DETECÇÃO DE VOLUME DE TRÁFEGO DE VEÍCULOS PROPORCIONADA POR VISÃO COMPUTACIONAL VIA REDES NEURAIS

Dissertação Submetida à

Universidade Federal da Bahia

como parte dos requisitos exigidos

para a obtenção do título de

Mestre em Engenharia Elétrica

por

Fernando Flores Santos Ribeiro

Janeiro 1999

Fernando Flores Santos Ribeiro

DETECÇÃO DE VOLUME DE TRÁFEGO DE VEÍCULOS PROPORCIONADA POR VISÃO COMPUTACIONAL VIA REDES NEURAIS

Dissertação apresentada ao Curso de Mestrado em Controle e Automação do Departamento de Engenharia Elétrica da Universidade Federal da Bahia, como requisito parcial à obtenção do título de Mestre em Engenharia Elétrica

Orientador: Prof. Antônio Cézar de Castro Lima
Universidade Federal da Bahia

Salvador

Departamento de Engenharia Elétrica da UFBA

Janeiro 1999

Agradecimentos

Ao Prof. Dr. Caiuby Alves da Costa, que desde o princípio creditou sua confiança irrestrita no meu trabalho.

Ao orientador desta dissertação, Prof. Dr. Antônio Cézar de Castro Lima, pelo seu apoio e acompanhamento paciente.

Ao Prof. André Renê Barboni, por algumas críticas e sugestões.

Ao grande amigo André Torres de Brito Daier, pelo apoio e sugestões.

À Fundação Politécnica, pela concessão da bolsa de estudos.

Aos meus pais por minha vida e educação.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

Neste trabalho são associadas as ferramentas de Redes Neurais e de Tratamento de Imagens, na forma de técnicas de detecção de bordas, com o objetivo de identificar corretamente o número de carros em imagens obtidas a partir de um mesmo ponto fixo. As redes neurais artificiais, por serem inspiradas no neurônio biológico, possuem características muito úteis no reconhecimento de padrões. A detecção de bordas reduz a imagem original a uma imagem que apresenta apenas as bordas dos objetos contidos nela. Visto que experimentos realizados com seres humanos e outros animais mostraram que bordas são as mais importantes pistas para interpretar imagens, tal associação de ferramentas busca facilitar ou possibilitar a detecção do número de carros em imagens, o que possibilitaria, por exemplo, a determinação do tempo de atuação de semáforos em resultado da comparação do fluxo de suas diversas vertentes de tráfego. Como resultado, demonstra-se que tal associação é bem sucedida e que, apesar de se ter obtido um erro absoluto médio de 0,8 carros no derradeiro experimento, podem ser obtidos resultados melhores à medida que se aumenta o número de imagens utilizadas no treinamento da rede neural.

Abstract

This work associates Neural Networks with the Image Processing technique of edge detection, having the objective of correctly identifying the number of cars in images obtained from a fixed place. Inspired in the biological neuron, the artificial neural network is a very useful tool for on pattern recognition. The edge detection reduces the original image to a image containing only the edges of the objects. Since experiences made with humans and others animals proved that edges are of fundamental importance for interpreting images, such association is proposed to facilitate or to make possible the detection of the number of cars in images so that the traffic light commutation time can be determined according to the tracks flow. The simulation results show that this association is successful and, despite obtaining a 0.8 absolute average error on the cars detection in the last experiment, this work demonstrates that better results can be obtained by increasing the number of images used in the neural network training.

Capítulo 1

Introdução

1.1 O Problema

O fluxo de carros nas cidades de médio e grande porte cada vez mais se torna um problema crítico. Diversos são os fatores que colaboram para isto, entre eles, o crescente número de carros, a falta de planejamento e manutenção das vias de tráfego e semáforos sem **adaptabilidade** às variáveis necessidades do dia. Este último fator mencionado será alvo deste estudo.

No semáforo pode ser selecionado o tempo de atuação, cedendo uma fração do tempo de livre tráfego para cada uma das direções de fluxo que se cruzam. Porém, a escolha deste tempo de atuação não é trivial, pois as necessidades de fluxo não são sempre as mesmas. Mesmo se for determinado este tempo de atuação a partir da necessidade média, ou por qualquer outro método, um único tempo de atuação fica claramente bem aquém da solução ideal.

Um segundo passo para uma solução ideal seria a determinação experimental de tempos de atuação diferentes para a necessidade média de diferentes faixas de horários. Porém, além do fato já comentado de que a necessidade média de cada intervalo não é uma boa solução por causa da variação de fluxo durante o intervalo, a própria necessidade média de determinada faixa de horários muda no

decorrer dos dias, a exemplo da diferença clara entre o fluxo no decorrer de um domingo e de uma segunda-feira.

Para aprimorar a atuação do semáforo, poderiam ser determinados tempos de atuação para intervalos de horários cada vez menores, gerando uma tabela com variação praticamente contínua de tempos de atuação. Quanto à variação da necessidade no decorrer dos dias, poderiam ser determinadas tabelas cíclicas, com variações semanais, mensais e anuais para melhorar a **adaptabilidade** do semáforo. Esta solução parece ser suficientemente boa, embora ainda não ideal. Porém, as dificuldades na obtenção de tais tabelas, específicas de cada semáforo existente nos cruzamentos de uma cidade, tornam esta solução inviável.

Apesar de inviável, a evolução que se realizou para chegar a esta solução seria um mecanismo automatizado que pudesse equivaler a um observador humano que manualmente determinasse o tempo de atuação conforme observasse a necessidade do fluxo mudar. Tal mecanismo teria a necessária **adaptabilidade**.

Buscando tal solução, propõe-se o uso da ferramenta “Redes Neurais” pela sua característica central: capacidade de adaptação na classificação de padrões e mesmo no estabelecimento de tais padrões.

1.2 A Ferramenta

A ferramenta “Redes Neurais” tem sua origem inspirada no cérebro humano. Ela é baseada em um modelo matemático do neurônio. Em termos simples, um neurônio tem vários receptores, ou canais de entrada de dados, e um emissor, ou canal de saída de dados. Cada um dos receptores tem uma capacidade de percepção característica, ou seja, um ganho. O somatório dos produtos das entradas pelos ganhos é submetido a um

processo no núcleo do neurônio (função de transferência) e enviado como resultado pelo emissor. Diferentes funções de transferência podem atender diferentes necessidades, conforme a função do neurônio.

Através da distribuição de quantidades determinadas de tais neurônios em um certo número de camadas, a exemplo do órgão humano que inspira tal ferramenta, obtemos uma ferramenta matemática capaz de em certa medida aprender e se adaptar.

1.3 Pré-processamento

Com freqüência, na busca de extrair informações de interesse de uma série de dados iniciais, a rotina básica de processamento se acha sobrecarregada por excesso de informação não relevante (que pode ser chamada de ruído) imiscuída em tais dados. Nestes casos, um pré-processamento que valorize a informação relevante, eliminando ou reduzindo o ruído, resultaria em economia de tempo na busca dos resultados desejados e algumas vezes possibilitando resultados que antes não podiam ser obtidos devido ao excesso de ruído.

No presente caso, as imagens obtidas através da câmera possuem muita informação caracterizada como ruído. É fácil discernir que tal imagem tem muitos elementos dispensáveis, a exemplo do cenário por detrás dos carros. Além disso, fatores tais como a variação da luminosidade durante o dia e a chuva podem dificultar o reconhecimento de padrões, ou seja, dos carros. Por tanto, é preciso trabalhar a imagem, padronizá-la e/ou filtrá-la para obter um melhor desempenho ou até mesmo possibilitar o sucesso do sistema.

1.4 Proposta de trabalho

Este trabalho visa modelar a solução ideal já mencionada para o semáforo: um observador humano que manualmente determinasse o tempo de atuação do semáforo conforme observasse a necessidade do fluxo mudar.

Um sensor que envia seus dados a um computador tem de limitar seu sinal a uma faixa de variação para que este, por sua vez, seja codificado em grupos de bits, à semelhança do olho humano que transforma as imagens em sinais padronizados que podem ser compreendidos pelo cérebro. O sensor do sistema em questão será uma câmera filmadora que enviará a imagem que seria vista por um observador humano que estivesse na posição do semáforo.

Para modelar o cérebro humano, será usada uma *Rede Neural*. Através da câmera, imagens do tráfego serão obtidas e processadas por tal rede com o objetivo de ter sucesso na detecção automática do número de carros contidos em cada imagem, possibilitando um futuro controle do tempo de atuação de um conjunto de semáforos.

Entretanto, a eficiência deste sistema depende muito dos dados de entrada. Como já foi delineado, um tratamento prévio dos dados é muitas vezes essencial. Dentro das ferramentas existentes de *Tratamento de Imagens*, rotinas de detecção de bordas serão propostas como valorizadoras da informação pertinente à detecção do número de carros, facilitando o trabalho da *Rede Neural*.

Neste interesse, este trabalho tem por objetivos determinar:

a) o número de carros em imagens obtidas a partir de um ponto fixo de filmagem.
b) a validade ou não de um pré-processamento destas imagens na obtenção correta do número de carros.

Capítulo 2

Redes Neurais

2.1 Introdução

Redes Neurais Artificiais, normalmente mencionadas como Redes Neurais, são técnicas computacionais que se inspiraram no cérebro humano, apresentando um modelo matemático baseado na estrutura neural deste. O poder desta ferramenta matemática está no reconhecimento de que o cérebro processa a informação de uma maneira completamente diferente de como se dá com o computador digital convencional, ou seja, na aquisição de conhecimento através da experiência.

Um grande passo para o atual entendimento do funcionamento do cérebro e sua subseqüente modelagem se deu com o trabalho pioneiro de Ramón y Cajál em 1911 (Haykin, 1994), onde foi introduzido o conceito de *neurônios* como elementos básicos da estrutura cerebral. O cérebro é constituído principalmente por um grande número de neurônios, na ordem de 10 bilhões, com um número ainda maior de interconexões ou sinapses, a saber, cerca de 60 trilhões. Cada neurônio é uma célula especializada que propaga um sinal eletroquímico, sendo composto de terminais de entrada, os dendritos, de um corpo central e de terminais de saída, os axônios. As mencionadas sinapses consistem em conexões não físicas entre axônios de uma célula com dendritos de uma outra, não havendo realmente um contato entre os neurônios. Quando um neurônio é ativado, ele emite um sinal eletroquímico através de seus axônios; sinal este que

atravessa as sinapses existentes com os dendritos de outros neurônios, ativando-os ou não. Esta ativação, que resultará novamente em uma emissão de sinal, ocorre se o sinal recebido pelo corpo central da célula exceder um determinado valor.

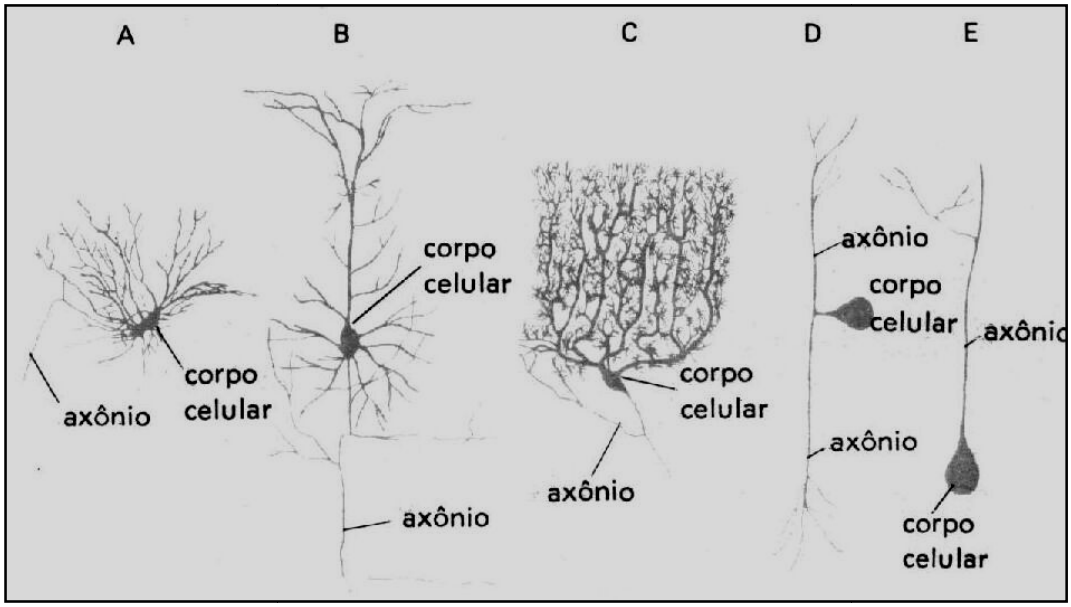


Figura 2.1 – Exemplos de várias formas de neurônios.

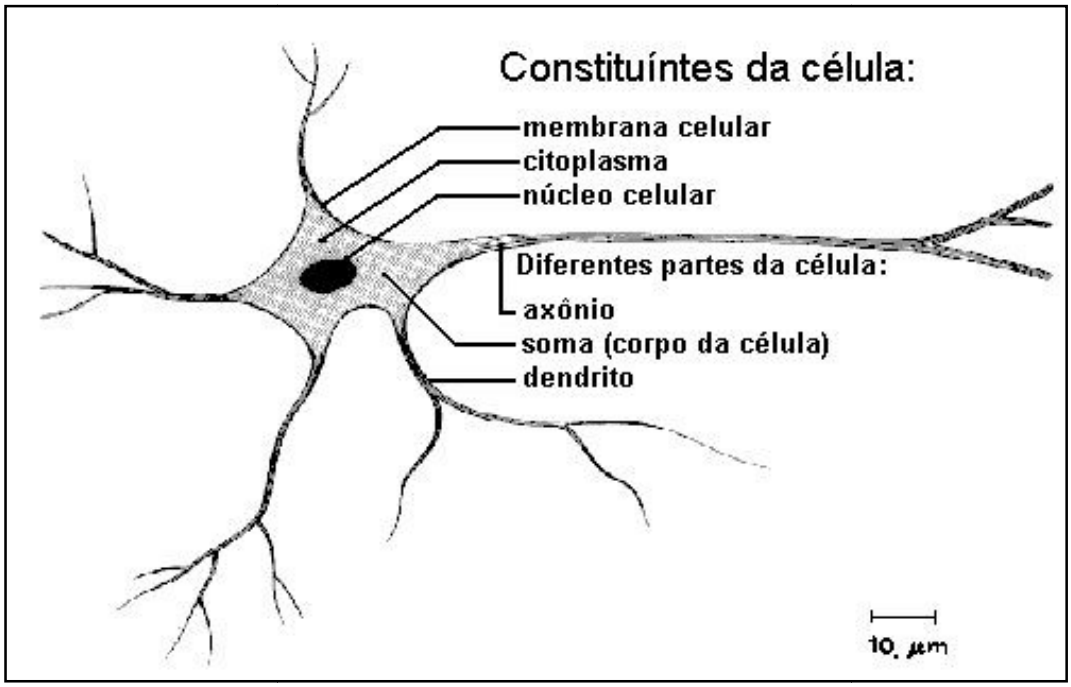


Figura 2.2 – Constituintes da célula neuronal – esquema.

O nível do sinal recebido e sua chance de reemissão dependem da eficácia das sinapses, que por sua vez dependem das substâncias químicas neurotransmissoras que preenchem a pequena distância ou *gap* que existe em uma sinapse. Esta característica define um ganho, ou seja, um fator de multiplicação que afetará o sinal antes dele ser recebido pelo corpo central da célula. Um dos mais influentes pesquisadores em sistemas neurológicos, Donald Hebb (Hebb, 1949), postulou que o aprendizado consistia principalmente de alterações dos ganhos das conexões sinápticas. Por exemplo, no clássico experimento condicionador de Pavlov, onde um sino era tocado pouco antes de se servir a refeição a um cachorro, o cachorro rapidamente aprendeu a associar o toque do sino à proximidade da refeição. As conexões sinápticas entre a parte apropriada do cérebro e as glândulas salivares foram fortalecidas, levando o cachorro a salivar toda vez que ouvia um sino tocar.

Outro aspecto interessante surge da comparação entre o cérebro e o computador digital convencional. Tipicamente, neurônios são cinco ou seis ordens de grandeza mais lentos do que portas lógicas de silício; eventos em um *chip* de silício ocorrem na ordem de nanosegundos (10^{-9} s), enquanto que em um neurônio tais eventos ocorrem na ordem de milisegundos (10^{-3} s). Entretanto, o cérebro trabalha melhor apesar da relativa lentidão de processamento por ter um enorme número de células numa massiva rede de conexões (Shepherd and Koch, 1990). O resultado disto é que o cérebro é uma estrutura extremamente eficiente. Além disso, no que diz respeito à eficiência energética, o cérebro consome aproximadamente 10^{-16} joules (J) por operação por segundo, enquanto o valor correspondente para os melhores computadores em uso hoje é em torno de 10^{-6} joules por operação por segundo (Faggin, 1991).

Mais um aspecto positivo do cérebro advém de sua estrutura altamente complexa, não linear e paralela de realizar o processamento das informações. Ele tem a capacidade de organizar neurônios de forma a realizar processamentos específicos, tais como: reconhecimento de padrões, percepção e controle motor muitas vezes mais rápido que o mais rápido dos supercomputadores existentes hoje. Considere, por exemplo, a visão humana, que é uma tarefa de processamento de informações (Churchland and Sejnowski, 1992; Levine, 1985; Marr, 1982). Esta pode ser definida como a função de prover uma *representação* do ambiente ao nosso redor e, mais importante ainda, de suprir-nos a informação de que precisamos para *interagir* com tal ambiente. Em termos comparativos, o cérebro rotineiramente realiza tarefas de reconhecimento de padrão, de forma contínua (a exemplo do reconhecimento de rostos familiares dentro de um ambiente completamente diferente do esperado ou não familiar), em um tempo da ordem de 100-200 ms, enquanto tarefas muito menos complexas levariam dias em um poderoso computador convencional (Churchland, 1986).

Um exemplo adicional é o do *sonar* do morcego. O sonar é um sistema ativo de localização de objetos através da recepção de ecos de um sinal emitido. Além de prover informação sobre quão longe um alvo (um inseto voador) está, o sonar do morcego também gera informação sobre a relativa velocidade do alvo, seu tamanho, bem como seu azimute e elevação (Suga, 1990a, b, c). Todo este complexo processamento necessário para extrair todas essas informações de um eco refletido por um alvo ocorre em um cérebro do tamanho de uma ameixa. De fato, através de tal sistema de localização, um morcego é capaz de perseguir e capturar seu alvo com uma facilidade e taxa de sucesso tão grandes que deixaria com inveja qualquer sistema de radar ou de sonar produzido e utilizado hoje pelo homem.

Diante disso tudo, a questão realmente é: como o cérebro humano ou de qualquer outro animal é capaz de realizar tais tarefas? Ao nascer, o cérebro já possui uma grande estrutura e a habilidade de construir suas próprias regras através do que usualmente chamamos de *experiência* (Haykin, 1994). Especialmente nos primeiros dois anos de vida, o cérebro humano realiza um vasto trabalho de *aprendizagem*, realizando em média cerca de 1 milhão de sinapses por segundo. Mesmo após este período, o cérebro humano continua a realizar novas sinapses, além de alterar as antigas. Esta característica pode ser defina como *Plasticidade*, sendo de vital importância para a adaptabilidade do ser humano ao ambiente que o cerca (Churchland and Sejnowski, 1992; Eggermount, 1990), bem como para as redes neurais artificiais ou simplesmente redes neurais.

Como definição simplificada (Haykin, 1994), uma rede neural é um processador de informações, distribuída de forma massiva e paralela, que tem a habilidade natural de armazenar conhecimento e torná-lo disponível ao uso. Tal ferramenta matemática retém duas características do cérebro: o conhecimento é adquirido através de um processo de aprendizado; e a força das interconexões neurais, conhecidas como pesos sinápticos ou *ganhos,* são utilizadas para armazenar conhecimento.

Redes neurais são também referidas na literatura como neurocomputadores, redes coneccionistas, processadores paralelamente distribuídos, etc. Nesta dissertação será feita uso simplesmente do termo “redes neurais”.

2.2 Características do modelo matemático

Cada unidade de processamento ou neurônio de uma rede neural possui canais de comunicação com outros neurônios, aos quais estão associados determinados pesos. Apesar do comportamento inteligente de uma rede neural vir das interações entre as unidades de processamento da rede, tais unidades fazem operações apenas sobre as informações recebidas localmente por seus canais receptores. A operação de uma unidade de processamento, proposta por McCullock e Pitts em 1943, foi de que os sinais são apresentados às suas respectivas entradas, de forma que cada sinal é multiplicado por um número, ou peso, que indica a sua influência na saída da unidade. Em seguida, é feita a soma ponderada dos sinais que produz um nível de atividade que, se exceder um certo limite (threshold), a unidade produz uma determinada resposta de saída.

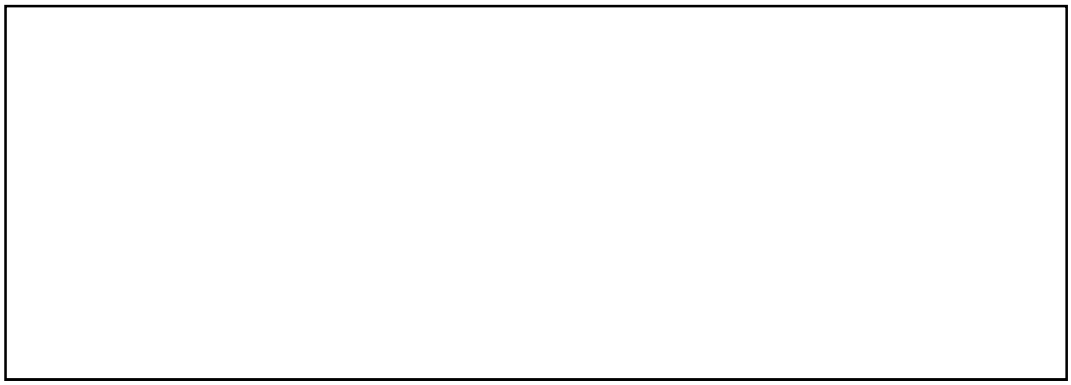


Figura 2.3 - Esquema de unidade McCullock - Pitts.

Conforme pode ser visto na Figura 2.3, se tivermos p sinais de entrada X_1, X_2, ..., X_p, pesos W_1, W_2, ..., W_p e limitador T (threshold), com sinais assumindo valores booleanos (0 ou 1) e pesos valores reais, teremos o nível de atividade A e a saída y dados pelas equações a seguir:

$$A = W_1.X_1 + W_2.X_2 + ... + W_p.X_p \quad (2.1)$$

y = 1, se A >= T ou

y = 0, se A < T

A maioria dos modelos de redes neurais possui alguma regra de treinamento, onde os pesos de suas conexões são ajustados de acordo com os padrões apresentados. Em outras palavras, elas aprendem através de exemplos. Além disso, arquiteturas neurais são tipicamente organizadas em camadas, com unidades que podem estar conectadas às unidades da camada posterior, conforme ilustrado na Figura 2.4.

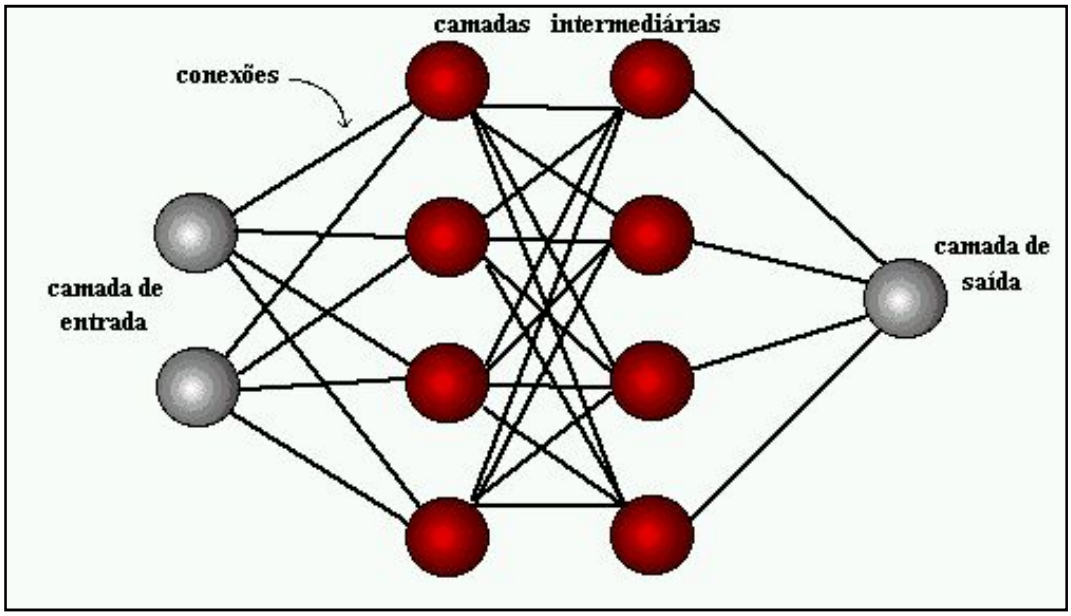


Figura 2.4 – Organização das camadas.

Usualmente as camadas são classificadas em três grupos:

- Camada de Entrada: onde os padrões são apresentados à rede;
- Camadas Intermediárias ou Escondidas: onde é feita a maior parte do processamento, através das conexões ponderadas; podem ser consideradas como extratoras de características;

- Camada de Saída: onde o resultado final é concluído e apresentado.

Além de diferirem pela camada onde estão, os neurônios podem ter funções de ativação diferentes. No exemplo da Figura 2.3, foi mencionado que a soma ponderada dos sinais produz um nível de atividade que, se exceder um certo limite (threshold), gera uma determinada resposta de saída. Este é um caso particular de função de ativação chamado de função Threshold. Podemos ver, na Figura 2.5, alguns outros exemplos de funções de ativação.

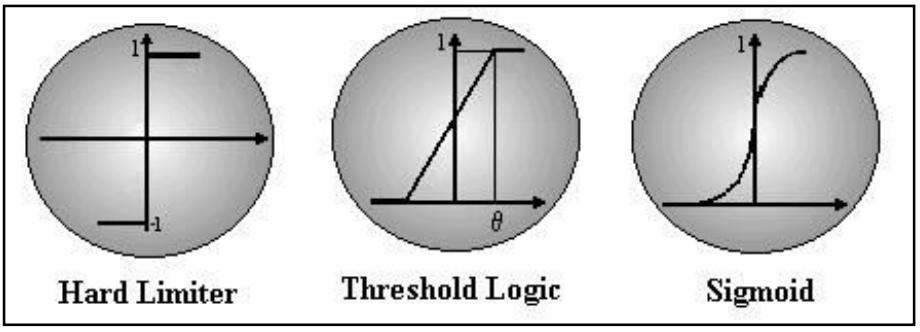


Figura 2.5 – Exemplos de funções de ativação.

É importante mencionar que cada camada tem os seus neurônios com a mesma função de ativação, ficando a escolha desta a depender da necessidade da aplicação em questão. Alguns tipos de neurônios e suas funções de ativação serão discutidos a seguir.

2.2.1 Perceptron

Uma rede neural composta de *perceptrons* é a mais simples forma de redes usadas para a classificação de padrões linearmente separáveis. Criado por Rosenblatt (1961), o *perceptron* consiste basicamente de um neurônio com ganhos sinápticos ajustáveis e um threshold, à semelhança da Figura 2.3. Foi provado por Rosenblatt que, se os padrões a

serem reconhecidos puderem ser separados linearmente em classes, então a rede composta de perceptrons terá sucesso em realizar esta classificação. Um único neurônio destes é capaz de classificar dois padrões diferentes que atendam a esse requisito, sendo necessários mais neurônios na camada para resolver problemas de classificação com mais de dois padrões.

Uma possibilidade de vencer a dificuldade de classificação de padrões não separáveis linearmente é a utilização de um método de pré-processamento que resultaria em padrões separáveis linearmente.

2.2.2 Linear

Este neurônio, fruto de trabalhos pioneiros realizados durante os anos sessenta, difere do anterior por ter uma função linear por função de ativação, podendo realizar aproximações de funções lineares ou associação de padrões. Apesar de estar limitado a resolver problemas lineares ou linearizáveis, uma rede linear pode ser facilmente treinada pelo método de Widrow-Hoff (Widrow and Hoff, 1960) também conhecido por Método dos Mínimos Quadrados ou *Least-Mean-Square Algorithm* (LMS). Este tipo de rede tem diversas aplicações em controle e sistemas de comunicações, mas limita-se a problemas lineares ou a aproximações lineares aceitáveis de problemas não-lineares.

2.2.3 Sigmoid

Vencendo as limitações do neurônio linear, o neurônio sigmoid possui uma função de ativação não-linear (Figura 2.5c), que pode assumir diferentes funções matemáticas, todas caracterizadas por suavidade, a exemplo da Log-Sigmoid e da Tan-Sigmoid, definidas a seguir:

a) Log-Sigmoid: $\varphi(v) = 1 / (1 + \exp(-a.v))$ (2.2)

b) Tan-Sigmoid: $\varphi(v) = (1 - \exp(-v)) / (1 + \exp(-v))$ (2.3)

O parâmetro “a” na primeira equação define a inclinação da curva sigmoid, sendo que valores deste parâmetro, tendendo ao infinito, levam esta função a se aproximar da função threshold.

Como mencionado anteriormente, uma grande vantagem deste tipo de rede vem do fato dela poder resolver problemas não-lineares e de ser diferenciável.

2.3 Desenvolvimento de aplicações

Na busca de solucionar um problema, os dois primeiros passos do processo de desenvolvimento de redes neurais artificiais são: a coleta de dados relativos ao problema e a sua separação em um conjunto de treinamento, e um conjunto de testes. Esta tarefa requer uma análise cuidadosa sobre o problema para minimizar ambigüidades e erros nos dados. Além disso, os dados coletados devem ser significativos e cobrir amplamente o domínio do problema; não devem cobrir apenas as operações normais ou rotineiras, mas também as exceções e as condições nos limites do domínio do problema.

Normalmente, os dados coletados são separados em duas categorias: dados de treinamento, que serão utilizados para o treinamento da rede e dados de teste, que serão utilizados para verificar sua performance sob condições reais de utilização. Além dessa divisão, pode-se usar também uma subdivisão do conjunto de treinamento, criando um conjunto de validação, utilizado para verificar a eficiência da rede quanto a sua capacidade de generalização durante o treinamento, podendo ser empregado como critério de parada do treinamento.

Depois de determinados estes conjuntos, eles são, geralmente, colocados em ordem aleatória para prevenção de tendências associadas à ordem de apresentação dos dados. Além disso, pode ser necessário pré-processar estes dados, através de normalizações, escalonamentos e conversões de formato para torná-los mais apropriados à sua utilização na rede.

O terceiro passo é a definição da configuração da rede, que pode ser dividida em três etapas:

1) Seleção da função de ativação (de cada camada) apropriada à aplicação;

2) Determinação da topologia da rede a ser utilizada - o número de camadas, o número de unidades em cada camada, etc.;

3) Determinação de parâmetros do algoritmo de treinamento e das funções de ativação. Este passo tem um grande impacto na performance do sistema resultante.

Na condução destas tarefas, não existem métodos exatos, mas regras gerais que levam normalmente estas escolhas a serem feitas de forma empírica. A definição da configuração de redes neurais é ainda considerada uma arte, que requer grande experiência dos projetistas.

O quarto passo é a realização do treinamento da rede. Nesta fase, seguindo o algoritmo de treinamento escolhido, serão ajustados os pesos das conexões. Nesta etapa, é importante considerar alguns aspectos tais como: a inicialização da rede, o modo de treinamento e o tempo de treinamento. Uma boa escolha dos valores iniciais dos pesos da rede pode diminuir o tempo necessário para o treinamento. Normalmente, os valores iniciais dos pesos da rede são números aleatórios uniformemente distribuídos, em um intervalo definido. Porém, a escolha errada destes pesos pode levar a uma saturação

prematura. Os pesquisadores Nguyen e Widrow (1989) encontraram uma função que pode ser utilizada para determinar valores iniciais melhores que valores puramente aleatórios.

Quanto ao modo de treinamento, na prática, é mais utilizado o modo padrão devido ao menor armazenamento de dados, além de ser menos suscetível ao problema de mínimos locais, devido à pesquisa de natureza estocástica que realiza. Por outro lado, no modo *batch*, que leva em conta a memória do que foi processado, tem-se uma melhor estimativa do vetor gradiente, tornando o treinamento mais estável. A eficiência relativa dos dois modos de treinamento depende do problema que está sendo tratado.

Quanto ao tempo de treinamento, vários fatores podem influenciar a sua duração, porém sempre será necessário utilizar algum critério de parada. O critério de parada do algoritmo *backpropagation* não é bem definido e, geralmente, é utilizado um número máximo de ciclos. Mas devem ser consideradas a taxa de erro médio por ciclo e a capacidade de generalização da rede. Além disso, pode ocorrer que, em um determinado instante do treinamento, a generalização comece a degenerar, causando o problema de *over-training*, ou seja, a rede se especializa no conjunto de dados do treinamento e perde a capacidade de generalização.

Por fim, o treinamento deve ser interrompido quando a rede apresentar uma boa capacidade de generalização e quando a taxa de erro for suficientemente pequena, ou seja, menor que um erro admissível. Assim, deve-se encontrar um ponto ótimo de parada com erro mínimo e capacidade de generalização máxima.

O quinto passo é o teste da rede. Durante esta fase, o conjunto de teste é utilizado para determinar a performance da rede com dados que não foram previamente utilizados. A performance da rede, medida nesta fase, é uma boa indicação de sua

performance real. Devem ser considerados ainda outros testes, como análise do comportamento da rede, utilizando entradas especiais, e análise dos pesos atuais da rede, pois, se existirem valores muito pequenos, as conexões associadas podem ser consideradas insignificantes e assim serem eliminadas (prunning). De modo inverso, valores substantivamente maiores que os outros poderiam indicar que houve *over-training* da rede.

Finalmente, com a rede treinada e avaliada, ela pode ser integrada em um sistema do ambiente operacional da aplicação. Para maior eficiência da solução, este sistema deverá conter facilidades de utilização como interface conveniente e facilidades de aquisição de dados através de planilhas eletrônicas, interfaces com unidades de processamento de sinais, ou arquivos padronizados. Além disso, uma boa documentação do sistema e o treinamento de usuários são necessários para o sucesso do mesmo.

Durante a fase de utilização prática, o sistema deve periodicamente monitorar sua performance e fazer a manutenção da rede, quando for necessário, ou indicar aos projetistas a necessidade de retreinamento. Outras melhorias poderão ainda ser sugeridas quando os usuários forem se tornando mais familiarizados com o sistema. Estas sugestões poderão ser muito úteis em novas versões ou em novos produtos.

Para facilitar o entendimento deste tópico, podem ser vistos, a seguir, exemplos de métodos de treinamento.

2.3.1 Regra Delta

Quando um padrão é inicialmente apresentado à rede, ela produz uma saída. Após medir a distância entre a resposta atual e a desejada, são realizados os ajustes apropriados nos

pesos das conexões de modo a reduzir esta distância. Este procedimento é conhecido como Regra Delta.

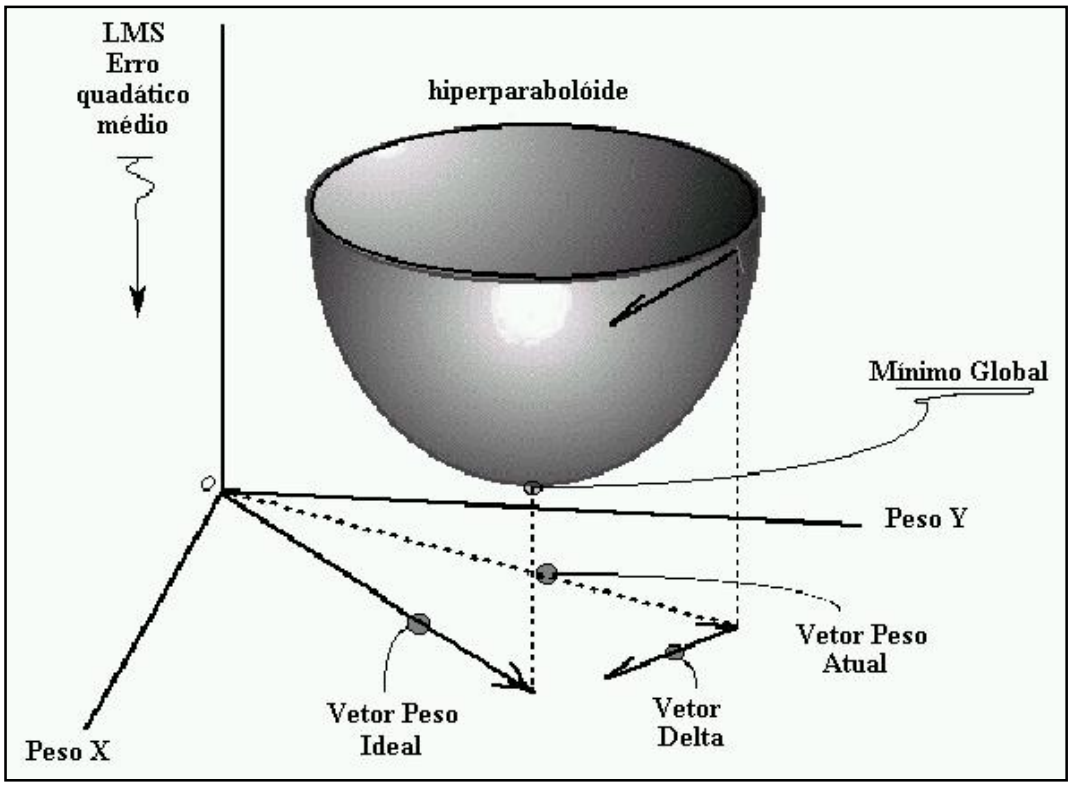


Figura 2.6 – Regra Delta.

A princípio, todas as conexões são iniciadas com pesos aleatórios e são definidos os pares de treinamento (X,d), que são formados pelo padrão de entrada e a sua respectiva resposta desejada. Calcula-se, então, a resposta obtida "O" e o erro "E" que consiste na diferença entre o desejado e o obtido (d-O). A partir deste erro, atualizam-se os pesos, conforme pode ser visto na Figura 2.7. Nesta atualização, é utilizada uma constante positiva (η) que pode ser descrita como taxa de aprendizado que corresponde à velocidade do mesmo. Sendo assim, este procedimento de correção dos pesos repetir-se-á até que o erro obtido seja satisfatório (E = e; onde "e" é o erro desejado).

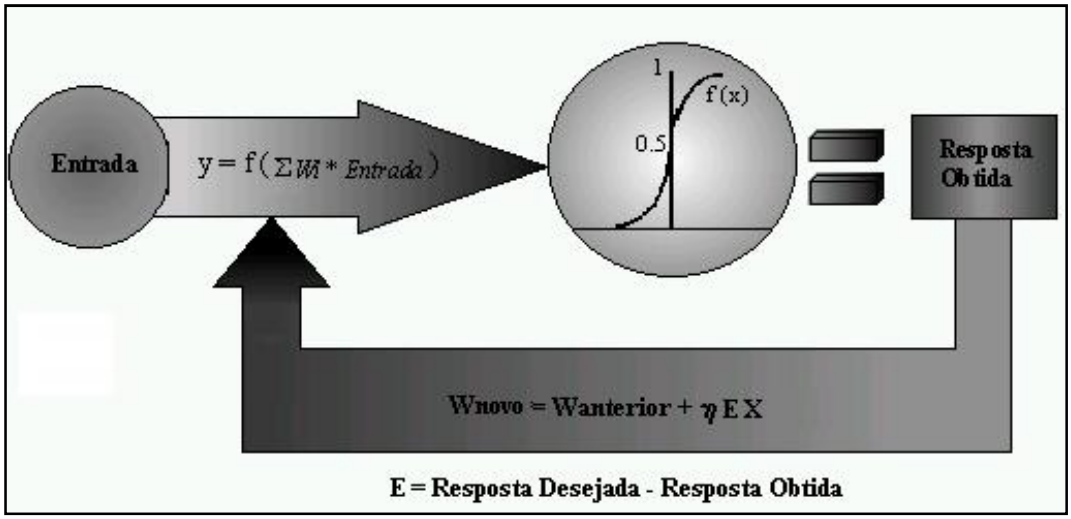


Figura 2.7 – Esquema de treinamento.

2.3.2 BackPropagation

Durante o treinamento com o algoritmo *backpropagation*, a rede opera em uma seqüência de dois passos. Primeiro, um padrão é apresentado à camada de entrada da rede, e a atividade resultante flui através da rede, camada por camada até que a resposta seja 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 e propagado a partir da camada de saída até a camada de entrada, sendo os pesos das conexões das unidades das camadas internas modificados, conforme o erro é retropropagado.

As redes que utilizam *backpropagation* trabalham com uma variação da Regra Delta apropriada para redes multi-camadas: a Regra Delta generalizada. A Regra Delta padrão implementa, essencialmente, um gradiente descendente no quadrado da soma do erro para funções de ativação lineares.

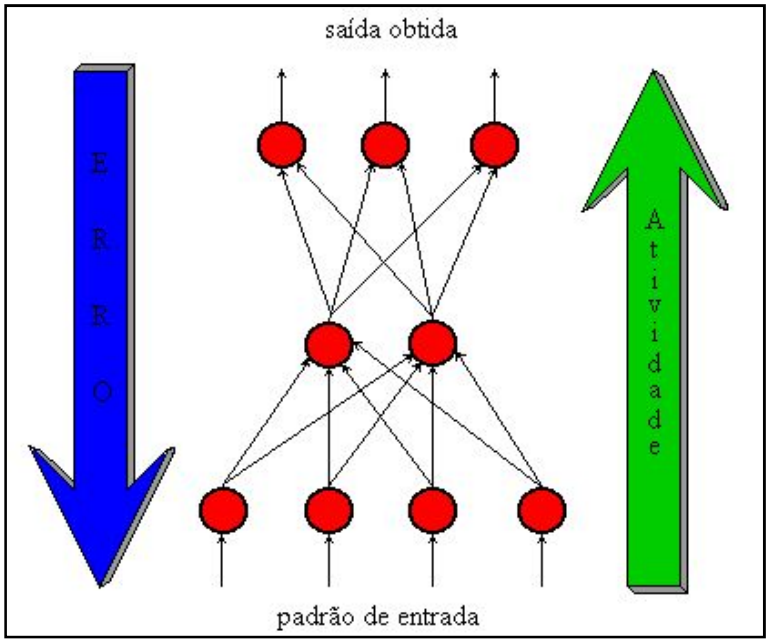


Figura 2.8 – Treinamento BackPorpagation.

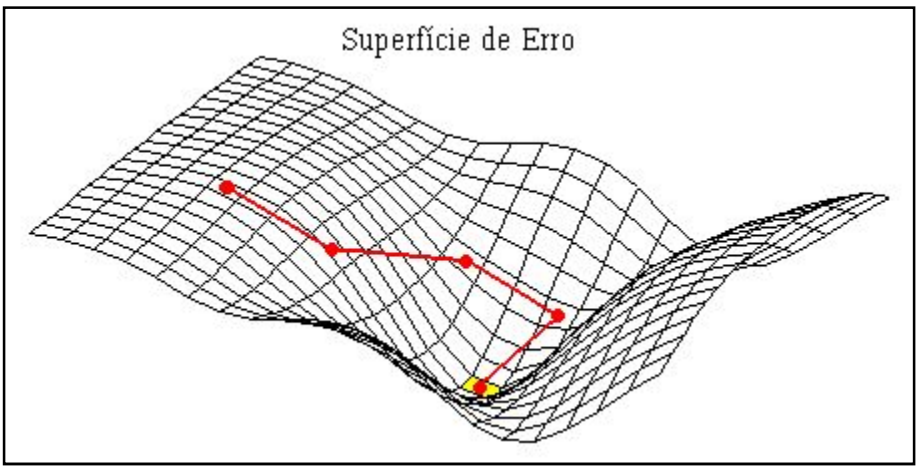
Redes sem camadas intermediárias podem resolver problemas onde a superfície de erro tem a forma de um parabolóide com apenas um mínimo. Entretanto, a superfície do erro pode não ser tão simples, como ilustrada na Figura 2.9, bem como suas derivadas mais difíceis de serem calculadas. Nestes casos, devem ser utilizadas redes com camadas intermediárias. Ainda assim, as redes ficam sujeitas aos problemas de procedimentos "hill-climbing", ou seja, problema de mínimos locais.

Figura 2.9 – Exemplo de busca do mínimo em uma superfície de erro.

Entretanto, a Regra Delta generalizada funciona, quando são utilizadas na rede, unidades com uma função de ativação semi-linear, que é uma função diferenciável e não decrescente. Note que a função *threshold* não se enquadra nesse requisito. Uma função de ativação, amplamente utilizada nestes casos, é a função sigmoid.

No que se refere à rapidez do treinamento, foi visto anteriormente que ela depende muito da taxa de aprendizado, que é uma constante de proporcionalidade no intervalo [0,1], pois este procedimento requer apenas que a mudança no peso seja proporcional à η. Entretanto, o verdadeiro gradiente descendente requer que sejam tomados passos infinitesimais. Porém, quanto maior for essa constante, maior será a mudança nos pesos, aumentando a velocidade do aprendizado, o que pode levar a uma oscilação do modelo na superfície de erro. O ideal seria utilizar a maior taxa de aprendizado possível que não levasse a uma oscilação, resultando em um aprendizado mais rápido.

Há ainda o problema de se encontrar um mínimo local, fazendo o erro parar de diminuir e estacionar em um valor maior que o aceitável. Para que se evite esta situação e ainda por cima gerar uma maneira de aumentar a taxa de aprendizado sem



levar à oscilação, a regra delta generalizada pode ser modificada para incluir o termo *momentum*: uma constante que determina o efeito das mudanças passadas dos pesos na direção atual do movimento no espaço de pesos. Desta forma, o termo *momentum* leva em consideração o efeito de mudanças anteriores de pesos na direção do movimento atual no espaço de pesos. O termo *momentum* torna-se útil em espaços de erro que contenham longas gargantas, com curvas acentuadas ou vales com descidas suaves.

2.4 Limitações

As redes neurais que utilizam *backpropagation*, assim como muitos outros tipos de redes neurais artificiais, podem ser vistas como "caixas pretas", nas quais quase não se sabe 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.

Uma outra limitação refere-se ao tempo de treinamento de redes neurais utilizando *backpropagation*, que tende a ser muito lento. Algumas vezes são necessários milhares de ciclos para se chegar a níveis de erros aceitáveis, principalmente se estiver sendo simulado em computadores seriais, pois a CPU deve calcular as funções para cada unidade e suas conexões separadamente, o que pode ser problemático em redes muito grandes ou com grande quantidade de dados. Muitos estudos estão sendo realizados para implementação de redes neurais em computadores paralelos, além de construção de *chips* neurais como Intel 80170NX Eletronically Trainable ANN ou placas aceleradoras como BrainMaker Accelerator Board CNAPS.

Um terceiro problema advém do fato de que é muito difícil definir a arquitetura ideal da rede, de forma que ela seja tão grande quanto for preciso para conseguir obter as representações necessárias, ao mesmo tempo pequena o suficiente para se ter um treinamento mais rápido. Além disso, não existem regras claras para se definir quantas unidades devem existir nas camadas intermediárias, quantas camadas, ou como devem ser as conexões entre essas unidades.

Capítulo 3

Tratamento de Imagens

3.1 Introdução

Quando se trabalha com reconhecimento de padrões, os objetos a serem classificados vêm com freqüência de imagens que precisam ser digitalizadas para serem trabalhadas por algoritmos diversos (Gose, 1996). Uma imagem digital é o resultado de uma amostragem realizada a intervalos regulares a partir da imagem original. Assim, uma *imagem digital* é simplesmente uma matriz de duas dimensões onde cada elemento numérico, conhecido como pixel, representa o nível de brilho ou nível de cinza que aquela região da imagem possui. Portanto, o intervalo de amostragem define quão bem aquela amostragem representa a imagem original, resultando no conceito chamado resolução geométrica, que é inversamente proporcional ao intervalo de amostragem.

Para se obter imagens digitais, Digitalizadores ou *Scanners* podem ser utilizados para fazer a conversão de fotografias para imagens digitais com a resolução desejada. Outra forma de se obter tais imagens é através da utilização de placas de captura de vídeo que interligam uma câmera de vídeo a um computador. Neste caso, o brilho dos pixels, antes representado por uma tensão variável no tempo, é convertido para sua representação digital por um conversor analógico-digital.

Independente da forma pela qual se obtenha as imagens digitais, o procedimento de reconhecimento de padrões a partir delas pode ser árduo por demandar

tempo e recursos matemáticos devido ao tamanho das matrizes de pixels e da quantidade de imagens com as quais se trabalha. Por isso, mecanismos de tratamento de imagem podem levar a uma valorização da informação que se deseja extrair, poupando tempo precioso da rotina matemática de reconhecimento de padrões propriamente dita.

Existem vários métodos de tratamento de imagens visando os mais diversos objetivos (Jain, 1989; Pratt, 1978). A seguir serão mencionados alguns destes para por fim dar destaque ao método utilizado neste trabalho, a saber, detecção de bordas.

3.2 Métodos Básicos de Tratamento de Imagens

3.2.1 Inversa

A transformação inversa reverte luz em escuridão. Um exemplo familiar é o negativo de uma fotografia onde o que era branco torna-se preto e vice-versa. Matematicamente falando, se a imagem original é definida por $g_1(x,y)$ e a nova imagem por $g_2(x,y)$, então a transformada inversa é definida por

$$g_2(x,y) = K - 1 - g_1(x,y) \tag{3.1}$$

onde K define o número de níveis de cinza (256 por exemplo)

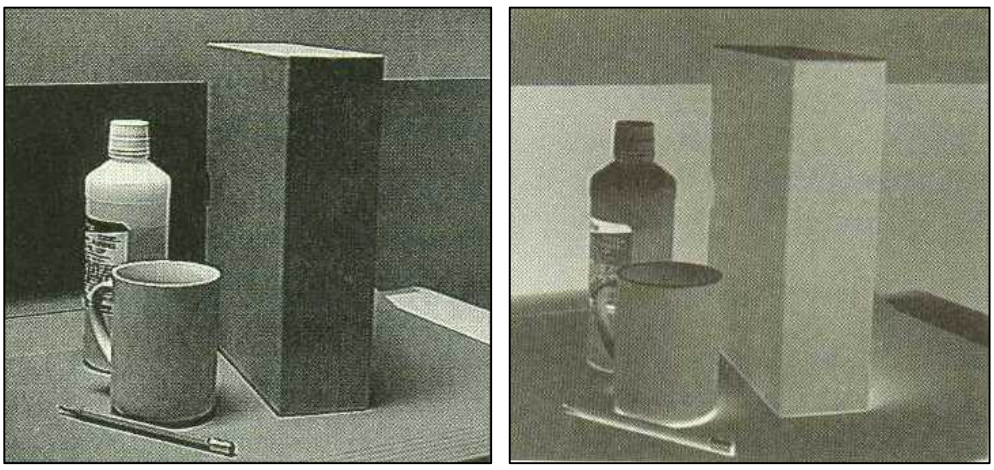


Figura 3.1 – Imagem original e sua transformada inversa.

3.2.2 Thresholding

A transformação chamada thresholding atribui o valor zero (preto) a qualquer nível de cinza menor ou igual a **T** (chamado valor de treshold). Além disso, os níveis de cinza de valor superior a **T** recebem o valor **K – 1**, que equivale ao branco. Esta transformação é muito útil quando se pretende valorizar ou separar objetos claros de um fundo escuro ou vice-versa. Portanto, a transformação thresholding é definida por

$$g_2(x,y) = 0 \qquad \text{se } g_1(x,y) \leq T \tag{3.2}$$

$$g_2(x,y) = K - 1 \qquad \text{se } g_1(x,y) > T$$

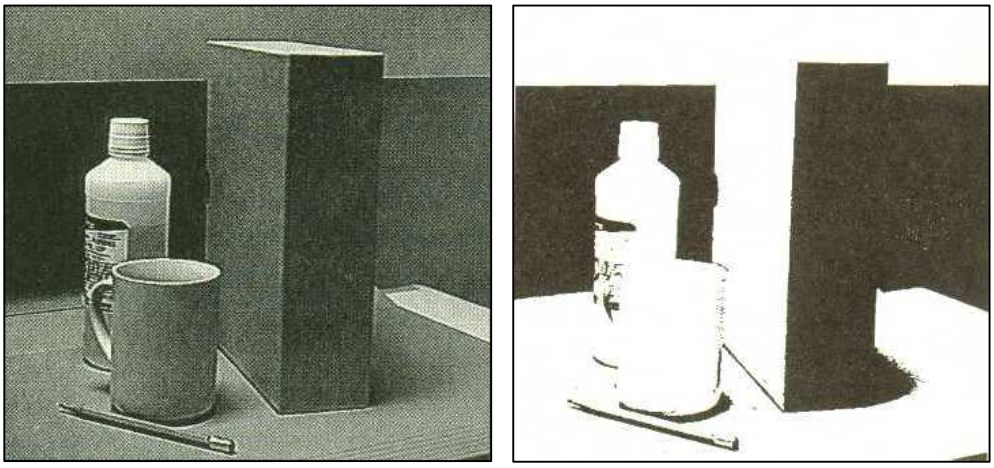


Figura 3.2 – Imagem original e sua transformada thresholding (T=128 K=256).

3.2.3 Redução de níveis de cinza

Para reduzir o número possível de níveis de cinza de uma imagem sem mudar a essência do seu brilho e contraste, divide-se os níveis por uma constante **G**, arredonda-se o resultado, e, por fim, multiplica-se tal resultado pela mesma constante **G** para restaurar a faixa original. Este procedimento resultará em uma quantização grosseira dos níveis de cinza se a constante **G** for maior 1. Portanto, esta redução é definida por

$$g_2(x,y) = \; | \, g_1(x,y).G/K \, | \; . \; K/G \tag{3.3}$$

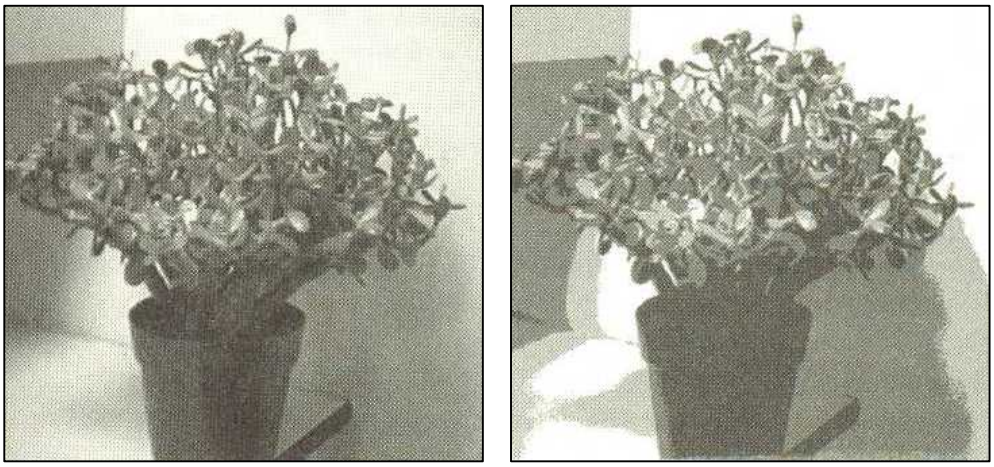


Figura 3.3 – Imagem original e após redução de níveis de cinza(K=8, G=4).

Uma razão para efetuar uma redução de níveis de cinza é obter uma compressão dos dados. Pode-se ver da Figura 3.3 que, mesmo reduzindo de 8 para 4 níveis possíveis de cinza, o vaso ainda pose ser bem reconhecido, obtendo-se uma economia de um terço, ou seja, 1 bit por pixel, pois

$8 = 2^3$, três bits por pixel

$4 = 2^2$, dois bits por pixel

3.2.4 Detecção de bordas

Experimentos realizados com seres humanos e outros animais mostraram que bordas são as mais importantes pistas para interpretar imagens (Gose, 1971; Gose, 1974). Se uma imagem consiste de objetos de interesse sobre um fundo contrastante, uma borda é a transição do fundo para o objeto e vice-versa (Krueger, 1989). A mudança ou diferença de intensidade nesta transição é chamada de *intensidade da borda*.

A taxa de mudança nos níveis de cinza no que diz respeito à distância horizontal numa imagem contínua é igual à derivada parcial de **g(x,y)** em **x**. Conforme

pode ser visto em (3.4), se Δx for substituído por 1 (um), que é o menor valor diferente de zero que Δx pode assumir em uma imagem, obtém-se (3.5).

$$\frac{\partial g(x,y)}{\partial x} = \lim_{\Delta x \to 0} \frac{g(x+\Delta x, y)-g(x,y)}{\Delta x} \tag{3.4}$$

$$g'_x(x,y) = g(x+1, y) - g(x,y) \tag{3.5}$$

onde g'_x representa a primeira derivada parcial de g em x

Da mesma forma, temos (3.6):

$$g'_y(x,y) = g(x, y+1) - g(x,y) \tag{3.6}$$

Estas derivadas parciais finitas representam a mudança no nível de cinza de um pixel para o próximo e podem ser usadas para reforçar ou detectar mudanças abruptas no nível de cinza de uma imagem. Pelo fato de bordas de objetos em uma cena produzirem tais mudanças, estes operadores são chamados de detetores de bordas.

Os detetores de bordas mono-dimencionais em (3.5) e (3.6) podem ser representados pelos operadores

-1	1

1
-1

(3.7)

Tais detetores indicam quão rápido os o nível de cinza cresce ou decresce em função de **x** e **y**. Um valor positivo de $g'_x(x,y)$ indica uma transição de escuro para claro quando movendo para a direita, enquanto um valor negativo indica uma transição de claro para escuro (Brooks, 1978; Torre, 1986). Estes detetores de bordas são chamados de **detetores de ruptura** porque são definidos para marcar as rupturas entre pixels e não para marcar pixels. Se forem desejados detetores que marquem pixels, operadores com tamanho excedente tais como

-1	0	1

1
0
-1

(3.8)

Na digitalização de cenas reais, alguns pixels na borda de objetos repousam parcialmente no objeto e parcialmente sobre o fundo da imagem, produzindo um anel de pixels ao redor do objeto com níveis de cinza entre o tom do objeto e o tom do fundo. Quando os operadores (3.8) são centralizados em uma borda, seus valores não são afetados por estes níveis intermediários por causa do peso zero. Em alguns casos, uma falta de clareza adicional pode ser introduzida pelo sistema, fazendo com que a transição da borda se dê ainda mais gradualmente. Sendo assim, operadores tais como

-1	0	0	0	0	0	0	0	1

(3.9)

podem ser usados. Estes dão resultados melhores para bordas de ampla graduação do que os (3.8), e sua resposta a ruído randômico não é incrementada, fato que ocorreria se simplesmente (3.8) fossem multiplicados por uma constante para produzir tal efeito.

Podem ser vistos a seguir detetores de borda popularmente utilizados.

Detetor de borda de Robert:

Este detetor se baseia em derivadas diagonais tais como

-1	0
0	1

0	1
-1	0

-1	0	0
0	0	0
0	0	1

0	0	1
0	0	0
-1	0	0

(3.10)

devendo ser utilizado para valorizar bordas diagonais.

Detetor de borda de Prewitt:

Este detetor combina uniforme suavidade em uma direção com detecção de borda na direção perpendicular, produzindo

-1	0	1
-1	0	1
-1	0	1

-1	-1	-1
0	0	0
1	1	1

(3.11)

que podem ser fatorados em dois simples operadores:

-1	0	1
-1	0	1
-1	0	1

=

1
1
1

x

-1	0	1

e

-1	-1	-1
0	0	0
1	1	1

=

-1
0
1

x

1	1	1

Detetor de borda de Sobel:

Este combina suavidade binomial com detecção de contornos, e também é definido por operadores que podem ser fatorados:

-1	0	1
-2	0	2
-1	0	1

=

1
2
1

x

-1	0	1

=

1
1

x

1
1

X

-1	0	1

e

-1	-2	-1
0	0	0
1	2	1

=

-1
0
1

x

1	2	1

=

-1
0
1

x

1	1

x

1	1

Assim, estes operadores podem ser representados como um número de deslocamentos, somas e subtrações da imagem inteira, que podem ser efetuados rapidamente usando equipamento adequado.

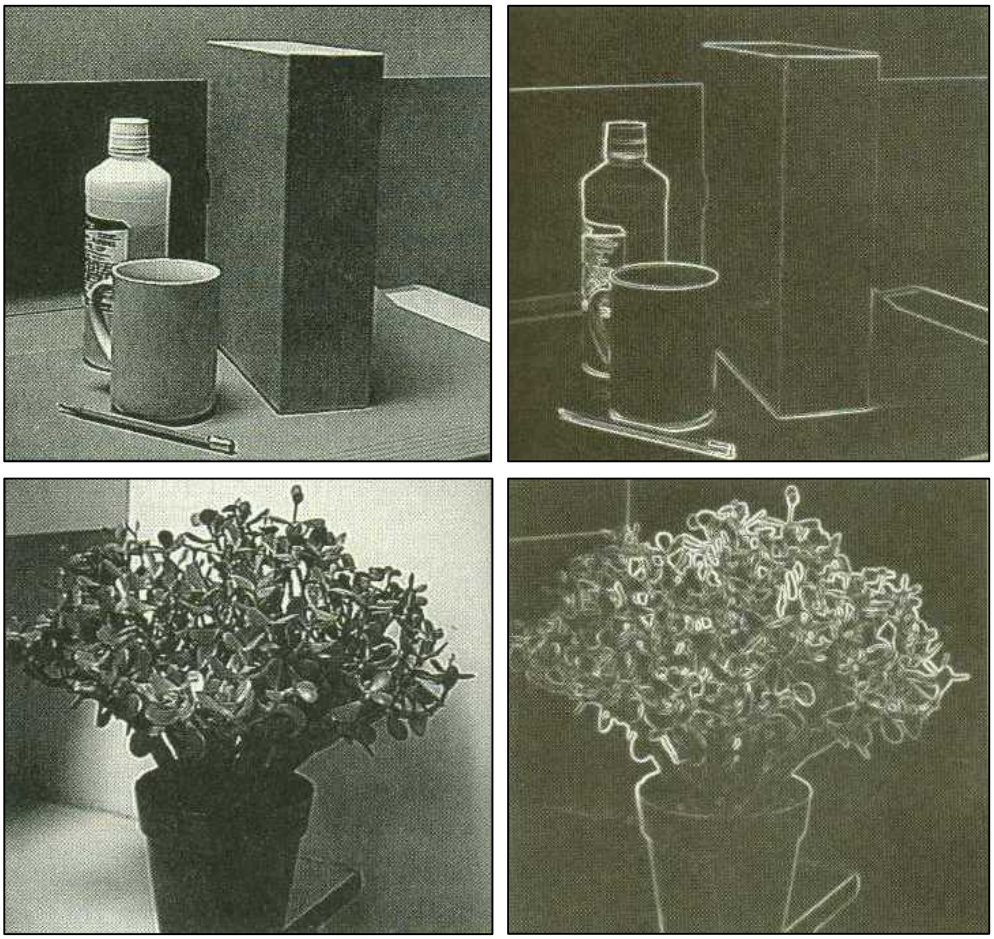


Figura 3.4 – Imagens originais e após detecção de borda por Sobel.

Na Figura 3.4 podem ser vistos exemplos da aplicação do método de Sobel na detecção das bordas de duas imagens. Neste trabalho, detecção de bordas desempenhará um papel fundamental na obtenção dos resultados desejados, em associação com redes neurais.

Capítulo 4

Material e Métodos

4.1 Equipamentos e Programas

O equipamento necessário à realização deste trabalho consistiu de uma câmera de vídeo, de um microcomputador padrão PC e de uma placa de captura de vídeo para conexão da câmera ao microcomputador. Suas especificações técnicas são:

- Câmera Sony HandyCam, Vídeo 8, Modelo CCD-TRV11 NTSC;
- Microcomputador PC K6-300-II, 256 Mb RAM, HD 6Gb;
- Placa de Captura de Vídeo LG, chipset Cirrus Logic CL-GD5446TV, 4Mb RAM.

Os programas utilizados foram:

- Sistema Operacional *Windows 95 OSR2*;
- *Adobe Premiere 4.2* com driver **indel**;
- *AVI Constructor 2.3* trial version, por Michael Caracena;
- *IrfanView 32 bit version 2.83*, por Irfan Skiljan, estudante da Universidade de Tecnologia de Viena;
- *MATLAB basic toolbox version 5.1*;
- *MATLAB Neural Network toolbox version 2.04*;
- *MATLAB Image Processing toolbox version 2.0*.

4.2 Geração das imagens digitais

O primeiro passo se deu na escolha de um local adequado à filmagem no sentido de conseguir um ponto de filmagem elevado e central em relação à pista de tráfego de automóveis, simulando as imagens que seriam obtidas a partir de uma câmera situada em um semáforo central. Por possuir as características desejadas, o local escolhido foi um trecho da avenida Garibaldi, na cidade de Salvador-Ba, trecho este perto do encontro da citada avenida com a avenida Vasco da Gama.

Da posse de um filme de aproximadamente dez minutos, a câmera foi conectada à placa de captura de vídeo com o objetivo de obter as imagens digitais equivalentes. Para tanto, três programas gráficos foram utilizados.

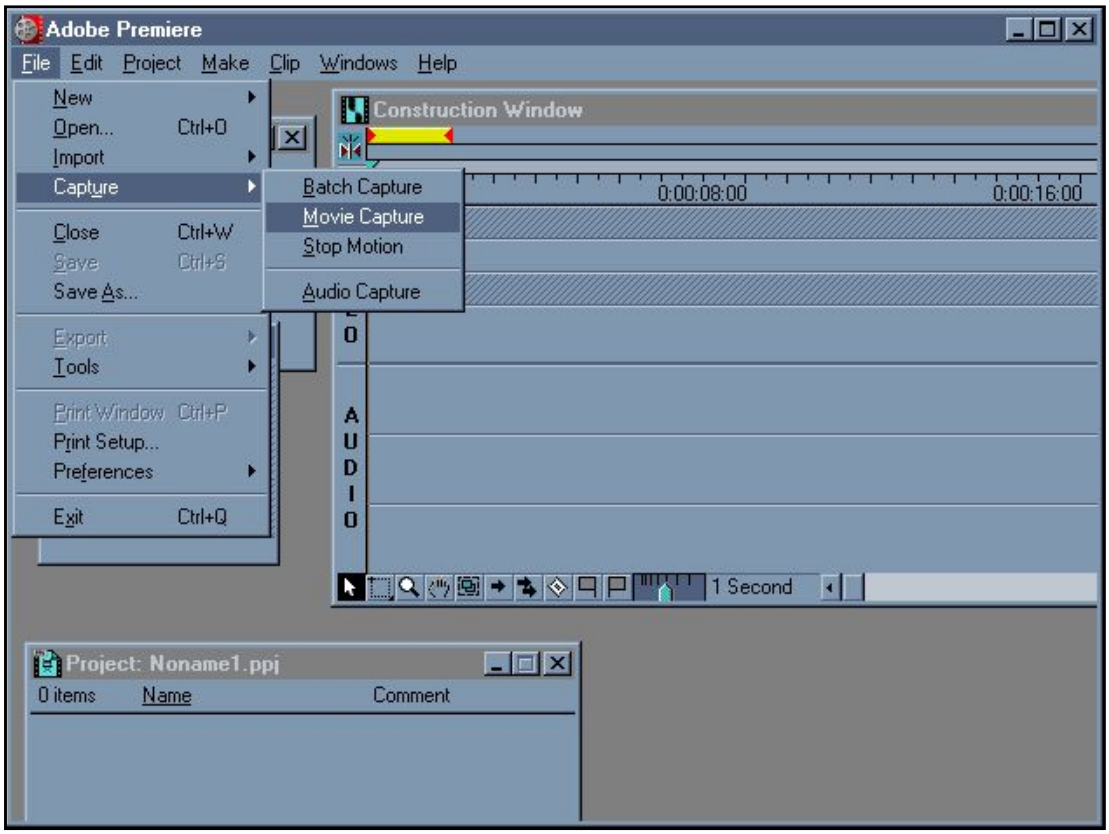


Figura 4.1 - Tela do programa Adobe Premiere.

Inicialmente, utilizou-se o programa de computador Adobe Premiere para gerar um arquivo AVI (clipe de vídeo) que consiste numa conversão do sinal emitido pela câmera para uma seqüência imagens digitais agrupadas na forma de um vídeo, incorporando inclusive o som, embora no presente caso desnecessário. Dentro do Premiere, no menu *file* - *capture* - *video capture* (Figura 4.1), foram definidas as opções de captura, a saber, resolução 160x120 e taxa de captura de 15 fps (frames por segundo). Estas opções resultam em termos 15 imagens geradas a cada segundo de filme e no fato de que cada imagem equivalerá a uma matriz de 160 colunas por 120 linhas. Visto que quanto menor a resolução mais leve se tornará o processamento das imagens, uma resolução maior não foi escolhida por esta se mostrar suficiente para posterior aplicação da técnica de detecção de bordas. É importante mencionar que a versão do Premiere a ser utilizada deve conter o *driver INDEL* sem o qual a captura não é possível. A versão 5.0 deste programa, por exemplo, não pode ser utilizada por não possuir este *driver*.

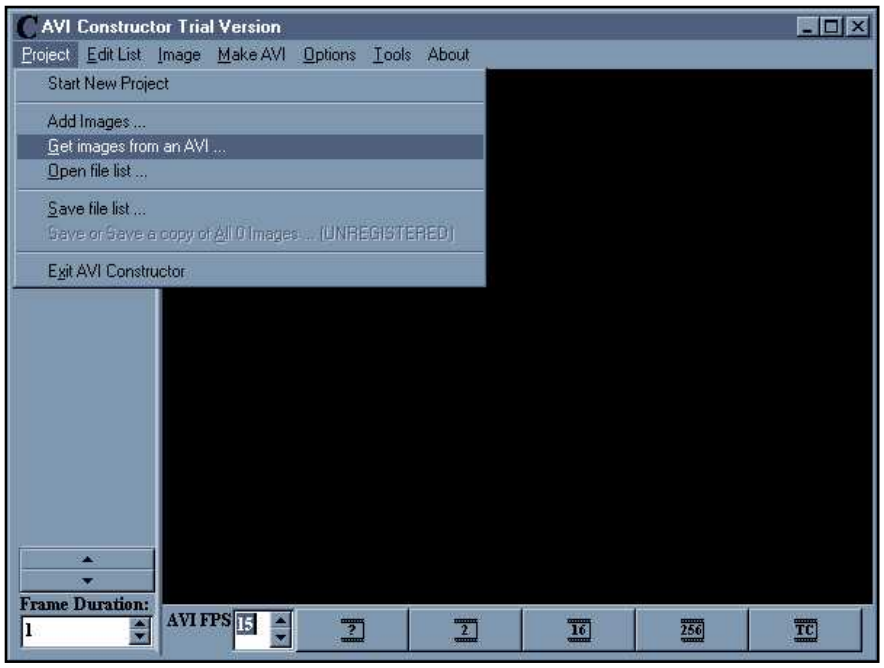


Figura 4.2 - Tela do programa AVI Constructor.

A seguir, o programa AVI Constructor foi utilizado para desmembrar o arquivo de filme digital AVI em uma série de arquivos de imagens digitais. O procedimento foi muito simples, embora não trivial. Isto porque este programa não tem por opção direta desmembrar o filme de um arquivo AVI para gerar as imagens digitais equivalentes. Entretanto, quando o usuário seleciona o menu *project - get images from an AVI* (figura 4.2), ele carrega o arquivo AVI escolhido para iniciar sua edição, gerando para isso arquivos temporários que nada mais são do que as imagens digitais desmembradas a partir do filme. Sem dar término à execução do AVI Constructor (pois senão os temporários são apagados), basta copiar tais arquivos temporários para um outro diretório ou pasta e, então, pode-se fechar o AVI Constructor, ou seja, terminar a sua execução. Isto gerou 8402 imagens em arquivos distintos e autonumerados, pois haviam aproximadamente dez minutos de filme que, à taxa de captura mencionada anteriormente de 15 imagens por segundo, teria de gerar aproximadamente 9000 fotos.

Das fotos obtidas, foram selecionadas aquelas que não continham imagens parciais de carros para permitir uma definição correta do número de fotos em cada foto, possibilitando um treinamento bem definido e uma verificação exata dos erros resultantes do reconhecimento de imagens não utilizadas no treinamento. Imagens com ônibus, caminhões ou motos foram retiradas também por dificultarem a definição exata de sua equivalência em número de carros. Em resultado, das 8402 imagens originais, restaram 3200 fotos que se mostraram mais que suficientes para a realização deste experimento. Além disso, cabe mencionar que dentre os vários padrões existentes de arquivos de imagens digitais, estas fotos foram geradas automaticamente no padrão BMP (Imagem de Bitmap). Se não fosse assim, teriam de ser convertidas para tal padrão para sua posterior utilização no MATLAB.

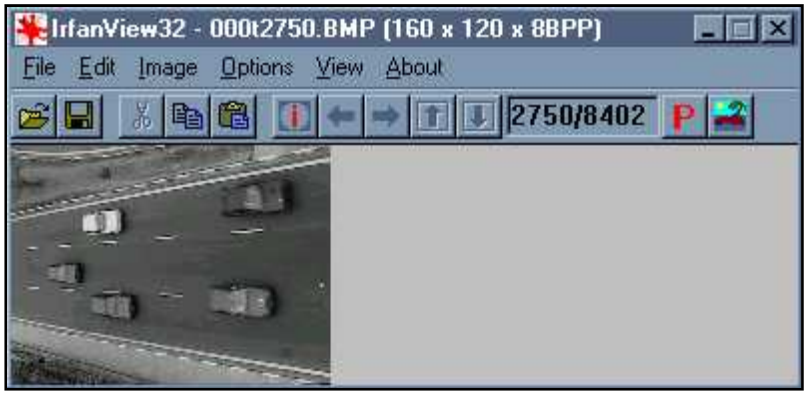


Figura 4.3 - Tela do Programa IrfanView.

O derradeiro passo na obtenção das imagens digitais é a utilização do programa IrfanView para converter as imagens que até então são coloridas para imagens em preto e branco, ou seja, em tons de cinza. Isto foi feito devido ao fato de que as cores são claramente irrelevantes no processo de decisão da quantidade de carros, até mesmo dificultando tal decisão por acrescentar informação dispensável. O IrfanView foi utilizado por possuir uma opção no menu *file* - *batch conversion* (Figura 4.3) que possibilita a conversão de todas as imagens através de um único comando, enquanto outros programas fazem esta conversão apenas arquivo a arquivo.

4.3 Pré-processamento das imagens

A partir deste ponto, de posse dos 3200 arquivos de imagens BMP em preto e branco no formato 160x120 pixels, todo o processo fica a cargo do programa MATLAB e seus *toolboxes* de Redes Neurais e de Processamento de Imagens, que acrescentam as rotinas necessárias ao MATLAB para a obtenção dos resultados desejados.

Foi mencionado anteriormente que um pré-processamento que vise favorecer a informação pertinente, eliminando ou reduzindo ruídos indesejáveis, alivia a

rotina principal de tratamento destes dados na busca do resultado final. Por que utilizar técnicas de detecção de bordas como um pré-processamento das imagens? A idéia surgiu da possibilidade provável de que informações tais como a cor do veículo, nível de iluminação variável durante o dia, presença de chuva ou não, detalhes no modelo do carro, entre outros, sobrecarregariam o processo de identificação de carros através de um treinamento a partir de um número finito de imagens. Visto que estas características variam muito, tal número finito de imagens teria de ser muito grande para levar a rede neural a elaborar uma "regra geral" para fazer o reconhecimento do número de carros de qualquer imagem retirada daquele mesmo ponto de filmagem. Isto se dá porque a rede neural, que será responsável pelo reconhecimento da quantidade de carros em uma imagem, terá antes que ser "treinada" por receber um número finito de fotos e a informação da quantidade de carros que cada imagem possui. Após esta etapa de treinamento, fotos que não participaram nele devem ser reconhecidas dentro de uma margem de erro aceitável, mostrando que a regra gerada pelo treinamento se tornou suficientemente geral para atender às necessidades de reconhecimento. Quanto maior o nível de variação de detalhes, maior será o número de imagens a serem utilizados no treinamento para gerar a desejada "regra geral".

No presente caso, as imagens foram divididas em lotes de treinamento e lotes de verificação, sendo que os lotes de treinamento na quantidade de três e na forma de um primeiro lote com 400 imagens, um segundo com 200 imagens e um terceiro com 200 imagens. Já os lotes de verificação foram divididos na quantidade de seis lotes de 400 imagens cada. Esta divisão em vários lotes visa possibilitar a manipulação de tamanho volume de informações em vista da grande quantidade de memória RAM que este procedimento de treinamento necessitará.

As imagens foram então pré-processadas pelo MATLAB, que possui uma rotina de detecção de bordas implementada em seu *toolbox* de Tratamento de Imagens e denominada **edge**. Esta rotina possui basicamente quatro métodos matemáticos para se detectar as bordas de uma imagem: *sobel*, *prewitt*, *roberts* e *log*. O método *log* foi apontado como o mais apropriado através da realização de um teste visual que consiste da rotina "testevisual.m" (ANEXO), podendo ser observado em parte na Figura 4.4.

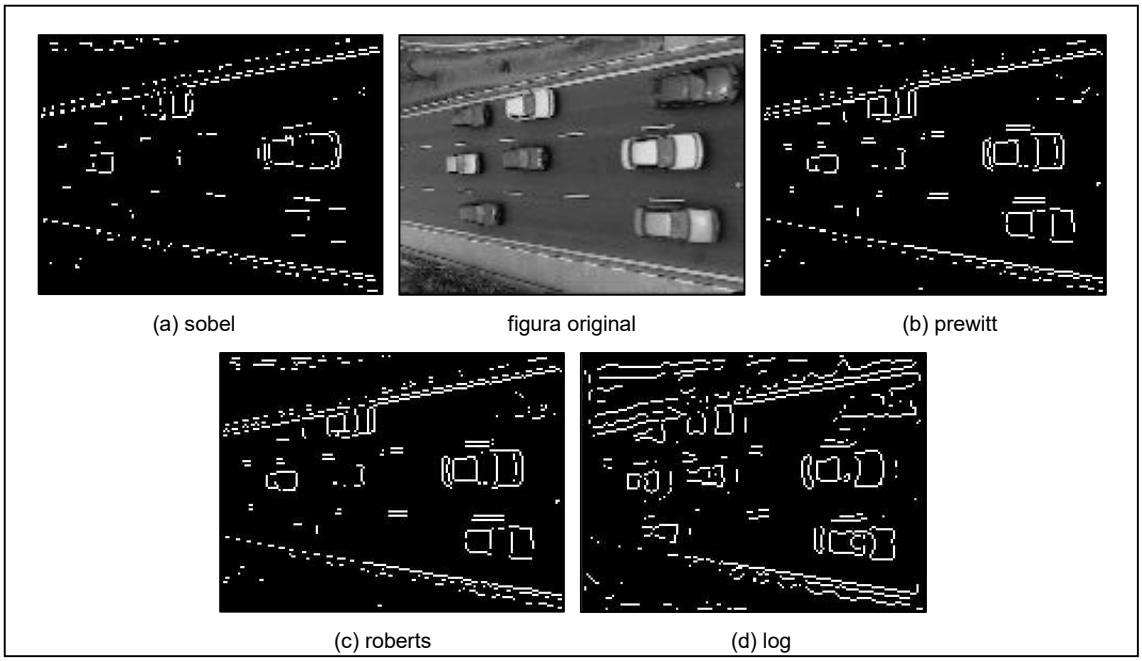


Figura 4.4 - Teste visual comparativo dos métodos de detecção de bordas.

Além disso, a área fora da pista de passagem de carros, ou seja, a área da calçada em diante, foi excluída no momento da entrada de dados na rede neural, resultando numa redução por imagem de anteriores 19200 pixels (160x120) para 12585 pixels, uma eliminação de 34,45% de informação claramente sem utilidade.

No MATLAB, este pré-processamento foi realizado através do arquivo "def400p1.m" (ANEXO), definindo no final uma matriz de entrada com cada imagem na forma de uma de suas colunas, ou seja, uma matriz *Inp* (input) de 12585x400 pixels, visto que foram utilizadas 400 fotos para treinamento inicial. Também ficou definida a

matriz de resposta correta do número de carros, ou seja, uma matriz *Tar* (target) de 1x400 elementos, que contém o número de carros de forma direta. Isto se dá por causa do fato de que a camada final da rede neural consiste de um neurônio do tipo *log-sigmoidI*, cuja saída varia de 0 a 1. Devido ao fato de que o número de carros nas imagens variou de 0 a 8, foi utilizada a seguinte forma de adequar esta variação

$$Tar = Tar /10 + 0,1 \quad (4.1)$$

o que fez a faixa de respostas corretas variar de 0,1 a 0,9.

Desta forma, foram utilizados arquivos similares para definir todos os nove lotes de figuras utilizados seja para treinamento como para verificação.

4.4 Definição da rede neural

A definição da rede neural mais apropriada a determinado problema é essencialmente empírica. Na escolha do número de camadas, definiu-se duas como estritamente necessárias, sendo a primeira para receber diretamente cada pixel da imagem e a última para totalizar e emitir o resultado (o número de carros) na forma de um valor único e portanto possuindo um único neurônio. Uma camada adicional, intermediária, foi utilizada por se entender que esta traria a possibilidade de um reprocessamento do resultado da primeira camada, aumentando as chances de obtenção de uma regra geral que modele o reconhecimento de carros. Além disso, esta camada intermediária pesa muito pouco no tempo de processamento pois ela só gera conexões com um pequeno número de neurônios, os da primeira camada, enquanto que na primeira camada cada neurônio tem uma conexão com cada pixel. Como forma escolhidos vinte neurônios para a primeira camada e outros vinte para a segunda camada, são 800 conexões entre a

primeira e a segunda camadas contra 12585 conexões entre a entrada de dados e a primeira camada de neurônios.

A quantidade de neurônios foi definida em função da quantidade de dados utilizada no treinamento. Embora para determinar tal quantidade não haja uma regra simples, pode-se observar que a necessidade de mais neurônios cresce à medida que se aumenta o universo de treinamento, pois com este aumento a regra geral a ser obtida tende a ser mais complexa e detalhada, gerando a necessidade de um aumento na quantidade de neurônios. Assim, em um treinamento realizado com 50 imagens, 8 neurônios podem ser suficientes para as duas primeiras camadas, enquanto que treinando com 200 imagens a necessidade de neurônios pode subir para 12. Um outro aspecto é o de que a regra geral obtida no primeiro treinamento não vai ser tão geral quanto a obtida com as 200 imagens no segundo exemplo. Apesar disso e de acordo com as necessidades de margem de erro, ambas as redes podem ser satisfatórias para solucionar o problema.

Tendo sido definidos o número de camadas e a quantidade neurônios em cada camada (20x20x1), foi escolhida a função de transferência *log-sigmoid* (2.2) para os neurônios de todas as camadas. Isto foi feito por se desejar que a saída dos neurônios fosse limitada a uma pequena faixa, diminuindo a possibilidade de resultados muito espúrios, e que não atingissem valores negativos por não serem resultados úteis para o problema.

No MATLAB, toda esta definição é realizada pela rotina **initff**, que define valores iniciais randômicos para os ganhos de cada conexão de cada neurônio de uma rede *feed-forward*, ou seja, de redes que não possuem realimentação de dados. Seu uso pode ser visto no arquivo “train400p1.m” (ANEXO).

4.5 Treinamento da rede neural

Para realizar o treinamento da rede neural *feed-forward* foi escolhido o método *backpropagation*. Este método é implementado no MATLAB através das rotinas descritas a seguir:

- **trainbp**: treinamento básico com o método *backpropagation*;
- **trainbpa**: treinamento aprimorado por aprendizado adaptativo;
- **trainbpm**: treinamento aprimorado por aprendizado com *momentum*;
- **trainbpx**: treinamento aprimorado tanto por aprendizado adaptativo como pela utilização de *momentum*;
- **trainlm**: treinamento utilizando avanços do método variante de Levenberg-Marquardt, sendo veloz e necessitando de muito mais memória;

A característica *momentum* diminui a sensibilidade do método a pequenos detalhes na superfície de erro, ajudando a rede a evitar mínimos locais que poderiam impedi-la de encontrar um ponto de erro satisfatório. O tempo de treinamento também pode ser diminuído por se utilizar aprendizado adaptativo, pois este tenta manter a taxa de aprendizado tão grande quanto possível, ao mesmo tempo que mantém o treinamento estável. Sendo tais características desejáveis, a rotina utilizada foi a **trainbpx**. A **trainlm** não foi cogitada por causa de sua superior necessidade de quantidade de memória, fator limitante no processo de treinamento devido ao grande volume de informação em processo. Para se ter uma idéia, só a matriz de entrada do treinamento *tar* possui 12585x400 pixels, cada pixel sendo uma variável do tipo *real double*, ou seja, ocupando 8 bytes de memória. Portanto, só esta matriz ocupa 40272000 bytes da memória RAM do microcomputador!

O treinamento da rede ocorre em etapas denominadas épocas. Estas épocas consistem em se aplicar todos os dados de entrada de treinamento na rede, verificar o erro de cada um desses dados e ajustar a rede para diminuir o erro médio. Isto evita que, ao ajustar a rede para uma entrada, se aumente o erro das outras. Assim, um vetor parâmetro de fundamental importância na rotina **trainbpx** é chamado de TP (training parameters), pois define os parâmetros que ajustam as necessidades do treinamento. Dois dos parâmetros úteis de TP definem o número máximo de épocas de treinamento e o erro mínimo desejado. O treinamento perdurará enquanto nenhuma destas duas condições for atingida. Também existem parâmetros em TP para ajustar as características de *momentum* e de aprendizado adaptativo. A utilização da rotina **trainpbx** bem como os valores utilizados em tais parâmetros podem ser vistos no arquivo "train400p1.m" (ANEXO).

4.6 Simulação da rede neural

Após a obtenção da rede neural treinada, vem a necessidade de verificar se os resultados obtidos foram satisfatórios, ou seja, se a rede neural conseguiu definir uma regra suficientemente geral para reconhecer corretamente a quantidade de fotos não só das imagens utilizadas para treinamento mas também das imagens que lhe são até então desconhecidas. Este procedimento é chamado de simulação e é realizado com base na rotina **simuff** do MATLAB, que pode ser vista em sua forma de utilização no arquivo "sim400p4p1.m" (ANEXO). O seu nome vem trazendo a informação de que as 400 imagens do lote 4 estão sendo simuladas na rede neural resultante do treinamento com apenas o lote 1. Além de realizar a simulação, este arquivo também determina os erros

médios absolutos tanto do lote de treinamento como do lote de verificação, determinando também a distribuição do erro deste último lote por faixas de variação.

4.7 Estrutura geral do experimento

Tendo sido as imagens obtidas, pré-processadas e a definidas no espaço de trabalho do MATLAB e salvas separadamente em nove arquivos definidos como lotes, foi realizada a inicialização da rede neural seguida do primeiro procedimento de treinamento com o lote denominado p1, possuidor de 400 imagens. Após este processamento, obteve-se a rede neural treinada e pronta para ser submetida à simulação separadamente com os lotes denominados p4, p5, p6, p7, p8 e p9, cada um de 400 fotos.

Igual procedimento de definição e treinamento foi realizado com as imagens não pré-processadas pela técnica de detecção de bordas, para determinação da validade de utilizar-se de tal técnica. Neste caso, não se chegou a realizar nenhum procedimento de simulação.

Mantendo-se a linha de trabalho com detecção de bordas e partindo-se por ponto inicial do treinamento resultante do lote p1, acrescentou-se o lote p2, possuidor de 200 imagens, com o objetivo de obter uma rede neural com um treinamento mais aprimorado. Novamente, obteve-se a rede neural treinada e pronta para ser submetida à simulação separadamente com os lotes de p4 a p9.

Por fim, acrescentou-se o lote p3, possuidor de 200 imagens, com o objetivo de obter um aprimoramento ainda maior, determinado ou não uma tendência. Assim, obteve-se a rede neural treinada e pronta para ser submetida à simulação separadamente com os lotes de p4 a p9.

Capítulo 5

Resultados e Discussão

5.1 Validação do uso de detecção de bordas

O treinamento do lote p1 de 400 imagens, sem utilização de técnicas de detecção de bordas foi realizado com limite de 1000 épocas ou até o momento em que o erro quadrático médio (Sum-Squared Error - SSE) atingisse o valor de 0,01 equivalente a um décimo de carro devido à anteriormente mencionada adequação das faixas de resposta.

Figura 5.1 - Treinamento do lote p1 sem detecção de bordas.

Este treinamento levou cerca de seis horas de processamento e iniciou com SSE = 21,7737 e, ao término das 1000 épocas sem atingir o erro desejado, obteve SSE = 0,6771, como pode ser visto na Figura 5.1.

Em seqüência, treinamento similar foi realizado com a única diferença consistindo da utilização de técnicas de detecção de bordas. Este treinamento levou cerca de três horas de processamento e iniciou com SSE = 107,1609 e, ao término de 372 épocas atingiu o erro desejado, obtendo-se SSE = 0,0097, como pode ser visto na Figura 5.2.

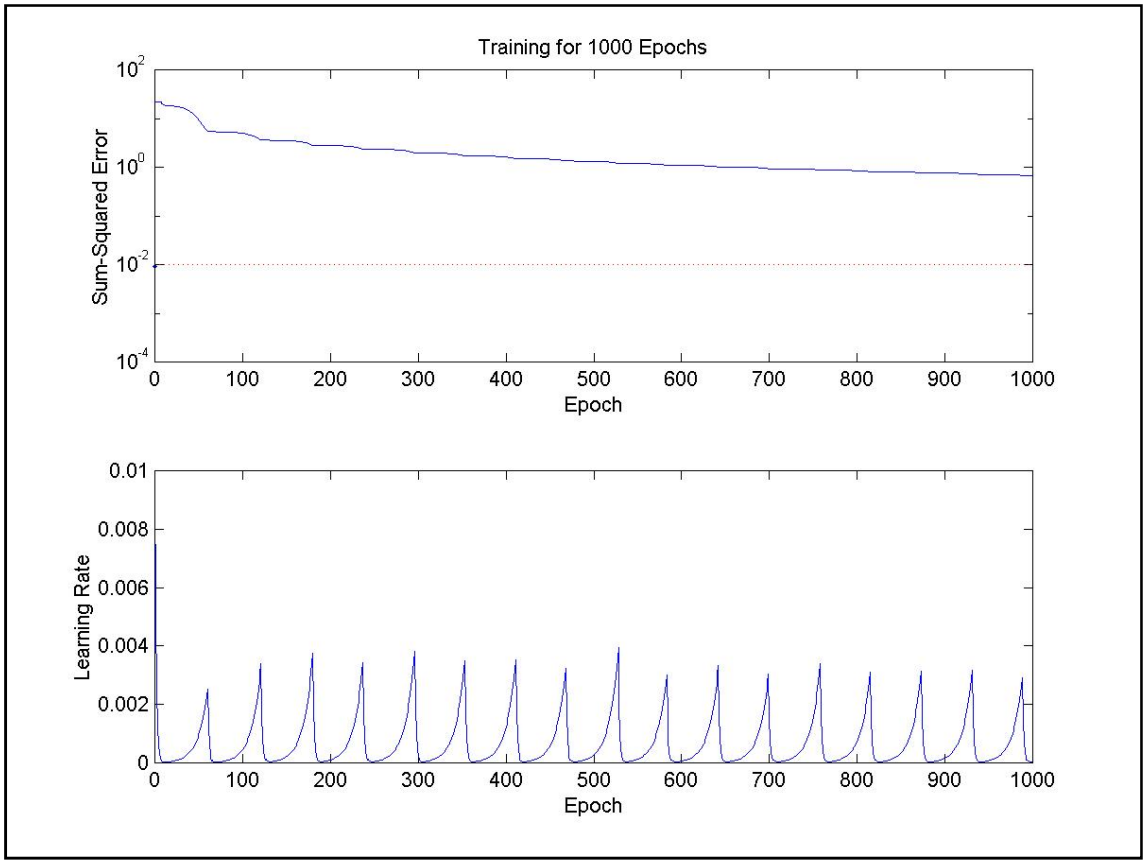
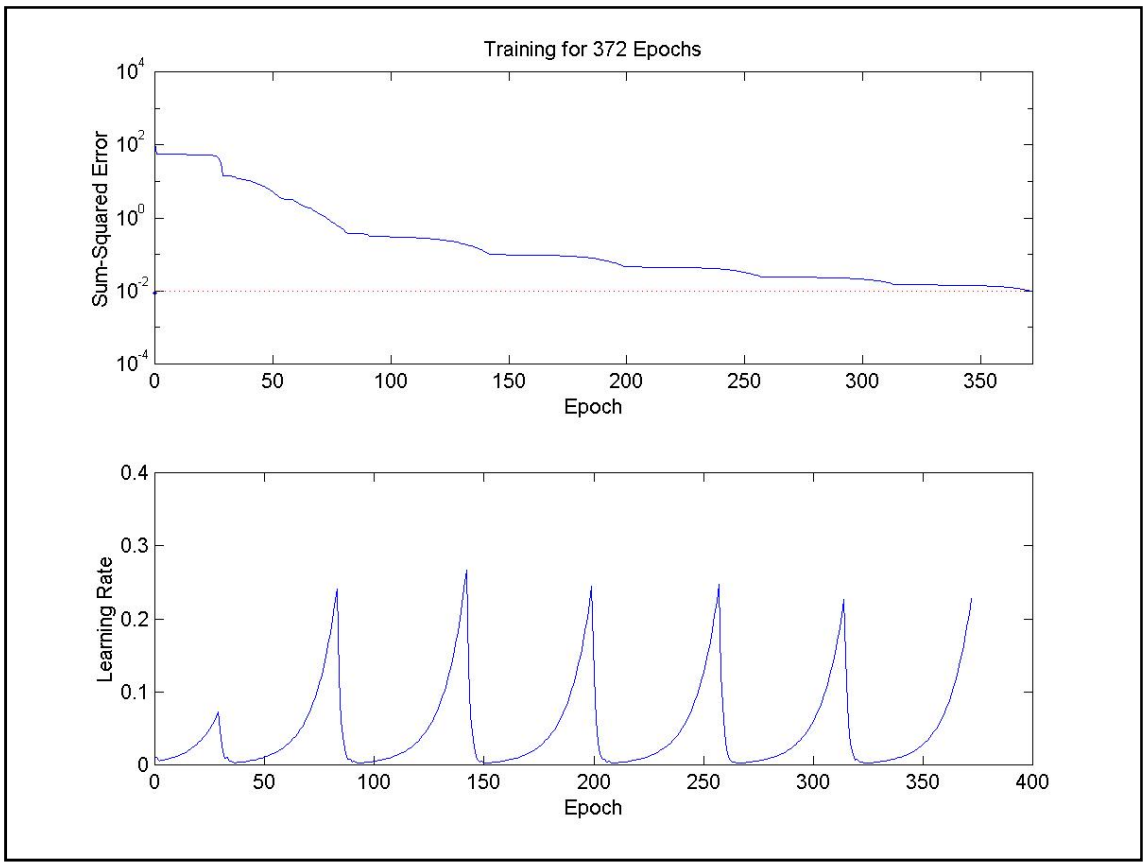


Figura 5.2 - Treinamento do lote p1 com detecção de bordas.

Comparando-se tais resultados, fica clara a superioridade do método que utilizou detecção de bordas. Testes adicionais mostraram que mesmo realizando o treinamento com as imagens na sua forma original com 4000 épocas de treinamento, o que demorou cerca de vinte horas de treinamento, não se pode reduzir o valor de SSE de 0,5. Este é um erro muito grande, da ordem de 5 carros por imagem!

Também fica clara a superioridade da velocidade de convergência do



treinamento que utilizou detecção de bordas. Embora ambos os métodos mostrem tendência de queda do erro, a lentidão desta queda no experimento sem detecção de bordas demonstra que mesmo em 10000 épocas de treinamento é improvável que se consiga o resultado desejado, além disto provavelmente vir a demorar dias!

Devido ao fato de que o método de treinamento utilizado possui técnicas de aprendizado adaptativo, podem ser vistos os valores que a da taxa de aprendizado (Learning Rate) assumiu durante ambos os processos. Tal taxa sobe à medida que percebe que o erro continua a diminuir e cai no momento que verifica que o erro aumenta ou se mantém fixo. Comparando-se Os valores médios desta taxa visualizados nas Figuras 5.1 e 5.2, vê-se mais uma vez a superior velocidade de convergência do experimento com detecção de bordas.

5.2 Desempenho da rede neural

Mantendo a utilização de detecção de bordas, foram realizados os treinamentos adicionais descritos no tópico 4.7, a saber, um treinamento com 600 imagens (lotes p1 e p2) apresentado na Figura 5.3 e outro com 800 imagens (lotes p1, p2 e p3) apresentado na Figura 5.4. Ambos partiram do ponto inicial do experimento imediatamente anterior , ou seja, sem reinicializar a rede neural. Assim, obteve-se um total de três redes neurais diferentes, embora similares e possuidoras da mesma estrutura básica.

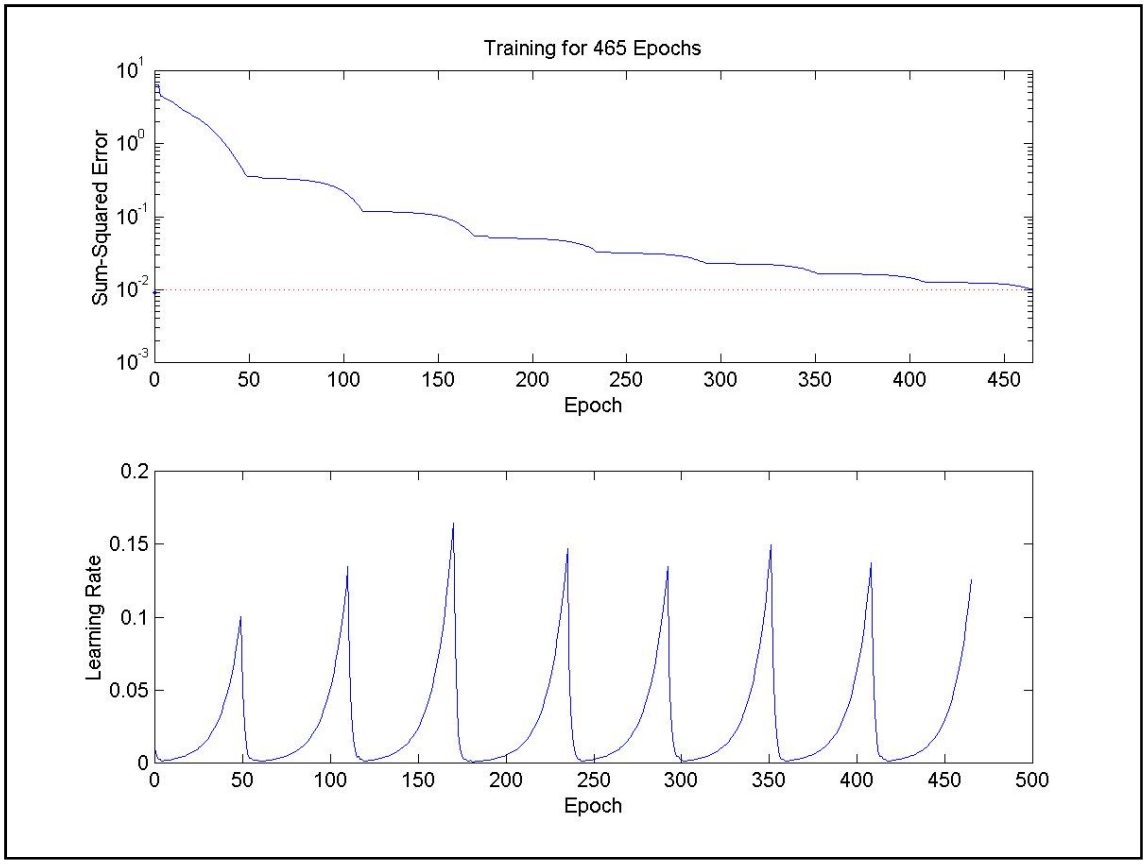


Figura 5.3 - Treinamento dos lotes p1+p2 com detecção de bordas.

O treinamento com 600 imagens (p1+p2) levou cerca de cinco horas de processamento e iniciou com SSE = 6,3689 e, ao término de 465 épocas, atingiu o erro desejado, obtendo SSE = 0,0100, como pode ser visto na Figura 5.3. O erro inicial aqui obtido é bem menor do que no experimento da Figura 5.2, tendo em vista que se utilizou por ponto inicial o resultado do treinamento do lote p1. Apesar disso, o número de épocas necessário para que se atingisse o erro desejado aumentou, pois o universo de imagens a ser modelado pela rede neural na forma de uma regra geral se tornou mais complexo. Também podem ser comparados os tempos de duração, levando em conta não a demora total mas o tempo gasto por época. Este deveria aumentar pois cada época teve de processar 200 figuras a mais e, de fato, aumentou.

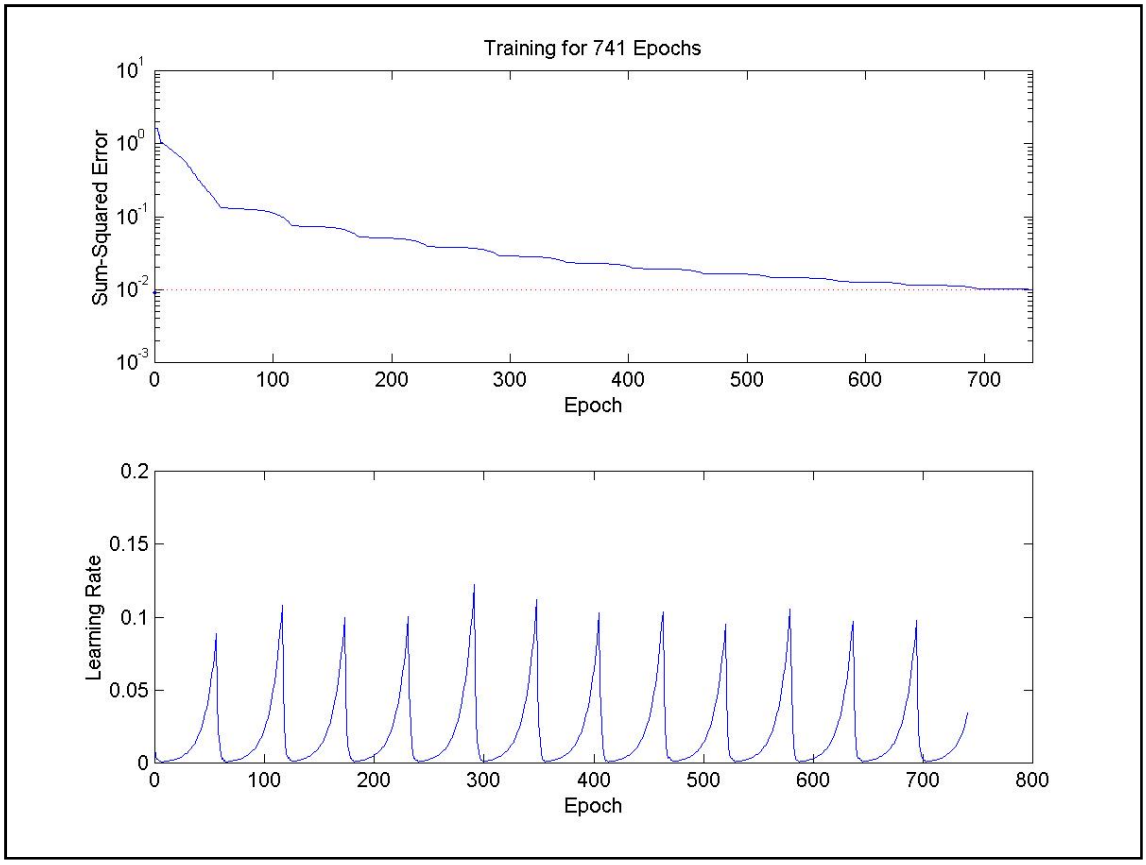


Figura 5.4 - Treinamento dos lotes p1+p2+p3 com detecção de bordas.

Já o treinamento com 800 imagens (p1+p2+p3) demorou cerca de nove horas e meia de processamento, iniciando seu processo com SSE = 1,6093 e, ao término de 741 épocas, atingiu o erro desejado, obtendo SSE = 0,0100, como pode ser visto na Figura 5.3. O erro inicial aqui obtido é ainda menor do que no experimento da Figura 5.3, tendo novamente em vista que se utilizou por ponto inicial o resultado do treinamento do lote p2, sendo novidade para a rede neural apenas um quarto das fotos apresentadas para treinamento. Mais uma vez, o número de épocas necessário para que se atingisse o erro desejado aumentou, bem como o tempo gasto por época de treinamento, por causa das mesmas razões mencionadas anteriormente quando se comparou o experimento de treinamento com os lotes “p1 e p2” e com apenas “p1”.

Para comparar os resultados obtidos com os três treinamentos realizados com o auxílio de detecção de contornos, foram realizadas as simulações com os seis lotes de verificação, cujos erros resultantes em **número de carros** por imagem podem ser vistos nas tabelas e gráficos a seguir.

Lote Simulado	Quant. de Imagens	Erro Médio Absoluto	Erro absoluto distribuído em faixas						
			até 0,25	≥0,25 a 0,5	≥0,5 a 1,0	≥1,0 a 2,0	≥2,0 a 3,0	≥3,0 a 4,0	≥ 4
p1	400	0,0492	-	-	-	-	-	-	-
p4	400	2,2503	17	18	37	105	149	74	0
p5	400	1,4315	40	43	78	175	56	8	0
p6	400	1,2476	58	53	100	156	31	2	0
p7	400	0,9820	116	65	118	83	12	4	2
p8	400	1,0009	62	66	137	118	16	1	0
p9	400	0,8189	126	85	117	57	14	1	0
p4 a p9	2400	**1,2885**	419	330	587	694	278	90	2

Tabela 5.1 - Desempenho da rede neural após treinamento com lote p1.

Lote Simulado	Quant. de Imagens	Erro Médio Absoluto	Erro absoluto distribuído em faixas						
			até 0,25	≥0,25 a 0,5	≥0,5 a 1,0	≥1,0 a 2,0	≥2,0 a 3,0	≥3,0 a 4,0	≥ 4
p1+p2	600	0,0407	-	-	-	-	-	-	-
p4	400	1,3358	48	36	94	175	44	2	1
p5	400	1,0237	69	77	113	125	15	1	0
p6	400	0,9208	95	72	117	102	12	2	0
p7	400	0,8815	130	66	124	67	10	2	1
p8	400	0,8915	60	81	146	106	7	0	0
p9	400	0,7158	133	91	125	44	7	0	0
p4 a p9	2400	**0,9615**	535	423	719	619	95	7	2

Tabela 5.2 - Desempenho da rede neural após treinamento com lotes p1+p2.

Lote Simulado	Quant. de Imagens	Erro Médio Absoluto	Erro absoluto distribuído em faixas						
			até 0,25	≥0,25 a 0,5	≥0,5 a 1,0	≥1,0 a 2,0	≥2,0 a 3,0	≥3,0 a 4,0	≥ 4
p1+p2+p3	800	0,0353	-	-	-	-	-	-	-
p4	400	0,9347	109	86	111	78	13	2	1
p5	400	0,6784	101	125	134	35	4	1	0
p6	400	0,7648	99	88	145	61	7	0	0
p7	400	0,8674	122	79	120	66	10	3	0
p8	400	0,8453	71	90	144	89	6	0	0
p9	400	0,7096	133	86	120	57	4	0	0
p4 a p9	2400	**0,8000**	635	554	774	386	44	6	1

Tabela 5.3 - Desempenho da rede neural após treinamento com lotes p1+p2+p3.

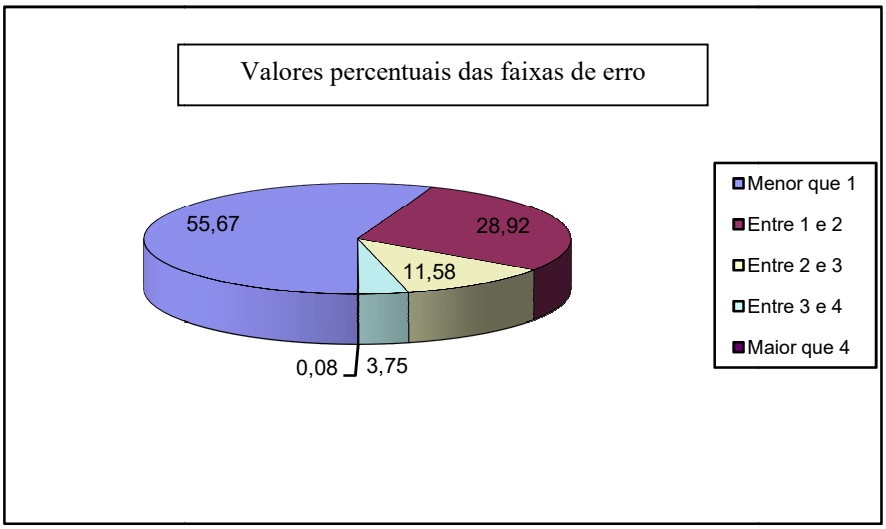


Figura 5.5 - Gráfico de desempenho do treinamento com lote p1.

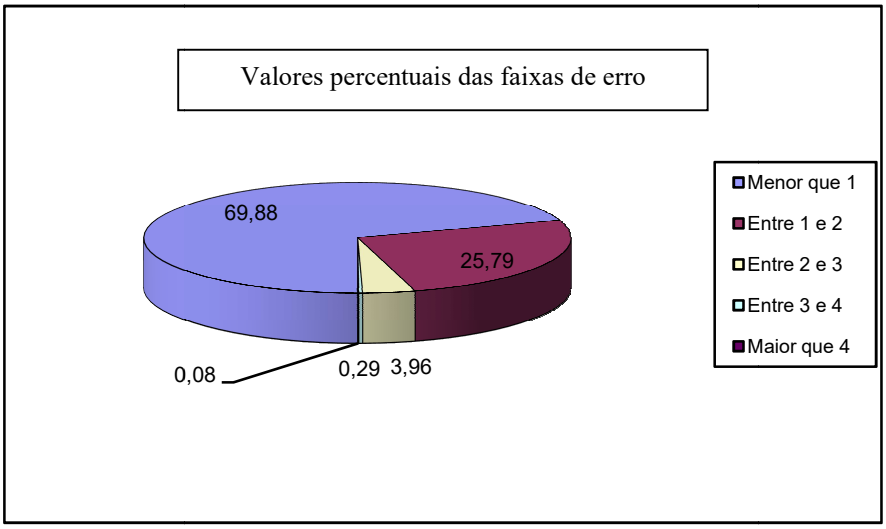


Figura 5.6 - Gráfico de desempenho do treinamento com lote p1+p2.

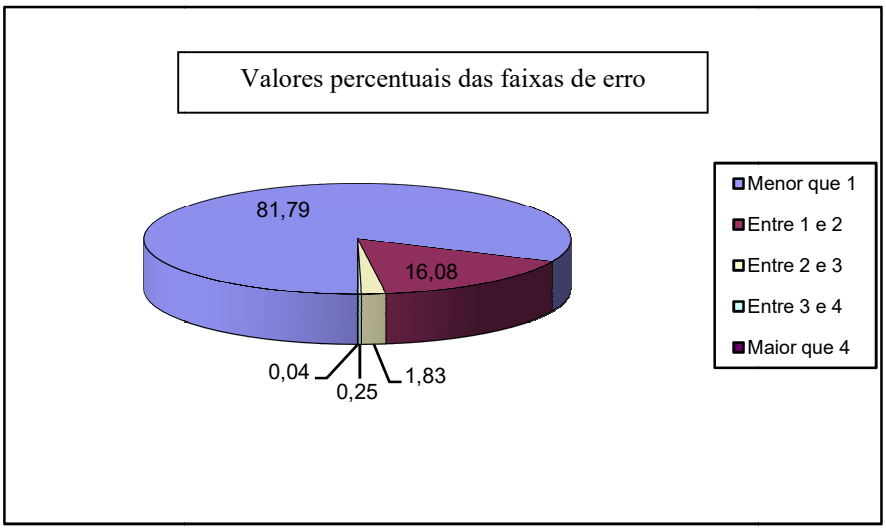


Figura 5.7 - Gráfico de desempenho do treinamento com lotes p1+p2+p3.

Para uma melhor análise percentual da evolução do erro, seguem as mesmas tabelas anteriores com o erro normalizado (divisão imagem a imagem do erro pela resposta desejada), bem como uma tabela que expõe o conteúdo dos lotes de treinamento e de verificação no tocante à quantidade de carros em cada imagem.

Lote Simulado	Quant. de Imagens	Erro Médio Absoluto Normalizado	Erro normalizado distribuído em faixas						
			até 0,10	≥0,10 a 0,25	≥0,25 a 0,5	≥0,5 a 0,75	≥0,75 a 1,0	≥1,0 a 1,5	≥ 1,5
p1	400	0,0038	-	-	-	-	-	-	-
p4	400	0,4645	33	45	66	36	28	90	102
p5	400	0,4088	37	68	77	59	47	42	70
p6	400	0,4114	56	64	90	45	36	27	82
p7	400	0,2965	97	98	127	65	11	0	2
p8	400	0,3393	53	78	122	51	27	49	20
p9	400	0,2713	108	116	74	57	28	16	1
p4 a p9	2400	**0,3653**	284	469	556	313	177	224	277

Tabela 5.4 - Desempenho normalizado da rede neural após treinamento com lote p1.

Lote Simulado	Quant. de Imagens	Erro Médio Absoluto Normalizado	Erro normalizado distribuído em faixas						
			até 0,10	≥0,10 a 0,25	≥0,25 a 0,5	≥0,5 a 0,75	≥0,75 a 1,0	≥1,0 a 1,5	≥ 1,5
p1+p2	600	0,0034	-	-	-	-	-	-	-
p4	400	0,3004	60	66	103	58	34	50	29
p5	400	0,3033	57	87	118	58	18	28	34
p6	400	0,3134	74	91	96	44	30	48	17
p7	400	0,2679	96	115	139	40	8	0	2
p8	400	0,3001	61	81	131	39	31	46	11
p9	400	0,2311	122	115	72	51	28	11	1
p4 a p9	2400	**0,2860**	470	555	659	290	149	183	94

Tabela 5.5 - Desempenho normalizado da rede neural após treinamento com lotes p1+p2.

Lote Simulado	Quant. de Imagens	Erro Médio Absoluto Normalizado	Erro normalizado distribuído em faixas						
			até 0,10	≥0,10 a 0,25	≥0,25 a 0,5	≥0,5 a 0,75	≥0,75 a 1,0	≥1,0 a 1,5	≥ 1,5
p1+p2+p3	800	0,0028	-	-	-	-	-	-	-
p4	400	0,2367	97	132	113	36	20	2	0
p5	400	0,2211	86	123	124	43	16	8	0
p6	400	0,2705	85	96	99	56	32	27	5
p7	400	0,2635	90	121	139	42	5	1	2
p8	400	0,2834	55	105	125	37	32	36	10
p9	400	0,2226	124	110	69	44	31	21	1
p4 a p9	2400	**0,2496**	537	687	669	258	136	95	18

Tabela 5.6 - Desempenho normalizado da rede neural após treinamento com lotes p1+p2+p3.

Lote	Quant. de	Número de carros								
	Imagens	0	1	2	3	4	5	6	7	8
p1	400	89	108	46	58	49	29	11	0	10
p2	200	25	84	39	17	19	16	0	0	0
p3	200	95	105	0	0	0	0	0	0	0
p4	400	77	148	52	54	51	12	5	1	0
p5	400	117	187	65	1	15	15	0	0	0
p6	400	131	205	46	8	5	5	0	0	0
p7	400	81	173	74	54	13	5	0	0	0
p8	400	135	148	82	27	8	0	0	0	0
p9	400	93	197	65	29	16	0	0	0	0
p1 a p9	3200	843	1355	469	248	176	82	16	1	10

Tabela 5.7 - Distribuição das imagens por quantidade de carros.

Tanto o desempenho geral como de cada lote de verificação demonstram a clara tendência de melhora do desempenho da rede neural à medida que o treinamento é reforçado com um maior número de imagens. Portanto para obter resultados melhores, a depender da necessidade de reconhecimentos mais aprimorados, basta tornar mais numerosos os lotes de treinamento.

O desempenho final obtido de um erro absoluto médio de 0,8 carros por imagem não conhecida já é suficiente para atingir o objetivo de comparar fluxos de duas ou mais vertentes de um semáforo. Em tempo de execução, imagens geradas a intervalos relativamente longos podem dar a noção suficiente do fluxo real em cada vertente. Comparando o resultado visual da aplicação de detecção de bordas no experimento aqui desenvolvido (Figura 4.4) com o resultado visual obtido na explanação teórica (Figura 3.4), pode-se ver que o resultado é espantoso tendo em vista o relativo baixo nível de bordas geradas pela rotina contida no MATLAB. De posse de rotinas superiores ou de maior resolução de imagens, este resultado certamente seria ainda melhor.

Capítulo 6

Conclusão e Perspectivas Futuras

Com a associação das ferramentas de Tratamento de Imagem, na forma de detecção de contornos, e de Redes Neurais, pode ser obtido um resultado que não seria possível somente com uma destas ferramentas. A comparação entre os resultados do treinamento de uma rede neural com fotos não pré-processadas e uma outra com fotos pré-processadas demonstrou no mínimo uma superioridade no desempenho desta última, superioridade esta que leva a uma economia de tempo de treinamento. Além disso, sequer houve sucesso comprovado da rede neural sem pré-processamento, sucesso este que talvez possa ser obtido com treinamentos excessivamente longos, com sua duração da ordem de dias.

Apesar de não se ter obtido um resultado com erro cem por cento inferior a um carro, ficou demonstrado que o erro relacionado ao reconhecimento correto do número de carros em uma imagem obtida a partir de um mesmo ponto fixo depende de quão bem o lote de imagens utilizado no treinamento represente o variação possível de todo o universo de imagens. Com o aumento do número de imagens utilizadas no treinamento, aumentam as chances de se aproximar de um lote cuja representatividade seja boa.

Estudos futuros podem ser realizados no sentido de comparar a eficiência de diferentes arquiteturas de redes neurais, bem como de diferentes rotinas de treinamento.

Devido ao fato de que o programa MATLAB só trabalha bem com variáveis reais de dupla precisão, ocupando assim 8 bytes de memória por variável, a quantidade de memória RAM necessária à definição das matrizes de entrada de dados e de seu posterior treinamento ficou muito elevada. Estudos a serem realizados com treinamentos envolvendo um número ainda maior de imagens ou imagens de maior resolução devem experimentar outras ferramentas que utilizem variáveis inteiras ou reais de precisão simples. Por exemplo, no experimento aqui realizado, a matriz de entrada de dados só continha valores zero ou um, mas tiveram de ser definidas como reais de dupla precisão pois o MATLAB não multiplica variáveis inteiras.

Além disso, as rotinas de detecção de bordas possuem parâmetros que regulam sua sensibilidade automaticamente ou manualmente. Portanto, estudos sobre a influência desta sensibilidade no desempenho do sistema podem levar a uma mais rápida convergência no treinamento da rede neural.

Anexo

Listagem das Rotinas elaboradas no MATLAB

I - Arquivo “testevisual.m”:

```
% Carregando a figura
[X,map]=bmpread('c:\tesebmp\000t1000.bmp');
I=ind2gray(X,map);
figure(1);
imshow(I,64)
pause
% Mostrando os métodos de detecção de bordas
BW1=edge(I,'roberts');
BW2=edge(I,'prewitt');
BW3=edge(I,'sobel');
BW4=edge(I,'log');
figure(2);
subplot(2,2,1);
imshow(BW1,2)
subplot(2,2,2);
imshow(BW2,2)
subplot(2,2,3);
imshow(BW3,2)
subplot(2,2,4);
imshow(BW4,2)
```

II - Arquivo “train400p1.m”:

```
datestr(datenum(now),0)
mm=repmat([0 1],12585,1);
% Inicializando a rede
[Wc1,Bc1,Wc2,Bc2,Wc3,Bc3]=initff(mm,20,'logsig',20,'logsig',1,'logsi
g');
clear mm;
% Treinando a rede
tp=[5 1000 0.01 0.01 1.1 0.5 0.9 1.0];
[Wc1,Bc1,Wc2,Bc2,Wc3,Bc3,te,tr]=trainbpx(Wc1,Bc1,'logsig',Wc2,Bc2,'l
ogsig',Wc3,Bc3,'logsig',Inp,Tar,tp);
datestr(datenum(now),0)
% Salva em um arquivo todas as variáveis atuais
save postrain400p1
```

III - Arquivo “def400p1.m” em forma resumida (o original tem 5214 linhas):

```
datestr(datenum(now),0)
nFig=0;
% Carregando a figura e reformatando
[X,map]=bmpread('c:\tesebmp\000000t1.bmp');
I=ind2gray(X,map);
J=edge(I,'log');
K=J(37:86,1);
for j=2:160,
  i1=round((205-j)/5.55);,
  i2=round((j+475)/5.55);,
  K=[K;J(i1:i2,j)];,
end
Inp=K;
Tar=0;
nFig=nFig+1
% Carregando a figura e reformatando
[X,map]=bmpread('c:\tesebmp\000000t2.bmp');
I=ind2gray(X,map);
J=edge(I,'log');
K=J(37:86,1);
for j=2:160,
  i1=round((205-j)/5.55);,
  i2=round((j+475)/5.55);,
  K=[K;J(i1:i2,j)];,
end
Inp=[Inp,K];
Tar=[Tar,0];
nFig=nFig+1
% Carregando a figura e reformatando
[X,map]=bmpread('c:\tesebmp\000000t3.bmp');
I=ind2gray(X,map);
J=edge(I,'log');
K=J(37:86,1);
for j=2:160,
  i1=round((205-j)/5.55);,
  i2=round((j+475)/5.55);,
  K=[K;J(i1:i2,j)];,
end
Inp=[Inp,K];
Tar=[Tar,0];
nFig=nFig+1

(...)

Inp=double(Inp);
Tar=Tar/10+.1;
clear X;
clear map;
clear I;
clear J;
clear K;
clear i1;
clear i2;
clear j;
clear nFig;
```

IV - Arquivo “sim400p4p1.m”:

```
% Simulação com fotos do lote de treinamento
[An1,An2,An3]=SIMUFF(Inp,Wc1,Bc1,'logsig',Wc2,Bc2,'logsig',Wc3,Bc3,'lo
gsig');
Erro=(Tar-An3)*10;
mErro=sqrt(Erro*Erro'/400)
% Simulação com fotos do lote de verificação
[An1p4,An2p4,An3p4]=SIMUFF(Inp4,Wc1,Bc1,'logsig',Wc2,Bc2,'logsig',Wc3,
Bc3,'logsig');
Erro4=(Tar4-An3p4)*10;
% Obtenção do erro absoluto distribuido em faixas
mErro4=sqrt(Erro4*Erro4'/400)
modErro4=Erro4.*Erro4;
mEd=[0 0 0 0 0 0 0];
for j=1:400,
  modErro4(j)=sqrt(modErro4(j));
  if modErro4(j)<0.25;,
    mEd(1)=mEd(1)+1;,
  elseif modErro4(j)<0.5;,
    mEd(2)=mEd(2)+1;,
  elseif modErro4(j)<1;,
    mEd(3)=mEd(3)+1;,
  elseif modErro4(j)<2;,
    mEd(4)=mEd(4)+1;,
  elseif modErro4(j)<3;,
    mEd(5)=mEd(5)+1;,
  elseif modErro4(j)<4;,
    mEd(6)=mEd(6)+1;,
  else mEd(7)=mEd(7)+1;,
  end,
end
mEd
```

Referências Bibliográficas

Brooks, M. J., 1978. *Rationalizing Edge Detectors*. pp. 277-285.

Churchland, P. S., 1986. *Neurophilosophy: Toward a Unified Science of the Mind/Brain*. Cambridge, MA: MIT Press.

Churchland, P. S., and T. J. Sejnowski, 1992. *The Computational Brain*. Cambridge, MA: MIT Press.

Eggermount, J. J., 1990. *The Correlative Brain: Theory and Experiment in Neural Interaction*. New York: Springer-Verlag.

Faggin, F., 1991. "VLSI implementation of neural networks," Tutorial Notes. *International Joint Conference on Neural Networks*, Seattle, WA.

Gose, E. E., J. W. Bacus, and L. Ackerman, 1971. "A comparison of some computer-measured and human-measured pattern recognition properties". *Journal of Cybernetics*, 1.

Gose, E. E., S. Werner, and R. G. Bickford, 1974. "Computadorized EEG spike detection". *Proceedings of the San Diego Biomedical Symposium*, February.

Gose, E., 1996. *Pattern recognition & image analysis*. Prentice Hall, Upper Saddle River, N.J..

Haykin, S., 1994. *Neural Networks: A Comprehensive Foundation.* New York: Macmillan Publishing.

Hebb, D. O., 1949. *The Organization of Behavior: A Neuropsychological Theory.* New York: Wiley.

Jain, A. K., 1989. *Fundamentals of Digital Image Processing*. Prentice Hall, Englewood Cliffs, N.J..

Krueger, W. M., and K. Phillips, 1989. *The Geometry of Differential Operators with Application to Image Processing*. No. 12, December.

Levine, M., 1985. *Man and Machine Vision*. New York: McGraw-Hill.

Marr, D., 1982. *Vision*. New York: Freeman.

MaCulloch, W. S., and W. Pitts, 1943. "A logical calculus of ideas immanent in nervous activity." *Bulletin of Mathematical Biophysics* **5**, 115-133.

Nguyen, D., and B. Widrow, 1989. "The truck backer-upper: An example of self-learning in neural networks." *International Joint Conference on Neural Networks*, Vol. 2, pp. 357-363, Washington, DC.

Pratt, W. K., 1978. *Digital Image Processing*, Wiley, New York.

Torre, V., and Poggio, T. A., 1986. *On Edge Detection*. No. 2, March, pp. 147-163.

Ramón y Cajál, S., 1911. *Histologie du systéme nerveux de l'homme et des vertébrés.* Paris: Maloine; Edition Francaise Revue: Tome I, 1952; Tome II, 1955; Madrid: Consejo Superior de Investigaciones Cientificas.

Rosenblatt, F., 1961. *Principles of Neurodynamics*, Washington D.C.: Spartan Press.

Shepherd, G. M., and C. Koch, 1990. "Introduction to synaptic circuits." In *The Synaptic Organization of the Brain* (G. M. Shepherd, ed.), pp 3-31. New York: Oxford University Press.

Suga, N., 1990a. "Cortical computational maps for auditory imaging." *Neural Networks* **3**, 3-21.

Suga, N., 1990b. "Computations of velocity and range in the bat auditory system for echo location." In *Computational Neuroscience* (E.L. Schwartz, ed.), pp. 213-231. Cambridge, MA: MIT Press.

Suga, N., 1990c. "Biosonar and neural computation in bats." *Scientific American* **262**(6), 60-68.

Widrow, B., and M.E. Hoff, Jr., 1960. "Adaptative swiching circuits." IRE WESCON *Convention Record*, pp. 96-104.