASIL; OLIVEIRA, 2000). O algoritmo de treinamento supervisionado mais conhecido é a regra delta (WIDROW; HOFF, 1960) e sua generalização para redes de múltiplas camadas o back-propagation (RUMELHART; HINTON; WILLIAMS, 1986).

- Aprendizado Não-Supervisionado: ao contrário do aprendizado supervisionado este tipo de aprendizado não necessita de um supervisor e não há saída desejada. Durante o treinamento os padrões de entrada são apresentados continuamente à rede, esta por sua vez organiza e classifica os padrões de acordo com regularidades encontradas nos dados. O aprendizado não-supervisionado é aplicado na resolução de problemas que tem por objetivo descobrir características estatisticamente relevantes nos dados de entrada, como a

descoberta de classes, ou agrupamentos(BRAGA; CARVALHO; LUDERMIR, 2007). Citam-se como modelos conhecidos deste tipo de aprendizado os mapas auto-organizáveis (SOM - Self Organizing-Maps) de Kohonen (KOHONEN, 1982) e os modelos ART(Adaptive Resonancy Theory) (CARPENTER; GROSSBERG, 1988).

2.5 Perceptron

O modelo perceptron foi proposto por Rosenblatt, em 1958, sua topologia original era composta por unidades de entrada(retina), por um nível intermediário formado pelas unidades de associação e por um nível de saída formado pelas unidades de resposta. Apesar dessa topologia possuir três níveis, ela ficou conhecida como perceptron de uma camada, pois somente o nível de saída possui propriedades adaptativas. De forma geral podemos definir a retina como unidades sensoras, e as unidades intermediárias de associação, que apesar de formadas por neurônios MCP, possuem pesos fixos, definidos antes do treinamento(BRAGA; CARVALHO; LUDERMIR, 2007). O modelo descrito pode ser observado na figura 3.

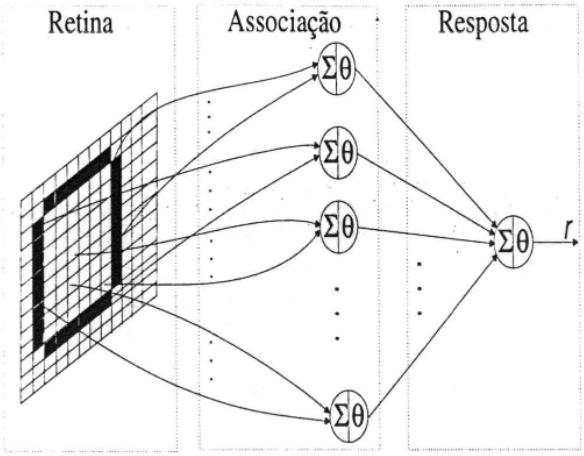


Figura 3: Topologia do Perceptron simples com uma única camada (BRAGA; CARVALHO; LUDERMIR, 2007).

Dado um neurônio arbitrário da camada de resposta de um perceptron, e seus vetores de entrada x' e de pesos w', sua ativação é dada por $\sum w'_i + x'_i = w'.x'$, onde $w'.x'$ é produtos dos vetores w' e x'. Portanto a condição crítica para o disparo do neurônio pode ser representada por $w'.x' = \theta$, ou $w'.x' - \theta = 0$, o que equivale a se adicionar um peso w_0 com valor $-\theta$ às entradas do neurônio e conectá-lo a uma entrada com valor fixo $x_0 = 1$. A nova condição crítica de disparo passa a ser $w.x = 0$, onde $w = \{-\theta, w_1, w_2, ..., w_n\}^T$ e $x = \{1, x_1, x_2, ..., x_n\}^T$. Considere um neurônio arbitrário da rede, submetido ao par de treinamento $\{x, y_d\}$, onde x é o valor de entrada e y_d a saída desejada. Considere a saída atual da rede em resposta a entrada x

como y, assim pode-se definir o erro devido à saída atual como sendo $e = y_d - y$. No caso do perceptron tem-se sempre que $y \in \{0,1\}$ e $y_d \in \{0,1\}$, portanto $e \in \{-1,0,1\}$. No caso de $e = 0$, não será necessário ajuste no vetor de pesos w, pois $y = y_d$, quando $e = 1$, um vetor deve ser somado ao vetor de pesos w, com intuito de aproximá-lo do resultado, porém quando $e = -1$, um vetor deve ser subtraído do vetor de pesos w. Assim o treinamento do perceptron pode ser representado pela Equação 2.3, onde $w(n)$ e $w(n+1)$ representam os valores dos vetores de pesos nos instantes n e $n+1$, respectivamente, η é a taxa de aprendizado, uma constante que indica com qual velocidade o vetor de pesos será ajustado, e é o erro no instante n, que indicará em qual direção os pesos do vetor w será ajustado e x é o vetor de entrada. De acordo com o Teorema da Convergência(ROSENBLATT, 1958), atualização dos pesos através da Equação 2.3 leva sempre a uma solução, caso as classes em questão sejam linearmente separáveis.

$$w(n+1) = w(n) + \eta e x(n) \tag{2.3}$$

Segundo Braga (2007), a implementação do algoritmo de treinamento do perceptron pode ser descrito da seguinte forma:

1. Inicialize n;
2. Inicialize o vetor de pesos w com valores aleatórios;
3. Aplique a regra de atualização de pesos $w(n+1) = w(n) + \eta e x(n)$ para todos os p pares (x^i, y_d^i) do conjunto de treinamento $\Gamma = \{(x^i, y_i^i)\}_{i=1}^{p}$;
4. Repita o passo anterior até que $e = 0$ para todos os p elementos de Γ;

2.6 Perceptron Múltiplas Camadas

Como visto anteriormente as RNAs de uma única camada estão limitadas a resolver problemas linearmente separáveis. Contudo a não-linearidade está presente na maioria dos problemas reais, sendo necessária a utilização de uma estrutura com características não lineares para a resolução de problemas com maior complexidade.

Minsky e Papert (1972) em seu livro "Perceptrons"abordaram principalmente a incapacidade de redes do tipo perceptrons de uma única camada em resolver problemas não linearmente separáveis. Uma solução para o problema seria a utilização de mais uma camada de pesos adptativos, o que equivaleria ao aumento da dimensionalidade do problema. Contudo seria necessário um algoritmo de aprendizagem que tratasse todas as camadas, segundo Minsky e Papert seria

impossível implementar tal algoritmo, anos mais tarde foi provado que eles estavam enganados sobre isto (AZEVEDO; BRASIL; OLIVEIRA, 2000).

O Perceptron de Múltiplas Camadas (MLP - Multilayer Perceptron) é uma rede de múltiplas camadas, composta por um conjunto de unidades sensoriais que formam a camada de entrada, uma ou mais camadas ocultas e uma camada de saída, uma representação gráfica de uma rede MLP pode ser observada na figura 4. Estas redes se caracterizam por possuir neurônios com função de ativação sigmoidais nas camadas intermediárias e sigmoidais ou lineares na camada de saída. As camadas intermediárias devem utilizar funções de ativação não lineares para possibilitar o resolução de problemas de maior ordem no espaço de entrada. A utilização de funções lineares em todas as camadas resultaria em uma rede de uma única camada, já que transformações lineares sucessivas podem ser descritas como uma única transformação linear (BRAGA; CARVALHO; LUDERMIR, 2007).

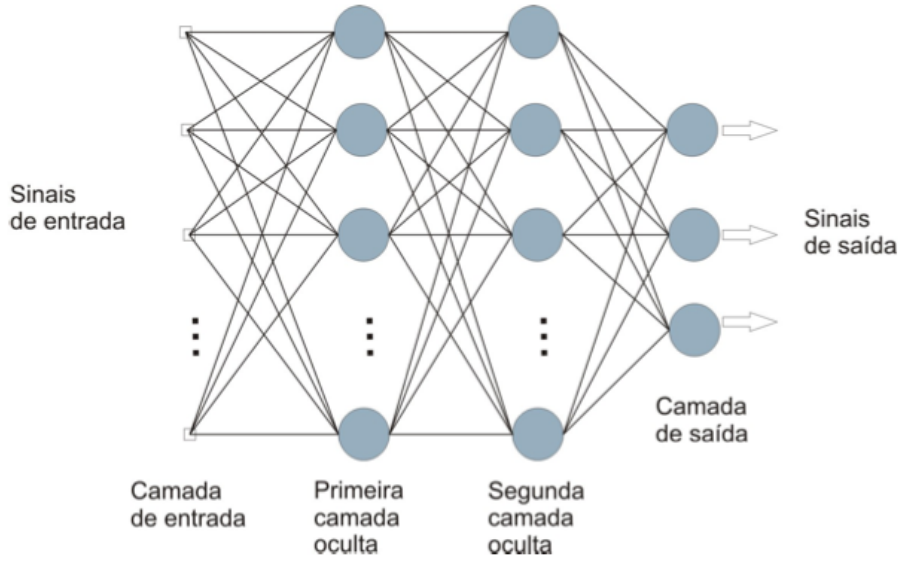


Figura 4: Topologia da rede MLP - Multilayer Perceptron (HAYKIN, 2001)

De uma maneira geral a função das múltiplas camadas de uma rede feedforwad como a MLP, é transformar o problema descrito pelo conjunto de dados de entrada em uma representação tratável para a camada de saída da rede. Desta forma pode-se considerar que um problema não-linearmente separável seja transformado pela camada intermediária e um problema linearmente separável.

Uma questão muito importante que deve ser considerada ao criar uma rede de múltiplas camadas é a definição do número de neurônios em cada camada. A quantidade de neurônios influência muito no desempenho da rede principalmente em sua capacidade de generalização. Quanto mais neurônios maior a complexidade da rede e maior a capacidade de resolver proble-

mas. Porém deve-se ter cuidado ao dimensionar uma rede, pois uma rede com muitos neurônios por conter uma grande número de soluções, possui uma pequena quantidade de soluções ótimas em relação ao total de soluções. Apesar de várias pesquisas em redes neurais voltadas para o dimensionamento de redes, não existe nenhuma fórmula para calcular o número de neurônios que uma rede deve ter, pois a quantidade necessária de neurônios está diretamente ligada a complexidade do problema a ser resolvido.

2.6.1 Treinamento de Redes MLP

A solução para o problema do treinamento de redes MLPs surgiu na década de 80, com o algoritmo de retropropagação de erros ou back-propagation. Este algoritmo é conhecido também como regra delta generalizada, pois é uma generalização da regra delta utilizada para o treinamento de redes Adaline(WIDROW; HOFF, 1960). O back-propagation é um algoritmo de treinamento supervisionado, que durante o treinamento submete a rede à pares de dados, formados pela entrada e saída desejada, para o ajuste dos pesos da rede. O treinamento ocorre em duas fases, forward e backward, onde cada uma percorre a rede em determinado sentido. Segundo Braga, Carvalho e Ludermir (2000) o algoritmo de aprendizado back-propagation e suas fases, podem ser descritos da seguinte forma:

Fase forward:

1. A entrada é apresentada à primeira camada da rede.
2. Para cada camada C^i a partir da camada de entrada
   2.1 Após os nodos da camada $C^i(i > 0)$ calcularem seus sinais de saída, estes servem como entrada para a definição das saídas produzidas pelos nodos da camada C^{i+1}.
3. As saídas produzidas pelos nodos da última camada são comparadas às saídas desejadas.

Nessa fase, são determinadas as saídas para um dado padrão de entrada.

Fase backward:

1. A partir da última camada até a camada de entrada.
   1.1 Os nodos da camada atual ajustam seus pesos de forma a reduzir seus erros.
   1.2 O erro de um nodo das camadas intermediárias é calculado utilizando os erros dos nodos da camada seguinte conectados a ele, ponderados pelos pesos das conexões entre eles.

Nessa fase, as saídas desejada e fornecida são comparadas e os pesos ajustados.

O algoritmo do back-propagation une as duas fases apresentadas acima e percorre os seguintes passos:

1. Inicialização de pesos e parâmetros.
2. Repita até que o erro mínimo ou a realização de um dado número de ciclos:
   2.1 Para cada padrão de treinamento X:
      2.1.1 Definir saída da rede através da fase forward.
      2.1.2 Comparar saídas produzidas com saídas desejadas.
      2.1.3 Atualização de pesos dos nodos através da fase backward.

Esse algoritmo propõe que o ajuste de pesos seja realizado através do método do gradiente descendente. Apesar deste algoritmo caminhar na direção correta para a correção de erros e proporcionar bons resultados, ele deixa um tanto a desejar na velocidade de convergência em busca da minimização dos erros. A principal dificuldade está relacionada com sua sensibilidade às características da superfície de erro, o que dificulta a sua convergência em áreas de baixo gradiente e mínimos locais (BRAGA; CARVALHO; LUDERMIR, 2007).

2.6.2 Variações do Algoritmo Back-propagation

A implementação original do back-propagation é muito lenta em diversas situações, e seu desempenho piora na medida em que aumenta o tamanho e a dificuldade dos problemas submetidos a rede. Visando acelerar o tempo de treinamento e melhorar o desempenho, foram criadas algumas variações do algoritmo back-progation, como o back-propagation com momentum, Quickprop e Rprop, que serão abordados com mais detalhes a seguir:

- Back-propagation com momentum: Este método consiste na adição de um termo para o ajuste dos pesos, proporcional ao valor da mudança de peso anterior, este termo é chamado de momentum (AZEVEDO; BRASIL; OLIVEIRA, 2000). De acordo com Zurada (1992) a adição do momentum pode ser feita conforme a equação 2.4, onde, t e $t-1$ indicam a etapa de treinamento atual e a anterior, β é o termo de momento e $\nabla E(t)$ correspondem ao gradiente descendente do erro.

$$\Delta w(t) = -\eta \nabla E(t) + \beta \Delta w(t-1) \tag{2.4}$$

- Quickprop: Este algoritmo utiliza informações sobre a curvatura da superfície de erro para acelerar a convergência da rede. Comparado com o back-propagation a principal diferença é que o Quickprop utiliza as inclinações anterior e atual do erro para cada peso. Quickprop considera que a superfície de erro é localmente quadrática, formando uma parábola, e procura saltar da posição atual na superfície para o ponto mínimo da parábola(BRAGA; CARVALHO; LUDERMIR, 2007). A equação para ajuste dos pesos pode ser observada na Equação 2.5.

$$\Delta w_{ji}(t) = \frac{\frac{\partial E}{w_{ji}}(t)}{\frac{\partial E}{w_{ji}}(t-1) - \frac{\partial E}{w_{ji}}(t)} \Delta w_{ji}(t-1) \tag{2.5}$$

- Rprop: O algoritmo back-propagation padrão realiza o ajustes dos peso utilizando o valor da derivada parcial, o problema é que em alguns casos esse valor é muito próximo de zero, fazendo com que o ajuste de pesos seja mínimo. O Rprop(Resilient back-propagation) elimina esse problema, pois utiza apenas o sinal da derivada e não o seu valor. O sinal da derivada indica apenas a direção do ajuste de pesos, o tamanho do ajuste de pesos é definido por um "valor de atualização"Δ^t_{ji}, de acordo com a equação 2.6.

$$\Delta w_{ji}(t) = \begin{cases} -\Delta_{ji}(t), & se \quad \frac{\partial E}{\partial w_{ji}}(t) > 0 \\ +\Delta_{ji}(t), & se \quad \frac{\partial E}{\partial w_{ji}}(t) < 0 \\ 0, & se \quad \frac{\partial E}{\partial w_{ji}} = 0 \end{cases} \tag{2.6}$$

Onde o valor de atualização Δ^t_{ji} é definido por um processo de adaptação que depende sinal da derivada do erro com relação ao peso a ser ajustado, conforme a equação 2.7, onde $0 < \eta^- < 1 < \eta^+$.

$$\Delta_{ji}(t) = \begin{cases} \eta^+ \Delta_{ji}(t-1), & se \quad \frac{\partial E(t-1)}{\partial w_{ji}} \frac{\partial E(t)}{\partial w_{ji}} > 0 \\ \eta^- \Delta_{ji}(t-1), & se \quad \frac{\partial E(t-1)}{\partial w_{ji}} \frac{\partial E(t)}{\partial w_{ji}} < 0 \\ \Delta_{ji}(t-1), & se \quad \frac{\partial E}{\partial w_{ji}} = 0 \end{cases} \tag{2.7}$$

De acordo com a regra de adaptação utilizado pelo Rprop, quando a derivada parcial do erro em relação a um peso w_{ji} mantém o seu sinal, mostra que o ajuste anterior reduziu o erro cometido, o valor de atualização Δ_{ji} é aumentado pelo fator η^+, acelerando a convergência do treinamento. Porém quando a derivada parcial muda de sinal, significa que o ajuste anterior foi demasiadamente grande, o valor de atualização Δ_{ji} é reduzido pelo fator η^-, mudando a direção do ajuste (BRAGA; CARVALHO; LUDERMIR, 2007).

3 Desenvolvimento

Neste capítulo será apresentado o desenvolvimento da proposta, onde será abordada a metodologia escolhida e o banco de faces utilizado. Em seguida será discutido a implementação do sistema, os resultados obtidos e as considerações sobre os mesmos.

3.1 Método Proposto

As redes neurais artificiais tem contribuído muito no desenvolvimento de sistemas de reconhecimento e classificação de padrões em imagens e são utilizadas em vários trabalhos voltados ao reconhecimento de expressões faciais (XIAO et al., 1999). O presente trabalho aborda um método de reconhecimento de faces baseado em redes neurais do tipo MLP (Multilayer Perceptron) submetida a um treinamento com o algoritmo Rprop (Resilient back-propagation). A rede neural foi implementada utilizando o Encog, framework JAVA para implementação de redes neurais artificiais. A proposta consiste em treinar a rede a partir de fotografias da face de indivíduos, após treinada a rede é submetida ao teste de reconhecimento onde recebe uma outra imagem de um dos indivíduos com o qual a rede foi treinada anteriormente.

3.2 Banco de Imagens

Como visto anteriormente existem diversos fatores que influenciam no reconhecimento da face humana tais como variação da iluminação, pose, expressão facial, óculos, mudança nos cabelos, barba ou bigode, etc. Levando isto em conta, para avaliar um sistema de reconhecimento facial de forma eficiente é necessário utilizar um banco de dados que possua imagens com essas variações.

Devido ao reconhecimento facial ser uma área de pesquisa fortemente ativa temos diversos bancos de faces disponíveis para esta finalidade, tais como, AR, BANCA, CAS-PEAL, CMU Hyper , CMU PIE, Equinox IR , FERET, KFDB , MIT, MPI, ND HID, ORL, UMIST, U. Texas,

U. Oulu, XM2VTS, Yale, Yale B, sendo que alguns destes são disponibilizados de maneira gratuita (LI; JAIN, 2005).

Neste trabalho o banco de imagens escolhido foi o ORL que foi produzido pela Olivetti Research Laboratory em Cambridge, UK. Este banco é gratuito/público e possui um total de 400 imagens sendo 40 indivíduos e 10 imagens diferentes para cada indivíduo. As imagens possuem variações na expressão facial (olhos abertos/fechados, sorrindo/sem sorrir), iluminação e detalhes faciais (com ou sem óculos). As imagens foram obtidas sob um fundo escuro e homogêneo, e estão em escala de cinza com uma resolução de 92x112 pixels. Este banco está disponível publicamente em http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html, a figura 5 fornece uma amostra deste banco de faces.

Figura 5: Amostra de imagens encontrados no banco de faces ORL (AT&T, 1992).

Este banco de dados foi escolhido por ser público, conter uma quantidade considerável de imagens, possuir as variações necessárias para o experimento (iluminação, pose, expressão facial, etc), outro motivo é o fato deste banco de dados ser bastante utilizado em trabalhos na área possibilitando uma comparação de resultados com trabalhos correlatos.

3.3 Encog Framework

O Encog é um avançado framework de redes neurais e aprendizado de máquina. Encog contém classes para criar uma grande variedades de redes neurais, também possui classes para normalizar e processar dados para essas redes, possibilita também o treinamento das redes usando multithreaded resilient propagation, permite o uso do GPU para aumentar a velocidade de processamento. Encog fornece conjuntos de dados especializados para tornar mais fácil o processamento de diferentes tipos de dados. Uma mesma rede neural é capaz de lidar com diferentes tipos. O ImageNeuralDataSet faz parte desses conjuntos de dados especializados, esta classe aceita uma lista de imagens que serão carregadas e processadas para uma forma útil para o Encog. O Framework Encog está em desenvolvimento ativo desde 2008 e está disponível para

JAVA e .NET(HEATON, 2010).

3.4 Implementação do sistema

O sistema de reconhecimento facial desenvolvido neste trabalho foi implementado na linguagem JAVA com o auxílio do Encog (Framework para redes neurais). A rede neural deverá ser treinada e receberá como entrada as imagens do banco de faces, para isso será utilizado o ImageNeuralDataSet, uma classe do Encog específica para tratamento de dados fornecidos através de imagens. Antes de instanciar o ImageNeuralDataSet um objeto downsampled deve ser criado, no Encog todos objetos downsampled deve implementar a interface Dowsample. Atualmente o Encog suporta duas classes downsample, RGBDownsample, SimpleIntensity-Downsample. Para este projeto foi utilizado a classe RGBDownsample, por ser mais precisa na representação das cores ou níveis de cinza presente na imagem. O RGBDownsample é o mais avançado dos dois, ele converte para a resolução especificada e transforma cada pixel em uma entrada de três cores (RGB). O número total de valores de entrada dos neurônios produzidos por este objeto será altura vezes largura vezes três. Primeiramente será instanciado o objeto downsample e depois o ImageNeuralDataset passando o downsample como parâmetro no construtor, conforme o códido 3.1.

Código 3.1: Instanciando Downsample e ImageNeuralDataset

```
1 downsample = new RGBDownsample();
2
3 this.training = new ImageNeuralDataSet(downsample, false, 1, -1);
```

Os valores 1 e -1 especificam o intervalo em que as cores serão normalizadas, o valor false significa que não se deseja tentar detectar bordas, se fosse true, Encog tentaria detectar as bordas, porém a detecção de bordas atual do Encog não é muito avançada, sendo que se for necessário uma detecção de bordas avançada as imagens deve ser processadas antes de serem enviadas ao objeto ImageNeuralDataSet. Agora que o objeto ImageNeuralDataSet foi criado, deverá ser acrescentado algumas imagens a ele. Para adicionar imagens a este conjunto de dados, deve ser criado um objeto ImageNeuralData para cada imagem. A linhas de código 3.2 demonstram como uma imagem de um arquivo pode ser adicionada ao ImageNeuralData.

Código 3.2: Adicionando uma imagem ao ImageNeuralData.

```
1 Image img = ImageIO.read( [filename of image] );
2 ImageNeuralData data = new ImageNeuralData( img );
3 this.training.add( data, [ideal output] );
```

As imagens são carregadas a partir de um arquivo usando a classe Java ImageIO, qualquer tipo de imagem aceito pelo java pode ser utilizado pelo dataset. A saída ideal deve ser especificada no caso de treinamento supervisionado, para treinamento não supervisionado esse parâmetro pode ser omitido. Após o objeto ImageNeuralData ser instanciado ele é adicionado ao dataset, esses passos são repetidos para cada imagem que for adicionada. Depois de todas as imagens terem sido carregadas, elas estão prontas pra serem redimensionadas. Para diminuir a resolução das imagens deve ser chamado o método downsample e especificar a altura e largura para qual serão redimensionadas, conforme as linhas de código 3.3.

Código 3.3: Redimensionando as imagens com downsample.

```
1 this.training.downsample(this.downsampleHeight, this.downsampleWidth);
```

Depois que o método downsample foi chamado os dados para treinamento foram obtidos e podem ser utilizados para treinar a rede neural. Para finalizar a criação da rede deve ser chamado o método simpleFeedFoward da classe EncogUtility, que irá de fato criar a rede neural. Finalmente a rede será criada conforme pode ser observado no código 3.4, os parâmetros passados para o método simpleFeedFoward, são "this.training.getInputSize()" que é o tamanho da entrada da rede, "hidden1" e "hidden2" correspondem a quantidade de neurônios na primeira e segunda camada oculta respectivamente, o parâmetro true especifica que seja usada uma função de ativição tangente hiperbólica.

Código 3.4: Criando a Rede Neural.

```
1 this.network = EncogUtility.simpleFeedForward(this.training.getInputSize(),
      hidden1, hidden2, this.training.getIdealSize(), true);
```

Neste ponto a rede neural já esta criada e pronta para ser treinada. O Encog possui vários algoritmos de treinamento tanto supervisionado quanto não supervisionado, nesse projeto foi utilizado o algoritmo RProp. A rede neural é iniciada com pesos e limiares aleatórios. Algumas vezes, o conjunto aleatório de pesos e limiares fará o treinamento da rede neural estagnar, nesta situação o melhor é repor um novo conjunto de valores aleatórios e começar a treinar novamente. Para lidar com situações como essa o Encog permite que estratégias de treinamento sejam adicionadas. Nesse projeto foi utilizada a estratégia ResetStrategy, esta possui dois parâmetros, o primeiro afirma que o erro mínimo que a rede deve atingir antes de ser zerado automaticamente para novos valores aleatórios. O segundo parâmetro especifica o número de ciclos permitidos para a rede atingir essa taxa de erro. Se o numero especificado de ciclos é atingido, então os pesos e limiares serão randomizados. O código 3.5 adiciona a estratégia reset ao treinamento.

Código 3.5: Adicionando a Strategia para o treinamento.

```
train.addStrategy(new ResetStrategy(strategyError, strategyCycles));
```

Após definido o algoritmo de treinamento e a estratégia que será utilizada, o treinamento da rede deve ser iniciado, com um código como o 3.6, onde a cada iteração o método iteration é chamado atualizando os pesos da rede, e as iterações continuam enquanto o erro atual da rede for menor que o erro especificado ou que o número de épocas (epoch) seja menor que o limite.

Código 3.6: Iniciando o treinamento da rede neural.

```
int epoch = 0;
do {
            train.iteration();
            System.out.println("Epoch \#"
                    + epoch + " Error:" + train.getError());
            epoch++;
} while (train.getError() > erro \&\& epoch < limite);
```

Após isto a rede neural está treinada e pronta para reconhecer as imagens. Primeiramente a imagem a ser reconhecida deve ser carregada para um objeto ImageNeuralData, e depois redimensionada para as dimensões corretas utilizando o método downsample, veja código 3.7.

Código 3.7: Carregando a imagem e redimensionando.

```
final File file = new File(filename);
final Image img = ImageIO.read(file);
final ImageNeuralData input = new ImageNeuralData(img);

input.downsample(this.downsample, false, this.downsampleHeight, this.
    downsampleWidth, 1, -1);
```

Depois de redimensionada a imagem está pronta para ser reconhecida pela rede, através do método winner, conforme pode ser observado no código 3.8.

Código 3.8: Reconhecendo a Imagem.

```
final int winner = this.network.winner(input);
System.out.println("What is: " + filename + ", it seems to be: "
this.neuron2identity.get(winner));
```

O sistema utilizado nesse projeto foi criado com base nas especificações acima.

3.5 Especificação dos Testes

Para a realiazação dos testes foi utilizado um computador pessoal macBook Pro, com processador Intel Core i5, memória de 4 GB 1333 MHz DDR3 e sistema operacional Mac OS X Versão 10.6.8. Não foi utilizado outra arquitera computacional para os testes, pois o objetivo é avaliar o desempenho do sistema em relação a taxa de acertos e não o desempenho de processamento ou tempo. Os testes foram realizados a partir do banco de imagens ORL, do qual foram selecionadas 100 imagens, sendo 10 indivíduos, onde cada indíviduo possui 10 imagens distintas. Para cada indíviduo foi selecionado aleatoriamente 5 imagens para treino e 5 imagens para o reconhecimento.

Objetivando alcançar a melhor estrutura de rede para a aplicação proposta, este trabalho testou várias estruturas possíveis para a rede, variando a entrada da rede, quantidade de camadas intermediárias(ocultas) e número de neurônios em cada camada intermediária. Para a entrada da rede foi considerada três dimensões de imagens, 23x23, 46x46 e 92x92. A dimensões das imagens também são responsáveis pelo dimensionamento da camada de entrada, por exemplo a rede com entrada 23x23, a camada de entrada deve ter 1587 neurônios, que correnssponde altura x largura x 3(que representa os níveis RGB), as outras dimensões de entrada seguem o mesmo cálculo, obtendo 6.348 neurônios para entrada de 46x46 e 25.392 neurônios para a rede com entrada de 92x92. A camada de saída deve ser dimensionada de acordo com a quantidade de padrões a ser reconhecidos, sendo um neurônio para cada padrão a ser reconhecido. Considerando que este trabalho será testado para reconhecer 10 indíviduos, a camada de saída sempre possuirá 10 neurônios. A quantidade de neurônios nas camadas intermediárias foram escolhidos manualmente, para as redes com uma camada intermediária foram utilizados 100, 200, 300, 400, e 500 neurônios na camada intermediária. Enquanto que para as redes com duas camadas intermediárias, para na primeira rede foi utilizado 200 neurônios na primeria camada e 100 na segunda camada, na segunda 300 e 150, na terceira 400 e 200, na quarta 500 e 250. No total foram testadas 27 variações da rede, onde cada variação foi testada quatro vezes e feita a média aritmética simples dos resultados.

3.6 Resultados Obtidos

Os testes foram realizados seguindo as especificações acima, os resultados obtidos foram relacionados na forma de tabelas e gráficos. Primeiramente foi testado a rede com uma camada intermediária, variando a quantidade de neurônios, para as dimensões 23x23, 46x46 e 92x92, onde os resultados podem ser observados nas tabelas 2, 3 e 4, respectivamente.

Tabela 2: Resultado com imagens de 23x23 pixels e uma camada oculta de neurônios.

Neuronios	Qtd. Imagens	Acertos	Acertos %	Tempo Treinamento (ms)
100	50	35	70,00	2884,50
200	51	37	74,00	9363,50
300	52	37,5	75,00	33012,25
400	53	37	74,00	79919,75
500	54	36,75	73,50	98864,50

Tabela 3: Resultado com imagens de 46x46 pixels e uma camada oculta de neurônios.

Neurônios	Qtd. Imagens	Acertos	Acertos %	Tempo Treinamento (ms)
100	50	35,75	71,50	47780,75
200	50	37,00	74,00	79363,25
300	50	37,25	74,50	161516,25
400	50	37,25	74,50	329070,50
500	50	36,50	73,00	693410,50

Tabela 4: Resultado com imagens de 92x92 pixels e uma camada oculta de neurônios.

Neurônios	Qtd. Imagens	Acertos	Acertos %	Tempo Treinamento (ms)
100	50	34,75	69,50	553136,00
200	50	36,00	72,00	1116350,25
300	50	36,25	72,50	3074872,75
400	50	36,75	73,50	5602922,50
500	50	36,25	72,50	5475441,75

Na figura 6 pode ser observado o gráfico com a relação dos "números de neurônios"X "porcentagem de acertos", para cada uma das dimensões testadas, considerando as redes de uma camada intermediária.

Em seguida pode ser observado os testes para a rede com duas camadas intermediárias, variando a quantidade de neurônios, para as dimensões 23x23, 46x46 e 92x92, onde os resultados podem ser observados nas tabelas 5, 6 e 7, respectivamente.

Tabela 5: Resultado com imagens de 23x23 pixels e duas camadas oculta de neurônios.

Neurônios	Qtd. Imagens	Acertos	Acertos %	Tempo Treinamento (ms)
200-100	50	34,50	69,00	10277,50
300-150	50	36,25	72,50	39728,00
400-200	50	35,75	71,50	108500,50
500-250	50	34,75	69,50	216042,00

A figura 7 mostra o gráfico como a relação "números de neurônios"X "porcentagem de acertos", para as redes com duas camadas intermediárias.

Figura 6: Gráfico "número de neurônios"X "porcentagem de acertos", para a rede com uma camada intermediária.

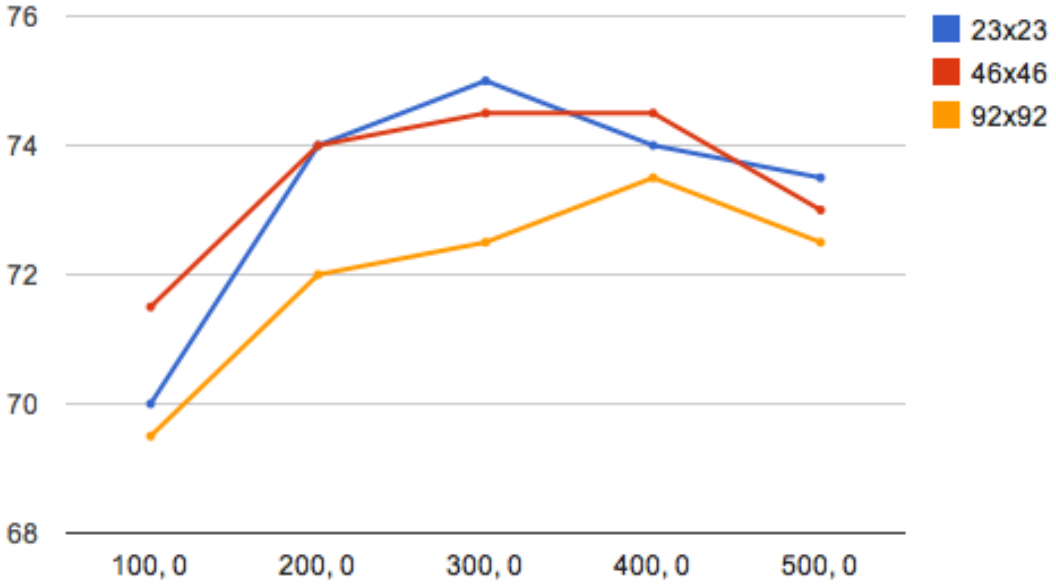


Tabela 6: Resultado com imagens de 46x46 pixels e duas camadas oculta de neurônios.

Neurônios	Qtd. Imagens	Acertos	Acertos %	Tempo Treinamento (ms)
200-100	50	34,25	68,50	106505,00
300-150	50	34,75	69,50	488760,25
400-200	50	35,75	71,50	1442746,25
500-250	50	35,00	70,00	1113016,50

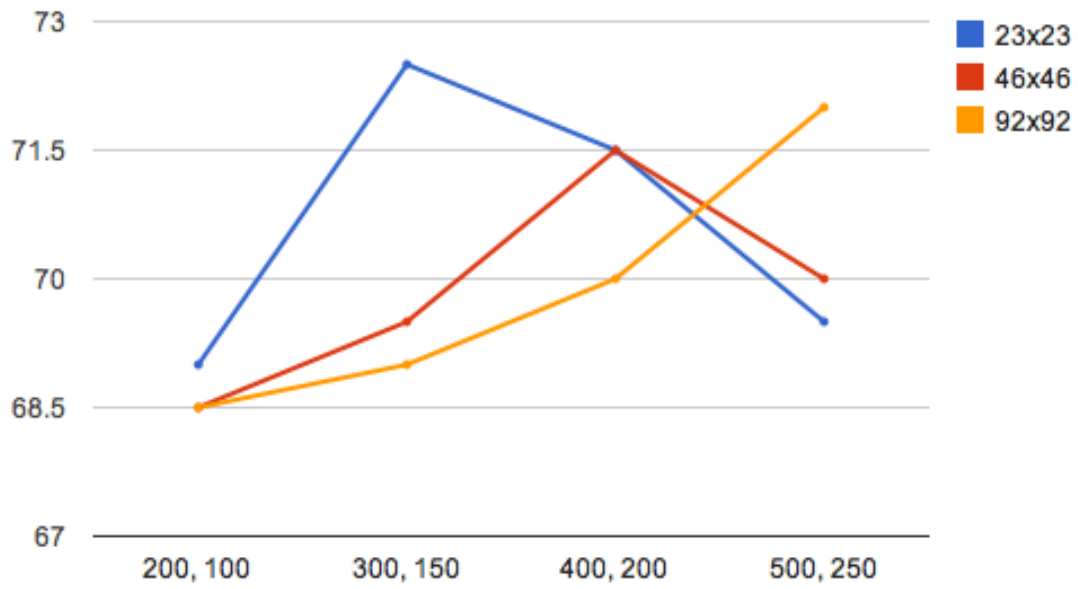
Tabela 7: Resultado com imagens de 92x92 pixels e duas camadas oculta de neurônios.

Neurônios	Qtd. Imagens	Acertos	Acertos %	Tempo Treinamento (ms)
200-100	50	34,25	68,50	1333327,25
300-150	50	34,50	69,00	1791875,00
400-200	50	35,00	70,00	3607281,00
500-250	50	36,00	72,00	3349097,50

3.7 Considerações sobre os Testes

Com base nos testes realizados, pode ser observado uma taxa de acerto variando entre 68.5% e 75%, apesar do resultado pouco satisfatório, este já se encontra dentro da faixa acertos média obtidas em trabalhos de área, que está entre 75 – 98% de acertos(BARRETT, 1998). A maior taxa de acerto obtida foi de 75%, alcançada pela rede com entrada de 23x23 e uma camada intermediária de 300 neurônios.

Figura 7: Gráfico "número de neurônios"X "porcentagem de acertos", para rede com duas camadas intermediárias.



Observando os resultados apresentados para as redes com uma camada intermediária, pode-se notar que conforme é aumentado a quantidade de neurônios na camada intermediária aumenta também a taxa de acertos da rede, porém isso acontece até um determinado ponto, onde ocorre o pico das taxas de acertos, após esse ponto as taxas de acertos da rede voltam a cair. Com isso conclui-se que no início a rede possui uma baixa taxa de acertos pois existem poucos neurônios na camada intermediária para representar o padrão de entrada, conforme vai aumentando o número de neurônios, o padrão de entrada fica melhor representado pela rede obtendo uma taxa de acertos maior. No entando quando o número de neurônios começa a subir demais a rede começa a perder capacidade de generalização fazendo com que a taxa de acertos díminua. Para todas dimensões de entrada dados foi observado esse mesmo comportamento, diferindo apenas o ponto onde ocorre o pico da taxa de acertos. Para as redes com entrada 23x23 a maior taxa de acertos acontece com 300 neurônios na camada intermediária e para as redes com entradas 46x46 e 92x92 o pico acontece com 400 neurônios na camada intermediária. Assim é possível chegar na conclusão de que quanto maior a entrada da rede, maior deve ser a quantidade de neurônios nas camadas intermediárias para representação do padrão de entrada. As redes com duas camadas intermediárias demonstraram um comportamento semelhante aos das redes com uma camada intermediária, porém apresentaram uma maior diferença entre os valores e picos de acertos comparando as diferentes dimensões de entrada.

É importante também resaltar que para essa aplicação as redes com uma camada intermediária obteviveram um resultado melhor que as redes com duas camadas intermediárias. Tanto nas redes com uma, quanto nas redes com duas camadas intermediárias, a dimensão para os dados de entrada que apresentou melhor desempenho foi 23x23, sendo que para as redes com uma camada intermediária a que apresentou maior taxa acerto foi a rede com 300 neurônios na camada intermediária com uma taxa de acerto de 75% enquanto que para as redes com duas camadas a que apresentou maior taxa de acerto foi a rede com 300 e 150 neurônios nas camadas intermediárias, com taxa de acerto de 72.5%.

4 Conclusões

Este trabalho teve como objetivos principais, validar uso das redes neurais aplicadas no problema do reconhecimento facial e implementar um sistema capaz de reconhecer a face humana. Estes objetivos foram alcançados com a implementação da rede neural e os testes de desempenho para o problema proposto.

Neste trabalho foi possível demonstrar a eficência de uma rede do tipo MLP(Multilayer Perceptron) aplicada na resolução do problema do reconhecimento facial. Considerando o melhor caso, a rede proposta conseguiu uma taxa de acerto de 75% o que está na faixa média encontrada nos trabalhos da àrea. Foi possível também demonstrar alguns pontos interessantes em relação as variações da rede proposta. O primeiro ponto interessante notado foi o fato de que uma imagem maior como entrada da rede não implica em um melhor desempenho. Também foi possível demonstrar que para este tipo de abordagem as redes com uma camada intermediária demonstaram um desempenho melhor do que as redes com duas camadas intermediárias. Outro ponto interessante é a curva da taxa de acertos que aumenta com o número de neurônios das camadas intermediária e depois de certo ponto a taxa de acertos começa a diminuir mostrando uma diminuição na capacidade de generalização da rede.

Para trabalhos futuros sugere-se buscar uma variação da rede que possa obter uma maior taxa de acertos, uma abordagem interessante seria combinar a rede existente com alguma técnica de extração de características como Discrete Wavelet Transform (DWT), Discrete Radon Transform (DRT), Discrete Cosine Transform (DCT).

Referências

ABATE, A. et al. 2D and 3D face recognition: A survey. *Pattern Recognition Letters*, Elsevier Science Inc., New York, NY, USA, v. 28, n. 14, p. 1885–1906, out. 2007. ISSN 01678655. Disponível em: <http://dx.doi.org/10.1016/j.patrec.2006.12.018>.

AT&T. *The Database of Faces, Cambridge University Computer Laboratory. Disponível em: <http://www.cl.cam.ac.uk/ research/dtg/attarchive/facedatabase.html>. Acesso em 10 Maio 2011.* 1992. Disponível em: <http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html>.

AZEVEDO, F. M.; BRASIL, L. M.; OLIVEIRA, R. C. L. *Redes Neurais com Aplicações em Controle e em Sistemas Especialistas.* [S.l.]: VISUAL BOOKS, 2000. ISBN 8575020056.

BARRETT, W. A. A survey of face recognition algorithms and testing results. *ACM Transactions on Computer Systems*, 1998.

BARTLETT, M. S.; LADES, H. M.; SEJNOWSKI, T. J. *Independent Component Representations for Face Recognition*. 1998.

BELHUMEUR, P. N.; HESPANHA, J. a. P.; KRIEGMAN, D. J. *Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection*. 1997. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.10.3247>.

BRAGA, A.; CARVALHO, A. C.; LUDERMIR, T. B. *Redes Neurais Artificiais: Teoria e aplicações*. [S.l.]: LTC Editora, 2000.

BRAGA, A.; CARVALHO, A. C.; LUDERMIR, T. B. *Redes Neurais Artificiais: Teoria e aplicações*. LTC Editora, 2007. ISBN 9788521615644. Disponível em: <http://www.worldcat.org/isbn/9788521615644>.

BRUNELLI, R.; POGGIO, T. Face recognition: Features versus templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 15, p. 1042–1052, 1993.

CARPENTER, G. A.; GROSSBERG, S. The art of adaptive pattern recognition by a self-organizing neural network. *Computer*, IEEE, v. 21, n. 3, p. 77–88, 1988. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=33>.

CHELLAPPA, R.; WILSON, C. L.; SIROHEY, S. Human and machine recognition of faces: a survey. *Proceedings of the IEEE*, v. 83, n. 5, p. 705–741, maio 1995. ISSN 00189219. Disponível em: <http://dx.doi.org/10.1109/5.381842>.

COOTES, T. F. et al. Active Shape Models-Their Training and Application. *Computer Vision and Image Understanding*, v. 61, n. 1, p. 38–59, jan. 1995. Disponível em: <http://www.sciencedirect.com/science/article/B6WCX-45NJT56-1K/2/1933a32b2a89e4c3f032ba0718f4ef03>.

COTTRELL, G.; FLEMING, M. Face recognition using unsupervised feature extraction. In: *International Symposium on Neural Networks*. [S.l.: s.n.], 1990.

COX, I. J.; GHOSN, J.; YIANILOS, P. Feature-based face recognition using mixture-distance. *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE Comput. Soc. Press, p. 209–216, 1996. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=517076>.

HALLINAN, P. W. Recognizing human eyes. In: ____. *Geometric Methods in Computer Vision*. SPIE, 1991. v. 1570, n. 1, p. 214–226. Disponível em: <http://link.aip.org/link/?PSI/1570/214/1>.

HAYKIN, S. *REDES NEURAIS - PRINCIPIOS E PRATICA*. BOOKMAN COMPANHIA ED, 2001. ISBN 9788573077186. Disponível em: <http://books.google.com.br/books?id=lBp0X5qfyjUC>.

HEATON, J. *Programming Neural Networks with Encog 2 in Java*. [S.l.]: Heaton Research, Inc., 2010. ISBN 1604390077, 9781604390070.

HEINEN, M. R. Autenticação de Assinaturas Online utilizando Redes Neurais Artificiais. In: ____. *Autenticação de Assinaturas Online utilizando Redes Neurais Artificiais*. São Leopoldo: Universidade do Vale do Rio dos Sinos, 2002. cap. 3, p. 24–40.

KANADE, T. Picture processing by computer complex and recognition of human faces. 1973.

KOHONEN, T. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, Springer, v. 43, n. 1, p. 59–69, 1982. Disponível em: <http://www.springerlink.com/content/m8816487026vh131>.

LAWRENCE, S. et al. Face recognition: a convolutional neural-network approach. *Neural Networks, IEEE Transactions on*, v. 8, n. 1, p. 98–113, ago. 2002. Disponível em: <http://dx.doi.org/10.1109/72.554195>.

LI, S.; JAIN, A. *Handbook of face recognition*. Springer, 2005. (Springer eBooks collection: Computer science). ISBN 9780387405957. Disponível em: <http://books.google.com.br/books?id=amVDaTdgKYcC>.

MENDEL, J. M.; MCLAREN, R. W. Reinforcement learning control and pattern recognition systems. In: *Adaptive, Learning and Pattern Recognition Systems: Theory and Applications*. [S.l.]: Academic Press, 1970. p. 287–318.

MINSKY, M.; PAPERT, S. *Perceptrons: an introduction to computational geometry*. MIT Press, 1972. Disponível em: <http://books.google.com.br/books?id=36E0QgAACAAJ>.

PIAZENTIN, D.; DUARTE, M. Troca de chaves criptográficas com redes neurais artificiais. *Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg)*, p. 310–319, 2011.

RIZK, M.; TAHA, A. Analysis of neural networks for face recognition systems with feature extraction to develop an eye localization based method. *IEEE Electronics, Circuits and Systems, 2002. 9th International Conference on*, v. 3, p. 847 – 850, 2002.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organisation in the brain. *Psychological Review*, v. 65, p. 368–408, 1958.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *Nature*, Nature Publishing Group, v. 323, n. 6088, p. 533–536, out. 1986. Disponível em: <http://dx.doi.org/10.1038/323533a0>.

RUSSELL, S.; NORVIG, P. *Artificial intelligence: a modern approach*. Prentice Hall, 2003. (Prentice Hall series in artificial intelligence). ISBN 9780137903955. Disponível em: <http://books.google.com.br/books?id=KI2WQgAACAAJ>.

SWETS, D. L.; WENG, J. Discriminant analysis and eigenspace partition tree for face and object recognition from views. In: *IEEE International Conference on Automatic Face and Gesture Recognition*. [S.l.: s.n.], 1996. p. 192–197.

SWETS, D. L.; WENG, J. Using discriminant eigenfeatures for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 18, p. 831–836, 1996.

TURK, M.; PENTLAND, A. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, v. 3, p. 71–86, 1991.

TURK, M. A.; PENTLAND, A. P. Face recognition using eigenfaces. In: *Computer Vision and Pattern Recognition*. [S.l.: s.n.], 1991.

WIDROW, B.; HOFF, M. E. Adaptive switching circuits. *1960 IRE WESCON Convention Record*, IRE, Part 4, n. 104, p. 96–104, 1960. Disponível em: <http://isl-www.stanford.edu/ widrow/papers/c1960adaptiveswitching.pdf>.

XIAO, Y. et al. Recognition of facial expressions using 2d dct and neural network. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, John Wiley Sons, Inc., v. 82, n. 7, p. 1–11, 1999. ISSN 1520-6440. Disponível em: <http://dx.doi.org/10.1002/(SICI)1520-6440(199907)82:7<1::AID-ECJC1>3.0.CO;2-E>.

ZHAO, W. et al. Face recognition: A literature survey. *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 35, n. 4, p. 399–458, dez. 2003. ISSN 0360-0300. Disponível em: <http://dx.doi.org/10.1145/954339.954342>.

ZURADA, J. *Introduction to artificial neural systems*. West, 1992. ISBN 9780314933911. Disponível em: <http://books.google.com/books?id=wLFQAAAAMAAJ>.

APÊNDICE A – Implementação do Sistema de Reconhecimento Facial

Código A.1: Implementação do Sistema de Reconhecimento Facial

```
1  # -*- coding: utf-8 -*-
2
3  package face.recog;
4
5
6  import java.awt.Image;
7  import java.io.File;
8  import java.io.IOException;
9  import java.util.ArrayList;
10 import java.util.HashMap;
11 import java.util.List;
12 import java.util.Map;
13
14 import javax.imageio.ImageIO;
15
16 import org.encog.neural.data.NeuralData;
17 import org.encog.neural.data.basic.BasicNeuralData;
18 import org.encog.neural.data.image.ImageNeuralData;
19 import org.encog.neural.data.image.ImageNeuralDataSet;
20 import org.encog.neural.networks.BasicNetwork;
21 import org.encog.neural.networks.training.propagation.resilient.
       ResilientPropagation;
22 import org.encog.neural.networks.training.Train;
23
24 import org.encog.neural.networks.training.strategy.ResetStrategy;
25 import org.encog.util.downsample.Downsample;
26 import org.encog.util.downsample.RGBDownsample;
27 import org.encog.util.downsample.SimpleIntensityDownsample;
28 import org.encog.util.simple.EncogUtility;
29
```

```
30 public class FaceRecog {
31
32     private final List<ImagePair> imageList = new ArrayList<ImagePair>();
33     private final Map<String, Integer> identity2neuron = new HashMap<String
           , Integer>();
34     private final Map<Integer, String> neuron2identity = new HashMap<
           Integer, String>();
35     private ImageNeuralDataSet training;
36     private int outputCount;
37     private int downsampleWidth;
38     private int downsampleHeight;
39     private BasicNetwork network;
40     private Downsample downsample;
41     private String msg;
42
43     class ImagePair {
44
45         private final File file;
46         private final int identity;
47
48         public ImagePair(final File file, final int identity) {
49             super();
50             this.file = file;
51             this.identity = identity;
52         }
53
54         public File getFile() {
55             return this.file;
56         }
57
58         public int getIdentity() {
59             return this.identity;
60         }
61     }
62
63     public static void main(final String[] args) {
64         try {
65             final ImageNets program = new ImageNets();
66             program.createTraining(23, 23, "RGB");
67             program.input("./faces/id2-f1.png", "Id2");
68             program.input("./faces/id3-f1.png", "Id3");
69             program.input("./faces/id1-f1.png", "Id1");
70             program.network(256, 16);
```

```
71              program.train("console", 1, 0.25, 20);
72              program.whatIs("./faces/id1-f2.png");
73              program.whatIs("./faces/id3-f2.png");
74              program.whatIs("./faces/id2-f2.png");
75          } catch (final Exception e) {
76              e.printStackTrace();
77          }
78      }
79
80      private int assignIdentity(final String identity) {
81
82          if (this.identity2neuron.containsKey(identity.toLowerCase())) {
83              return this.identity2neuron.get(identity.toLowerCase());
84          }
85
86          final int result = this.outputCount;
87          this.identity2neuron.put(identity.toLowerCase(), result);
88          this.neuron2identity.put(result, identity.toLowerCase());
89          this.outputCount++;
90          return result;
91      }
92
93      public void createTraining(int width, int height, final String strType)
            {
94
95          this.downsampleHeight = width;
96          this.downsampleWidth = height;
97
98          if (strType.equals("RGB")) {
99              this.downsample = new RGBDownsample();
100             System.out.println("rgb");
101         } else {
102             this.downsample = new SimpleIntensityDownsample();
103             System.out.println("simple");
104         }
105
106         this.training = new ImageNeuralDataSet(this.downsample, false, 1,
                -1);
107         System.out.println("Training set created");
108     }
109
110     public String input(final String image, final String identity) {
111
```

```
112         final int idx = assignIdentity(identity);
113         final File file = new File(image);
114 
115         this.imageList.add(new ImagePair(file, idx));
116 
117         System.out.println("Added input image:" + image + " idx: " + idx);
118         return "Added input image:" + image + " idx: " + idx;
119     }
120 
121     public void network(final int hidden1, final int hidden2) throws
        IOException {
122         System.out.println("Downsampling images...");
123 
124         for (final ImagePair pair : this.imageList) {
125             final NeuralData ideal = new BasicNeuralData(this.outputCount);
126             final int idx = pair.getIdentity();
127             for (int i = 0; i < this.outputCount; i++) {
128                 if (i == idx) {
129                     ideal.setData(i, 1);
130                 } else {
131                     ideal.setData(i, -1);
132                 }
133             }
134 
135             final Image img = ImageIO.read(pair.getFile());
136             final ImageNeuralData data = new ImageNeuralData(img);
137             this.training.add(data, ideal);
138         }
139 
140         this.training.downsample(this.downsampleHeight, this.
            downsampleWidth);
141         System.out.println(this.training.getInputSize() + " - size");
142 
143 
144         this.network = EncogUtility.simpleFeedForward(this.training.
            getInputSize(), hidden1, hidden2,
145                 this.training.getIdealSize(), true);
146         System.out.println("Created network: " + this.network.toString());
147     }
148 
149     public void train(final String mode, final int minutes, final double
        strategyError, final int strategyCycles) {
150 
```

```
151         System.out.println("Training Beginning... Output patterns="
152                 + this.outputCount);
153
154
155         final Train train = new ResilientPropagation(this.network, this.
                training);
156         train.addStrategy(new ResetStrategy(strategyError, strategyCycles))
                ;
157
158         long ini = System.currentTimeMillis();
159         int epoch = 0;
160         do {
161             train.iteration();
162             System.out.println("Epoch #"
163                     + epoch + " Error:" + train.getError());
164             epoch++;
165         } while (train.getError() > 0.0001 && epoch < 1000);
166
167         long fim = System.currentTimeMillis();
168         this.msg = "Treino > Duracao: "+ (fim - ini) + " Epocas:"+epoch;
169         System.out.println(msg);
170
171         System.out.println("Training Stopped...");
172     }
173
174     public String whatIs(final String filename) throws IOException {
175         final File file = new File(filename);
176         final Image img = ImageIO.read(file);
177         final ImageNeuralData input = new ImageNeuralData(img);
178         input.downsample(this.downsample, false, this.downsampleHeight,
179                 this.downsampleWidth, 1, -1);
180         final int winner = this.network.winner(input);
181         String id = filename.substring(filename.lastIndexOf("/") + 1);
182         id = id.substring(0, id.indexOf("-"));
183
184         String out = "What is: " + filename + ", it seems to be: " + winner
                + " - "
185                 + this.neuron2identity.get(winner) + " - ideal: " + id;
186         if (id.equals(this.neuron2identity.get(winner))) {
187             out += " Sucess!";
188         } else {
189             out += "Erro!";
190         }
```

```
191             System.out.println(out);
192             return this.neuron2identity.get(winner);
193         }
194 
195         public String getMsg(){
196             return this.msg;
197         }
198 }
```