Universidade do Porto

Faculdade de Engenharia

FEUP

RECONHECIMENTO DE SINAIS DE TRÂNSITO UTILIZANDO REDES NEURONAIS ARTIFICIAIS

Relatório Final

Inteligência Artificial

3º ano de Mestrado Integrado em Engenharia Informática e Computação

Elementos do Grupo:

João Henriques, 110509026, joao.henriques@fe.up.pt

Wilson Pimentel, 090509052, pimentel.wilson@gmail.com

Porto, 20 de Maio de 2012

Índice

1. Objectivo

Este trabalho enquadra-se na realização dos objectivos pedagógicos da disciplina de Inteligência Artificial, do 3º ano, 2º semestre, do Mestrado Integrado em Engenharia Informática e de Computação da Faculdade de Engenharia da Universidade do Porto.

O projecto escolhido trata-se da elaboração de uma aplicação que permite o reconhecimento de sinais de trânsito através de Redes Neuronais Artificiais. O sinal é apresentado à rede sob a forma de um ficheiro de imagem, e consequentemente analisado e classificado com base em imagens conhecidas, utilizadas na fase de treino. A fase de treinos é conseguida através de um algoritmo de treino supervisionado, "*backpropagation*".

2. Descrição

2.1. Funcionalidades

O programa disponibiliza através da interface gráfica, a possibilidade de criação e treino de Redes Neuronais Artificiais com o intuito de serem utilizadas no reconhecimento de imagens, previamente conhecidas na fase de treinos. Estas redes podem ser guardadas num ficheiro e carregadas posteriormente para poderem ser utilizadas, ou mesmo para dar continuidade ao treino previamente iniciado.

Como métodos de treino, é oferecido ao utilizador, o treino normal, através de épocas, ou um treino inteligente. Em qualquer altura estes treinos podem ser interrompidos e recomeçados.

É também possível pedir para a aplicação escrever um relatório com o estado actual da rede para as imagens de teste, apresentado o valor de saída, bem como o erro obtido na última camada.

2.2. Estrutura do Programa

O projecto foi dividido em 3 módulos principais:

- Módulo da "Rede Neuronal", onde estão presentes todas as funcionalidades necessárias para a criação de uma rede neuronal artificial, multicamada e com ligações directas. É possível também propagar os valores da camada de entrada e corrigir a rede de forma a se adaptar aos valores esperados na camada de saída. Também fornece a estrutura básica para enviar os conjuntos de valores a serem utilizados no treino da mesma. Foi decidido isolar o módulo desta forma, tornando-o o mais genérico possível e permitindo que o mesmo possa ser utilizado para qualquer fim necessário, estando ou não relacionado com reconhecimento de imagens.

- Módulo de "Tratamento de Imagens", oferece as ferramentas necessárias para a extracção das "*features*" das imagens a serem utilizadas tanto na fase de treino como na fase de reconhecimento. Está disponível também uma classe que permite transformar o conjunto destas "features" no formato compatível com o esperado pelo módulo "Rede Neuronal" para efectuar o treino da rede.

- Módulo de "Interface", onde está presente toda a lógica associada à interface com o utilizador, fazendo uso dos dois módulos anteriormente descritos.

2.3. *Esquema de Representação do Conhecimento*

As Redes Neuronais Artificiais são extremamente úteis no reconhecimento de padrões, voz, sinais, classificação de forma, etc., ou seja, em situações em que não são possíveis utilizar os meios convencionais, sendo assim, o meio ideal para utilizar no reconhecimento de sinais de trânsito. Estas redes são baseadas no processo cognitivo do cérebro humano, que é constituindo por 10^{11} neurónios, capazes de comunicar entre si através da sinapse e efectuando decisões com base nos conhecimentos adquiridos. Este mesmo modelo foi adaptado de forma a poder ser simulado em computadores, e poder ser utilizada na Inteligência Artificial, sendo a sinapse feita tendo em conta um peso associado à ligação entre os neurónios.

Embora existam vários tipos de configurações possíveis diferentes de Redes Neuronais Artificiais, neste trabalho apenas são utilizadas redes multicamada com ligações directas aos neurónios da camada posterior. Não foi adoptada nem testada nenhuma técnica de "*brain damage*", onde poderiam haver neurónios que não tivessem qualquer ligação com os neurónios da camada seguinte.

Estas redes funcionam, colocando um conjunto de valores nos neurónios da primeira camada (a camada de entrada) e fazendo a sua propagação para os neurónios da camada seguinte, assim sucessivamente até se atingir a última camada. Cada neurónio calcula a sua saída, aplicando na função sigmóide o somatório do valor da saída dos neurónios da camada anterior multiplicados pelo peso da ligação associada ao neurónio corrente.

Para a aprendizagem de rede, foi utilizado o algoritmo supervisionado "backpropagation".

Este algoritmo actua após a propagação dos valores da camada de entrada terem ocorrido até à última camada. De seguida é calculado o erro da propagação nesta última camada, resultante do valor esperado pelo valor obtido em cada neurónio da camada. Este erro é de seguida propagado para as camadas intermédias, onde são calculados os novos erros dos seus neurónios. Este cálculo é feito através do produto da derivada (da função sigmóide) do valor de saída no neurónio corrente pelo somatório de todas as ligações do mesmo neurónio para os neurónios da camada seguinte, multiplicados pelo erro obtido no neurónio de destino. Este processo é repetido para todos os neurónios de todas as camadas intermédias. Por fim, os pesos das ligações da rede são reajustados com base nos nestes erros obtidos.

De forma à rede poder aprender novos dados, e mesmo assim não esquecer os anteriores, é utilizado um valor (constante) como taxa de aprendizagem, em que cada peso da rede, apenas varia uma pequena quantidade do valor obtido.

Relativamente à implementação, é importante focar que as redes criadas pelo programa têm sempre 19 neurónios na camada de entrada e o número de tipos de imagens diferentes como o número de neurónios da camada de saída. O número de camadas intermédias, bem como a quantidade de neurónios presentes em cada uma, é decidido pelo utilizador.

A explicação quanto aos 19 neurónios de entrada deve-se ao facto da ferramenta que extrai as features, classificar em cada imagem 8 + 8 + 3 features diferentes. A primeira fase parte por dividir a imagem em 8 tiras horizontais, e 8 tiras verticais, somando o valor numéricos da cor de cada pixel da mesma tira, e dividindo pelo valor total possível de se obter. Desta forma são calculados factores que podem ser utilizados para comparar com outras imagens. Por fim, é calculado o factor da quantidade de vermelho, verde e azul presente na imagem.

Para treinar a rede, é possível utilizar o clássico método de treino por épocas, onde é apresentada à rede todo o conjunto das imagens, um número definido de vezes (épocas).

Foi implementado como método adicional, um treino mais inteligente, onde tenta treinar a rede até todos os outputs estiverem como eram esperados, e todos os erros serem superiores a -0.5, e inferiores a 0.5. Caso seja utilizado este método, nada impede que a rede continue a ser treinada com o método normal.

É importante focar, que dependendo das imagens de entrada, bem como número de neurónios na(s) camada(s) intermédia(s) podem fazer este método de treino nunca terminar, significando que com as imagens actuais, não é possível chegar a um estado óptimo para todas as imagens, e sendo necessário ou modificar o número de neurónios na(s) camada(s) intermédia(s), ou trocar a imagem que não permite à rede estabilizar para outra ordem.

O programa espera que as imagens se encontrem com o nome num formato próprio, de forma a poderem ser dispostas em conjuntos de uma imagem por cada tipo. Isto permite que a rede não seja treinada continuamente com várias imagens do mesmo tipo, separando-as de forma igual pelo teste. O formato do nome é o seguinte "#_$$$$$[_default].ext", sendo o # um número em comum com as outras imagens que partilham do mesmo tipo, e caso desejemos especificar uma imagem padrão, que será apresentada na utilização do programa quando a mesma for reconhecida, deve ser colocado "_default" antes da extensão do ficheiro.

2.4. Análise dos Algoritmos usados

Os principais algoritmos utilizados no programa foram:

- Propagação dos valores: O(a*b*c).
- "Retro propagação" do erro: O(2*a*b*c)
- Treino da rede: O(x*y*3*a*b*c)

Sendo "a" o número de camadas, "b" o número de neurónios da camada e "c" o número de neurónios da camada seguinte, "x" o número de épocas de treino e "y" o número de imagens de treino.

2.5. Ambiente de Desenvolvimento

A linguagem de programação escolhida foi o JAVA, uma linguagem de alto nível, mas extremamente eficiente e capaz de lidar com o universo de situações que lhe possam colocar.

Na máquina de desenvolvimento foi utilizado o sistema operativo Windows 7 64 bit, mas este não foi um factor decisivo para a conclusão do projecto, uma vez que o JAVA corre numa máquina virtual, que independentemente do sistema operativo o resultado final deverá ser sempre o mesmo.

O IDE utilizado para desenvolver o projecto foi o eclipse para Java, versão Indigo.

Para elaborar a interface gráfica em SWING com o utilizador, foi usado como recurso adicional, o "Window Builder", uma ferramenta disponível como plug-in para o eclipse, relevando-se extremamente útil para o planeamento do layout de cada "janela", e funcionando à base de drag-and-drop dos componentes.

Não foi utilizada nenhuma biblioteca, além do SWING para java, na elaboração do projecto, tendo todo o trabalho presente sido desenvolvido de raiz.

2.6. ***Avaliação do Programa***

O programa avalia as imagens com base nas imagens previamente conhecidas na fase de testes e apresenta ao utilizador as duas imagens com maior probabilidade de se parecerem com a imagem de entrada.

O programa foi testado com uma camada intermédia de 8 neurónios, sendo treinada apenas com o método inteligente, com o total de 72 imagens diferentes, para 24 tipos de sinais de trânsito. Os resultados para imagens novas, adquiridas da internet foram extremamente favoráveis, tendo o reconhecimento tido uma grande percentagem de sucesso, por volta de umas 7/10.

3. Conclusões

Com a elaboração e conclusão deste trabalho, foi-nos possível aprender sobre a finalidade e como utilizar as redes neuronais artificiais, tendo-se relevado num excelente e eficaz meio para resolver problemas em que os meios convencionais não se aplicam. A fase mais complicada do trabalho passou pela recolha das imagens, bem como não haver uma fórmula exacta que nos diga o número exacto de camadas necessárias para cada situação específica, nem o número de testes exactos necessários. Foi necessário testar diferentes configurações até tentarmos achar à ideal para as imagens que tínhamos

presentes. Foi possível concluir também que a ordem em que as imagens são apresentadas à rede na fase de treino, modifica por completo o resultado final da rede.

O programa poderia ser melhorado com um método mais aperfeiçoada da extracção das *features* de cada imagem, que infelizmente devido ao tempo que tínhamos disponível para o projecto não nos permitiu aprofundar demasiado na pesquisa e implementação mais correcta do mesmo. Pensamos que se tivéssemos algumas noções de Computação por Visão, seria claramente uma mais-valia para a elaboração deste algoritmo, mas uma vez, que esta mesma área já saía um pouco do programa da disciplina achamos mais correcto dedicarmos o tempo na implementação dos algoritmos das redes neuronais.

Um ponto que achamos importante de focar sobre a utilização do programa, passa por o mesmo ter sido exclusivamente testado e treinado com o tipo de imagens a que o trabalho se destinava, sinais de trânsito. Estas imagens foram sempre apresentadas com o sinal centrado, e com o fundo branco. Não foi estudado nem é sabido qual será a resposta da rede a imagens que não respeitem este modelo. De forma a estas imagens poderem ser utilizadas com o estado actual de implementação dos algoritmos presentes no trabalho, seria necessário as mesmas passarem por uma fase de pré processamento em quer seriam eliminadas as áreas desnecessárias, e com o sinal centrado.

4. Recursos

4.1 Software utilizado

- *Eclipse*, versão Indigo, http://www.eclipse.org/
- *Window Builder,* Plugin para Eclipse versão Indigo, https://developers.google.com/java-dev-tools/download-wbpro

2.7. Bibliografia

- Slides das aulas, presentes na página da disciplina, http://paginas.fe.up.pt/~eol/IA/1112/ia_.html
- "*Artificial Intelligence: A modern approach*". S.Russel, P.Norvig, Prentice-Hall, 3rd Ed., 2010
- http://en.wikipedia.org/wiki/Backpropagation

- http://takinginitiative.net/2008/04/23/basic-neural-network-tutorial-c-implementation-and-source-code/

5. Anexos

5.1 Manual do Utilizador

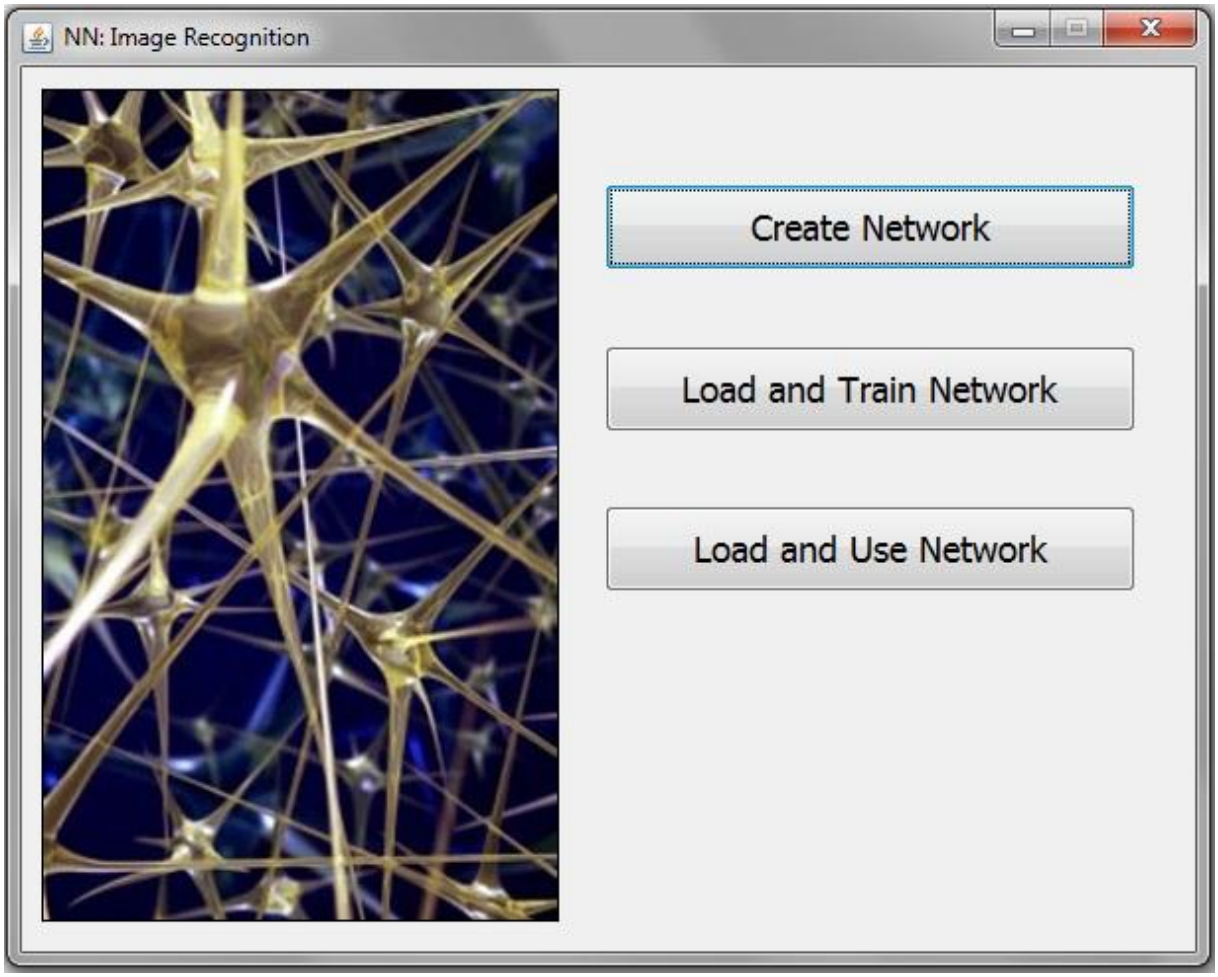

Fig1. Janela Principal

É possível escolher nesta janela qual será a tarefa a desempenhar pelo programa.

- Criar uma rede, opção "Create Network":

Quando esta janela é aberta, é possível adicionar ou remover imagens, mostrando sempre o *preview* da imagem seleccionada.

É possível escolher a configuração da rede: o número de camadas intermédias e a localização do ficheiro para salvar a rede.

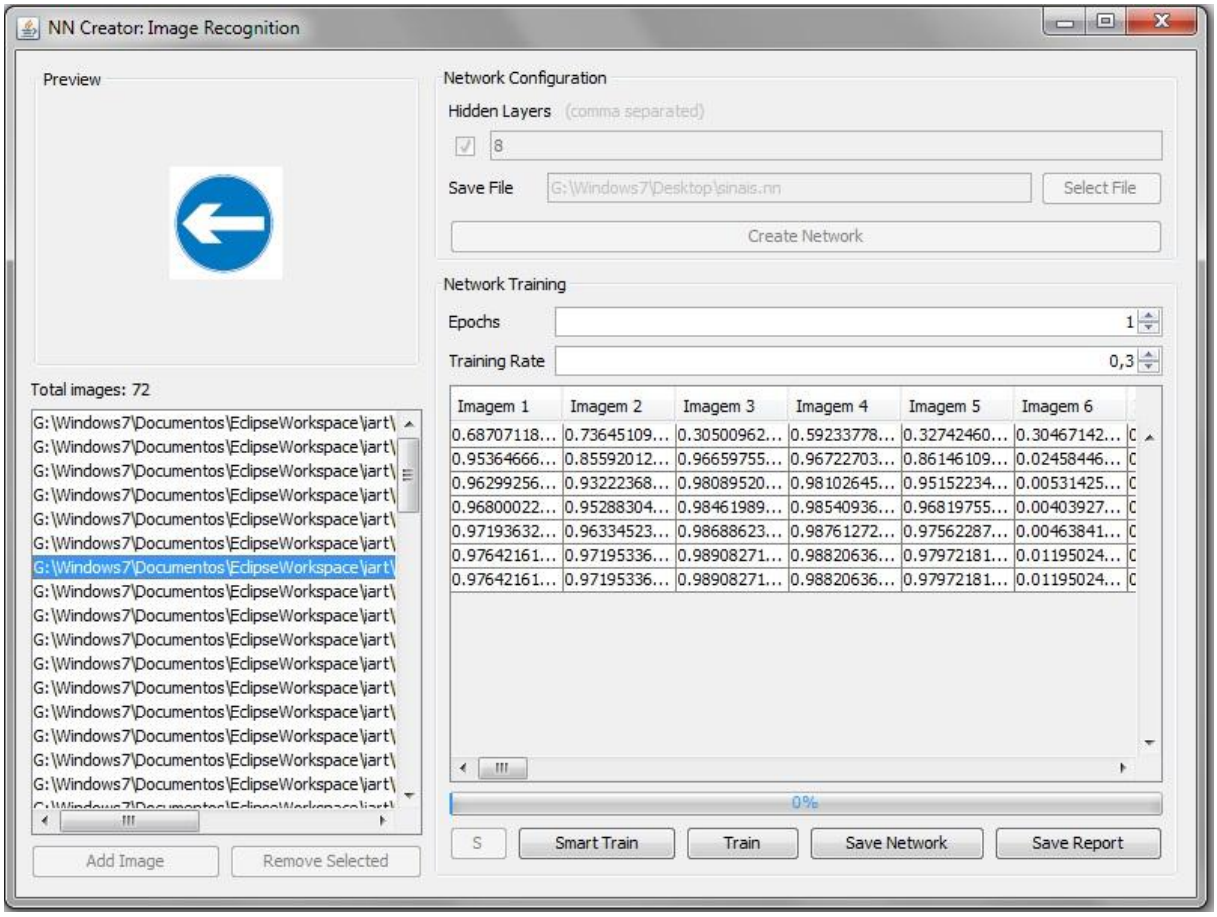

Fig 2: Criação/treino de uma rede

Para criar a rede basta clicar em "Create Network", passando o programa para o modo de treino, desactivando a adição/remoção de imagens e modificação das camadas intermédias. Neste modo, o painel de treino é activado, sendo possível seleccionar o número de épocas para o treino normal, bem como a taxa de aprendizagem global da rede.

Está disponível o botão "Smart Train" que activa o modo de treino inteligente, descrito anteriormente, e o botão "Train" que treina a rede, o número de épocas especificadas. Quando qualquer um dos treinos está a decorrer é possível carregar no botão "S", de forma a interromper o mesmo.

O botão "Save Network" pode ser utilizado para guardar o estado actual da rede e o "Save Report" guarda um relatório com o estado actual de reconhecimento, bem como o erro, para cada imagem do conjunto de testes.

Está presente também uma tabela com o output do estado actual da rede para cada imagem.

- <u>Treinar rede existente, opção "Load and Train Network":</u>

Aparece uma caixa para seleccionar a rede a carregar, e de seguida aparece o ecrã (Fig. 2), no modo de treino, como já descrito no ponto anterior.

- Utilizar rede existente, opção "Load and Use Network":

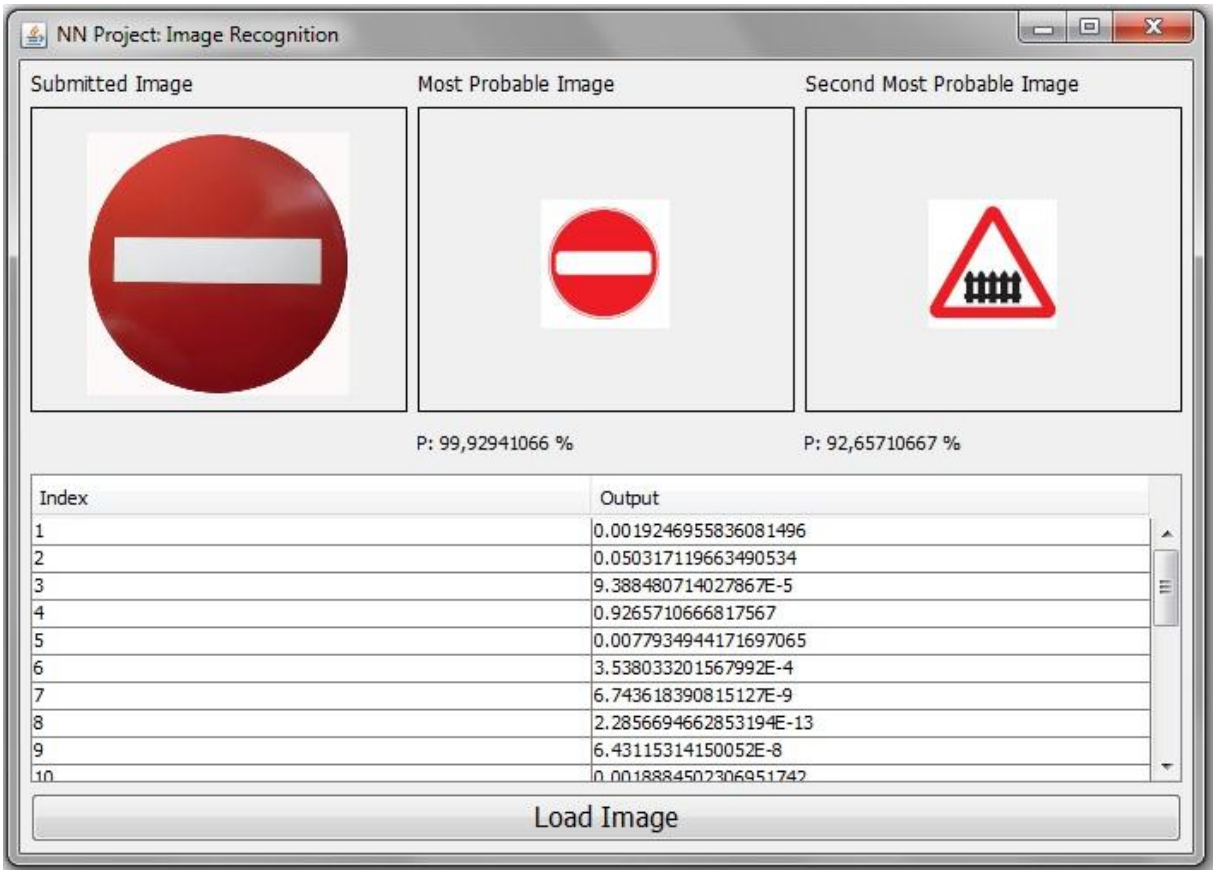

FIG 3: Utilização do programa

Quando carregada uma rede pronta a ser utilizada, é possível clicar em "Load Image" e seleccionar uma imagem para a rede testar se consegue identificar, ou então arrastar directamente para o quadrado de "Submitted Image", sendo a mesma carregada automaticamente.

Neste ecrã é apresentada a imagem carregada, bem como a imagem com maior e segunda maior taxa de reconhecimento.

É apresentado também, o output numa tabela mais abaixo que esta imagem gerou em todas as saídas da rede.

5.2 Exemplo de uma execução

Uma vez que já apresentamos algumas imagens no subcapítulo 5.1, que resultaram de um exemplo de execução, iremos simplesmente descrever os passos pela ordem correcta:

1) Carregar em "Create Network". (Fig 1).
2) Carregar em "Add Image", e seleccionar as 72 imagens dos 24 tipos de sinais diferentes, presentes na pasta "sinais". (Fig2)
3) Activar a *checkbox* por baixo da label "Hidden Layers" de modo a activar a caixa para inserir as camadas intermédias, e inserir o número 8. (É possível não utilizar qualquer camada intermédia, ou até inserir o número de neurónios de várias camadas intermédias, separados por vírgula).
4) Carregar em "Select File" e seleccionar a localização de onde o ficheiro com a rede deverá ser guardado.
5) Carregar em "Create Network"
6) Em seguida clicar em "Smart Train", para treinar a rede com o modo inteligente (Dependendo das imagens seleccionadas, bem como os neurónios e o número de camadas intermédias, este método de treino pode nunca terminar, como já explicado anteriormente).
7) Clicar em "Save Network" para salvar o estado actual da rede.
8) Retornar ao menu anterior, clicando no "X" da janela e seleccionar a opção "Load and Use Network", seleccionando o ficheiro em que a rede foi salva no ponto 7). (Fig 1)
9) Arrastar as imagens para o quadrado da imagem submetida, ou clicar em "Load Image" para escolher que imagem carregar. (Fig 3)