UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE SISTEMAS DE INFORMAÇÃO – BACHARELADO

PROTÓTIPO DE SISTEMA PARA AUTENTICAÇÃO DE IMPRESSÕES DIGITAIS UTILIZANDO O MODELO DE REDES NEURAIS ARTIFICIAIS *CASCADE CORRELATION*.

RICARDO INÁCIO MAIOLA

BLUMENAU
2005

2005/2-09

RICARDO INÁCIO MAIOLA

PROTÓTIPO DE SISTEMA PARA AUTENTICAÇÃO DE IMPRESSÕES DIGITAIS UTILIZANDO O MODELO DE REDES NEURAIS ARTIFICIAIS *CASCADE CORRELATION*.

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Sistemas de Informação — Bacharelado.

Prof. Jomi Fred Hübner, Orientador

BLUMENAU
2005

2005/2-12

PROTÓTIPO DE SISTEMA PARA AUTENTICAÇÃO DE IMPRESSÕES DIGITAIS UTILIZADO O MODELO DE REDES NEURAIS ARTIFICIAIS *CASCADE CORRELATION*.

Por

RICARDO INÁCIO MAIOLA

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Jomi Fred Hübner – Orientador, FURB
Membro:	Prof. Mauro Marcelo Mattos – FURB
Membro:	Prof. Roberto Heinzle – FURB

Blumenau, 16 de dezembro de 2005

Dedico este trabalho a toda minha família e a todos meus amigos, especialmente os que me apoiaram durante o período de realização deste trabalho.

AGRADECIMENTOS

Aos meus pais, Jaime Maiola e Inês Mª dos Santos Maiola pela oportunidade de existência e, juntamente com minha irmã Carolina dos S. Maiola e minha tia Teresinha Maiola, pela educação dada durante toda minha vida e pelo incentivo dado durante a realização deste trabalho.

Aos meus amigos, em especial ao Jorge Luis Pamplona e Juliano José Depiné, que se demonstraram preocupados com o andamento deste projeto e me motivaram para continuar seguindo em frente.

Ao professor Mauro Marcelo Mattos pela sugestão deste tema e por todo o auxílio concedido durante todo este ano letivo.

Ao professor Jomi Fred Hübner pela orientação dada durante a realização deste trabalho e pela boa disposição em me atender.

Ao professor Fernando Santos Osório pela disponibilização do algoritmo *Cascade Correlation.*

“Os homens perdem a saúde para juntar dinheiro e depois perdem o dinheiro para recuperar a saúde. Por pensarem ansiosamente no futuro, esquecem o presente de tal forma que acabam por nem viver no presente nem no futuro. Vivem como se nunca fossem morrer e morrem como se nunca tivessem vivido”.

Confúcio.

RESUMO

Este trabalho apresenta o desenvolvimento de um protótipo de reconhecimento de impressões digitais que utiliza o modelo de redes neurais artificiais *Cascade Correlation*. Os dados de entrada apresentados à rede são obtidos através de imagens de impressões digitais pré-processadas, possibilitando o reconhecimento de uma impressão digital em meio a uma base de dados. Ao final do trabalho é feita uma análise dos resultados obtidos pelo desenvolvimento deste trabalho, concluindo se *Cascade Correlation* é um modelo de rede neural propício para a identificação de impressões digitais.

Palavras-chave: Redes neurais artificiais. Impressões digitais. *Cascade* Correlation. Treinamento. Reconhecimento.

ABSTRACT

This work presents the development of a fingerprints recognization software that uses the Cascade Correlation neural network model. The input data introduced to the net are gotten by preprocessed figerprints pictures, making possible the fingerprint recognization in a database. At the end of this work, it's made an analysis of the results gotten by the development of this work, concluding that Cascade Correlation is a suitable neural network model to fingerprints identification.

Key-words: Artificial neural networks. Fingerprints. Cascade Correlation. Trainning. Recognition.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

ID – Impressões Digitais

IID – Imagem de Impressão Digital

MLP – *Multi Layer Perceptron*

RN – Redes Neurais

RNA – Redes Neurais Artificiais

UML – *Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO

Há indícios de que a dactiloscopia (processo de leitura impressões digitais com a finalidade de identificação individual) já era utilizada no ano 650 durante a dinastia de Tang na China, no qual se determinava que o marido desse um documento à divorciada, autenticando-o com a sua impressão digital. Aproximadamente no ano de 1300, os chineses empregavam a impressão digital não só nos divórcios, como também nos casos de crimes. (Bombonatti, 2005).

Inicialmente a identificação através das impressões digitais (IDs) era feita por indivíduos treinados, ou seja, uma ou mais pessoas tinham arquivos de IDs impressos em papel e, quando conhecido o nome do indivíduo, pegava-se sua ficha e comparavam-se as minúcias e outras características da ID já destacadas nesta ficha com a ID fornecida, utilizando para isto lupa e materiais como régua e lápis. Quando não se tinha o nome do indivíduo a ser identificado com sua ficha correspondente, como por exemplo, um crime sem testemunha, este trabalho se tornava demorado e de paciência, onde muitas vezes o indivíduo não podia ser identificado entre tantas fichas (Gumz, 2002, p. 4).

A biometria é o ramo da ciência que estuda as medidas físicas dos seres vivos, daí o termo identificação biométrica para indicar as tecnologias que permitem a identificação das pessoas através dos traços físicos característicos e únicos de cada ser humano: os traços faciais, a íris, a impressão digital entre outros.

Durante os últimos anos as pessoas têm usado chaves, cartões, assinaturas e senhas para validar sua identidade. O grande problema destes artefatos é que podem ser esquecidos, roubados, perdidos, copiados, armazenados de maneira insegura e até utilizados por uma pessoa que não tenha autorização. Desta forma, não é surpreendente que o novo campo de atrações seja a biometria (SIM, 2005).

Com a evolução da tecnologia na era da informação, a sociedade vem se tornando um sistema eletronicamente conectado. Daí a necessidade de desenvolver sistemas que facilitem a vida das pessoas no cotidiano. Percebe-se nitidamente que as máquinas ocupam espaço considerável no mundo atual, substituindo o trabalho do ser humano. A implantação de uma tecnologia que contemple o uso da informática ao uso de bases de dados de impressões digitais e de seus dados cadastrais, tornará possível uma maior resposta à sociedade, como por exemplo, na elucidação de crimes. O método de identificação dactiloscópio é extremamente eficiente, seguro, de baixo custo, de aplicação potencialmente massificada e tem apresentado avanços principalmente com a incorporação dos recursos da informática. Porém, a comparação de impressões digitais não está restrita à área criminal, exercendo papel fundamental no reconhecimento de pessoas. Atualmente a verificação de impressões digitais pode ser empregada em sistemas de segurança, transações financeiras, controle de acesso a locais restritos, controle de freqüência de funcionários, acesso em redes corporativas, validação de documentos, autenticação de portadores de cartões e comprovação de identidade, entre muitas outras aplicações. (Costa, 2001, p. 15).

O reconhecimento das imagens dos dedos é feito de forma aproximada, fazendo com que determinado percentual dos reconhecimentos possa não ser confiável e até mesmo incorreto. Sendo assim, diversos estudos são realizados para que o percentual de erros nos reconhecimentos torne-se cada vez mais insignificante.

Redes Neurais Artificiais (RNAs) são técnicas computacionais que apresentam um modelo matemático inspirado na estrutura neural de organismos inteligentes e que adquirem conhecimento através da experiência. As Redes Neurais Artificiais surgiram como uma tentativa de reproduzir o funcionamento do cérebro humano, e assim se desenvolver máquinas capazes de realizar muitas funções que antes só eram possíveis de serem realizadas através da intervenção humana (Rocha, 2003).

Um dos modelos de Redes Neurais Artificiais (RNA) mais utilizados na atualidade é o *Multi Layer Perceptron* (MLP) com *Backpropagation*, também chamado simplesmente de *Backpropagation*, proposto em 1986 por David Rumelhart através do livro *"Parallel Distributed Processing"* (Processamento Distribuído Paralelo) (HEINEN, 2002, p.19).

Parafraseando Fahlman (1988), "o algoritmo do *Backpropagation* possui diversos problemas e limitações e que tornam este método não totalmente confiável. Além disso, o modelo de *Backpropagation* exige que sejam feitas inúmeras simulações para serem obtidos os primeiros resultados confiáveis, sendo assim, nunca se sabe com antecedência se o aprendizado irá ocorrer de forma satisfatória com a configuração desenvolvida". Sentindo a necessidade de criar um modelo que solucionasse alguns dos problemas identificados no modelo *Backpropagation*, Fahlman propôs um novo modelo de RNA chamado de *Cascade Correlation*.

Este trabalho foca principalmente no desenvolvimento de um sistema de identificação de impressões digitais que utilize o algoritmo de RNA *Cascade Correlation*. Ao final do trabalho é feita uma análise em relação aos resultados obtidos nos trabalhos de Gumz (2002) e Matias (2004), nos quais utilizam o modelo de *Backpropagation* para solução deste mesmo problema.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é desenvolver um protótipo de sistema de reconhecimento de impressões digitais utilizando o modelo de RNA *Cascade Correlation.*

Os objetivos específicos do trabalho são:

a) desenvolver um sistema de identificação de impressões digitais que utilize o algoritmo de RNA *Cascade Correlation*;
b) fazer uma análise dos resultados obtidos para concluir se *Cascade Correlation* é um

modelo de rede neural favorável para identificação de impressões digitais.

1.2 ESTRUTURA DO TRABALHO

Este trabalho está organizado como descrito a seguir:

a) no capítulo 2 serão apresentados assuntos relacionados aos conceitos da biometria, dactiloscopia, redes neurais artificiais enfatizando os modelos de *Backpropagation* e *Cascade Correlation* e, no final, um levantamento de trabalhos correlatos ao que será abordado neste projeto;
b) no capítulo 3 será apresentado o processo de desenvolvimento do protótipo deste projeto, como suas estruturas, desenvolvimento do algoritmo e da interface, a metodologia e aplicação do protótipo, fragmentos de códigos, entre outros.;
c) no capítulo 4, será apresentada uma conclusão sobre a aplicação de *Cascade Correlation* no reconhecimento de impressões digitais juntamente com uma análise comparativa dos resultados dos trabalhos de Matias(2004) e Gumz(2002).

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados conceitos da biometria e da dactiloscopia para a compreensão do método de reconhecimento humano a partir de uma característica física própria e única de cada ser. Também serão apresentados conceitos de redes neurais, estendendo-se para os modelos de *Backpropagation* e *Cascade Correlation*. No final, serão citados alguns trabalhos correlatos e seus devidos resultados.

2.1 BIOMETRIA

Biometria é o estudo estatístico das características físicas dos seres vivos. Recentemente este termo também foi associado à medida de características físicas das pessoas como forma de identificá-las unicamente. Hoje a biometria é usada na identificação criminal, controle de ponto, controle de acesso, etc. Os sistemas chamados biométricos podem basear seu funcionamento em características de diversas partes do corpo humano, por exemplo: os olhos, a palma da mão ou as digitais do dedo (Wikipedia, 2005). O termo “biometria” tem origem grega que se forma através da junção das palavras: “bios” – vida e “métron” – medida ou comparação.

A Biometria ainda é vista como uma tecnologia futurista e distante do nosso cotidiano, porém, há centenas de anos, nossos antepassados já usavam os princípios básicos da verificação biométrica. Existem diversas referências históricas sobre indivíduos sendo identificados por características físicas e parâmetros como cicatrizes, critérios de mensuração física ou a combinação de características mais complexas como cor dos olhos, altura e assim por diante. Estes seriam freqüentemente utilizadas no setor de agricultura, onde grãos e provisões seriam estocados em uma central de reposições para movimentações futuras após

identificação dos proprietários (SEAMA, 2005).

2.1.1 Sistemas de Identificação Biométrica

Um sistema biométrico é essencialmente um sistema de reconhecimento de padrões que efetua a identificação pessoal pela determinação da autenticidade da característica biométrica registrada em posse do individuo (GUMZ, 2002).

Os sistemas biométricos identificam indivíduos pelas suas características comportamentais, como o reconhecimento de voz, de assinaturas manuscritas e da dinâmica da digitação; e pelas características físicas, como o reconhecimento da face, da íris, da retina, da geometria das mãos e das impressões digitais na qual é o foco deste trabalho.

A identificação pelas impressões digitais tem sido utilizada em várias aplicações como controle de acesso, caixas automáticos de bancos, registros de saúde, entre outras. Algumas de suas principais vantagens são a rapidez e a confiança, o baixo preço e o pequeno tamanho dos leitores e o fato de ele ser considerado pelos usuários como pouco intrusivo. Entretanto, algumas pessoas acham que, sendo requerido sua impressão digital, estão sendo tratadas como criminosas. (FIORESE, 2005);

Segundo Boreki (2003), sempre que falamos em sistemas de biometria, tem-se que ter em mente dois termos que indicarão o índice de segurança do sistema, são eles:

a) taxa de falsa rejeição: esta taxa indica qual é o percentual de indivíduos que não conseguirão passar pelo sistema, mesmo apresentando a biometria correta (a partir de uma aquisição da mesma pessoa, o sistema reacusará seu acesso). Um dos objetivos dos sistemas é sempre possuir uma pequena taxa de falsa rejeição;
b) taxa de falsa aceitação: este é um ponto muito perigoso nos sistemas, indica qual o percentual de tentativas, onde a amostra não corresponde ao mesmo indivíduo que

efetuou o cadastro, e mesmo assim, o sistema o identificará, ou seja, uma pessoa se passando pelo dono do código pode ser aceito em um ambiente protegido. Este valor deve ser o mais próximo possível de zero.

O nível de precisão configurado no algoritmo de comparação tem efeito direto nessas taxas. O modo como estas são determinadas é fundamental para a operação de qualquer sistema biométrico e assim deve ser considerado um fator primário na avaliação de sistemas biométricos. A configuração do valor limite para tolerância a estes erros é crítica no desempenho do sistema. A falsa rejeição causa frustração e a falsa aceitação causa fraude (PEREIRA, 2003).

A figura 1 retrata o ponto de exatidão do reconhecimento biométrico:

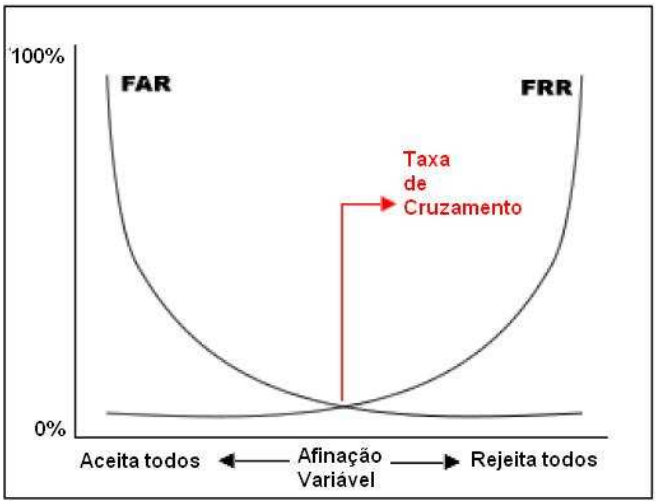

Figura 1: exatidão de cruzamento biométrico.
Fonte: adaptado de Pereira (2003).

2.1.2 Dactiloscopia

A Dactiloscopia é a ciência que trata a identificação de pessoas através da comparação de suas IDs. O termo Dactiloscopia deriva de dois elementos gregos *daktylos* = dedos e *skopêin* = examinar. A Dactiloscopia divide-se em civil, criminal e clínica. A Dactiloscopia civil trata da identificação das pessoas para expedição de documentos de identidade, tais

como cédula de identidade, passaporte, etc. A Dactiloscopia criminal reveste-se de três aspectos, a saber: indiciado em inquérito policial, expedição de documentos de idoneidade e aproveitamento das impressões papilares encontradas nos locais de crimes. Já a Dactiloscopia clínica estuda as perturbações que se verificam nos desenhos digitais. Perturbações estas resultantes de algumas doenças, ou exercício de certas profissões (KEHDY, 1968, p. 25-27 apud MATIAS, 2004, p.7).

Segundo APPES (2005), as postulações das impressões digitais são as seguintes:

a) perenidade: é o surgimento das papilas dérmicas desde o sexto mês de vida intra-uterina até a putrefação cadavérica quando se dá o descolamento do derma. Durante nossa existência, os desenhos papilares se conservam sempre os mesmos, não desaparecem, não se modificam, ao contrário do que se sucede com outras partes do corpo;
b) imutabilidade: os desenhos digitais permanecem idênticos a si mesmo, não mudando jamais. Segundo Edmund Locard (1912 apud APPES, 2005), os desenhos digitais não são modificáveis, nem patologicamente nem por vontade de seu portador. Queimaduras superficiais, queimaduras do segundo grau, com flictenas, não prejudicam a fixidez da impressão digital;
c) variabilidade: os desenhos digitais variam de dedo para dedo e de pessoa para pessoa. Segundo Edmund Locard (1912 apud APPES, 2005), os desenhos digitais nunca são idênticos em dois indivíduos;
d) classificabilidade: é a determinação exata do tipo e subtipo digital por meio de código formado por símbolos literais e numéricos convencionais, dado a cada desenho digital. Este código consiste numa fórmula datiloscópica. João Evangelista Purkinge foi o primeiro a se preocupar de maneira científica em

ordenar e classificar os desenhos digitais em grupos e tipos segundo os caracteres formados pelas linhas papilares.

Segundo Appol (2005), o datilograma, nome técnico do desenho digital, divide-se em 3 áreas, limitadas pelas linhas diretrizes, a saber:

a) basilar: é formada pelo conjunto de linhas existentes entre a prega interfalangeana[1] e a terceira linha abaixo do ramo descendente e ascendente do delta[2];
b) região nuclear: é formada pelo conjunto de linhas que circunscrevem o centro do datilograma, ou seguindo a diretriz superior até o ramo ascendente do delta;
c) região marginal: é formada pelo conjunto de linhas do ápice e das laterais do datilograma até a linha imediata que acompanha a diretriz superior do delta.

Estas regiões podem ser identificadas na figura 2.

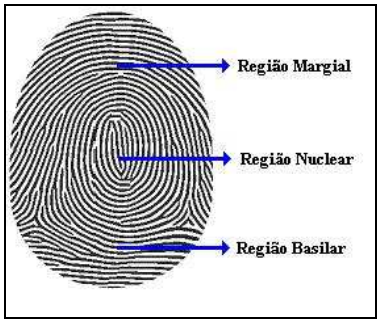

Figura 2: regiões dos sistemas de linhas.
Fonte: Matias (2004, p.9).

O que diferencia uma impressão digital de outra são saliências naturais da pele, formadas a partir do sexto mês de vida inter-uterina, que possuem o formato de linhas que terminam abruptamente ou se bifurcam.

Quando se entinta um dedo e o possua sobre uma superfície, a impressão criada é a das bordas entre os sulcos. A forma como essas bordas fluem é fator distintivo e criam características individualizadoras na impressão digital, chamadas minúcias, conforme

[1] Prega inerfalangeana é a região da derme que abrange a articulação da falange do dedo.

[2] Os deltas são ângulos formados pelas cristas papilares. Podem ser formados pela bifurcação de uma linha simples ou pela divergência de linhas paralelas.

demonstra a figura 3. Essas minúcias são os registros particulares do individuo que as possui, consistindo de terminações, bifurcações ou formas especificas. São essas minúcias que devem ser extraídas, mapeadas e comparadas com a impressão questionada para determinar o seu correspondente, registrado no banco de dado (Chang, 1999).

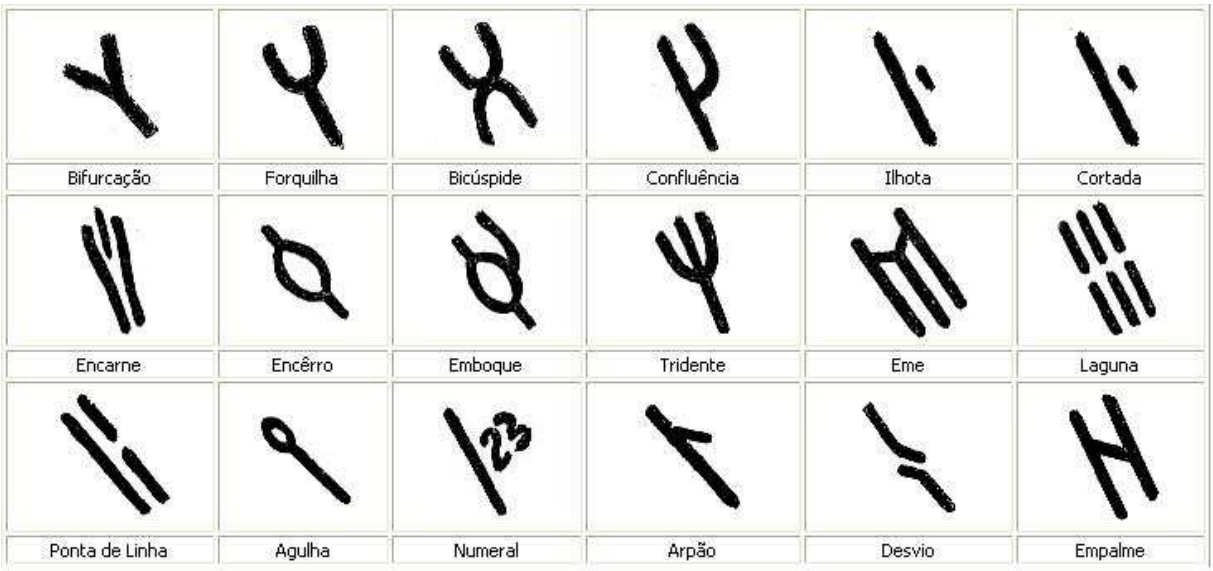

Figura 3: minúcias das impressões digitais
Fonte: adaptado de APPOL(2005).

Tavares Júnior (1991, p.32 apud MATIAS, 2004, p.6) afirma que para se confirmar a identidade de uma ID, deverão ser coincididos no mínimo doze minúcias, as quais devem ser encontradas da mesma forma, localização e mesma quantidade nas duas impressões em comparação. Também não pode haver minúcias que se encontram na impressão testemunha[3] e não estão presentes na impressão suspeita[4].

2.2 REDES NEURAIS

O cérebro humano é considerado o mais fascinante processador baseado em carbono existente, sendo composto por aproximadamente 10 bilhões neurônios. Todas as funções e

[3] Impressão testemunha é a encontrada em local de crime.
[4] Impressão suspeita é a tomada dos dedos do suspeito ou encontrada no arquivo datiloscópico.

movimentos do organismo estão relacionados ao funcionamento destas pequenas células. Os neurônios estão conectados uns aos outros através de sinapses, e juntos formam uma grande rede, chamada REDE NEURAL. As sinapses transmitem estímulos através de diferentes concentrações de Na+ (Sódio) e K+ (Potássio), e o resultado disto pode ser estendido por todo o corpo humano. Esta grande rede proporciona uma fabulosa capacidade de processamento e armazenamento de informação (DIN, 2005).

O sistema nervoso é formado por um conjunto extremamente complexo de neurônios. Nos neurônios a comunicação é realizada através de impulsos, quando um impulso é recebido, o neurônio o processa, e passado um limite de ação, dispara um segundo impulso que produz uma substância neurotransmissora o qual flui do corpo celular para o axônio (que por sua vez pode ou não estar conectado a um dendrito de outra célula). O neurônio que transmite o pulso pode controlar a freqüência de pulsos aumentando ou diminuindo a polaridade na membrana pós sináptica. Eles têm um papel essencial na determinação do funcionamento, comportamento e do raciocínio do ser humano. Ao contrário das redes neurais artificiais, redes neurais naturais não transmitem sinais negativos, sua ativação é medida pela freqüência com que emite pulsos, freqüência esta de pulsos contínuos e positivos. As redes naturais não são uniformes como as redes artificiais, e apresentam uniformidade apenas em alguns pontos do organismo. Seus pulsos não são síncronos ou assíncronos, devido ao fato de não serem contínuos, o que a difere de redes artificiais (DIN, 2005).

Os neurônios são compostos de três partes: dendritos, axônio e corpo celular ou soma. Os dendritos fazem o transporte de informações vindas de outros neurônios para dentro da célula. As informações são somadas no corpo celular, processando assim uma outra informação, que sai da célula através do axônio. Então esse axônio e outros axônios de outros neurônios vão se comunicar com os dendritos de uma determinada célula, montando uma rede. Essa comunicação recebe o nome de sinapse nervosa ou junção sináptica, que é a

unidade funcional básica para a construção de circuitos neurais biológicos e envolve a junção das membranas plasmáticas de dois neurônios de modo a formar uma junção pontual orientada do neurônio pré-sináptico para o pós-sináptico. O tamanho de uma junção sináptica é menor do que 1mm (MESONPI, 1999). A figura 4 trás uma representação ilustrativa de um neurônio humano e seus constituintes e a figura 5 ilustra o funcionamento de uma rede neural natural utilizando neurônios artificiais e suas respectivas conexões.

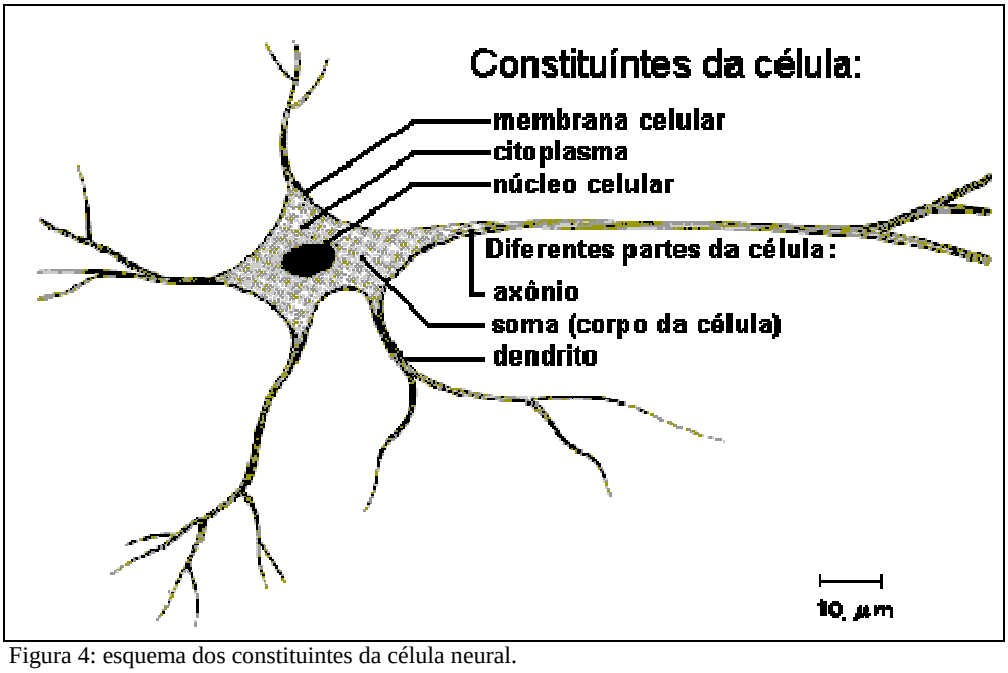

Figura 4: esquema dos constituintes da célula neural.
Fonte: DIN (2005).

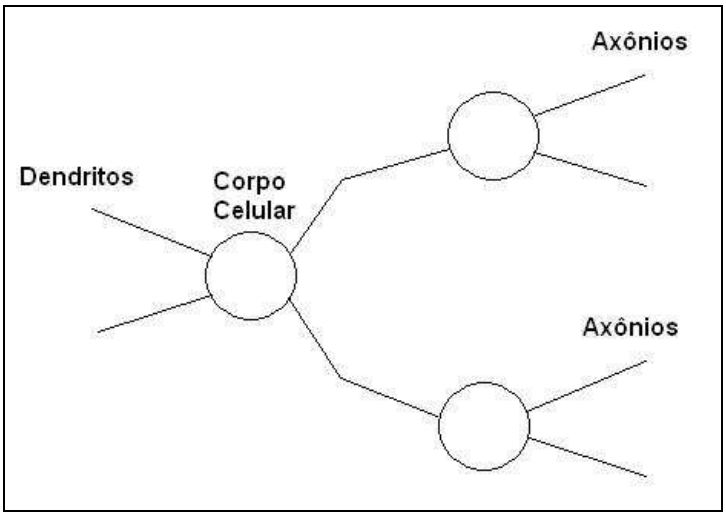

Figura 5: representação artificial de uma rede neural.
Fonte: adaptado de Mesonpi (1999).

Redes Neurais Artificiais são sistemas computacionais, de implementação em hardware ou software, que imitam as habilidades computacionais do sistema nervoso biológico, usando um grande número de simples neurônios artificiais interconectados. Os neurônios artificiais são emulações simplificadas dos neurônios biológicos, estes recebem informação de sensores ou de outros neurônios artificiais, produzindo operações simples sobre estes dados, e passam o resultado para outros neurônios artificiais (LOESCH, 1996 apud ZAPAROLI, 2002). Uma RN é uma coleção de neurônios dispostos de forma que configurem um aspecto específico. É com estes neurônios que a RN aprenderá as informações que serão fornecidas pelos canais de entrada dos neurônios. O aprendizado está distribuído por toda a rede, ou seja, por todos os neurônios (TAFNER, 1996).

O neurônio artificial (figura 6) é uma estrutura lógico-matemática que procura simular a forma, o comportamento e as funções de um neurônio biológico. Assim sendo, os dendritos foram substituídos por *entradas*, cujas ligações com o corpo celular artificial são realizadas através de elementos chamados de *peso* (simulando as sinapses). Os estímulos captados pelas entradas são processados pela *função de soma*, e o limiar de disparo do neurônio biológico foi substituído pela *função de transferência* (CEREBROMENTE, 2005).

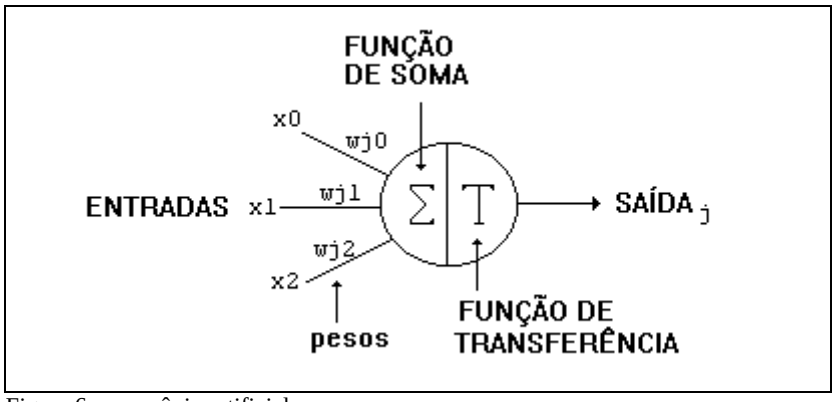

Figura 6: o neurônio artificial.
Fonte: Cerebromente (2005).

2.2.1 Perceptron

O modelo de RN *perceptron* foi o primeiro a ser desenvolvido. Possui um conceito de neurônio artificial, exemplificado na figura 6, que ainda hoje é usado. Cada neurônio computa uma soma ponderada de suas entradas, e passa esta soma em uma função não-linear com limiarização. Perceptrons tomam decisões, determinam se um padrão de entrada se encaixa ou não em um certo padrão (LOESCH, 1996 apud ZAPAROLI, 2002).

Segundo Lemes (2005), arquiteturas neurais são tipicamente organizadas em camadas. Usualmente as camadas são classificadas em três grupos:

a) Camada de entrada: onde os padrões são apresentados à rede;
b) Camadas Intermediárias: onde é feita a maior parte do processamento, através das conexões ponderadas; podem ser consideradas como extratoras ou moduladoras de características;
c) Camada de Saída: onde o resultado final é concluído e apresentado – veja a figura 7.

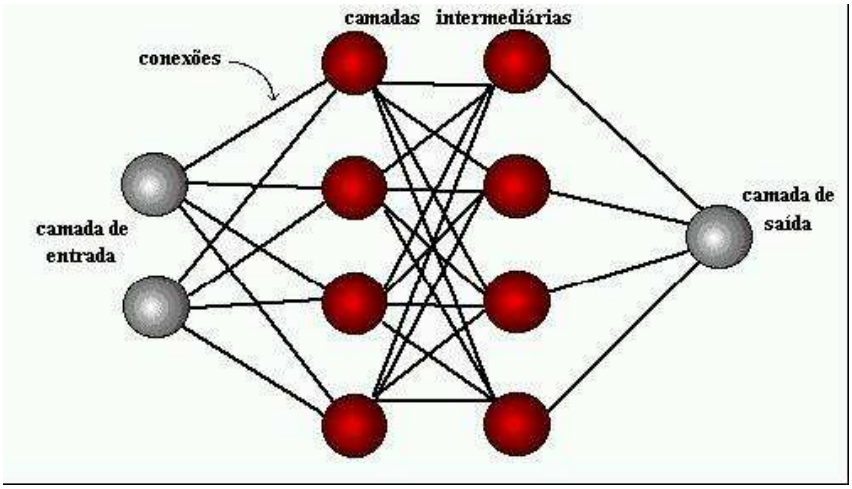

Figura 7: rede neural artificial.
Fonte: Lemes (2005).

O *Perceptron* trabalha da seguinte forma: quando um neurônio artificial é ativado, ele recebe os valores obtidos nas entradas, realiza a soma ponderada, envia o resultado para a

função de transferência, e assim se obtém o valor de saída. O valor de saída obtido é então comparado com a saída desejada, e o erro na saída é calculado diminuindo o valor da saída obtida pelo valor da saída desejada. Este erro é utilizado para o cálculo do ajuste dos pesos, dado pela fórmula:

$$\textbf{Peso\_Novo}(i) = \textbf{Peso\_Antigo}(i) + \frac{\beta * \textbf{Erro}(i) * \textbf{Entrada}(i)}{|\textbf{Entrada}(i)|}$$

onde Entrada é o valor recebido na entrada, Erro é o erro calculado na saída e β é o passo, que é um valor entre 0 e 1 que determina a velocidade do processo de aprendizado. Quando uma Rede Neural está sendo treinada, existe uma curva de erro a ela associada, como por exemplo, a curva mostrada na figura 8. Esta figura representa que para certos valores de pesos o erro é menor e para outros é maior, e conforme se altera os valores dos pesos para cima ou para baixo o erro vai sendo alterado de acordo com a curva de erro do problema que está sendo analisado. Cada problema tem a sua curva de erro própria, sendo que a curva da figura 7 apenas um exemplo ilustrativo (HEINEN, 2002).

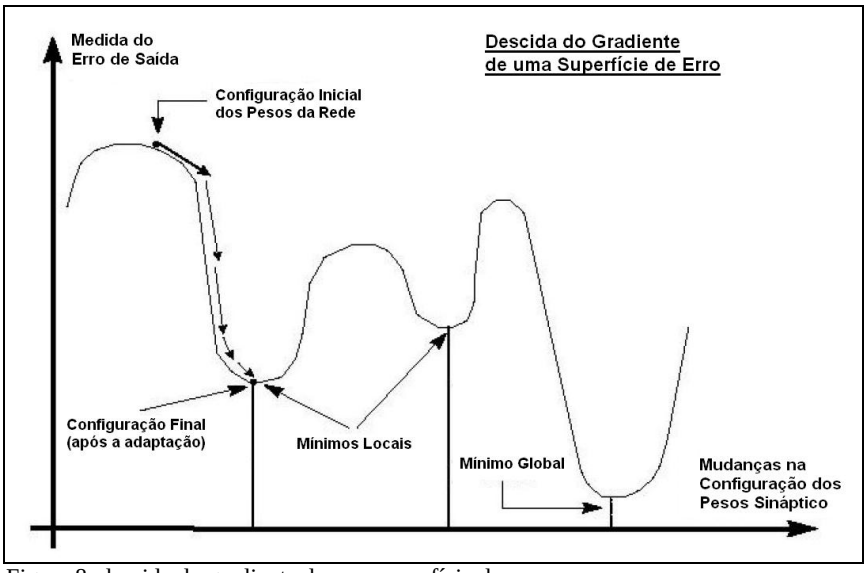

Figura 8: descida do gradiente de uma superfície de erro.
Fonte: adaptado de Heinen (2002).

Na figura 8 pode-se notar que existem espécies de “vales” na curva onde o erro é mais baixo que nos pontos vizinhos. Como os pesos são inicializados de forma aleatória, o erro inicial se situa em algum ponto da curva. A medida que os pesos vão sendo ajustados, vai se descendo na curva de erro, até que se atinja um ponto mínimo, de onde qualquer mudança que seja feita para baixo ou para cima fará com que o erro seja aumentado. Em uma curva de erro existe um mínimo global, que é o ponto onde ocorre o menor erro possível, e vários mínimos locais, que são mostradas na figura 8. O mínimo global é muito difícil de ser atingido, e mesmo que já se esteja nele não será possível de saber, pois é difícil conhecer o formato de toda a curva de erro do problema que está sendo analisado (HEINEN, 2002).

2.2.2 Backpropagation

O conhecimento das Redes Neurais Artificiais (RNA) se dá através da ponderação que os pesos da conexão entre os neurônios de diferentes camadas trocam entre si. Ou seja, encontrar solução para um determinado problema utilizando RNA seria, a grosso modo, encontrar a melhor topologia de rede, bem como ajustar corretamente os pesos das conexões entre os neurônios (VIEIRA; ROISENBERG, 2005, p.7).

Durante o treinamento com o algoritmo *backpropagation*, a rede opera em uma seqüência de dois passos (figura 9). Primeiro, um padrão é apresentado à camada de entrada da rede. A atividade resultante flui através da rede, camada por camada, até que a resposta seja comparada à saída produzida pela camada de saída. No segundo passo, a saída obtida é comparada à saída desejada para esse padrão particular. Se esta não estiver correta, o erro é calculado. O erro é propagado a partir da camada de saída até a camada de entrada e os pesos das conexões das unidades de camadas internos vão sendo modificados conforme o erro é retropropagado (CARVALHO, 2000 apud TONSIG, 2000).

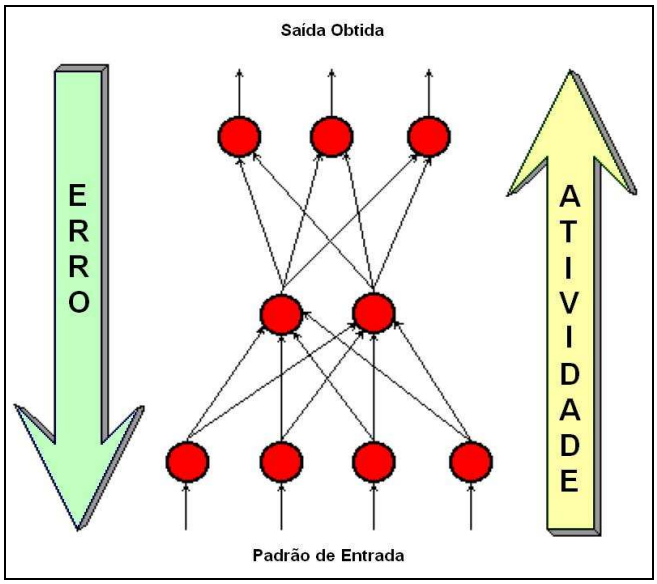

Figura 9: operação do algoritmo de *backpropagation.*
Fonte: adaptado de Tonsig (2000).

Depois que a rede estiver treinada e o erro estiver em um nível satisfatório, ela poderá ser utilizada como uma ferramenta para classificação de novos dados (figura 10). Para isto, a rede deverá ser utilizada apenas no modo progressivo (*feed-forward)*. Ou seja, novas entradas são apresentadas à camada de entrada, são processadas nas camadas intermediárias e os resultados são apresentados na camada de saída, como no treinamento, mas sem a retropropagação do erro. A saída apresentada é o modelo dos dados, na interpretação da rede (Carvalho, 2000 apud TONSIG, 2000).

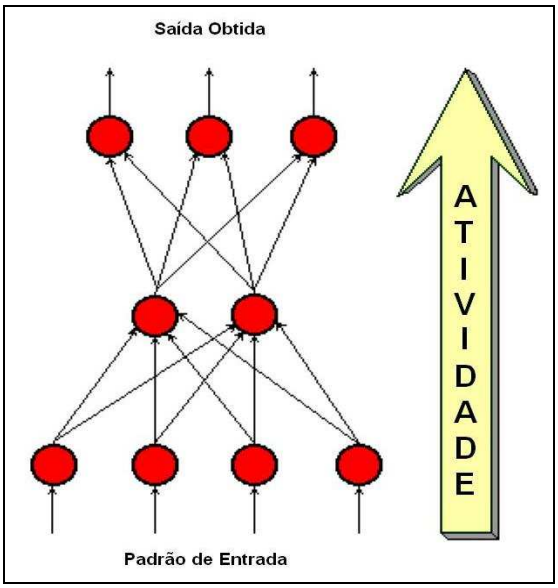

Figura 10: *backpropagation* em modo progressivo (*feed-forward*).
Fonte: Tonsig (2000).

Segundo Fahlman (1988 apud HEINEN, 2004, p.21-22), o algoritmo de *Backpropagation* possui diversos problemas e limitações como:

a) definição da arquitetura: no *Backpropagation*, é preciso definir manualmente o número de neurônios da Rede Neural, o número de camadas e as interconexões entre os neurônios destas camadas, o que exige que sejam feitas várias simulações até que se consiga chegar a uma arquitetura ideal para a solução de determinado problema;

b) dependência da inicialização dos pesos: como os pesos são iniciados de forma aleatória, nunca se sabe exatamente em que pondo da curva de erro o aprendizado irá iniciar, o que faz com que cada simulação realizada traga resultados diferentes, algumas vezes não sendo possível se chegar a valores satisfatórios;

c) minimização do erro lenta e incerta: são necessárias muitas épocas[5] de aprendizado para que o aprendizado ocorra, e nunca se sabe com antecedência se o aprendizado irá ocorrer de forma satisfatória com a configuração e a topologia utilizada;

d) plasticidade e elasticidade: depois que uma Rede Neural convergiu para uma solução, é muito difícil adicionar novos exemplos e continuar o aprendizado do ponto onde parou, e ao se tentar fazer isto, corre-se o risco de fazer com que a Rede Neural esqueça todos os conhecimentos adquiridos anteriormente (esquecimento catastrófico).

As redes neurais que utilizam *backpropagation*, assim como muitos outros tipos de redes neurais artificiais, podem ser vistas como “caixas pretas”, na qual quase não se sabe

[5] O ajuste do peso em uma Rede Neural ocorre ao longo de muitas épocas, onde em cada época são submetidos todos os exemplos da base de dados à Rede Neural.

porque a rede chega a um determinado resultado, uma vez que os modelos não apresentam justificativas para suas respostas. Neste sentido, muitas pesquisas vêm sendo realizadas visando a extração de conhecimento de redes neurais artificiais, e na criação de procedimentos explicativos, onde se tenta justificar o comportamento da rede em determinadas situações (TONSIG, 2000).

Com a finalidade de tentar resolver a maioria destes problemas, Fahlman propôs um novo modelo de Redes Neurais, chamado de *Cascade Correlation*, que corrige muitas das deficiências presentes no modelo *Backpropagation* (FAHLMAN, 1988 apud HEINEN, 2002, p.30).

2.2.3 Cascade-Correlation

Cascade Correlation combina duas idéias-chave. A primeira é a arquitetura em cascata, na qual cada unidade oculta é adicionada na rede uma por vez, sem mudanças após serem adicionadas. A segunda é o algoritmo de aprendizado, que cria e instala a nova unidade oculta. Para cada unidade oculta nova, tenta-se maximizar a magnitude da correlação entre a saída da nova unidade e o erro residual que se tenta eliminar (FAHLMAN, 1991).

Segundo Heinen (2002), o *Cascade Correlation* é um modelo otimizado de MLP (*Multi Layer Perceptron*), desenvolvido por Fahlman com a finalidade de solucionar a maioria dos problemas levantados anteriormente no algoritmo do Backpropagation. As principais vantagens que o *Cascade Correlation* apresenta em relação aos demais modelos de Redes Neurais são:

a) definição da arquitetura: no *Cascade Correlation*, não é necessário determinar o número de neurônios da camada oculta nem as interligações entre estes neurônios, pois o próprio algoritmo do *Cascade Correlation* se encarrega de

determinar a melhor topologia de rede possível para solucionar um determinado problema;

b) continuidade do aprendizado: quando surgirem novos exemplos, pode se treinar uma Rede Neural para aprender estes novos casos a partir de uma rede já treinada sem perder os conhecimentos adquiridos anteriormente;

c) velocidade de aprendizado: o *Cascade Correlation* converge muito rapidamente para um ponto mínimo da curva de erro, e pelo fato de não serem necessárias várias simulações anteriores para ajustar a arquitetura da rede e os parâmetros, o tempo necessário para o aprendizado de um problema é reduzido drasticamente.

A arquitetura em *Cascade Correlation* é ilustrada na figura 12. Começa-se com algumas entradas e uma ou mais unidade de saída, mas sem unidades ocultas. O número de entradas e saídas é imposto pelo problema e pela representação I/O que o experimento escolhe. Toda entrada é conectada a cada unidade de saída por uma conexão com pesos ajustáveis. Há também a entrada bias, permanentemente setada para +1 (FAHLMAN, 1991).

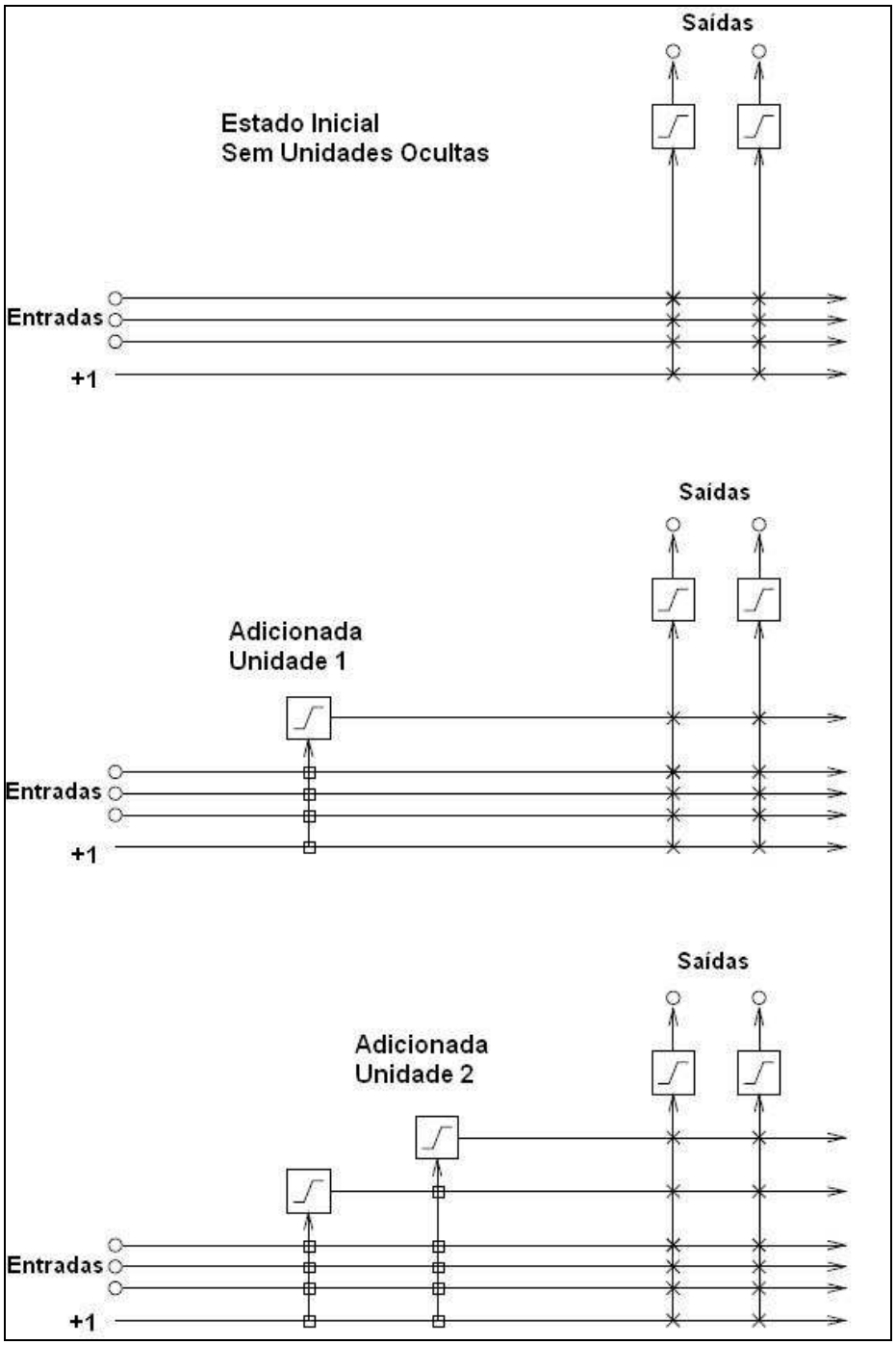

Figura 11: a arquitetura em cascada, estado inicial e após adicionar duas unidades ocultas. As linhas verticais somam toda a ativação entrante. As conexões em boxes são congeladas e as conexões "X" são treinadas repentinamente.
Fonte: adaptado de Fahlman (1991).

As unidades de saída podem produzir uma soma linear de seus pesos de entrada, ou podem empregar alguma função de ativação não-linear (FAHLMAN, 1991). Nas experiências de Fahlman, a foi usada a função de ativação sigmoidal hiperbólica tangente (figura 11), na qual é escalada de –1.0 até +1.0.

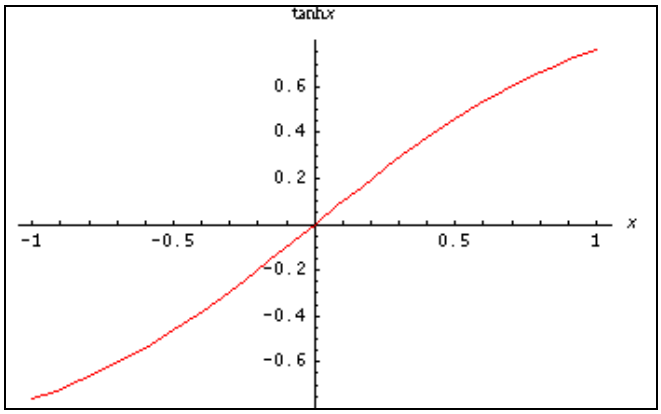

Figura 12: representação gráfica da função sigmoidal hiperbólica tangente.
Fonte: Mathworld (1999).

Durante o treinamento da base de conhecimento, a rede neural executa diversos ciclos de treinamento. Dentro de cada ciclo, o valor do erro residual é ajustado até atingir um ponto de estagnação. Quando o ajuste do erro chegar a esta estagnação, uma unidade candidata é adicionada à rede, na qual receberá os pesos de saída de todas as unidades de entrada e de unidades ocultas já existentes. A saída desta unidade ainda não é conectada à rede, pois primeiramente deve-se tentar maximizar a correlação entre a ativação da unidade candidata e o erro, treinando e ajustando os pesos da unidade candidata. Quando a correlação parar de melhorar, deve-se interromper o treinamento desta unidade candidata e instalá-la à rede, conectando-a com as unidades de saída e tornando-a uma unidade oculta ativa da rede.

Adicionando a unidade oculta à rede, seus pesos de entrada são congelados e todos os pesos de saída são treinados até o ajuste do erro estagnar novamente. Ao chegar à estagnação, outra nova unidade candidata é criada, repetindo todo o processo descrito acima. Este ciclo é contínuo até o erro ser aceitavelmente normal.

Quando uma nova unidade oculta está para ser criada, além de estabelecer a conexão com cada uma das entradas e saídas originais da rede, também é feita uma conexão entre a nova unidade e as unidades ocultas preexistentes. Cada nova unidade adiciona uma nova camada com um único neurônio à rede, gerando uma arquitetura em cascata. Geralmente as unidades ocultas são simples, mas esta estrutura em cascata pode ser implementada com

unidades ocultas mais complexas (CASTRO, 1999).

Em vez de uma única unidade candidata, é possível criar um *pool* de unidades candidatas, cada uma com uma configuração de pesos iniciais randomizados diferentemente. Todos recebem o mesmo sinal de entrada e vêem o mesmo erro residual para cada teste padrão de treinamento. Isto porque elas não interagem uma com as outras ou afetam a ativação na rede durante o treinamento, pois todas estas unidades candidatas podem ser treinadas em paralelo (FAHLMAN, 1991). A unidade candidata que apresentar a maior correlação, será incluída na rede como uma nova unidade oculta conectada a todas as camadas da rede, conforme ilustrado na figura 13.

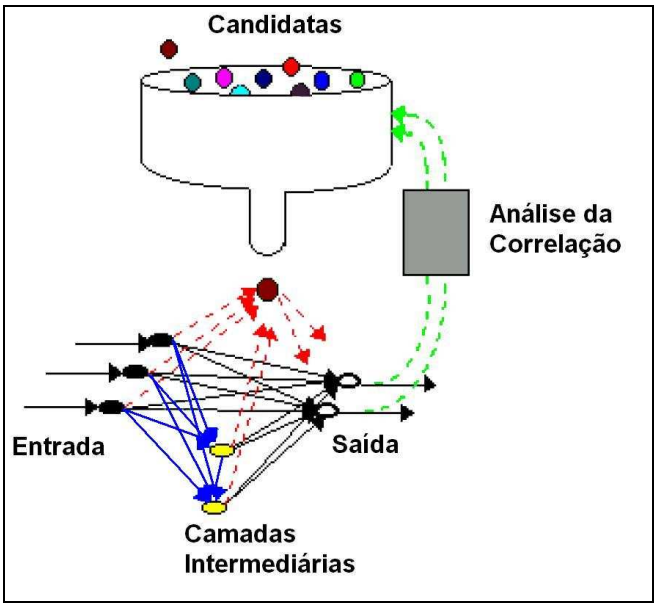

Figura 13: unidade candidata recém inserida na rede após ser selecionada no pool de unidades candidatas.
Fonte: adaptado de Vargas (1998).

Uma das limitações do *Cascade Correlation* é de somente poder ser utilizado para resolver problemas relacionados à classificação, porém, como a identificação de impressões digitais está diretamente ligada à classificações, este modelo se adequa plenamente aos objetivos do projeto.

2.3 TRABALHOS CORRELATOS

Alguns dos trabalhos correlatos encontrados foram:

a) O trabalho de Heinen (2002) trata o reconhecimento de assinaturas digitais on-line utilizando o modelo *Cascade Correlation*. A assinatura é obtida através de um *hardware* chamado *Tablet* que, além da imagem, extrai informações da pressão da caneta sobre o *hardware*, velocidade da assinatura, número de vezes que a caneta foi levantada, entre outros.

b) No trabalho de Gumz (2002), foi desenvolvido um protótipo de identificação de minúcias em uma impressão digital, classificando-as entre crista final, crista bifurcada e minúcia falsa, utilizando o modelo de rede neural *Backpropagation*. Os resultados desejados não foram obtidos, sendo que o autor acredita que o trabalho possuía limitações quanto às IIDs, que possuíam uma baixa qualidade e eram de tamanhos muito grandes e variáveis. Também se acredita que a quantidade de neurônios utilizados na camada de entrada da rede, que abrangia 10x10 *pixels* de uma área selecionada da IID, era muito pequena para abranger todos os possíveis exemplos de minúcias. Também entrou-se em questão o pré-processamento das imagens, a classificação incorreta das minúcias e a escassez de minúcias falsas no treinamento da rede como possíveis problemas para o atingimento de bons resultados do trabalho.

c) No trabalho de Matias (2004), foi desenvolvido um protótipo de sistema para identificação de deltas e núcleos nas impressões digitais também utilizando o modelo de redes neurais *Backpropagation*. O protótipo apresentou limitações no treinamento e validação da base de conhecimento, onde, conforme também concluído no trabalho de Gumz (2002), é possível que estas limitações estejam relacionadas ao pré-processamento e na qualidade das IID’s.

3 DESENVOLVIMENTO DO TRABALHO

Nas próximas seções serão apresentados os requisitos do protótipo, a especificação e toda a metodologia utilizada para o desenvolvimento do trabalho. Ao final, serão apresentados os resultados obtidos com o protótipo e a conclusão do trabalho.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Os requisitos para o desenvolvimento do protótipo são:

a) o protótipo deverá criar uma RNA *Cascade Correlation*;
b) o protótipo deverá possuir uma base de IID's pré-processadas para treinamento da base de conhecimento;
c) o protótipo deverá analisar IID's para obtenção dos dados de entrada do treinamento;
d) o sistema deverá realizar o treinamento de um grupo de IID's ou de uma IID específica;
e) o sistema deverá salvar em arquivos a topologia da rede e os pesos obtidos no treinamento;
f) o sistema deverá permitir carregar os pesos já obtidos em treinamento anteriores para serem treinados novamente;
g) o sistema deverá cadastrar pessoas e suas devidas IID's;
h) o sistema deverá extrair os pesos das IID's de pessoas e armazená-los em arquivo de texto;
i) o sistema deverá permitir carregar os pesos obtidos para realizar o reconhecimento da ID;
j) o sistema deverá carregar uma IID's e identificar o usuário no qual ela pertence.

3.2 ESPECIFICAÇÃO

Nesta seção será apresentado o caso de uso do protótipo e as principais operações das classes “TRedeNeural” e “TArquivos”, que são as duas maiores classes implementadas no protótipo.

3.2.1 Caso de Uso

A especificação do protótipo será apresentada através do caso de uso da notação da UML. É importante ressaltar que o desenvolvimento deste trabalho teve o foco voltado somente na viabilidade da utilização de *Cascade Correlation* no reconhecimento de impressões digitais, sem visão de negócio.

O desenho do diagramas foi obtido através da ferramenta Rational Rose 2002 versão demo da Rational Rose Corporation. Maiores informações sobre a ferramenta pode ser obtidas no site Rational (2002).

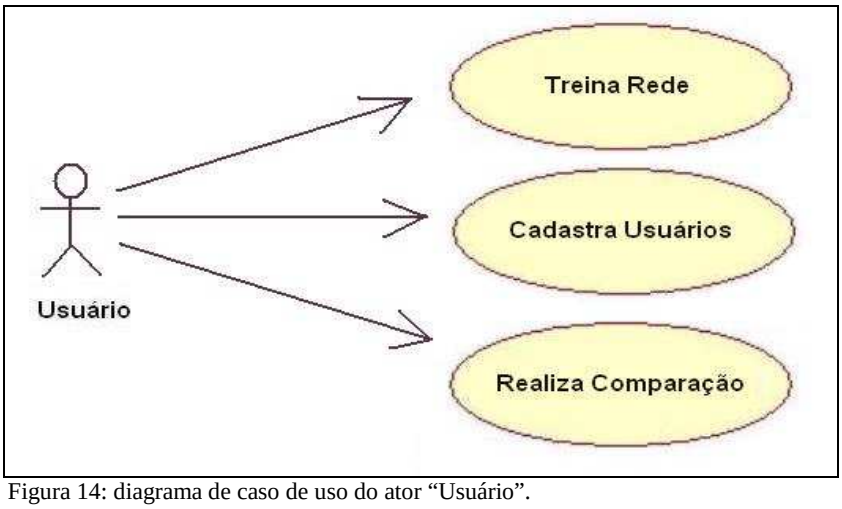

Figura 14: diagrama de caso de uso do ator “Usuário”.

Cada caso de uso, atende os seguintes requisitos:

a) Treina Rede: possibilita ao usuário treinar a rede neural, configurando os valores

de saída para cada IID da base de conhecimento;

b) Cadastra Usuários: possibilita ao usuário cadastrar os dados básicos dos usuários e suas respectivas IID’s para o futuro reconhecimento;

c) Realiza Comparação: possibilita ao usuário selecionar IID’s em uma base de dados, realizando comparações com IID’s previamente cadastradas. O sistema verificará se a IID selecionada está armazenada no banco de dados do sistema e retornará o resultado;

3.2.2 Classe TRedeNeural

A classe TRedeNeural é responsável em realizar todas as operações necessárias para o treinamento da rede neural através da base de conhecimento, para a extração de dados dos arquivos de pesos dos usuários cadastrados e para o reconhecimento da impressão digital. A seguir serão listados os principais procedimentos e funções que incluem nesta classe:

d) pCriaNovaRede: é o construtor da classe TRedeNeural e inicializador das variáveis globais do sistema;

e) pCriaRedeArquivos: cria a rede neural através dos arquivos de topologia e pesos que são gerados no treinamento da base de conhecimento;

f) pAbreEntradasSaidas: atribui os valores de entrada obtidos através das IID’s para os neurônios da camada de entrada e, também, os valores desejados para os neurônios da camada de saída da rede neural;

g) pAtivaNeuronios: realiza a ativação de todos os neurônios de rede. Para esta ativação, cada valor de entrada do neurônio é multiplicado com seu peso. Como cada neurônio pode possuir mais de uma entrada, os resultados desta multiplicação são acumulados. O total acumulado é submetido à função de ativação sigmóide

hiperbólica tangente, obtendo assim o valor de saída deste neurônio;

h) pTreinaSaidas: calcula o erro do neurônio subtraindo o valor desejado do valor obtido em sua ativação. O erro total obtido é submetido à função de ativação sigmóide hiperbólica tangente e multiplicado pelo parâmetro de velocidade do deslocamento da superfície do erro (ver figura 8). Neste procedimento também são calculados os valores delta, o bias e o erro do quadrado médio para cada época.

i) fFuncaoAtivacao: função que realiza o cálculo da função de ativação sigmoidal hiperbólica tangente;

j) pAjustaPesos: realiza o ajuste dos pesos das camadas de entrada através da técnica *quickprop*;

k) fTestaSaidas: verifica se o aprendizado da camada de saída estagnou. Caso estagnar, inicia-se o processo de alocação das unidades candidatas, senão todos os pesos das conexões da rede são submetidos ao treinamento novamente,

l) pIniciaPool: após a estagnação do aprendizado da rede neural entre as unidades de entrada, saída e ocultas já existentes, o sistema cria um grupo de unidades candidatas. Estas unidades candidatas serão treinadas até o aprendizado estagnar e, posteriormente, passarão por um processo de análise de correlação com as unidades externas. A unidade que tiver maior correlação será efetivada como uma nova unidade oculta da rede. A quantidade de unidades candidatas atribuídas ao pool é pré-definida e, neste trabalho, utilizam-se 8 unidades candidatas;

m) pTreinaOcultas: procedimento responsável pelo treinamento e pela análise de correlação das unidades candidatas. O treinamento das unidades candidatas é similar ao das unidades efetivas da rede. Primeiramente as entradas destes neurônios são submetidas à função de ativação para a obtenção de suas saídas que, posteriormente, serão utilizadas para o ajuste da correlação. Após este ajuste, as

unidades candidatas são ativadas novamente para realizar-se o cálculo do erro, o ajuste dos pesos através da técnica *quickprop* (FAHLMAN, 1988) e, novamente, o ajuste da correlação. Este procedimento é realizado diversas vezes até o aprendizado das unidades candidatas estagnar. Quando estagnar, a unidade candidata de maior correlação com as unidades externas torna-se uma unidade oculta efetiva da rede;

n) pTestaOcultas: procedimento que verifica a estagnação do aprendizado das unidades candidatas e que define, durante o processo de treinamento, se as unidades candidatas continuarão sendo treinadas ou se já está na hora de efetivar uma unidade candidata à rede;

o) pInsereCand: procedimento que efetivamente inclui a unidade candidata à rede, tornando-a uma unidade oculta. Esta etapa é muito importante para toda a rede neural, pois aqui são atualizadas todas as variáveis globais do sistema. Esta atualização é necessária, pois havendo um novo neurônio na rede, há um novo vetor de pesos, um novo vetor de valores de entrada, uma nova camada de neurônios, novas conexões com os demais neurônios, entre outros.

Os procedimentos aqui listados estão ordenados pela ordem de execução, portanto, pode-se ter um entendimento melhor de um algoritmo de identificação de impressões digitais que utilize *Cascade Correlation* analisando estes passos.

Para complementar, após a inserção da nova unidade oculta à rede, os exemplos de entrada são submetidos novamente ao treinamento. A diferença entre o próximo treinamento que será realizado e o treinamento que acabou de ser finalizado está na própria unidade oculta recém inserida, pois agora há um novo neurônio na rede e, conseqüentemente, novos pesos e conexões. Todos estes pesos são recalculados, obtendo-se novos valores de saída e um novo valor total do erro. Em cada valor de saída é decrementado o valor desejado e acumulado em

outra variável. Esta variável retrata o erro acumulado durante o treinamento, que deverá ser menor que o erro consideravelmente tolerante para concluir-se que a rede neural aprendeu. Enquanto não aprender, novas unidades ocultas são adicionadas à rede e novas épocas de treinamento são realizadas até a rede aprender ou até a paciência do usuário esgotar.

3.2.3 Classe TArquivos

Esta classe é responsável pela geração e leitura dos arquivos da topologia e dos pesos da rede neural. Os principais procedimentos realizados são:

a) pSalvaTopologia: gera um arquivo de texto com todas as definições da rede neural para que, ao executar o protótipo novamente, seja possível carregar as definições desta rede neural e seus pesos a partir do que foi salvo anteriormente. Neste arquivo não são armazenados os pesos de cada conexão, somente definições como: quantidade de neurônios de entrada e saída, quantidade de camadas de neurônios, quantidade de conexões de saída para cada neurônio, quantidade de neurônios em cada camada oculta e de saída (que no caso deste protótipo sempre será 1), quais unidades candidatas foram selecionadas do pool para serem efetivadas, etc;

b) pSalvaPesos: gera o arquivo de pesos das conexões da rede neural. Este arquivo conterá a listagem de todas as conexões de saída dos neurônios de entrada especificando a qual neurônio esta conexão está ligada e qual é o peso desta conexão. Ao final desta listagem, é inserida uma linha que possui o peso *bias* da camada de entrada. Este mesmo procedimento é realizado com as camadas ocultas, listando cada neurônio, suas conexões de saídas e os devidos pesos e, ao final de cada camada oculta, o *bias*;

c) pCriaRedePelaTopol: cria a rede neural através de um arquivo de topologia salvo

em um treinamento realizado anteriormente;

d) pLePesos: lê o arquivo de pesos gerado num treinamento passado e atribui às conexões dos neurônios o valor do seu devido peso.

3.2.4 Estrutura global das variáveis

Todas as variáveis que definem a rede neural deste protótipo são globais, pois os valores contidos nesta estrutura podem ser lidos e alterados em qualquer ponto do sistema. Estas variáveis são agrupadas em *record* e podem ser de tipos simples como “integer”, “double” e “string” ou de estruturas mais complexas como vetores, ponteiros para outros agrupamentos ou ponteiros para si mesmo. Segue abaixo uma listagem destes agrupamentos e suas principais variáveis:

a) Record_RedeNeural: possui as definições de alto nível da rede neural como a quantidade de neurônios na camada de entrada e saída, a quantidade de neurônios que possuem ligação com a camada de entrada, a quantidade de camadas da rede e diversas variáveis utilizadas no ajuste de pesos dos neurônios, na ativação e no treinamento da rede neural;

b) Record_Neurônios: armazena as definições relevantes aos neurônios, como a quantidade de conexões de entrada e saída, se o neurônio está congelado ou não, o valor do erro entre outros. Também possui vetores para o armazenamento dos valores de entrada do neurônio, o vetor de pesos, o vetor dos valores delta e também um ponteiro para o Record_Conexao que será mencionado a seguir;

c) Record_Conexão: consiste em ser um *record* simples de apenas duas variáveis: “Neuronio”, que indica que o neurônio está ligado a determinado neurônio e “Entrada”, que indica a qual entrada do outro neurônio este neurônio está ligado;

d) Record_NoCamada: utilizado para a intercalação de uma camada da rede para outra, realizando operações que requerem esta flexibilidade como a ativação dos neurônios e a geração e leitura dos arquivos de topologia e pesos. Possui como variáveis o número de neurônios da camada, uma variável utilizada somente como identificador da camada e duas variáveis que apontam para o próprio *record* denominadas “Anterior” e “Próximo” que, conforme o nome já diz, servem para realizar o trânsito entre uma camada e outra;

e) Record_NoEntrada: é uma estrutura simples que armazena os dados relacionados a camada de entrada da rede neural. Possui como variáveis o valor de entrada do neurônio obtido através da IID, a quantidade de conexões de saída, uma variável do tipo “string” para identificação do neurônio e um vetor que aponta para “Record_Conexao” que indica a qual neurônio um neurônio de entrada está ligado;

f) Record_Pattern: este *record* está relacionado aos exemplos de arquivos de IID’s submetidos ao treinamento da rede neural e ao reconhecimento de pessoas. Possui somente duas variáveis: o vetor “Entrada” que armazena os valores obtidos através da extração dos *pixels* de uma ID, e o vetor “Saída” que armazena os valores de saída desejados;

g) Record_Candidatas: armazena os dados relacionados às unidades candidatas da rede. Possui uma variável que indica a quantidade de entradas desta unidade, uma variável do tipo “string” usada somente para identificação, uma variável que indica qual foi a unidade selecionada das 8 candidatas do *pool*, e outra que indica qual foi o valor da melhor correlação.

Para melhor entendimento do algoritmo, pode-se verificar na figura 15 o detalhamento dos passos que devem ser respeitados na implementação do treinamento em *Cascade Correlation*.

```
procedimento TreinaRede;
Inicio
  pLeImagens;          // Lê IID's
  pLeSaidasDesejadas;  // Lê valor desejado para cada imagem

  Se (PossuiArquivoPesos = Verdadeiro)
    pLeArquivoPesos    // Lê arquivo de pesos salvo em um treinamento anterior
  Senao
    pCriaNovaRede;     // Cria nova estrutura de pesos zerada

  Enquanto (xSucesso = falso) faça
  Inicio

    Para x := 0 até xQtdImagens - 1 faça    // Para cada IID
    Inicio
      pAtivaNeuronios;                      // Ativa entradas das unidade de saída e ocultas já existentes
      pTreinaConexoesSaida;                 // Calcula erro obtido, deltas, bias e erro quadrado médio

      Para x := 0 até xQtdNeuroniosSaida - 1 faça
      Inicio
        Se Erro[x] > xMaiorErro    // Verifica neurônio com maior erro
          xMaiorErro = Erro[x]
      Fim;
    Fim;

    pAjustaPesos;    // Ajusta os pesos com Quickprop

    Se (xMaiorErro < xErroParada)  // Se maior erro do neurônio é menor do que o erro máximo permitido
    Inicio
      Mensagem('Treinamento realizado com sucesso');   // Finaliza treinamento
      pSalvaPesosArquivo;                              // Salva pesos no arquivo de texto
      xSucesso = Verdadeiro;                           // Sai do Loop
    Fim;

    Se (fErroEstagnou)     // Se o aprendizado estagnou
    Inicio
      pCriaPoolCandidatas;  // cria pool de unidades candidatas
      pTreinaCandidatas;    // Treina unidades candidatas e calcula suas correlações
      pInsereUnidadeOculta; // Insere candidata de maior correlação como unidade oculta
    Fim;
  Fim;
Fim;
```

Figura 15: algoritmo *Cascade Correlation* para o treinamento da rede neural.

3.2.5 Diagrama de Atividades

Para melhor entendimento dos passos que devem ser executados para o treinamento da base de conhecimento, foi-se desenvolvido um diagrama de atividades, da notação da UML, que pode ser visualizado na figura 16:

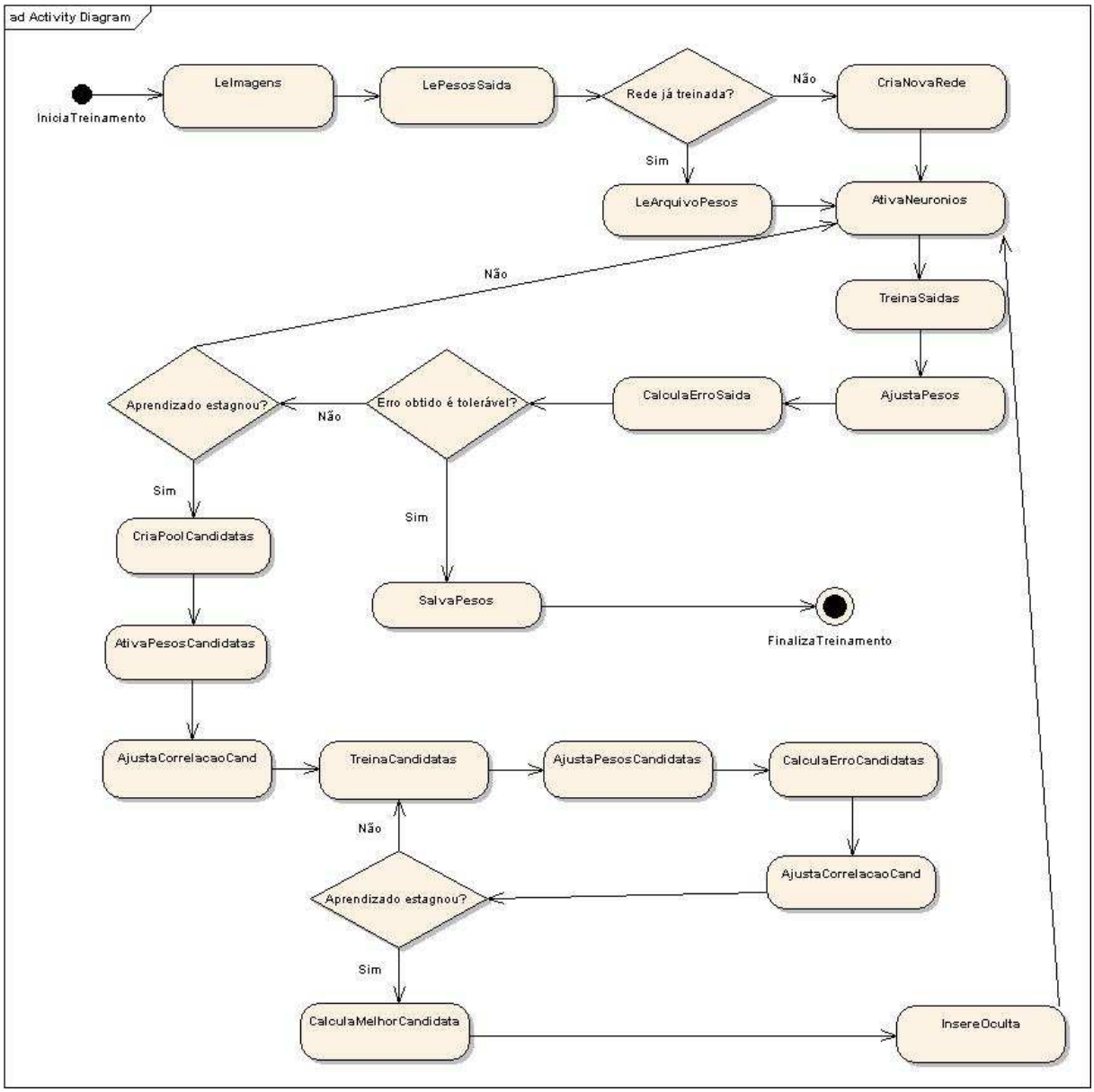

Figura 16: diagrama de atividades do treinamento da rede neural.

O Diagrama de atividades foi desenvolvido na ferramenta Enterprise Architect versão demo da Sparx Systems, obtida através do site Sparx (2006).

3.3 IMPLEMENTAÇÃO

Esta seção mostrará os detalhes do desenvolvimento do protótipo, como as telas de treinamento e reconhecimento e a maneira que se comportam, fragmentos de código fonte,

exemplos de IID's que podem ser utilizadas no treinamento, entre outros. Também serão mencionadas as ferramentas utilizadas para o desenvolvimento e softwares auxiliares utilizados para outras necessidades que não cabem a este trabalho sua implementação.

3.3.1 Técnicas e ferramentas utilizadas

Para realizar a implementação do protótipo utilizou-se a linguagem de programação *Object* Pascal no ambiente de desenvolvimento Borland Delphi, na versão 7.0. Para o armazenamento dos dados cadastrais dos usuários foi utilizado o SBDG Interbase versão 5.0. A seguir serão mostrados fragmentos de código fonte implementados para o protótipo e uma breve descrição de sua funcionalidade.

Na figura 17 é apresentada a declaração das variáveis globais contidas no protótipo. Entre estas variáveis, pode-se verificar todos os *record* que foram listados na seção 3.2.4. Alguns itens dos *records* foram retirados para que o quadro não ficasse muito extenso.

```
type TArrayEntradas = array [0..cMaxNeuronios - 1] of Double;
type TArrayCandidatas = array of ^Double;

type Record_RedeNeural = record
  Camadas       : Integer;  // quantidade de camadas
  Entradas      : Integer;  // quantidade de entradas
  Saidas        : Integer;  // quantidade de saidas
  Neuronios     : Integer;  // quantidade de neuronios
  NovasUnidades : Integer;  // número de unidades escondidas adicionadas
  FatorMU       : Double;   // Usado pelo Quickprop que verifica se está chegando muito rapido a uma estagnacao
  FatorSrk      : Double;   // Usado pelo Quickprop que verifica se está chegando muito rapido a uma estagnacao
  TotalEpocas   : Integer;  // total de epocas do treinamento
  TesteEpocas   : Integer;  // total de epocas do teste
  CasCorEpocas  : Integer;  // Total CasCor hidden epocas
  SigmOffset    : Double;   // valor de deslocamento da curva de erro na parábola
  Epsilon       : Double;   // Velocidade de Convergência
  MaxErro       : Double;   // correta saida estimada (erro aceitavel)
  SaidaPacien   : Double;   // Paciencia no aprendicado da saída
  OculPacien    : Double;   // Paciencia no aprendizado das ocultas (candidatas)
  ErroMedio     : Double;   // Erro medio de todas as saídas da rede
  ErroSqr       : Double;   // Soma ao quadrado do erro de todas as ativações
end;

type Record_Conexao = record
  Neuronio : Integer;    // Conectado ao neuronio...
  Entrada  : Integer;    // Conectado a entrada... desse neuronio
end;

type TPointerNoCamada = ^Record_NoCamada;                 // apontador para o próprio record
  TArrayPointerNoCamada = array of TPointerNoCamada;      // array de apontadores para este record
  NoProximo = TPointerNoCamada;                           // ponteiro para o proximo record
  NoAnterior = TPointerNoCamada;                          // ponteiro para o record anterior
  Record_NoCamada = record
  NumNeur : Integer;                                      // número de neurônios da camada
  NumInd  : array [0..cMaxNeuronios - 1] of Integer;      // label do neuronio da camada
  Proximo :  NoProximo;                                   // ponteiro para a proxima camada
  Anterior:  NoAnterior;                                  // ponteiro para a camada anterior
end;

type TArrayConexao = array [0..cNeuroniosEntrada - 1] of Record_Conexao;  // array de ponteiros para o record de conexao
  Record_NoEntrada = record
  Entrada  : Double;                // valor da entrada do neuronio
  QtdCon   : Integer;               // quantidade de conexoes do neuronio
  ConSaida : TArrayConexao;         // definições de cada conexão do neuronio
  Id       : array [0..4] of Char;  // label de identificação do neurônio
end;
type Record_Neuronios = record
  Id          : array [0..30] of Char;   // identificador do neuronio
  TipoId      : Byte;                    // tipo neuronio
  QtdEntradas : Integer;                 // quantidade de entradas do neuronio
  QtdSaidas   : Integer;                 // quantidade de saidas do neuronio
  Congelado   : Boolean;                 // Neuronio congelado ou nao
  ValorSoma   : Double;                  // soma do valor de entrada * erro
  ValorSaida  : Double;                  // valor saida apos a ativacao
  Correl      : TArrayCandidatas;        // utilizado para unidades candidatas na selecao da maior correlação
  CorrelAnter : TArrayCandidatas;        // utilizado para unidades candidatas na selecao da maior correlação
  Desejado    : Double;                  // valor desejado
  Erro        : Double;                  // valor do erro
  SomaErro    : Double;                  // soma do erro
  Bias        : Double;                  // bias
  Bias_delta  : Double;                  // bias delta
  Bias_epdt   : Double;                  // delta de aprendizado das epocas
  Bias_quick  : Double;                  // alteração do bias pelo quickprop
  Peso        : TArrayEntradas;          // vetor de pesos das conexoes
  Entrada     : TArrayEntradas;          // vetor de valores de entrada
  Delta       : TArrayEntradas;          // pesos delta
  Epdt        : TArrayEntradas;          // delta de aprendizado dos neuronios
  Quickdelta  : TArrayEntradas;          // alteração dos pesos quickprop
  Con         : TArrayConexao;           // ponteiro das definições das conexoes do neuronio
end;

type Record_Pattern = record
  Entrada : array [0..cNeuroniosEntrada-1] of Double;   // valores de entrada
  Saida   : array [0..cNeuroniosSaida-1] of Double;     // valores de saida desejados

end;

type Record_Candidatas = record
  QtdEntradas : Integer;   // numero de entradas candidatas
  Tipo        : array [0..cMaxNeuronios - 1] of Byte;      // 0=unidade de entrada, 1=unidade de neuronio
  UInd        : array [0..cMaxNeuronios - 1] of Integer;   // indice da unidade
  MelhorIndice: Integer;   // index da melhor candidata
  MelhorCorr  : Double;    // melhor correlação indexada do pool de candidatas
end;
type TArrayPointerPattern = array of Record_Pattern;
var
  GRedeNeural : Record_RedeNeural;
  Array_Neuronios  : array [0..cMaxNeuronios - 1] of Record_Neuronios;
  Array_GNoEntrada : array [0..cNeuroniosEntrada - 1] of Record_NoEntrada;
  GNoCamadaRaiz    : TArrayPointerNoCamada;     // ponteiro para lista de neuronios por camada
  GNoCamadaSaida   : TArrayPointerNoCamada;     // ponteiro para lista de neuronios de saída
  GNoCamada        : TArrayPointerNoCamada;     // ponteiro para lista de camadas
  GCandPool        : array [0..cQtdCandidatas - 1] of Record_Neuronios;   // registro do pool de candidatas
  GCandidatas      : Record_Candidatas;
  gDataL, gDataT   : TArrayPointerPattern;
```

Figura 17: declaração das variáveis globais do protótipo.

Na figura 18, é apresentada a rotina que chama todos os procedimentos para o treinamento da rede neural baseando-se nos exemplos de IID's. Este fragmento de fonte também teve que ser reduzido, devido a sua grande extensão.

```
procedure TfrTreinamentoRNA.btTreinarAutomaticoClick(Sender: TObject);
var
  xSucesso : Boolean;
  xAux: Integer;
  xOut: Integer;
  xNNeur: Integer;
  xErrMax: Double;
  xAcumErr: Double;
  // variáveis usadas na conversão da imagem para a matriz de entrada
  xSearchArq: TSearchRec;
  xImagem: TImage;
  xLinha, xColuna, xIndice, xIndArq: Integer;
begin
    xSucesso := false;
    while (GRedeNeural.EpMaxErro > GRedeNeural.StopErro) and (not xSucesso) do
    begin
      Inc(GRedeNeural.TotalEpocas);
      xAcumErr := 0;
      GRedeNeural.ErroMedio := 0;
      GRedeNeural.ErroSqr := 0;
      GRedeNeural.ErroRMS := 0;
      xErrMax := MinDouble;

      // procedure que converte as IID's e passa os valores para a variável gDataL que é do tipo Record_Pattern
      pConverteIID;

      wRNA.pInicializarPesos;
      wRNA.pInicializarEpocas;

      for xAux := 0 to Length(gDataL) - 1 do
      begin
        wRNA.pAbreEntradasSaidas(0, xAux);
        wRNA.pAtivacaoRede;
        wRNA.pTreinaSaidas;

        for xOut := 0 to GNoCamadaSaida[0]^.NumNeur -1 do
        begin
          xNNeur := GNoCamadaSaida[0]^.NumInd[xOut];
          if (not((Array_Neuronios[xNNeur].ValorSaida > Array_Neuronios[xNNeur].Desejado - GRedeNeural.MaxErro) and
                  (Array_Neuronios[xNNeur].ValorSaida < Array_Neuronios[xNNeur].Desejado + GRedeNeural.MaxErro))) then
            xErr := true;

          // retorna o valor absoluto do "valor obtido - valor desejado"
          xAcumErr := xAcumErr + (Abs(Array_Neuronios[xNNeur].ValorSaida - Array_Neuronios[xNNeur].Desejado));
        end;

        if (Abs(Array_Neuronios[xNNeur].Erro) > xErrMax) then
          xErrMax := Abs(Array_Neuronios[xNNeur].Erro);

      end;
      wRNA.pAjustaPesos;

      if (xErrMax < GRedeNeural.StopErro) then
      begin
        // treinamento realizado. Salva arquivos de pesos e topologia
        wArquivos.pSalvaArquivosTemporarios;
        MessageDlg('Treinamento realizado com sucesso!', mtInformation, [mbOK], 0);
        xSucesso := true;
        continue;
      end;

      // fTestaSaida: verifica se o aprendizado estagnou
      if (not wRNA.fTestaSaida(GRedeNeural.ErroSqr)) then
      begin
        wRNA.pIniciaPool;  // inicia pool de candidatas
        wRNA.pTreinaCandidatas; // tenta maximizar a correlacao das candidatas
        wRNA.pInsereCand;    // acha a melhor candidata e instala como uma nova unidade na rede
        wRNA.pInicializarEpocas;  // reinicializa todas as variaveis acumuladas
      end;
    end;
  finally
    xImagem.Free;
  end;
end;
```

Figura 18: rotina da chamada dos procedimentos para o treinamento da base

Na figura 19 é apresentada a rotina que realiza a ativação de todas as unidades ocultas e de todos os neurônios da camada de saída. Nota-se que nesta rotina há um exemplo de intercalação das camadas da rede que foi mencionada no subitem “d” do item 3.2.4.

```
{ procedure que realiza a ativação das unidades de saída e unidades ocultas existentes }
procedure TRedeNeural.pAtivacaoRede;
var
  xAux: Integer;
begin
  GNoCamada[0] := GNoCamadaRaiz[0];

  // se existir a camada selecionada...
  while not (GNoCamada[0] = nil) do
  begin
    // faz a ativação dos neuronios da camada
    for xAux := 0 to GNoCamada[0]^.NumNeur - 1 do
      pAtivacaoNeuronio(GNoCamada[0]^.NumInd[xAux]);

    // pega a proxima camada
    GNoCamada[0] := GNoCamada[0]^.Proximo;
  end;
end;

{ procedure que realiza a ativação do neurônio }
procedure TRedeNeural.pAtivacaoNeuronio(pNeuronio: Integer);
var
  xEntrada : TArrayEntradas;
  xPeso : TArrayEntradas;
  xAux : Integer;
  xSoma : Double;
  xFile: TextFile;
begin
  // retorna o vetor de pesos e de valores de entrada
  xPeso := Array_Neuronios[pNeuronio].Peso;
  xEntrada := Array_Neuronios[pNeuronio].Entrada;

  xSoma := 0;

  // soma ponderada das entradas
  for xAux := 0 to Array_Neuronios[pNeuronio].QtdEntradas - 1 do
    xSoma := xSoma + (xEntrada[xAux] * xPeso[xAux]);

  { Atribui o bias ao valor da soma da ativação }
  xSoma := xSoma + Array_Neuronios[pNeuronio].Bias;

  { Atribui ao registro do neurônio o valor da soma }
  Array_Neuronios[pNeuronio].ValorSoma := xSoma;

  // submete a soma ponderada à ativação da rede
  Array_Neuronios[pNeuronio].ValorSaida := fFuncaoAtivacao(xSoma);

  for xAux := 0 to Array_Neuronios[pNeuronio].QtdSaidas - 1 do
    Array_Neuronios[Array_Neuronios[pNeuronio].Con[xAux].Neuronio].
    Entrada[Array_Neuronios[pNeuronio].Con[xAux].Entrada] :=  Array_Neuronios[pNeuronio].ValorSaida;
end;
```

Figura 19: rotina de ativação dos neurônios efetivos da rede.

Na figura 20 é apresentada a rotina de treinamento dos valores de saída dos neurônios depois de realizada a ativação:

```
procedure TRedeNeural.pTreinaSaidas;
var
  xNErr: Double;
  xAux1, xAux2: Integer;
begin
  // retorna a diferença entre o valor de saída e o desejado
  for xAux1 := 0 to GNoCamadaSaida[0]^.NumNeur - 1 do
  begin
    xNErr := Array_Neuronios[GNoCamadaSaida[0]^.NumInd[xAux1]].ValorSaida -
             Array_Neuronios[GNoCamadaSaida[0]^.NumInd[xAux1]].Desejado;

    // multiplica o erro com o resultado da ativação do valor de saída
    Array_Neuronios[GNoCamadaSaida[0]^.NumInd[xAux1]].Erro := xNErr *
              fAtivaInicio(Array_Neuronios[GNoCamadaSaida[0]^.NumInd[xAux1]].ValorSaida);

    // acumula o valor do delta
    for xAux2 := 0 to Array_Neuronios[GNoCamadaSaida[0]^.NumInd[xAux1]].QtdEntradas - 1 do
      Array_Neuronios[GNoCamadaSaida[0]^.NumInd[xAux1]].Epdt[xAux2] :=
                                      Array_Neuronios[GNoCamadaSaida[0]^.NumInd[xAux1]].Epdt[xAux2] +
                                      (Array_Neuronios[GNoCamadaSaida[0]^.NumInd[xAux1]].Entrada[xAux2] *
                                      Array_Neuronios[GNoCamadaSaida[0]^.NumInd[xAux1]].Erro);

    // acumula o valor do bias
    Array_Neuronios[GNoCamadaSaida[0]^.NumInd[xAux1]].Bias_epdt := Array_Neuronios[GNoCamadaSaida[0]^.NumInd[xAux1]].Bias_epdt +
                                                                   Array_Neuronios[GNoCamadaSaida[0]^.NumInd[xAux1]].Erro;

    // acumula a soma do erro
    Array_Neuronios[GNoCamadaSaida[0]^.NumInd[xAux1]].SomaErro := Array_Neuronios[GNoCamadaSaida[0]^.NumInd[xAux1]].SomaErro +
                                                                  Array_Neuronios[GNoCamadaSaida[0]^.NumInd[xAux1]].Erro;

    // calcula o erro quadrado medio que será utilizado no algoritmo de ajuste de pesos
    GRedeNeural.ErroSqr := GRedeNeural.ErroSqr + (Array_Neuronios[GNoCamadaSaida[0]^.NumInd[xAux1]].Erro *
                           Array_Neuronios[GNoCamadaSaida[0]^.NumInd[xAux1]].Erro);
  end;
end;
```

Figura 20: rotina de treino dos valores de saída da ativação dos neurônios.

A figura 21 apresenta a rotina de inserção da unidade candidata selecionada como unidade oculta fixa da rede.

```
procedure TRedeNeural.pInsereCand;
var
  xAux1, xAux2 : Integer;
  xStrAux : array [0..10] of char;
begin
  // mova a candidata selecionada para uma unidade de rede oficial: copia pesos, conecta entradas e saidas
  StrCopy(Array_Neuronios[GRedeNeural.Neuronios].Id, GCandPOol[GCandidatas.MelhorIndice].Id);
  Array_Neuronios[GRedeNeural.Neuronios].Camada := GRedeNeural.Camadas - 1;
  Array_Neuronios[GRedeNeural.Neuronios].QtdEntradas := GCandPool[GCandidatas.MelhorIndice].QtdEntradas;
  Array_Neuronios[GRedeNeural.Neuronios].QtdSaidas := GRedeNeural.Saidas;

  // ajusta conexoes de entrada
  Array_Neuronios[GRedeNeural.Neuronios].Bias := GCandPool[GCandidatas.MelhorIndice].Bias;
  Array_Neuronios[GRedeNeural.Neuronios].Bias_delta := 0;
  Array_Neuronios[GRedeNeural.Neuronios].Bias_epdt := 0;

  for xAux1 := 0 to GCandidatas.QtdEntradas - 1 do
  begin
    if (GCandidatas.Tipo[xAux1] = 0) then  // é uma unidade de entrada
    begin
      Array_GNoEntrada[GCandidatas.UInd[xAux1]].
        ConSaida[Array_GNoEntrada[GCandidatas.UInd[xAux1]].QtdCon].Neuronio := GRedeNeural.Neuronios;
      Array_GNoEntrada[GCandidatas.UInd[xAux1]].
        ConSaida[Array_GNoEntrada[GCandidatas.UInd[xAux1]].QtdCon].Entrada := xAux1;
      Inc(Array_GNoEntrada[GCandidatas.UInd[xAux1]].QtdCon);
    end
    else
    begin
      Array_Neuronios[GCandidatas.UInd[xAux1]].
        Con[Array_Neuronios[GCandidatas.UInd[xAux1]].QtdSaidas].Neuronio := GRedeNeural.Neuronios;
      Array_Neuronios[GCandidatas.UInd[xAux1]].
        Con[Array_Neuronios[GCandidatas.UInd[xAux1]].QtdSaidas].Entrada := xAux1;
      Inc(Array_Neuronios[GCandidatas.UInd[xAux1]].QtdSaidas);
    end;
    Array_Neuronios[GRedeNeural.Neuronios].Peso[xAux1] := GCandPool[GCandidatas.MelhorIndice].Peso[xAux1];
    Array_Neuronios[GRedeNeural.Neuronios].Entrada[xAux1] := 0;
    Array_Neuronios[GRedeNeural.Neuronios].Delta[xAux1] := 0;
    Array_Neuronios[GRedeNeural.Neuronios].Epdt[xAux1] := 0;
  end;

  // ajusta conexões de saida
  GNoCamada[0] := GNoCamadaSaida[0];

  for xAux1 := 0 to Array_Neuronios[GRedeNeural.Neuronios].QtdSaidas - 1 do
  begin
    Array_Neuronios[GRedeNeural.Neuronios].Con[xAux1].Neuronio := GNoCamada[0]^.NumInd[xAux1];
    Array_Neuronios[GRedeNeural.Neuronios].Con[xAux1].Entrada := Array_Neuronios[GNoCamada[0]^.NumInd[xAux1]].QtdEntradas;
    Inc(Array_Neuronios[GNoCamada[0]^.NumInd[xAux1]].QtdEntradas);
  end;

  // ajusta ponteiros das camadas
  SetLength(GNoCamada, 1);
  new(GNoCamada[0]);
  if (GNoCamadaRaiz[0] = GNoCamadaSaida[0]) then
    GNoCamadaRaiz[0] := GNoCamada[0]
  else
    (GNoCamadaSaida[0]^.Anterior)^.Proximo := GNoCamada[0];

  GNoCamada[0]^.Anterior := GNoCamadaSaida[0]^.Anterior;
  GNoCamadaSaida[0]^.Anterior := GNoCamada[0];
  GnoCamada[0]^.Proximo := GNoCamadaSaida[0];
  GNoCamada[0]^.NumNeur := 1;
  GNoCamada[0]^.NumInd[0] := GRedeNeural.Neuronios;

  // congela todos os pesos da candidata selecionada
  Array_Neuronios[GRedeNeural.Neuronios].Congelado := true;

  // incrementa camada da unidade de saida - rede tem mais uma camada
  GNoCamada[0] := GNoCamadaSaida[0];
  for xAux1 := 0 to GRedeNeural.Saidas - 1 do
    Inc(Array_Neuronios[GNoCamada[0]^.NumInd[xAux1]].Camada);

  // Randomiza todos os pesos de saida
  for xAux1 := 0 to GRedeNeural.Saidas - 1 do
  begin
    Array_Neuronios[GNoCamada[0]^.NumInd[xAux1]].Bias := pRandom(GRedeNeural.EscalaAlea);
    for xAux2 := 0 to Array_Neuronios[GNoCamada[0]^.NumInd[xAux1]].QtdEntradas - 1 do
      Array_Neuronios[GNoCamada[0]^.NumInd[xAux1]].Peso[xAux2] := pRandom(GRedeNeural.EscalaAlea);
  end;

  for xAux1 := 0 to GRedeNeural.Saidas - 1 do
    Array_Neuronios[GNoCamada[0]^.NumInd[xAux1]].Peso[Array_Neuronios[GNoCamada[0]^.NumInd[xAux1]].QtdEntradas - 1] :=
                                          -GCandPool[GCandidatas.MelhorIndice].CorrelAnter[xAux1]^;

  // adiciona esta unidade no record GCandidata -> prepara o aprendizado da proxima candidata
  GCandidatas.Tipo[GCandidatas.QtdEntradas] := 1;   // é uma unidade oculta
  GCandidatas.UInd[GCandidatas.QtdEntradas] := GRedeNeural.Neuronios;
  Inc(GCandidatas.QtdEntradas);   // incrementa a quantidade de entradas das unidades candidatas da rede

  Inc(GRedeNeural.Neuronios);     // incrementa o total de camadas da rede
  Inc(GRedeNeural.Camadas);       // incrementa o total de neuronios da rede
  Inc(GRedeNeural.NovasUnidades); // altera numero de novas unidades adicionadas pelo CasCor
end;
```

Figura 21: inserção da unidade candidata à rede.

Na figura 22 é apresentada a rotina que gera o arquivo de pesos após o término do treinamento da base de conhecimento.

```
procedure TArquivos.pSalvaPesos;
var
  xInt, xNc, xNi: Integer;
  xFlag: Boolean;
  xFile: TextFile;
begin
  AssignFile(xFile, gDirAplicacao + '\Arquivos\Temp\Pesos_temp.txt');
  Rewrite(xFile);

  Writeln(xFile, Format('%d %d %d %d %s', [GRedeNeural.Entradas, GRedeNeural.Saidas, GRedeNeural.Camadas, GRedeNeural.Neuronios,
'CasCor']));

  for xNi := 0 to GRedeNeural.Entradas - 1 do
    for xNc := 0 to Array_GNoEntrada[xNi].QtdCon - 1 do
      Writeln(xFile, Format('I %d %d %3.18f', [xNi, Array_GNoEntrada[xNi].ConSaida[xNc].Neuronio,
        Array_Neuronios[Array_GNoEntrada[xNi].ConSaida[xNc].Neuronio].Peso[Array_GNoEntrada[xNi].ConSaida[xNc].Entrada]]));

  // escaneia todas as conexoes de saida dos neuronios
  for xNi := 0 to GRedeNeural.Neuronios - 1 do
  begin
    for xNc := 0 to Array_Neuronios[xNi].QtdSaidas - 1 do
    begin
      Writeln(xFile, Format('N %d %d %3.18f', [xNi, Array_Neuronios[xNi].Con[xNc].Neuronio,
        Array_Neuronios[Array_Neuronios[xNi].Con[xNc].Neuronio].Peso[Array_Neuronios[xNi].Con[xNc].Entrada]]));
    end;
    Writeln(xFile, Format('B %d %d %d %3.18f', [xNi, BooleanToInt(Array_Neuronios[xNi].Congelado), 4,
Array_Neuronios[xNi].Bias]));
  end;

  Writeln(xFile, 'End');
  CloseFile(xFile);
end;
```

Figura 22: geração do arquivo de pesos após o treinamento da rede neural.

3.3.2 Geração das Imagens de Impressões Digitais

Para o treinamento de uma rede neural, é necessário que uma quantidade satisfatória de exemplos de IID's sejam submetidas à este treinamento, possibilitando a rede a adquirir a capacidade de generalização e classificação. Para obtenção de uma base de conhecimento deste porte, as IID's deste trabalho foram obtidas através de um software de geração artificial de impressões digitais denominado SFinge (FINGERPRINTS, 2005).

Este software possui diversos recursos para gerar uma impressão digital que simule um exemplo bem próximo à realidade. Entre algumas destas funcionalidades, é possível definir atributos como a largura do lado esquerdo e do lado direito, e o comprimento do lado superior e inferior das impressões digitais. Outros itens configuráveis são: a direção e a espessura das papilas, a localização dos deltas, se a ID possui arranhões (cicatrizes), a pressão e a posição do ID na hora em que foi submetida, entre outros.

A figura 23 retrata algumas das telas que realizam este processamento nas IID's:

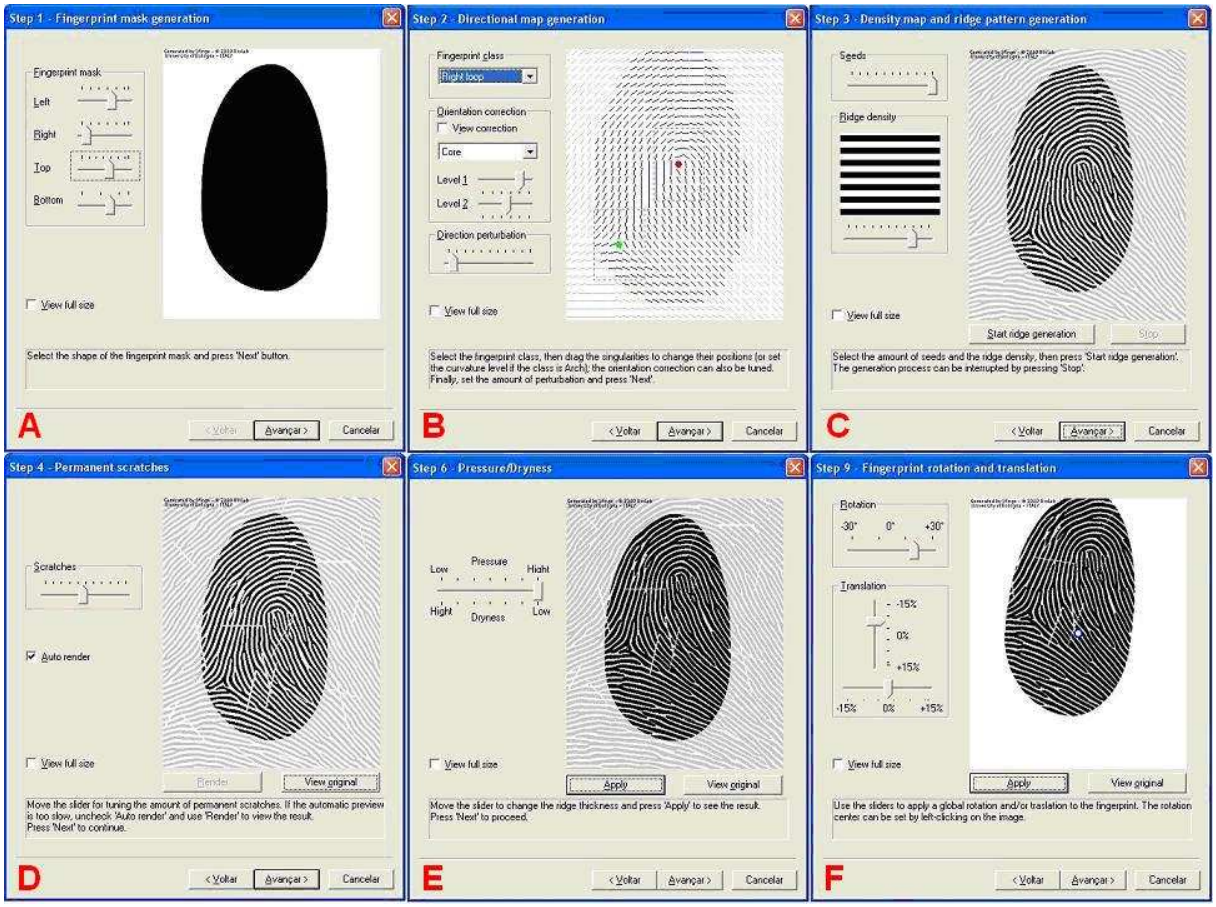

Figura 23: telas de definições de atributos para a geração das IID's.

A tela identificada com a letra "A" permite definir a largura e o comprimento da impressão digital. A tela "B" serve para configurar a direção das papilas, a quantidade de minúcias e a posição do delta. A tela "C" serve para configurar a espessura das papilas. A tela "D" permite incluir "arranhões" na imagem, simulando imperfeições ou cicatrizes do dedo. A tela "E" permite configurar a pressão do dedo no momento em que a impressão digital foi submetida e, finalmente, a tela "F" define a posição do dedo na imagem.

As IID aqui geradas também servirão para realizar os testes do desempenho da rede no reconhecimento de impressões digitais.

3.3.3 Processamentos das Imagens de Impressões Digitais

Como já se sabe, os valores de entrada dos neurônios da camada de entrada são obtidos através de um componente externo que, no caso deste trabalho, é a IID. As IID's servem como *input*[6] no treinamento da base de conhecimento e na extração dos dados da impressão digital do usuário, tanto no cadastro de usuários quanto na fase do reconhecimento da impressão digital.

As imagens geradas pelo SFinge não exigem muito processamento, pois já são bastante legíveis e livres de ruídos. Porém, este fato não necessariamente pode ser um ponto positivo para o reconhecimento das ID's. Os pesos obtidos no treinamento baseado nestas IID's "perfeitas" pode não retratar a realidade. Ou seja, no momento em que a rede for realmente submetida ao reconhecimento de uma IID verdadeira, o resultado poderá ser diferente do obtido no reconhecimento de uma IID artificial. Caso este fato realmente ocorra, é possível que a viabilização de *Cascade Correlation* para o reconhecimento de impressões digitais ainda seja válida, pois mesmo sendo IID's artificiais, a rede possuiu a capacidade de generalizar, classificar e aprender o reconhecimento destas impressões digitais.

Este trabalho não pretende treinar a base e nem realizar o reconhecimento das IID's utilizando algum tipo de processo de classificação de minúcias conforme realizado em Gumz (2002) e Matias (2004), visto que estes dois trabalhos não apresentaram resultados satisfatórios. Como não foram encontrados trabalhos relacionados ao reconhecimento de impressões digitais utilizando o algoritmo *Cascade Correlation*, achou-se que primeiramente deveria ser realizado um trabalho que analisasse toda ou grande parte da IID, para que após a obtenção dos resultados deste trabalho, fosse concluído se há necessidade de pesquisar outras alternativas para melhoria destes resultados.

---

[6] Diz-se *input* os valores externos que são utilizados como entrada dos neurônios na camada de entrada.

As imagens de maior resolução geradas pelo SFinge possuem 432x480 *pixels*. Considerando que cada pixel é um neurônio da camada de entrada, se estas IID's não fossem processadas, a rede neural possuiria 207.360 neurônios somente na camada de entrada. Este valor é bastante inviável, pois tornaria o treinamento da rede muito lento e exigiria uma base de dados muito grande para que a rede neural pudesse obter alguma capacidade de aprendizado.

A fim de diminuir a quantidade de neurônios da camada de entrada sem perder as características mais relevantes da impressão digital, o processamento das IID's para este trabalho foi feito da seguinte forma:

a) selecionou-se 50% da imagem, sendo que no sentido horizontal esta seleção é centralizada, ou seja, a partir do *pixel* 108 até o 324. Já, no sentido vertical, a região selecionada parte do *pixel* 130 até o 370. A figura 24 mostra a região selecionada de uma IID gerada pelo SFinge:

Figura 24: seleção de 50% da IID.

b) esta seleção resultou em um novo arquivo de 216x240 *pixels*, o que criaria uma camada de entrada da rede neural de 51.840 neurônios. Apesar do valor ter reduzido em ¼, este número ainda é muito alto. Como um segundo processamento,

a imagem de 216x240 *pixels* foi contraída em 50%, gerando uma imagem conforme o ilustrado na figura 25:

Figura 25: encolhimento de 50% da IID.

Após este processo de encolhimento, a IID passou a ter 108x120 *pixels*, totalizando 12960 neurônios na camada de entrada, que ainda é um número bastante alto. Após este passo, foram feitos diversos testes para reduzir ainda mais o tamanho da imagem, mas em nenhum deles obteve-se sucesso, pois prejudicavam a qualidade da IID. O processamento destas IID's foi feito manualmente, utilizando a ferramenta da Adobe Photoshop na versão 7.0.

Para solução deste problema, foi cogitada a idéia de que a redução do número de neurônios poderia ser feita no momento da criação da rede e da leitura dos arquivos de imagens. Sendo assim, foi decidido que somente 1 a cada 3 *pixels* da IID deveriam tornar-se um neurônio, fazendo com que a camada de entrada passasse a ter 4320 neurônios. Este número ainda é consideravelmente grande, mas em testes futuros no treinamento da base de conhecimento, concluiu-se que é uma quantidade bastante aceitável, pois os pesos gerados aparentavam ser concisos e o tempo para a finalização do treinamento não era muito grande.

A figura 26 apresenta o fragmento do código fonte onde a leitura da imagem é realizada e os *pixels* são selecionados:

```
xIndArq := 0;
while (xAux = 0) do
begin
  xIndice := 0;

  // verifica se foi escolhido somente um arquivo
  if (FileExists(edDiretorio.Text)) then
    xImagem.Picture.Bitmap.LoadFromFile(edDiretorio.Text)
  else
    xImagem.Picture.Bitmap.LoadFromFile(edDiretorio.Text + '\' + xSearchArq.Name);

  for xLinha := 0 to cAlturaImagem - 1 do
    for xColuna := 0 to cLarguraImagem - 1 do
    begin
      // se o indice do pixel está no modulo 3, converte a imagem.
      if ((xIndice mod 3) = 0) then
      begin
        if (xImagem.Canvas.Pixels[xColuna, xLinha] = clBlack) then
          gDataL[xIndArq].Entrada[xIndice div 3] := 0.3
        else
          gDataL[xIndArq].Entrada[xIndice div 3] := -0.2;
      end;

      Inc(xIndice);
    end;

  // passa o valor desejado para o vetorh de saida
  if (FileExists(edDiretorio.Text + '\S' + Copy(xSearchArq.Name, 2, 3) + '.txt')) then
  begin
    AssignFile(xFileSaida, edDiretorio.Text + '\S' + Copy(xSearchArq.Name, 2, 3) + '.txt');
    Reset(xFileSaida);
    FillChar(xBuffer, sizeOf(xBuffer), 0);
    Readln(xFileSaida, xBuffer);
    gDataL[xIndArq].Saida[0] := StrToFloat(Trim(xBuffer));
    CloseFile(xFileSaida);
  end
  else
  begin
    { Se não tiver arquivo de saida, joga as constantes 0.5 para
      indice de arquivo par, e -0.5 para os demais }
    if ((xIndArq mod 2) = 0) then
      gDataL[xIndArq].Saida[0] := 0.5
    else
      gDataL[xIndArq].Saida[0] := -0.5;
  end;
  xAux := FindNext(xSearchArq);
  Inc(xIndArq);
end;
```

Figura 26: leitura da IID e seleção de 1 a cada 3 *pixels* como neurônio de entrada.

3.3.4 Operacionalidade da implementação

O protótipo possui 3 telas: uma para o treinamento da base de conhecimento, uma para cadastro dos usuários e suas respectivas impressões digitais e a última para reconhecimento das impressões digitais. Os recursos e comportamentos destas telas serão detalhados nesta seção.

3.3.4.1 Tela de Treinamento da Rede Neural

A tela de treinamento da rede neural, conforme o nome já diz, serve para realizar o treino dos pesos das conexões entre os neurônios e salvá-los em um arquivo de texto após sua finalização. O caminho para acesso a esta tela, assim como a própria tela, podem ser visualizados na figura 27.

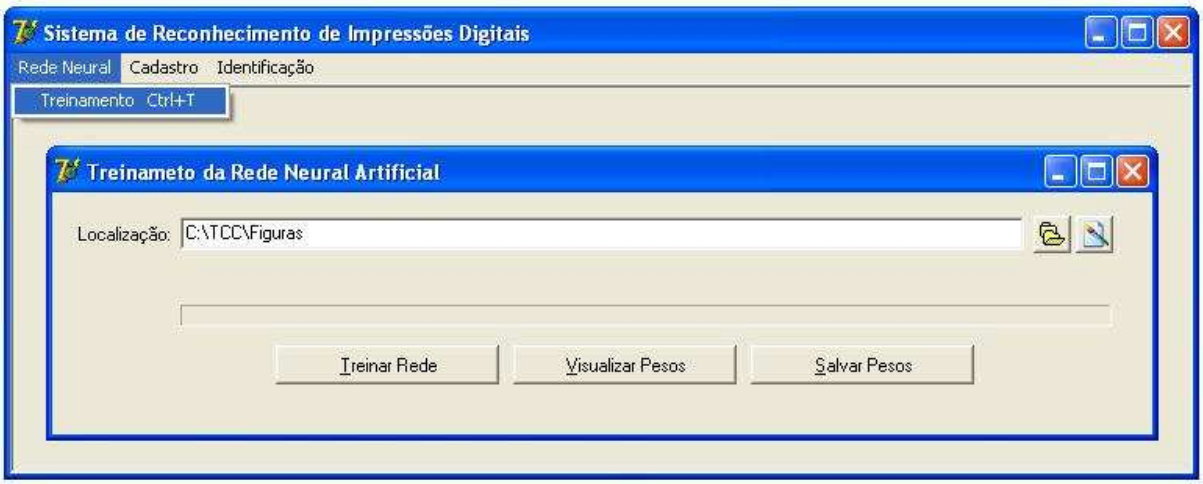

Figura 27: caminho e tela de treinamento da rede neural.

As imagens que serão utilizadas no treinamento devem possuir extensão “BMP” e podem ser selecionadas de duas maneiras: através de um diretório de IID’s desta mesma extensão ou selecionando uma imagem específica.

Para selecionar um grupo de IID’s a partir de um diretório, deve-se clicar no primeiro botão ao lado do campo “Localização”, que abrirá uma caixa de diálogo para a seleção de diretórios, conforme ilustrado na figura 28.

Figura 28: seleção do diretório de grupo de IID's.

Caso o usuário queira treinar somente uma IID especifica, deverá ser clicado no segundo botão ao lado do campo "Localização". Assim, será aberta uma caixa de diálogo para seleção de arquivos com filtro para extensões "BMP". Esta operação pode ser visualizada através da figura 29.

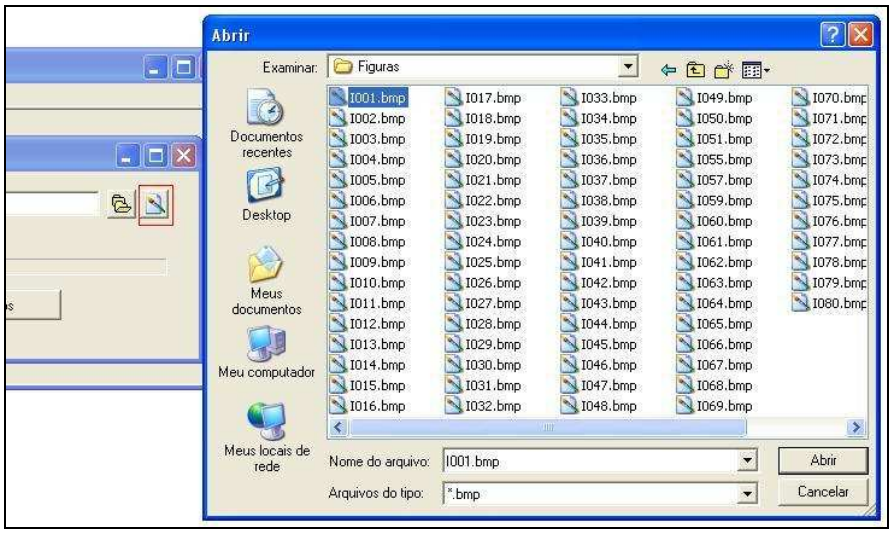

Figura 29: seleção individual de IID's.

O botão "Treinar Rede", que pode ser visualizado na figura 27, é responsável pelo processo de treinamento da rede neural baseando-se nas IID's indicadas no campo "Localização". Em nenhuma outra parte do sistema há possibilidade executar o treinamento, ou parte de um treinamento, que não seja através deste botão.

Para o funcionamento correto do treinamento e a geração de pesos concisos, as IID's

devem seguir o padrão descrito no item 3.3.3. Caso contrário, o funcionamento da rede neural e os resultados obtidos no reconhecimento podem não ser iguais ou aproximado aos obtidos neste trabalho.

A nomenclatura das IID's devem respeitar o seguinte padrão: o identificador "I" na primeira letra, um número seqüencial com 3 dígitos e a extensão BMP. O treinamento da base de conhecimento necessita que seja configurado um valor desejado de saída para cada IID que será treinada. A leitura desses valores de saída é feita através de arquivos de texto que deverão conter um valor entre –1 e 1 em seu conteúdo, valor este definido com base na função sigmoidal hiperbólica tangente, conforme visto no tem 2.2.3 deste trabalho. Cada IID deverá possuir um arquivo de saída no mesmo diretório, respeitando uma nomenclatura semelhante das imagens, com o identificador "S" na primeira letra, o número seqüencial de 3 dígitos relacionada a sua IID e a extensão "TXT". Portanto, ao realizar o treinamento da base, o sistema selecionará todas as IID's do diretório informado, irá ler o arquivo que contém o valor desejado de cada imagem e treinará as IID's até seus valores de saída chegarem próximo ao valor desejado.

Ao finalizar o processo de treinamento, o sistema criará uma pasta chamada "Temp" no diretório onde está situado o executável do software. Nesta pasta serão armazenados os arquivos de pesos e topologia gerados pelo treinamento. Em cada processo de treinamento que é finalizado, estes arquivos são atualizados. Porém, ao tentar realizar um reconhecimento de impressão digital, estes pesos não serão utilizados na comparação das duas imagens, pois ainda não foram efetivados. Para realizar a efetivação, é necessário clicar no botão "Efetivar Pesos" encontrado na tela de treinamento, no qual copiará os arquivos temporários para a pasta raiz do executável. A rotina de reconhecimento sempre utilizará o arquivo de pesos situado na pasta raiz do executável, e não na pasta "Temp".

Esta tela também possui o botão "Visualizar Pesos" que, ao ser clicado, abrirá um

arquivo de texto listando todos os pesos treinados. Esta funcionalidade facilita o processo de treinamento da base de conhecimento, pois permite o usuário acompanhar os pesos alterados entre cada treinamento.

3.3.4.2 Tela de Cadastro de Usuários

A tela de cadastro de usuários permite cadastrar os dados básicos e a IID de usuários, que futuramente serão carregados na tela de reconhecimento de impressões digitais. O caminho para o acesso, bem como a própria tela, podem ser vistos na figura 30.

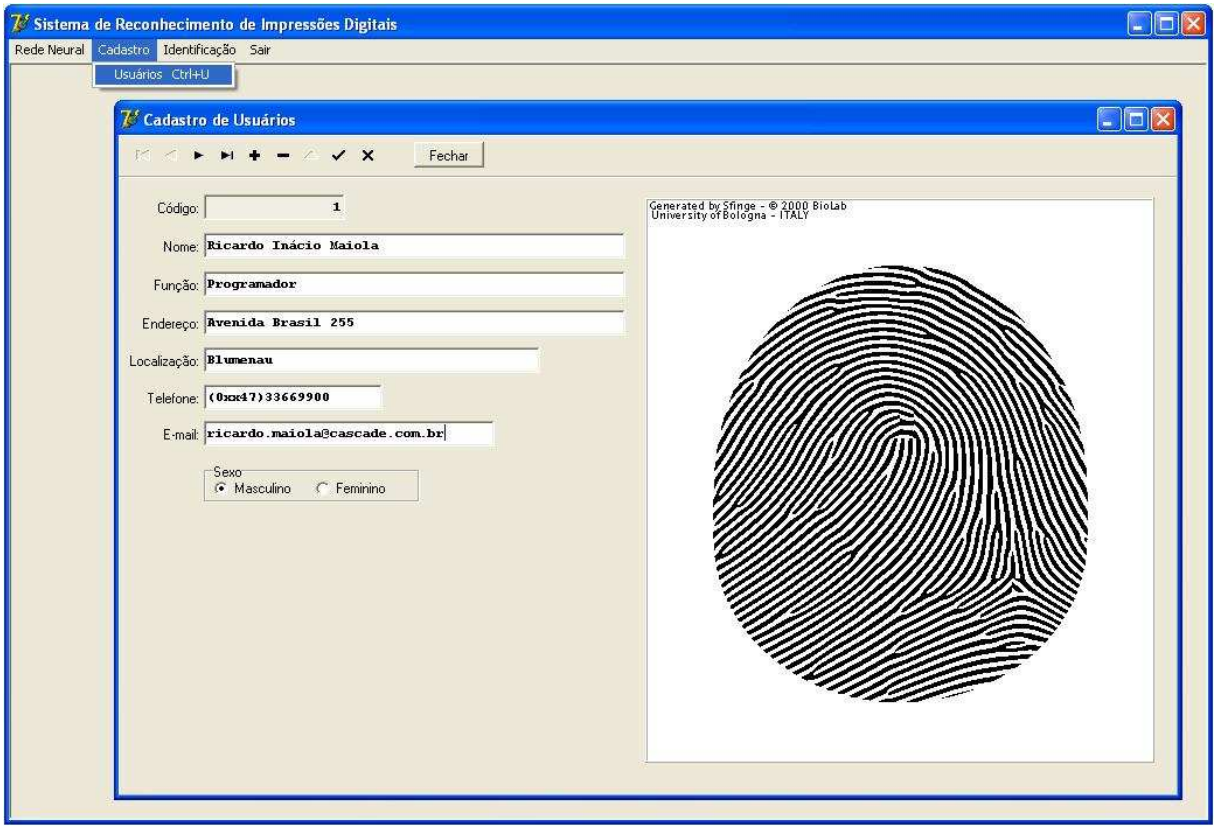

Figura 30: caminho e tela de cadastro de usuários.

Para cadastrar uma IID de um usuário, deve-se colocar a tabela em modo de inserção ou alteração, clicar no espaço reservado para a imagem e selecionar a IID do usuário através do dialog que será aberto. A IID selecionada será copiada para um diretório particular do usuário, que pode ser encontrado na pasta “base” localizada dentro do diretório da aplicação.

3.3.4.3 Tela de Identificação de Impressões Digitais

A tela de identificação de impressões digitais permite o usuário selecionar uma IID qualquer e retornar o usuário cadastrado no sistema portador desta impressão digital. O caminho e a tela de identificação de impressões digitais podem ser visualizados na figura 31.

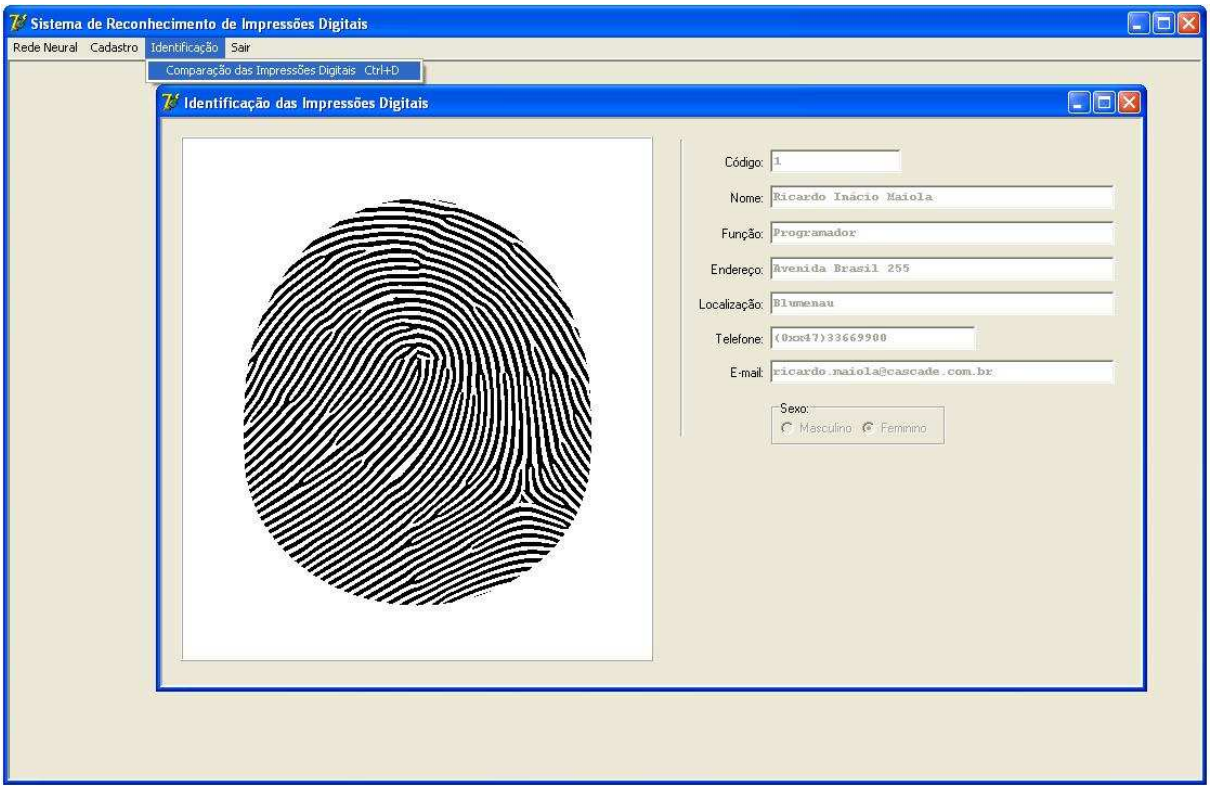

Figura 31: caminho e tela de identificação de impressões digitais.

Para o reconhecimento de uma impressão digital, deve-se clicar na área reservada à imagem, abrindo assim um dialog para seleção da IID. Ao selecionar uma IID, o sistema extrairá o valor de saída desta imagem e realizará uma comparação com o valor de saída de todas as IID's do cadastro de usuários. Quando uma IID de usuário possuir um valor de saída aproximado ao valor da IID selecionada, o sistema retornará os dados do usuário nos campos da tela.

3.3.4.4 Treinamento da base de conhecimento

A base de conhecimento utilizada no treinamento da rede neural foi obtida através da ferramenta SFinge, já mencionada no item 3.3.2 deste trabalho. Com esta ferramenta, foram geradas 75 IID's de diversas formas e configurações, para que a rede neural seja capaz de obter sua capacidade de generalização e classificação.

Além da base para o treinamento, é necessário criar outra base para a simulação de situações decorrentes na vida real, o chamado teste de validação. A base de validação contém 40 IID's que foram incluídas através do cadastro de usuários.

Nesta etapa, as IID's da base de conhecimento foram submetidas a treinamentos e validações continuamente, até a obtenção de um índice de acertos consideravelmente aceitável.

Primeiramente fez-se o treinamento da base de conhecimento para a obtenção dos pesos da rede neural. A partir destes pesos, foram realizados testes na tela de identificação de impressões digitais, verificando o índice de IID's que eram aceitas e rejeitadas indevidamente (falsa aceitação e falsa rejeição, conforme mencionado no item 2.1.1). Estes testes foram feitos repetidamente até adquirir-se o melhor e mais aceitável índice de acertos e rejeições corretos. Uma das vantagens do modelo *Cascade Correlation* é o rápido aprendizado em relação aos outros modelos de redes neurais, o que facilitou o processo de treinamento e validação da base. O número de épocas executadas em cada treinamento é bastante variável, o que impede a criação de uma estimativa numérica neste trabalho. Porém, a tendência é que este número reduza gradativamente após cada treinamento finalizado.

O treinamento da base de conhecimento possui diversos parâmetros que podem ser configurados internamente. Estes parâmetros são responsáveis pela diferença dos índices de aceitação e rejeição entre cada etapa de validação. Portanto, para a validação da base de

conhecimento, esses parâmetros foram alterados incessantemente até o momento que foram obtidos os resultados que serão apresentados nas seções 3.4.1 e 3.4.2.

Entre os parâmetros mais importantes no treinamento e reconhecimento de IID's está o parâmetro que define o valor de cada *pixel* da imagem que será treinada. No caso deste protótipo, uma IID pode gerar somente dois valores de entrada, que foram definidos como 0,30 para os *pixels* de cor preta e 0,15 para os *pixels* de cor branca. A definição destes valores foi feita após a realização de diversos testes com diferentes valores e intervalos. Através destes testes, notou-se que o intervalo muito grande entre os valores de cada cor causava baixos índices de reconhecimento, pois qualquer alteração mínima na IID resultava em um valor final muito além da tolerância.

Outro parâmetro importante para o reconhecimento de impressões digitais é o valor da tolerância, que já foi mencionado no item 2.1.1 deste trabalho. Houve uma certa dúvida na definição deste valor, pois uma tolerância muito grande pode aumentar consideravelmente o índice de falsa aceitação, comprometendo a segurança do sistema. Em compensação, caso setada uma tolerância muito baixa, o sistema poderá não reconhecer qualquer IID submetida, mesmo se esta existir na base de dados. Porém, uma falsa aceitação pode gerar conseqüências negativas muito grandes, comprometendo a visão do usuário quanto à segurança dos reconhecimentos do sistema. Dado este motivo, o valor de tolerância foi configurado como 0,001, um valor relativamente baixo que favorece o índice de aceites corretos.

Finalmente, o terceiro parâmetro mais importante no processo de treinamento da base de conhecimento é o erro máximo que um neurônio de saída pode possuir depois de treinado. Se o erro de um qualquer neurônio da camada de saída for inferior a 0,1, o processo de treinamento é finalizado.

Além dos parâmetros mencionados, existem outros que não precisaram ser alterados durante a validação da base de treinamento:

a) parâmetros *quickprop*: são parâmetros utilizados na técnica *quickprop*, desenvolvida por Fahlman (1988), e utilizada para ajustar os pesos dos neurônios. Entre eles, existe o “FatorMU” e o “FatorSrk”, setados inicialmente com os valores 1,75 e 0,6363 e que são alterados de acordo com o ajuste dos pesos. O “Epsilon”, que possui o valor 0,000001, é utilizado para realizar o ajuste dos pesos proporcionalmente à quantidade de exemplos da base de conhecimento;

b) parâmetros de paciência: são parâmetros utilizados na análise da estagnação do treinamento. Existem dois parâmetros de paciência: o “SaidaPacien” e o “OculPacien”, ambos setados com o valor 8. O parâmetro “SaidaPacien” verifica se as 8 últimas épocas do treinamento estão estagnadas, para então iniciar o processo de inserção de uma nova unidade oculta. O parâmetro “OculPacien” verifica se as 8 últimas épocas de treinamentos das unidades candidatas estagnaram, para então iniciar o processo de análise de correlação das candidatas;

c) parâmetro de randomização: na criação de uma nova rede neural, os pesos iniciais das conexões são obtidos através de uma processo de randomização. Para isto, foi criado um parâmetro com o valor 0,001, que é utilizado em um cálculo baseado na data atual. A randomização dos pesos pode ser feita de qualquer outra forma, desde que não gere valores muito altos que ultrapassem o limite suportado da variável de ponto flutuante.

3.4 RESULTADOS E DISCUSSÃO

Nesta seção serão descritos os testes de falsa aceitação e falsa rejeição realizados para a validação do protótipo. A partir desses testes, foram extraídos resultados que serão mencionados posteriormente.

3.4.1 Teste de falsa aceitação

O teste de falsa aceitação consiste em analisar o percentual de IID's que foram submetidas ao reconhecimento e foram identificadas como pertencentes à outra pessoa. A falsa aceitação pode causar graves inconvenientes em uma organização, pois compromete a segurança do pessoal e do patrimônio. Além disso, a visão da confiabilidade no *software* de reconhecimento pode ser afetada gravemente, dependendo do dano que esta falha pode causar.

Para a realização deste teste, foi utilizada a base de dados de 40 IID's cadastradas em diferentes usuários. Na tela de identificação, foram selecionadas as IID uma por uma, analisando o cadastro de usuário retornado após cada digital submetida. Um exemplo de reconhecimento pode ser visto na figura 32.

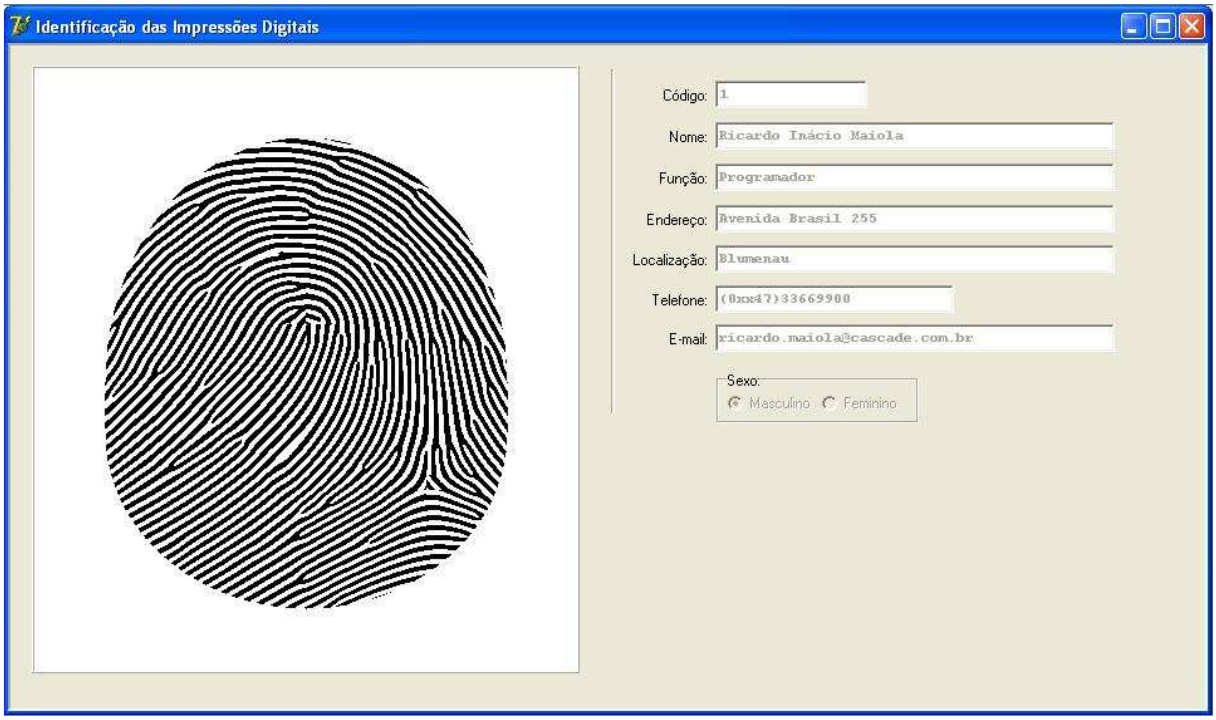

Figura 32: reconhecimento de uma IID.

Este tipo de reconhecimento apresentou bons resultados nos testes que foram

realizados. Nos primeiros treinamentos da base de conhecimento, foi identificada uma média de 3 erros no reconhecimento em testes realizados em toda as 40 IID’s. Pode-se verificar na figura 33 as digitais submetidas ao reconhecimento ao lado das identificadas indevidamente.

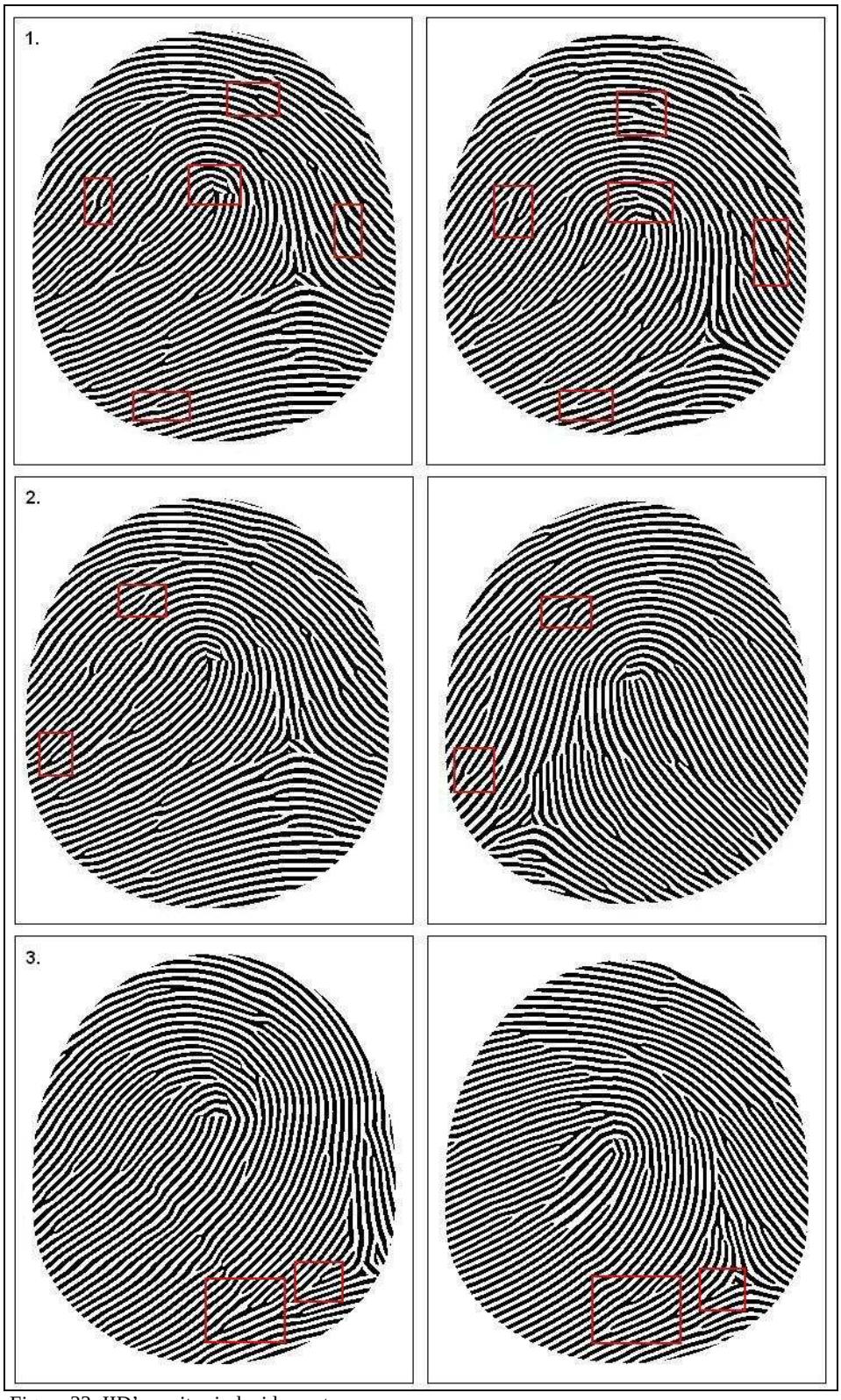

Figura 33: IID’s aceitas indevidamente.

A figura 32 apresenta as 3 comparações das IID’s que resultaram no reconhecimento incorreto, destacando os pontos semelhantes entre cada imagem comparada. Na linha 1, pôde-se verificar que em diversos pontos da impressão digital existem semelhanças nos desenhos das papilas, o que pode ter causado o reconhecimento incorreto da IID. Além desses pontos semelhantes, a impressão digital possui um padrão semelhante do desenho das papilas, como a posição do delta e do núcleo e a região da metade superior da imagem. Na 2ª e 3ª comparação, as impressões digitais reconhecidas incorretamentes não possuem muita semelhança. As causas mais prováveis destes reconhecimentos indevidos podem ser: a base de conhecimento, que pode ter seus pesos melhorados submetendo novos exemplos ao treinamento; ou o pré-processamento das imagens, que pode ter distorcido características importantes das IID’s.

Após a apuração destes resultados, passou-se a executar outros treinamentos. Enquanto estes treinamentos eram realizados, a média de erro foi-se reduzindo até tornar-se nula, ou seja, o treinamento da rede conseguiu classificar as IID’s, configurando pesos em que todas as IID’s puderam ser reconhecidas entre si, resultando 100% de aceites corretos.

3.4.2 Teste de falsa rejeição

O teste de falsa rejeição analisa o percentual de IID’s que são submetidas ao reconhecimento e são rejeitados indevidamente, ou seja, a IID submetida está armazenada na base de dados, mas o sistema não reconhece. É necessário que o sistema tenha uma determinada tolerância à rejeição, pois diversos fatores podem alterar o valor de saída de uma mesma IID, como a posição do dedo, a pressão em que é submetido, cortes, arranhões, cicatrizes, entre outros. Nos testes de rejeição realizados, o valor da tolerância foi definido com uma certa rigidez para que o índice de falsa aceitação não fosse prejudicado.

Para a realização dos testes, foi usada a mesma base de IID’s do teste de falsa aceitação, mas com algumas modificações que visam alterar o valor de saída da imagem, como riscos pretos e brancos, buracos e deslocamento do dedo. Assim como cicatrizes ou arranhões afetarão no valor de saída da imagem, estas simulações artificiais servirão para simular o mesmo comportamento. Alguns exemplos de IID’s utilizadas para os testes de falsa rejeição podem ser visualizadas na figura 34.

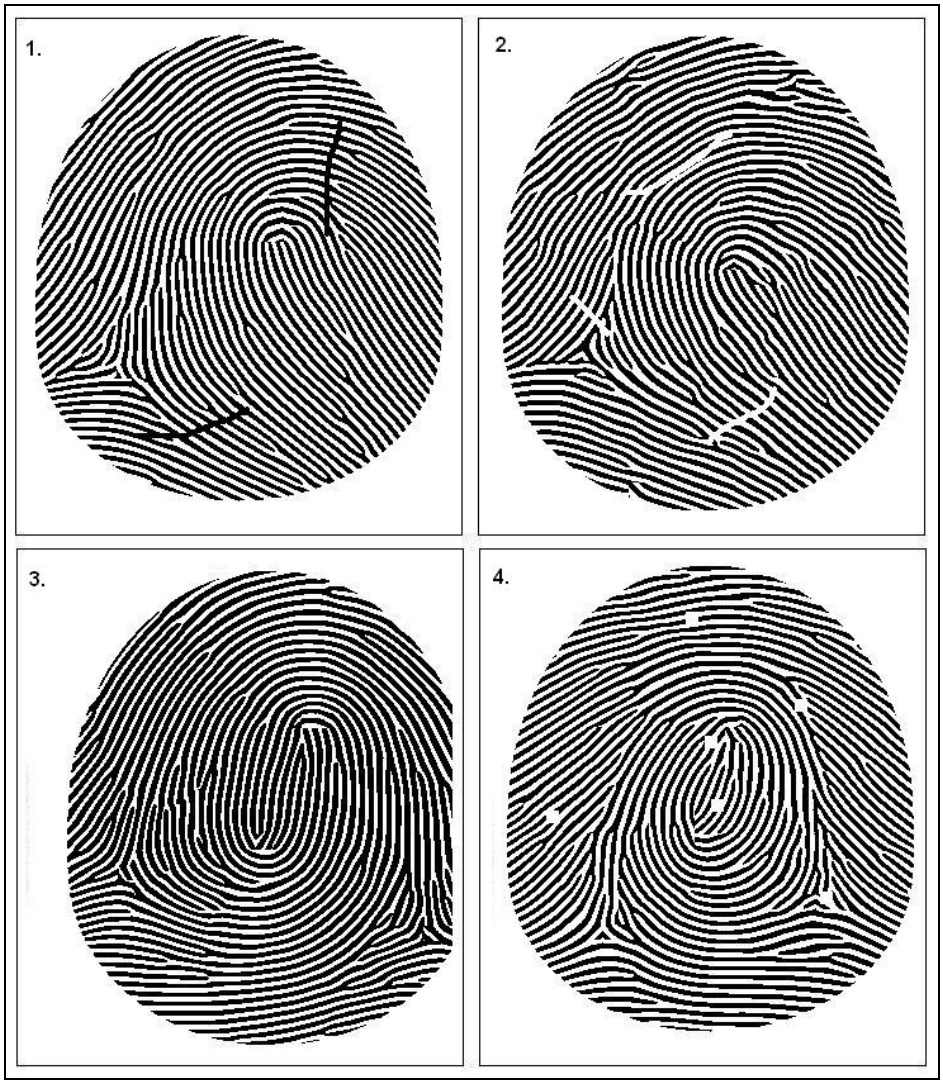

Figura 34: Exemplos de IID’s utilizadas no teste de falsa rejeição.

Os resultados obtidos nos testes de falsa rejeição não chegaram ao mesmo percentual dos testes de falsa aceitação, porém não deixaram de ser bons. Das 40 IID’s testadas, 9 não foram reconhecidas, totalizando-se 77,5% de falsas rejeições. Alguns destes casos de falsas

rejeições podem ser vistos na figura 35.

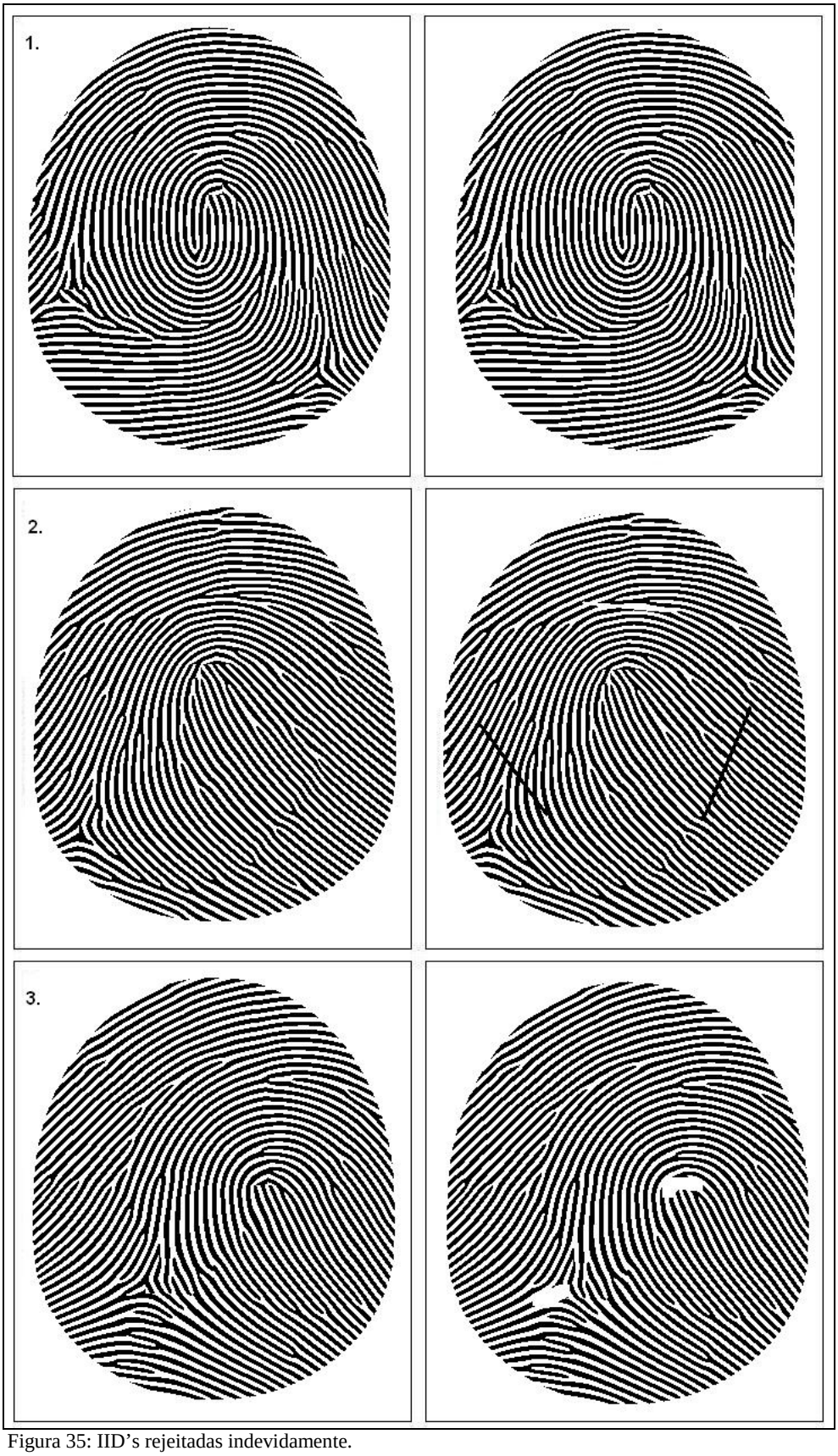

Figura 35: IID's rejeitadas indevidamente.

Na figura 24, são apresentados 3 casos de falsas rejeições identificados nos testes realizados. A linha 1 tenta simular o caso de deslocamento do dedo no momento da submissão da digital, que acaba perdendo bastante informação do lado direito da IID. A linha 2 simula cortes ou sujeiras que eventualmente podem existir no momento da identificação, o que fez a impressão digital ganhar um considerável detalhe em preto, que pode ter aumentado o valor de saída da IID. Na linha 3, tentou-se fazer um reconhecimento deformando o núcleo e o delta da impressão digital. Apesar das IID's pertencerem a usuários cadastrados, ficou claro que o reconhecimento não ocorreu pelo fato das imagens submetidas terem perdido informações muito importantes para reconhecimento, principalmente no 1º e 3º caso mencionado.

Acredita-se que parte destes 22,5% de falsas rejeições podem ter ocorrido pelo fato de não se saber até onde as imperfeições de uma IID podem ser toleradas. Isso faz com que as IID's com alterações mais grotescas utilizadas neste teste não entrem na estatística de falsas rejeições. Porém, durante a realização deste trabalho, não foi encontrado material bibliográfico que abordasse esta questão, o que manteve o resultado original do teste.

3.4.3 Resultados gerais obtidos

Após a realização dos testes relatados nos itens 3.4.1 e 3.4.2, verificou-se que o modelo de rede neural artificial *Cascade Correlation* é viável na utilização de reconhecimento de impressões digitais. Durante os primeiros treinamentos da base de conhecimento, os testes de falsa aceitação demonstraram que 7,5% das impressões digitais foram aceitas indevidamente, ou seja, 92,5% das IID's utilizadas no teste foram reconhecidas corretamente. Porém, realizando continuamente outros processos de treinamento, a rede neural passou a reconhecer todas as IID's entre si, resultando em 100% de acertos no teste de falsa aceitação. Este índice indica que a rede neural obteve capacidade de classificação, aprendizado e

generalização. No teste de falsa rejeição, 77,5% das IID’s foram aceitas corretamente, portanto, 22,5% das IID’s que possuíam imperfeições e deslocamentos consideráveis não foram reconhecidas pelo sistema. Este índice indica que a rede neural possuiu tolerância à falhas, pois mesmo tendo um percentual de falsa rejeição, a maioria das IID’s alteradas foram reconhecidas.

Em relação a Gumz (2002) e Matias (2004), os resultados obtidos foram bastante superiores, o que conclui-se que *Cascade Correlation* pode ser um modelo mais eficaz para o reconhecimento de impressões digitais do que o *Backpropagation*.

Apesar dos bons resultados conseguidos neste trabalho, é possível obter melhorias nos resultados criando uma base de conhecimento maior e mais generalizada, realizando testes em diferentes configurações de parâmetros. Além disso, há necessidade de criar uma base de impressões digitais reais, trabalhando o pré-processamento e colocando esta idéia em encontro com o que o mercado necessita.

O objetivo principal deste trabalho era verificar a viabilidade de *Cascade Correlation* para o reconhecimento de impressões digitais, o que foi comprovado que sim. Mas este trabalho é contínuo e ainda há muito que se fazer, pois este reconhecimento pode ser aprimorado para chegar mais perto possível da perfeição.

4 CONCLUSÕES

Este trabalho teve como objetivo o estudo da identificação de impressões digitais através de redes neurais artificiais. Através dele, foram estudados diversos sistemas de identificação biométrica, principalmente de reconhecimento de impressões digitais para que se conseguisse chegar a resultados satisfatórios.

A partir da proposta da criação de um sistema que verificasse a viabilidade em utilizar o modelo de rede neural artificial *Cascade-Correlation* no reconhecimento de impressões digitais, o trabalho apresentou bons resultados nos reconhecimentos. O destaque está no índice de falsa aceitação, onde se conseguiu treinar uma rede que todas as IID's foram reconhecidas entre si, obtendo o percentual de 100% de aceitações corretas. Houve uma certa dificuldade na realização dos testes de falsa rejeição, pois as IID's apresentadas à rede eram alteradas para a simulação de ruídos, deslocamento do dedo, entre outros e, conseqüentemente, tinha-se dúvida em qual seria a decisão mais segura: aceitar ou rejeitar a impressão digital. Utilizado um baixo índice de tolerância, obteve-se 77,5% de reconhecimento nos testes de falsa rejeição. Estes resultados foram superiores aos obtidos em Gumz (2002) e Matias (2004), que foram os principais trabalhos utilizados como referência.

Verificou-se como principal dificuldade para o desenvolvimento deste projeto, o levantamento bibliográfico, devido à escassez de documentação sobre *Cascade Correlation* e dificuldade na apropriação das informações contidas no material, retardando o tempo para o desenvolvimento da pesquisa. Praticamente não houve limitação tecnológica que dificultasse a finalização do mesmo.

Este trabalho apresentou *Cascade Correlation* como uma ferramenta viável para o reconhecimento de impressões digitais, que demonstrou capacidade de aprendizado, classificação e generalização das imagens apresentadas à rede. Em futuros trabalhos, poderão

ser criadas alternativas para obtenção de melhores resultados dos obtidos neste, pois é possível que *Cascade* Correlation seja mais viável para a identificação de impressões digitais do que qualquer outro modelo de rede neural já existente no mercado. O aprimoramento contínuo neste tipo de *software* é necessário, pois se investe a cada dia mais em soluções de segurança da informação e de pessoal.

REFERÊNCIAS BIBLIOGRÁFICAS

APPES, **Associação dos Papiloscopistas do Espírito Santo**. Espírito Santo, 2005. Disponível em: <http://www.appes.com.br> . Acesso em: 04 abr 2005.

APPOL – **Associação dos Papiloscopistas do Rio de Janeiro.** Rio de Janeiro, 2005. Disponível em: <http://www.appol.com.br>. Acesso em: 04 abr 2005.

BOMBONATTI, José. **História da Dactiloscopia.** São Paulo, 2005. Disponível em: <http://www.aguiarsoftware.com.br/bio_historia.shtml>. Acesso em: 04 abr 2005.

BOREKI, Guilherme. **Sistema de controle de acesso por *iButton* com verificação biométrica da geometria da mão.** Curitiba, 2003. Trabalho de Conclusão de Curso (Bacharelado em Engenharia da Computação), Centro Universitário Positivo.

CASTRO, Leandro Nunes de et al. **Redes Neurais Contrutivas: Uma Abordagem Comparativa.** Campinas, 1999. Disponível em < http://www.ele.ita.br/cnrn/artigos-4cbrn/4cbrn_025.pdf>. Acesso em 20 out 2005.

CEREBROMENTE. **The Brain and Artificial Intelligence.** [S.I.], 1998. Disponível em: <http://www.cerebromente.org.br/n07/opiniao/minsky/minsky_i.htm>. Acesso em 04 abr 2005.

CHANG, David H. **Fingerprint Recognition Through Circular Sampling**. Rochester, 1999. Disponível em: < http://www.cis.rit.edu/research/thesis/bs/1999/chang/thesis.html>. Acesso em 18 out 2005.

COSTA, Silvia M.F. **Classificação e verificação de impressões digitais.** 2001, 123f. Dissertação (Mestrado em Engenharia Elétrica) - Universidade de São Paulo, São Paulo.

DIN. **Uma Introdução às Redes Neurais.** Maringá, 2005. Disponível em: http://www.din.uem.br/ia/neurais/. Acesso em: 21 out 2005.

FAHLMAN, Scott E. **An empirical study of learning speed in back-propagation networks.** Pittsburg, 1988. Computer Science Technical Report CMU-CS-88-162, Carnegie Mellon University.

FAHLMAN, Scott E.; LEBIERE, Christian. **The Cascade-Correlation Learning Architecture.** Pittsburg, 1991. Disponível em: <http://www.cse.unsw.edu.au/~billw/cs9444/fahlman91cascadecorrelation.pdf>. Acesso em 28 set 2005.

FIORESE, Maurício. **Uma Proposta de Autenticação de Usuários para Ensino a Distância.** Porto Alegre, 2000. Universidade Federal do Rio Grande do Sul.

GUMZ, Rafael Araújo. **Protótipo de um sistema de identificação de minúcias em impressões digitais utilizado redes neurais artificiais multicamada**. Blumenau, 2002.

Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação), Universidade Regional de Blumenau.

HEINEN, Milton Roberto. **Autenticação On-line de assinaturas utilizando Redes Neurais.** São Leopoldo, 2002. Trabalho de Conclusão de Curso (Bacharelado em Informática), Universidade do Vale do Rio dos Sinos.

LEMES, Neslon H.T. **Redes Neurais: o *Perceptron*.** [S.I], 2005. Disponível em: <http://www.usuarios.unincor.br/nhtlemes/cpu/percepton.pdf>. Acesso em: 20 out 2005.

MATHWORLD. **Hyperbolic Tangent.** [S.I.], 1999. Disponível em: <http://mathworld.wolfram.com/HyperbolicTangent.html>. Acesso em 20 out 2005.

MATIAS, Caio R. S. **Protótipo de um sistema de identificação do(s) delta(s) e núcleo em impressões digitais utilizando Redes Neurais Artificiais.** Blumenau, 2004. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação), Universidade Regional de Blumenau.

MESONPI. **Introdução á Redes Neurais.** Rio de Janeiro, 1999. Disponível em: <mesonpi.cat.cbpf.br/naj/redesneurais.pdf>. Acesso em: 21 out 2005.

OSORIO, F. S.; HEINEN, M. R. **Biometria Comportamental: Pesquisa e desenvolvimento de um sistema de autenticação de usuários utilizando assinaturas manuscritas.** São Leopoldo, 2002?. Universidade do Vale do Rio dos Sinos.

PEREIRA, Leonardo de Pádua Costa. **Mapeamento de imagens binárias: um estudo sobre a biometria das mãos.** Belém, 2003. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação), Universidade da Amazônia.

RATIONAL. **Rational Rose**: a Rational suite product. Cupertino. Califórnia, 2002. Disponível em: <http://www.rational.com/products/rose/index.jsp>. Acesso em: 03 dez 2005. Rational Software Corporation.

ROCHA, Fabiana Z. F. **Proposta de um padrão manuscrito para reconhecimento automático dos símbolos do sistema SignWriting(SW).** Pelotas, 2003. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação), Universidade Católica de Pelotas.

SEAMA. **Seja bem-vindo ao futuro! Saiba mais sobre a Biometria e como ela será utilizada na instituição.** Maceió, 2005. Faculdade SAEMA.

SIM. **Linha de pesquisa – impressões digitais.** [S.I.], 2005. Disponível em: <http://sim.lme.usp.br/linhas/iinteligente/pimagem/idigital/idigital.html>. Acesso em: 08 out 2005.

SPARX. **Enterprise Architect.** [S.I.], 2006. Disponível em: <http://www.sparxsystems.com.au/products/ea_downloads.html>. Acesso em: 09 fev 2006.

TAFNER, M. A.; MARCOS X.; FILHO I.W.R. **Redes neurais artificiais: introdução e princípios da neurocomputação.** Blumenau: Editora da FURB, 1996.

TONSIG, Sérgio Luiz. **Redes Neurais Artificiais Multicamadas e o Algoritmo de *Backpropagation***. Campinas, 2000. Disponível em: <http://209.123.181.8/~archives/tutoriais/1243.zip >. Acesso em 18/10/2005.

FINGERPRINTS. **Small subset of our fingerprint database**. Cesena, 2005. Disponível em: <http://bias.csr.unibo.it/research/biolab/Fingdb.zip>. Acesso em: 13 dez 2005

VARGAS, Ernesto C.; SOUSA, Humberto C.; CARVALHO, André C. P. L. F. **Reconhecimento de Alvos Utilizando Redes Neurais Construtivas.** Petrópolis, 1998. Disponível em: <http://marte.dpi.inpe.br/col/sid.inpe.br/deise/1999/02.11.15.57/doc/10_229o.pdf>. Acesso em: 18 set 2005.

VIEIRA, R. C.; ROISENBERG, M. **Redes neurais artificiais :** um breve tutorial. Florianópolis, 2003?. Universidade Federal de Santa Catarina.

WIKIPEDIA. **Biometria**. [S.I.], 2005. Disponível em: <http://pt.wikipedia.org/wiki/Biometria>. Acesso em: 13 dez 2005;

ZAPAROLI, Alexsandra. **Protótipo de Software para Controle de Acesso de Funcionários Utilizando Redes Neurais Artificiais para Identificação de Impressão Digital**. Blumenau, 2002. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação), Universidade Regional de Blumenau.