DEPARTAMENTO DE ENGENHARIA INFORMÁTICA E DE SISTEMAS INSTITUTO SUPERIOR DE ENGENHARIA DE COIMBRA

**Conhecimento e Raciocínio- 2018/2019**



**Trabalho Elaborado Por:**

João Pedro Correia Machado Coelho (Nº 21270909)

Vitor Hugo Silva Fernandes (Nº 21270524)

Índice:

[1) Introdução 2](#_Toc13267672)

[2) Considerações/Obstáculos Encontrados 3](#_Toc13267673)

[3) Testes Pedidos 3](#_Toc13267674)

[4) Testes Adicionais 10](#_Toc13267675)

[5) Interface 14](#_Toc13267676)

[6) Conclusão 16](#_Toc13267677)

# Introdução

Este trabalho foi desenvolvido como trabalho prático para a cadeira de Conhecimento e Raciocínio, cadeira integrante do curso de Engenharia Informática do Instituto Superior de Engenharia de Coimbra.

Com este trabalho pretendemos adquirir conhecimentos relativos as redes neuronais e ao seu funcionamento, utilizando o *Matlab.* Assim com este trabalho temos o objetivo de criar e treinar uma rede neuronal, que consiga fazer a deteção de quadrados, triângulos, estrelas e círculos.

# Considerações/Obstáculos Encontrados

Abaixo encontra-se descrito algumas considerações/obstáculos encontrados neste trabalho:

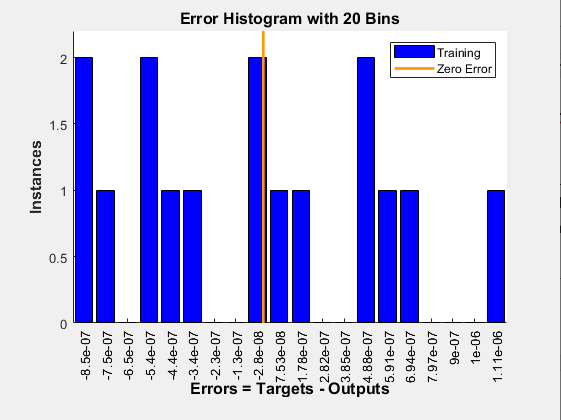
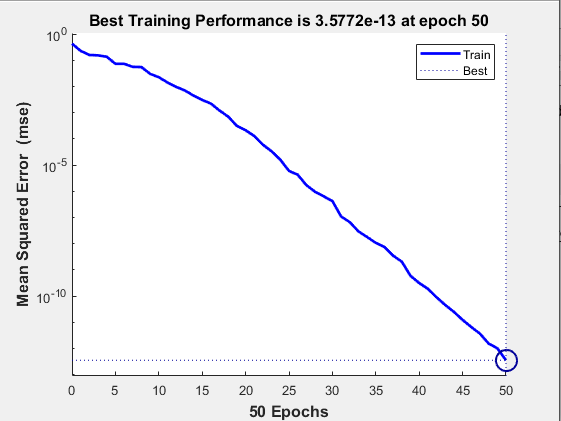
* **Necessidade de redimensionamento obrigatório:** Originalmente não utilizámos nenhum método de redimensionamento, no entanto isso faria que so fosse possível realizar a função de treino “trainscg”, por este ser mais leve, pois a função “recomendada” (“trainlm”) num conjunto grande de imagens de imagens pedia quantidades absurdas de memoria ram, dando erro. Assim pra combater esse problema decidimos realizar um redimensionamento com escala de 1:10, passando assim as imagens de 200x200 (com 40000 pixéis no total) para 20x20 (com 400 pixéis no total) que permite já a utilização de funções mais pesadas. Esta escala foi definida assim, porque como neste trabalho realizamos treinos de RN com imagens pouco complexas, estas imagens, mesmo com uma resolução baixa, permitem manter a sua formas, não afetando os resultados, como demostrado nos testes adicionais.
* **Utilização de binarização nas imagens:** Neste trabalho, nós optamos por binarizar as imagens as imagens (usando a função “imbinarize”). Esta decisão foi realizada pois as imagens fornecidas são a “preto e branco” sendo fácil assim a tradução das mesmas para uma matriz binaria (por estas terem um contraste bem definido, por serem preto e branco, imagens a core poderia dar o problema com certas cores), e também para não existir a necessidade de realizar uma normalização, que aumentava o tempo necessário para correr o código.

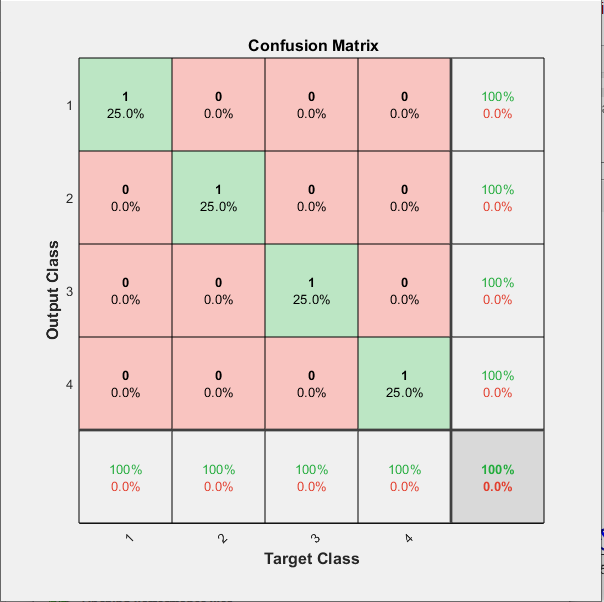
# Testes Pedidos

Neste tópico iremos realizar os testes pedidos no enunciado do trabalho, sendo estes testes descritos abaixo:

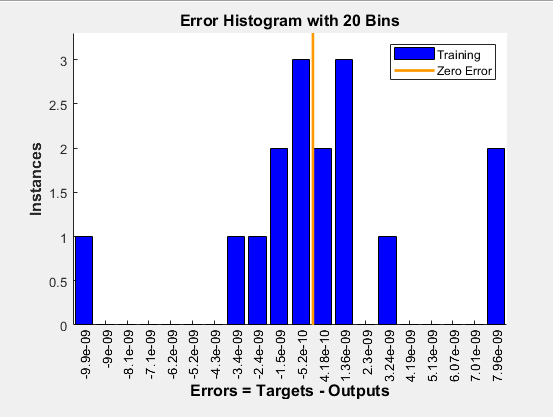
1. **Teste de diferentes configurações numa rede de 10 neurónios utilizando “Formas\_1”:**

Após criar uma rede com base em treinos nas imagens da pasta “Formas\_1” e se iniciar o treino com base nestas pode-se concluir que a melhor performance de treino é de 3.577e-13 na época 50, este treino teve um total de 50 épocas. Após a simulação, foi-se verificar a matriz de confusão que deu 100% positiva e acertou em todas as formas dadas pelo utilizador. As configurações usadas para esta função foram como função de treino trainscg e como função de ativação da rede foi tansig.



Outra configuração que pode ser dada a este teste é usar a função de treino “trainlm” com a função de ativação de “tansig”, que assim como a outra configuração feita acerta em todas as imagens, esta configuração tem como melhor prestação no treino 1.6942e-17 na época 5, esta fez um total de 5 épocas.



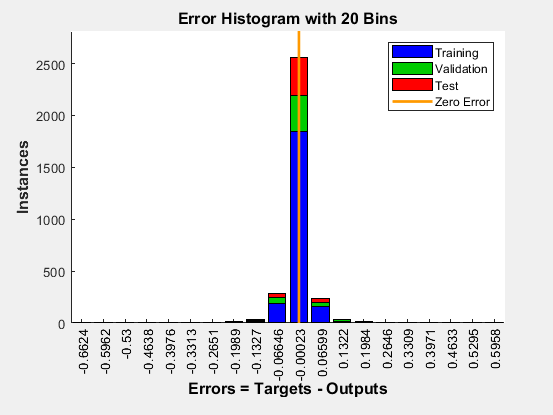
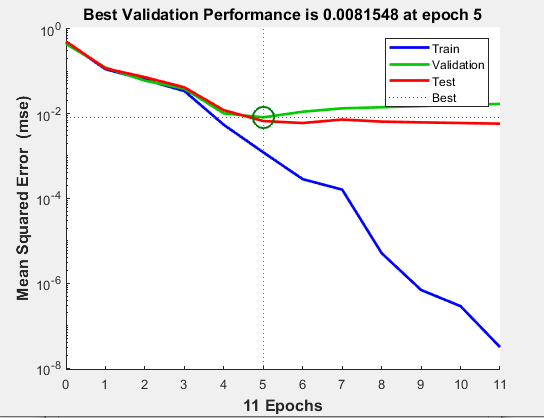
Outra configuração que pode ser dada é alterar a função de ativação para “purelin” e verifica-se que a melhor prestação num total de 4 épocas foi 2.822e-22 na 4ª época.

Com isto podemos concluir que a melhor configuração destas foi a última, pois para além de ter conseguido menos épocas, conseguiu um valor de prestação melhor (mais baixo) que os outros. No entanto, todas as configurações conseguiram fazer uma boa prestação tendo todas concluído o objetivo pretendido de acertar com todas as formas.

1. **Treino de RN utilizando “Formas\_2” com segmentação de dataset:**

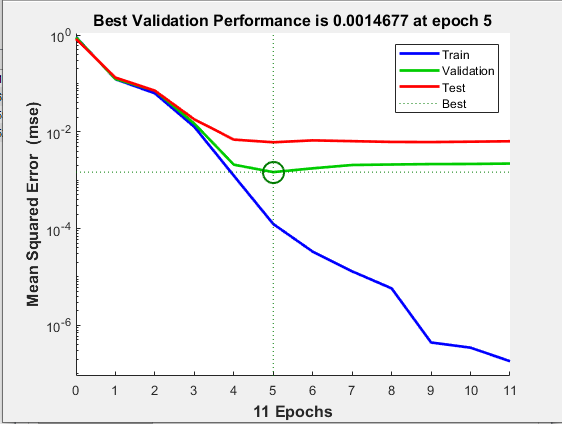
Para estes treinos irá ser usada a mesma configuração de funções de treino e de funções de ativação sendo que se irá apenas alterar os valores do dataset para verificar quais as principais alterações de desempenho que se verificam na prestação da análise da imagem e também irá ser utilizado as imagens da pasta Formas\_2 para testes em vez de Formas\_1. A configuração será a seguinte como função de treino “trainlm”, a rede terá apenas 1 layer e essa layer terá como função de ativação “tansig”.

A segmentação inicial do dataset será 70% para treinos, 15% para validações e de15% para testes.

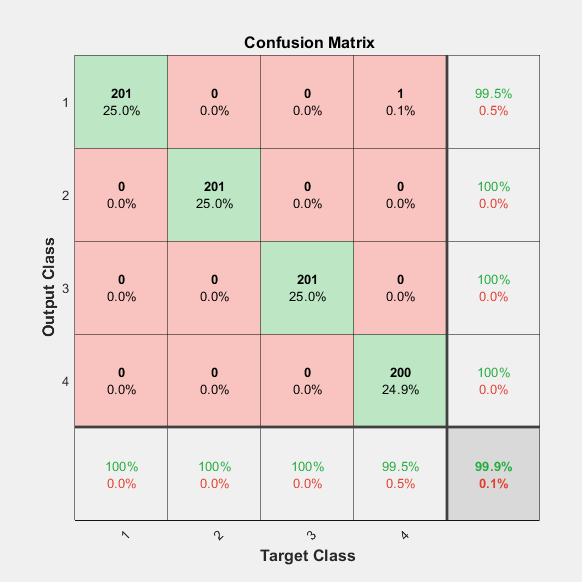


Com esta configuração de segmentação obteu-se um como melhor desempenho 0.0081548 na 5ª epoch apos 11 iterações.

Outra configuração da segmentação do dataset foram as de 80% para treinos, 10% para validações e de 10% para testes ao qual se obteu um desempenho de 0.0014677 na 5ªa epoch após 11 iterações e também a configuração de segmentação de dataset de 50% para treinos, 25% para validações e 25% para testes ao qual se obteu um desempenho 0.007835 na 7ª epoch após 13 iterações, tendo estas todos um gráfico de erro de aspeto semelhante.



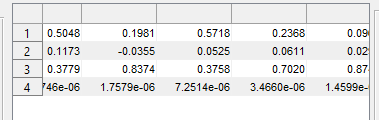
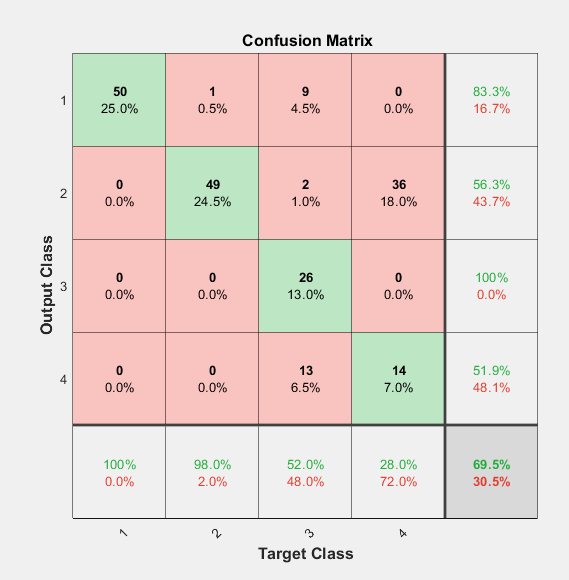
Em relação há matriz de confusão, apos correr a rede neuronal e comparando com as suas próprias imagens, podemos confirmar que este conseguiu reconhecer quase todas a imagens, falhando só numa, tendo assim uma percentagem de precisão de 99.9%. Com a realização de mais testes deparamo-nos que a percentagem de precisão varia entre 98 e 100%.



Com este estudo pode-se concluir que com um número grande de “amostras” existem pequenas variações nos valores do resultado entre os vários métodos testados, no entanto consideramos que a melhor configuração será a utilização de um número de treino alto de forma a rede neuronal ter amostras suficientes para os testes. Assim, utilizamos uma configuração de 70% para treinos, 15% para validações e de 15% para testes e após alguns treinos guardamos a melhor rede neuronal no ficheiro melhor.mat.

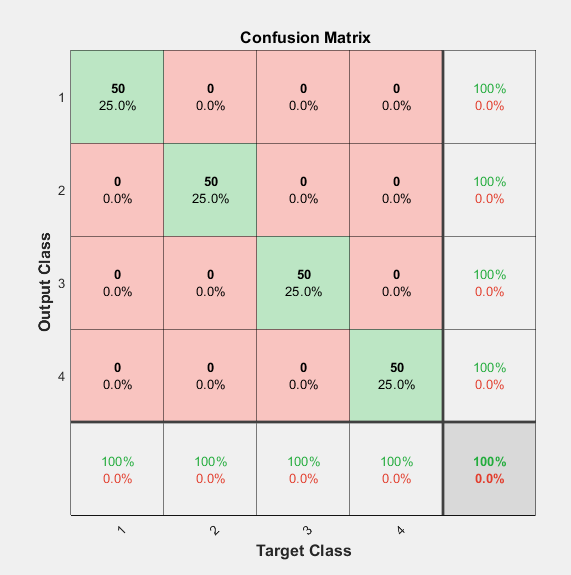
1. **Simulação/treino de rede neuronal usando “Formas\_3”:**

Para a análise da “Formas\_3” primeiro iremos usar a rede neuronal melhor.mat guardada no tópico anterior. Assim os resultados obtidos são os seguintes:

Assim, como podemos constatar pela matriz de confusão, este teste teve uma precisão de 69.5% tendo a rede neuronal errado um maior número de formas geométricas as formas correspondentes as estrelas e círculos (a direita encontra-se a tabela de saída equivalente a formas de estrela).

De seguida, iremos voltar a treinar a rede neuronal, desta vez utilizando a pasta Formas\_3, um redimensionamento das formas de 1:10, utilizando 1 layer com “tansig”, e a função de treino “trainlm” e como segmentação do dataset 70% nos treinos, 15% nas validações e 15% e 10 neurónios. Assim, apo correr a rede neuronal, comparando com as Formas\_3 obtivemos:



Assim, podemos concluir, que para a rede neuronal ter um resultado 100% correto, precisa de ser treinada com os mesmo objetos / objetos muito semelhantes. Assim, a precisão foi menor no teste com o melhor.mat pois a rede neuronal foi treinada para as imagens da pasta Formas\_2, e o teste é realizado com as Formas\_3, enquanto a precisão do teste da Formas\_3 (2º teste) foi de 100%, pois a rede neuronal foi treinada pelos objetos da pasta Formas\_3.

1. **Teste de rede neuronal utilizando imagem de utilizador:**

Para cumprir com a análise da pergunta desenhamos 4 imagens (utilizando o programa gimp), que estão guardadas na pasta ImagensLeitura. Assim, utilizando a rede neuronal criada no tópico anterior, iremos preceder ao treino. Abaixo encontram-se as imagens e o seu respetivo output no teste, tirado da tabela da interface grafica:

|  |  |
| --- | --- |
| Imagem | Tabela de dados |
|  |  |
|  |  |
|  |  |
|  |  |

*Legenda da tabela de dados: 1- Círculos, 2- Quadrados, 3- Estrelas, 4- Triangulos*

Assim, podemos concluir, que na simulação foi possível verificar qual a forma geométrica de todas as imagens. É possível chegar a essa conclusão observando a tabela de dados de cada imagem e, sabendo que a opção escolhida pela rede neuronal é o maior numero da tabela, podemos conferir que este rede neuronal conseguiu determinar a forma de um objeto , mostrando com 100% de certeza o objeto correto (poderia acontecer a rede neuronal “responder” que um objeto é 20% triangulo e 80% quadrado, por exemplo).

# Testes Adicionais

Neste tópico iremos mostrar os testes adicionais realizados, e a conclusão dos resultados testados. De forma a simplificar a leitura, cada teste terá uma pequena conclusão do mesmo:

* **Teste 1:** Este teste tem como objetivo verificar qual a variação de resultados utilizando diferentes números de layers, funções de layers, e diferentes números de neurónios. Neste teste foi utilizado: “Formas\_3” para treino, 1/10 de redimensionação, “trainscg” com função de treino e segmentação de dataset com os valores 70,15,15 (treino,Validação e testes, respetivamente). A tabela representa nas linhas os tipos de pre-conf. De layers, na vertical o número de neurónios e cada segmento tem a performance e a epoch:

|  |  |  |
| --- | --- | --- |
|  | nº de neurónios = 10 | nº de neurónios = 1000 |
| 1layer - tansig | 2,9577e-06, epoch 249 | 2,0217, epoch 30 |
| 1layer - logsig | 0,00022801, epoch 70 | 0,32511, epoch 38 |
| 1layer - purelin | 0,013695, epoch 44 | 39,4067, epoch 63 |
| 3layers - tansig | 0,0000096358, epoch 250 | 0,99294, epoch 26 |
| 3layers - todos | 0,0011865, epoch 36 | 82,1944, epoch 24 |

Com estes testes concluímos:

* + O número de neurónios utilizado na rede neuronal tem um efeito na rapidez do treino, no entanto, neste teste, a performance piorou com o nº de neurónios.
  + Diferentes funções de layers (e o número delas), altera os valores da rede neuronal, concluindo que a melhor função será a “tansig” tanto com 1 ou com 3 layers.
* **Teste 2:** Este teste tem como objetivo verificar se o redimensionamento das imagens tem um efeito negativo na RN. Neste teste foi utilizado: “Formas\_3” para treino, “trainscg” com função de treino e segmentação de dataset com os valores 70,15,15 (treino,Validação e testes, respetivamente) e 1layer de “transig”. A tabela representa nas linhas as imagens utilizadas, nas colunas o tipo de redimensionamento e cada segmento tem a performance e a epoch:

|  |  |  |
| --- | --- | --- |
|  | 1/1 de redimensiona | 1/10 de redimensiona |
| Formas\_1 | 1,8614, epoch 0 | 0,14638, epoch 1 |
| Formas\_2 | 0,0053148, epoch 111 | 0,00028963, epoch 125 |
| Formas\_3 | 0,013666, epoch 74 | 0,0044459, epoch 38 |

Com estes testes concluímos:

* + O redimensionamento das imagens não tem um efeito negativo na RN, sendo que os resultados estão todos dentro da margem de erro.
  + Que a utilização de formas 1, nos 2 teste, teve um resultado anormal. Concluímos que este resultado é devido à utilização de segmentação de dataset pois, no teste a do 3º tópico os resultados foram muito diferentes. Assim, também concluímos que a utilização de segmentação de dataset numa baixa quantidade de imagens é prejudicial a RN, sendo recomendado que não se utilizando, utilizando assim todas a imagens para treino, e não parte delas.
* **Teste 3:** Este teste tem como objetivo de comparar os diferentes tipos de funções de treino e como este afetam a RN. Neste teste foi utilizado: “trainlm” com função de treino, 1/10 de redimensionação, segmentação de dataset com os valores 70,15,15 (treino,Validação e testes, respetivamente) e 1layer de “transig”. A tabela representa nas linhas as Funções utilizadas, nas colunas tem o tipo de redimensionamento e cada segmento tem a performance e a epoch :

|  |  |
| --- | --- |
|  | nº de neurónios = 10 |
| trainlm | 4,3821e-05, epoch 9 |
| trainscg | 0,000043384, epoch 250 |
| trainbfg | 1,1076e-05, epoch 87 |
| trainbr | 8,1011e-14, epoch 49 |
| trainc | 0,9, epoch 0 |
| traincgb | 0,020524, epoch 15 |
| traincgf | 0,011844, epoch 20 |
| trains | 0,93372, epoch 0 |
| traingd | 0,10387, epoch 250 |
| traingda | 0,014097, epoch 137 |
| traingdm | 0,074447, epoch 250 |
| traingdx | 0,0062548, epoch 141 |
| trainoss | 6,2411e-05, epoch 64 |
| trainr | 0,001694, epoch 250 |
| trainrp | 0,00061552, epoch 22 |

Com estes testes concluímos:

* + Com este estudos nós confirmamos que as melhores funções são: trainlm, trainbr,trainoss,trainbfg.
* **Teste 4:** Este teste tem como objetivo de comparar os efeitos que diferentes segmentações do dataset têm na RN. Neste teste foi utilizado: “Formas\_3” para treino, “trainlm” com função de treino , 10 neurónios e 1layer de “transig”. A tabela representa nas linhas as imagens utilizadas, nas colunas os valores de Treino,validação e teste, respetivamente, e cada segmento tem a performance e a epoch:

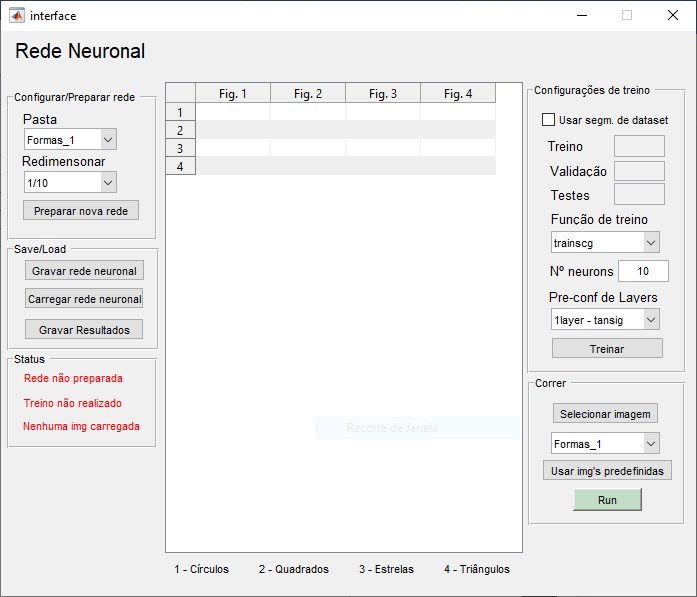
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 80,10,10 | 60,20,20 | 50,25,25 | 34,33,33 |
| Formas\_1 | 9,0928E-20, epoch 7 | 1,0798, epoch 3 | 0,52918, epoch 0 | 0,057884, epoch 1 |
| Formas\_2 | 0,0036298, epoch 5 | 0,0043503, epoch 5 | 0,011049, epoch 7 | 0,0071792, epoch 5 |
| Formas\_3 | 0,00017002, epoch 10 | 0,0093915, epoch 6 | 0,0024221, epoch 10 | 0,0009646, epoch 15 |

Com estes testes concluímos:

* + Os valores recomendados a utilizar na segmentação, são o, de neste caso, 80/10/10. Assim concluímos que se obtém melhores resultados quando se usa um maior número de figuras para treino do que os outros parâmetros, sendo os outros também necessários, mas em quantidade menor. Também tivemos confirmação da conclusão obtida no teste 2, que a utilização de segmentação em números pequenos de figuras tem um impacto negativo. Assim os valores recomendados serão 80/10/10 ou 75/10/10 para amostras de imagens maiores que 50, e utilização de 100% das imagens pra treino para amostras menores que 50.

# Interface

Neste tópico iremos mostrar de forma breve o funcionamento da interface:



* **Configurar/Preparar Rede**
  + **Pastas**: Permite escolher qual das pastas/figuras serão utilizadas no treino
  + **Redimensionar**: Permite escolher entre utilizar o tamanho original das imagens (que devido a limitações de hardware, pode limitar o número de funções de treino) ou fazer o redimensionamento das imagens numa relação 1:10 (10% do tamanho original), que permite a utilização de todas as funções (explicado melhor no 2 tópico).
  + **Preparar nova rede**: Procede a preparação das imagens para a utilização no treino da rede neuronal.
* **Save/Load**
  + **Gravar rede neuronal:** Permite guardar uma rede neuronal, já treinada, num ficheiro .mat.
  + **Carregar rede neuronal:** Permite carregar uma rede neuronal já guardada. Após o carregamento esta rede já estará treinada, sendo que se realizar o treino outra vez, esse irá fazer override ao treino carregado.
  + **Gravar Resultados:** Exporta para um ficheiro Excel a informação sobre os resultados da simulação realizada.
* **Status:** Mostra de forma fácil as ações já realizadas, sendo recomendado, durante o uso da aplicação, o utilizador seguir a ordem de ações demonstrada.
* **Tabela Central:** Mostra a informação sobre a última simulação realizada.
* **Configuração de treino**
  + **Usar segm. de dataset:** Permite escolher se pretende utilizar segmentação ou não, sendo “desbloqueado” as opções de treino, validação e testes, se este estiver ativo
  + **Treino / Validação / Testes:** Se a opção de segmentação estiver ativa, o utilizador escrever a percentagem de treino, validação e testes , respetivamente, não podendo a soma conjunta destes valores ser maior que 100%.
  + **Função de treino:** Permite escolher qual das funções de treino utilizar no treino
  + **Pre-cong de Layers:** Permite escolher, de entre varias pré-configurações, as configurações aplicadas as Layres, como o numero das mesmas e as funções utilizadas por estas
  + **Treinar:** Procede ao treino da rede neuronal com os parâmetros selecionados.
* **Correr**
  + **Selecionar Imagem:** Permite ao utilizador utilizar uma imagem guardada previamente para a simulação
  + **Usar img´s predefinidas:** Permite a utilização de imagens já pré-configuradas (escolhidas no pop-up menu acima) na simulação de RN.
  + **Run:** Corre a simulação/teste da rede neuronal com as imagens definidas, mostrando depois os resultados na tabela central e numa matriz de confusão.

# Conclusão

Com este trabalho podemos concluir que redes neuronais são algo que tem uma potencialidade muito grande para a análise de padrões com base nuns outros valores (valores de entrada) que são fundamentais para que o estudo das redes seja suficiente para efetuar as tarefas de análise a que estas são submetidas. Também pode ser concluído que as redes neuronais têm um potencial muito grande a nível de processamento lógico, mas este processamento envolve muitos recursos e para isto quem as prepara tem de ter o cuidado de gerir recursos de modo a não afetar a qualidade de análise até que esta seja demasiado corrompida. Bons métodos que podem resolver este problema de memória é conversões de tamanho no caso de imagens, ou a utilização de funções de baixo uso de memória (“trainscg”). Também podemos concluir, que o melhor método a aplicar a uma rede neuronal é o “trainlm“ , o melhor método para os layers (“tansig”) entre outras conclusões.

Em suma uma rede neuronal é um poderoso modelo computacional que pode ajudar em diversas atividades, estas têm de ser tratadas de modo a que o seu funcionamento não coloque em causa ou o desempenho de uma máquina, pois estas consomem bastantes recursos, e não coloque em causa o seu processamento lógico.