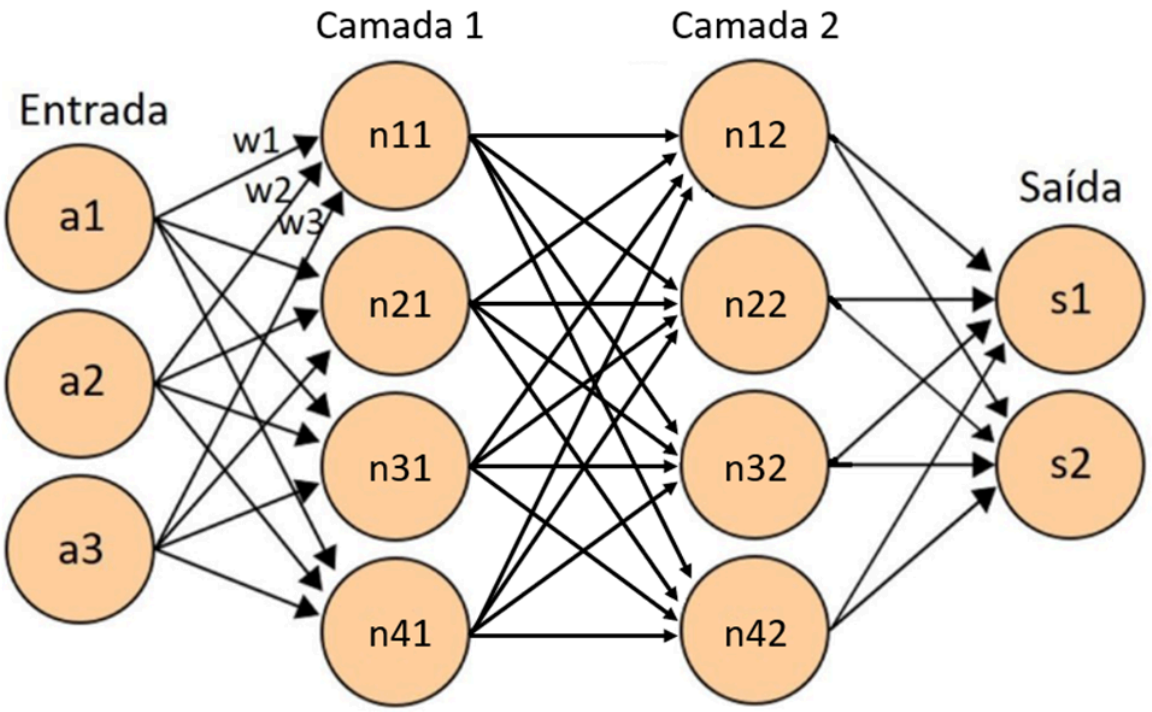
Instituto Superior de Engenharia de Coimbra

Conhecimento e Raciocínio



Trabalho Prático

Estudo de Redes Neuronais Feedforward

Licenciatura em Engenharia Informática

2024 / 2025

Nuno Tomás Paiva
a2023131763@isec.pt

Rui Martins dos Santos
a2023145822@isec.pt

ÍNDICE

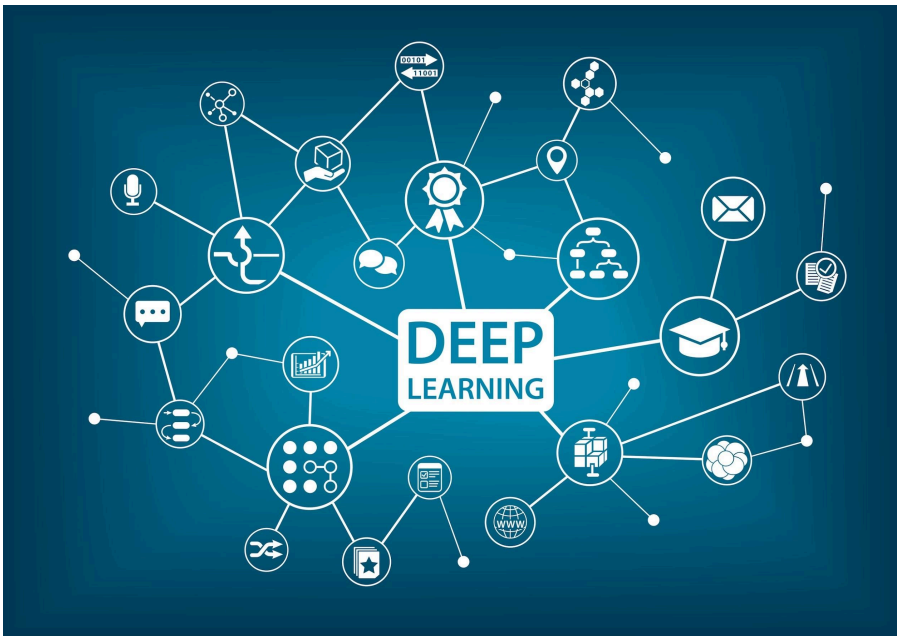
1. Introdução

O presente trabalho prático foi desenvolvido no âmbito da unidade curricular de Conhecimento e Raciocínio, tendo como principal objectivo explorar e aprofundar os conceitos associados às redes neuronais *feedforward*, recorrendo ao ambiente de desenvolvimento MATLAB e à toolbox de Deep Learning.

Neste projecto, o foco incide na aplicação de redes *feedforward* para a classificação de formas geométricas a partir de imagens binárias, representando seis classes distintas: círculo, papagaio (kite), paralelogramo, quadrado, trapézio e triângulo.

O trabalho foi desenvolvido em várias fases experimentais, onde se testaram diferentes configurações de redes neuronais ao variar-se a topologia (número de camadas e neurônios), as funções de ativação, os algoritmos de treino e os rácios de divisão de treino, validação e teste. Foi igualmente desenvolvida uma aplicação gráfica que permite configurar, treinar e aplicar redes de forma simples e intuitiva.

Com este projecto pretendeu-se não só implementar redes com bom desempenho, mas também compreender a influência dos diferentes parâmetros no comportamento e na capacidade de raciocínio das redes neuronais.



2. Alínea A) - Start

Na alínea A), utilizou-se a pasta start, contendo apenas 5 imagens de cada forma geométrica para treino. Inicialmente, as imagens foram convertidas em matrizes binárias de 25×25 pixels.

Na alínea ii) é nos pedido para usar todos os exemplos para treino, e então, a divisão automática dos dados em conjuntos de treino, validação e teste tornou-se irrelevante, uma vez que todas as amostras foram utilizadas exclusivamente para treinamento.

Alínea A) - Mudar topologia									
Conf1	1	10	tansig, tansig, purelin	trainlm	100% Treino	20		100.00	NaN
Conf2	1	30	tansig, tansig, purelin	trainlm	100% Treino	20		100.00	NaN
Conf3	2	10, 10	tansig, tansig, purelin	trainlm	100% Treino	20		100.00	NaN
Conf4	2	30, 30	tansig, tansig, purelin	trainlm	100% Treino	20		100.00	NaN
Conf5	2	10, 30	tansig, tansig, purelin	trainlm	100% Treino	20		100.00	NaN

Como era de esperar obteve-se sempre 100% de treino e não houve nenhuma precisão de teste, pois não usamos nenhuma percentagem nos parâmetros de divisão para o teste.

Na alínea iii) já não nos restringe a 100% de uso de todos os exemplos para treino para podermos fazer a comparação.

Alínea A) iii. - Mudar topologia										
Conf1	1	10	tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	20		80.00% (±6.67)	40.00% (±0.00)	40.00%
Conf2	1	30	tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	20		79.33% (±3.65)	28.00% (±22.80)	60.00%
Conf3	2	10, 10	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	20		66.00% (±14.79)	28.00% (±17.89)	40.00%
Conf4	2	30, 30	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	20		78.67% (±1.83)	32.00% (±10.95)	40.00%
Conf5	2	10, 30	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	20		76.00% (±6.83)	32.00% (±30.33)	80.00%

Como era de esperar já que agora estamos a usar 15% (valor default) para os exemplos para teste, já conseguimos obter uma percentagem para o teste, mas em comparação à percentagem global fica um pouco abaixo da alínea anterior.

Em termos de topologia, foi possível observar que o aumento do número de camadas escondidas tende a piorar o desempenho da rede, especialmente no caso da pasta Start, que contém um número muito reduzido de imagens por classe.

Por outro lado, aumentar o número de neurónios por camada geralmente contribui para uma melhoria nos resultados, até certo ponto. No entanto, quando o número de neurónios é excessivo, a rede torna-se propensa a overfitting, ou seja, ajusta-se demasiado aos dados de treino e perde capacidade de generalização para novos dados.

3. Alínea B) - Train

Nesta alínea vamos comparar várias configurações de rede com diferentes mudanças em todos os parâmetros e verificar o seu desempenho, deixando os outros valores default.

Ao utilizar **diferentes topologias** na pasta *Train*, verificou-se que, ao contrário do que acontece na pasta *Start*, o aumento do número de camadas e de neurónios não teve um impacto negativo nos resultados. No entanto, também não se registou uma melhoria significativa no desempenho. Isto poderá indicar que, apesar de haver mais imagens por classe, a complexidade adicional da rede não traz benefícios claros na tarefa de classificação das formas geométricas.

Alínea B) - diferentes topologias: número de neurónios e número de camadas - Train									
Conf1	1	10	tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	20	90.33% (±2.32)	68.44% (±10.93)	86.67%
Conf2	1	30	tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	20	90.47% (±0.80)	63.11% (±6.96)	73.33%
Conf3	2	10, 10	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	20	89.53% (±2.12)	70.67% (±4.82)	77.78%
Conf4	2	30, 30	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	20	90.87% (±1.97)	68.89% (±6.85)	80.00%
Conf5	2	10, 30	tansig, tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	20	89.20% (±2.58)	60.44% (±9.08)	75.56%

Ao utilizar diferentes **funções de ativação**, não se verificaram diferenças significativas na melhor repetição de teste. No entanto, observou-se uma discrepância acentuada na precisão global e na média de teste, com as funções *logsig* e *tansig* a apresentarem um desempenho substancialmente superior em comparação com *purelin* e *softmax*.

As funções logsig e tansig são mais eficazes neste contexto porque introduzem não-linearidade e produzem saídas compatíveis com a codificação one-hot, essencial para classificação. Em contraste, purelin (linear) é mais adequada

para regressão, enquanto softmax pode destabilizar o treino quando combinada com o algoritmo trainlm.

Alínea B) - diferentes funções de ativação - Train									
Conf1	1	10	logsig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	20	90.47% (±2.91)	72.89% (±3.98)	77.78%
Conf2	1	10	tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	20	91.33% (±2.35)	69.33% (±6.74)	77.78%
Conf3	1	10	purelin, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	20	71.07% (±24.29)	57.33% (±15.51)	77.78%
Conf4	1	10	softmax, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	20	65.80% (±20.21)	52.44% (±16.67)	73.33%

Nesta alteração das **funções de treino**, observamos que a função **'trainlm'** se destaca em relação às demais, apresentando desempenhos significativamente superiores tanto nos resultados globais quanto nos de teste.

Vale destacar, contudo, que a função **'trainlm'** também foi a que apresentou o maior tempo de execução, sendo consideravelmente mais lenta do que as demais. Esse maior tempo de processamento pode ter contribuído para o seu melhor desempenho, uma vez que permite uma otimização mais precisa durante o processo de aprendizagem.

Alínea B) - diferentes funções de treino - Train									
Conf1	1	10	tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	20	89.07% (±2.13)	68.44% (±3.65)	71.11%
Conf2	1	10	tansig, purelin	traingd	dividerand = {0.7, 0.15, 0.15}	20	21.40% (±5.02)	23.56% (±6.40)	33.33%
Conf3	1	10	tansig, purelin	trainscg	dividerand = {0.7, 0.15, 0.15}	20	46.60% (±3.52)	40.00% (±3.51)	44.44%
Conf4	1	10	tansig, purelin	trainrp	dividerand = {0.7, 0.15, 0.15}	20	25.13% (±7.73)	21.33% (±10.95)	35.56%

Ao alterarmos os valores dos **rácios de divisão**, observamos que os melhores resultados são obtidos quando atribuímos uma percentagem maior ao conjunto de treino.

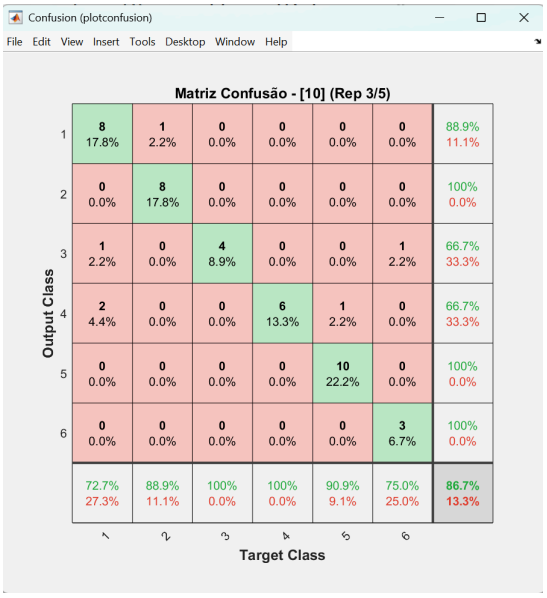
No entanto, é importante salientar que esta percentagem não pode ser de 100%, pois, nesse caso, não haveria dados reservados para validação e teste, o que resultaria em valores '**NaN**' e inviabilizaria a avaliação do desempenho da rede.

Alínea B) - diferentes rácios de divisão em treino/validação/teste - Train									
Conf1	1	10	tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	20	89.07% (±2.13)	68.44% (±3.65)	71.11%
Conf2	1	10	tansig, purelin	trainlm	dividerand = {0.8, 0.1, 0.1}	20	93.67% (±1.13)	74.67% (±7.30)	80.00%
Conf3	1	10	tansig, purelin	trainlm	dividerand = {0.6, 0.2, 0.2}	20	86.00% (±1.75)	64.33% (±5.35)	68.33%
Conf4	1	10	tansig, purelin	trainlm	dividerand = {0.4, 0.3, 0.3}	20	76.53% (±1.35)	61.56% (±4.94)	68.89%

Nesta alínea é nos pedido também para gravar as 3 melhores redes neuronais. As três melhores redes neuronais foram:

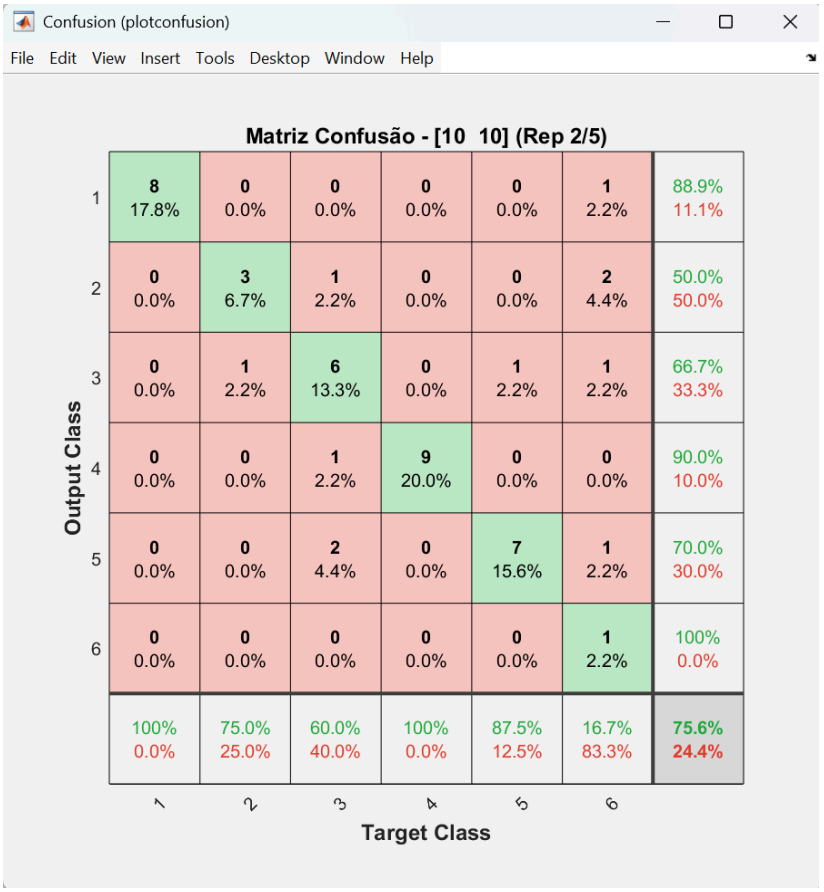
- alineaB_diferentTopo_Conf1.mat - Conf1 das diferentes topologias com 86.67% de teste:
  - número de camadas escondidas: 1
  - número de neurónios: 10
  - funções de ativação: tansig, purelin
  - função de treino: trainlm
  - divisão dos exemplos: dividerand = {0.7, 0.15, 0.15}
  - número de épocas: 20

Matriz de confusão:



- alineaB_diferentTopo_Conf3.mat - Conf3 das diferentes topologias com 77.78% de teste:
    - número de camadas escondidas: 2
    - número de neurónios: 10 10
    - funções de ativação: tansig, tansig, purelin
    - função de treino: trainlm
    - divisão dos exemplos: dividerand = {0.7, 0.15, 0.15}
    - número de épocas: 20

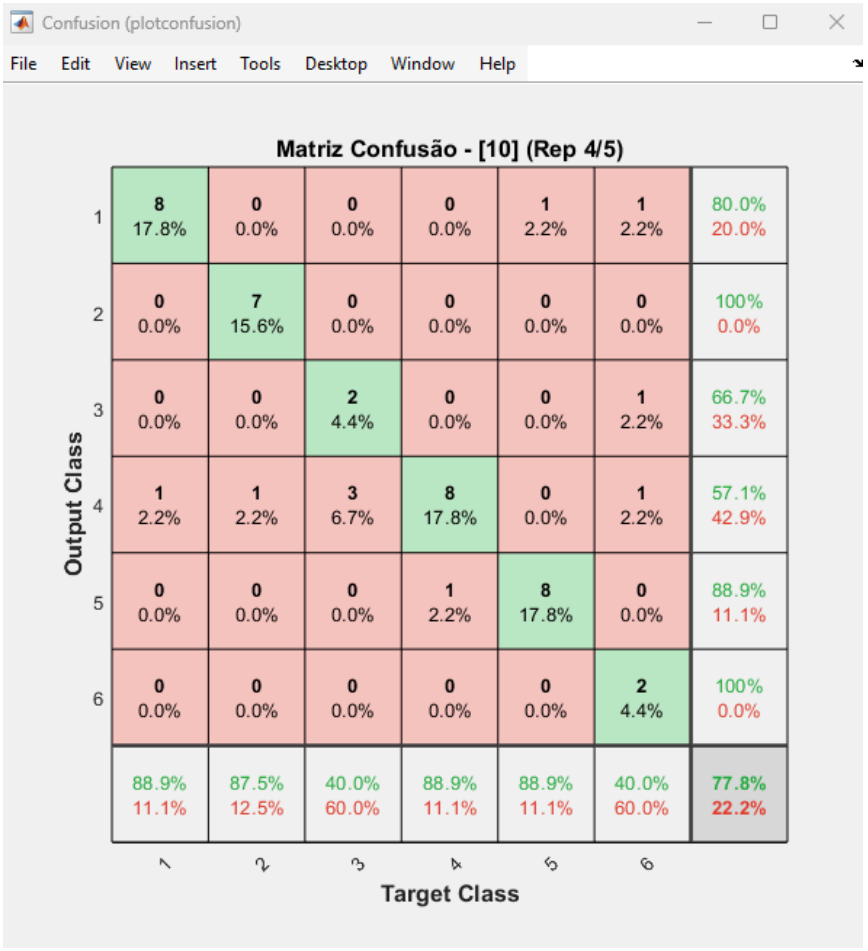
Matriz de confusão:



- alineaB_diferentFuncAtiv_Conf2.mat - Conf2 das diferentes funções de ativação com 77.78% de teste:
    - número de camadas escondidas: 1

- número de neurónios: 10
- funções de ativação: purelin, purelin
- função de treino: trainlm
- divisão dos exemplos: dividerand = {0.8, 0.1, 0.1}
- número de épocas: 20

Matriz de confusão:

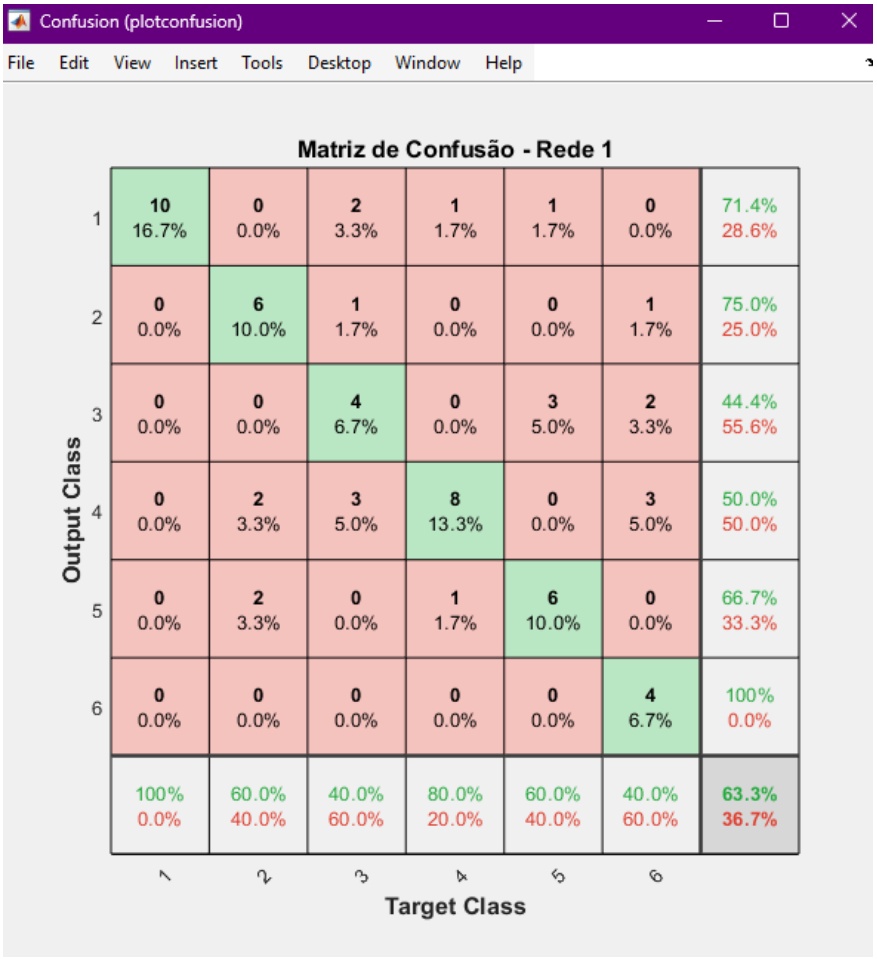


4. Alínea C) - Start, Train, Test

Nesta alínea i), é solicitado que utilizemos as melhores redes previamente guardadas na alínea anterior. Sem treinar as redes, devemos compará-las com os desempenhos obtidos na alínea b). Para essa comparação, as imagens a serem

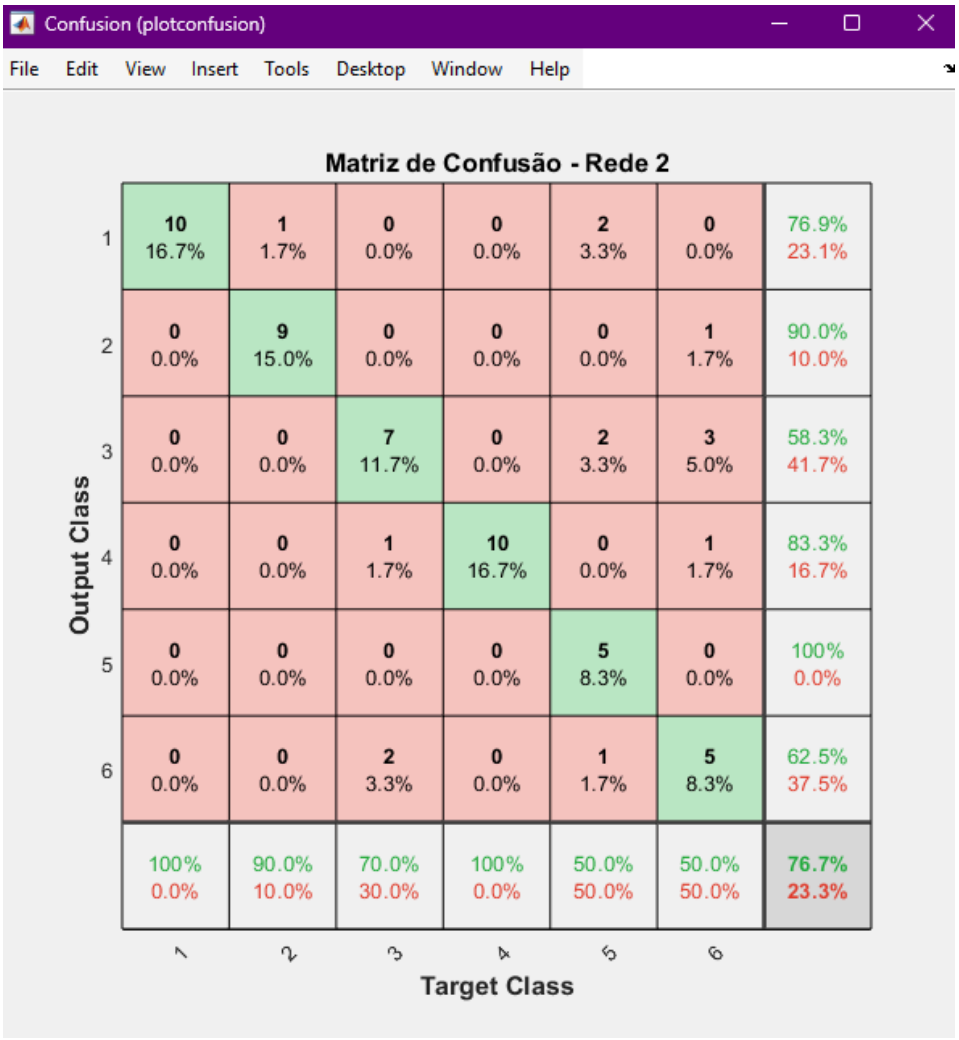
utilizadas encontram-se na pasta 'test', sendo importante garantir que as redes carregadas mantenham os pesos e parâmetros previamente ajustados, de forma a permitir uma avaliação justa e objetiva da sua capacidade de generalização.

Com a rede: *alineaB_diferentRatios_Conf3.mat* e a seguinte matriz de confusão:



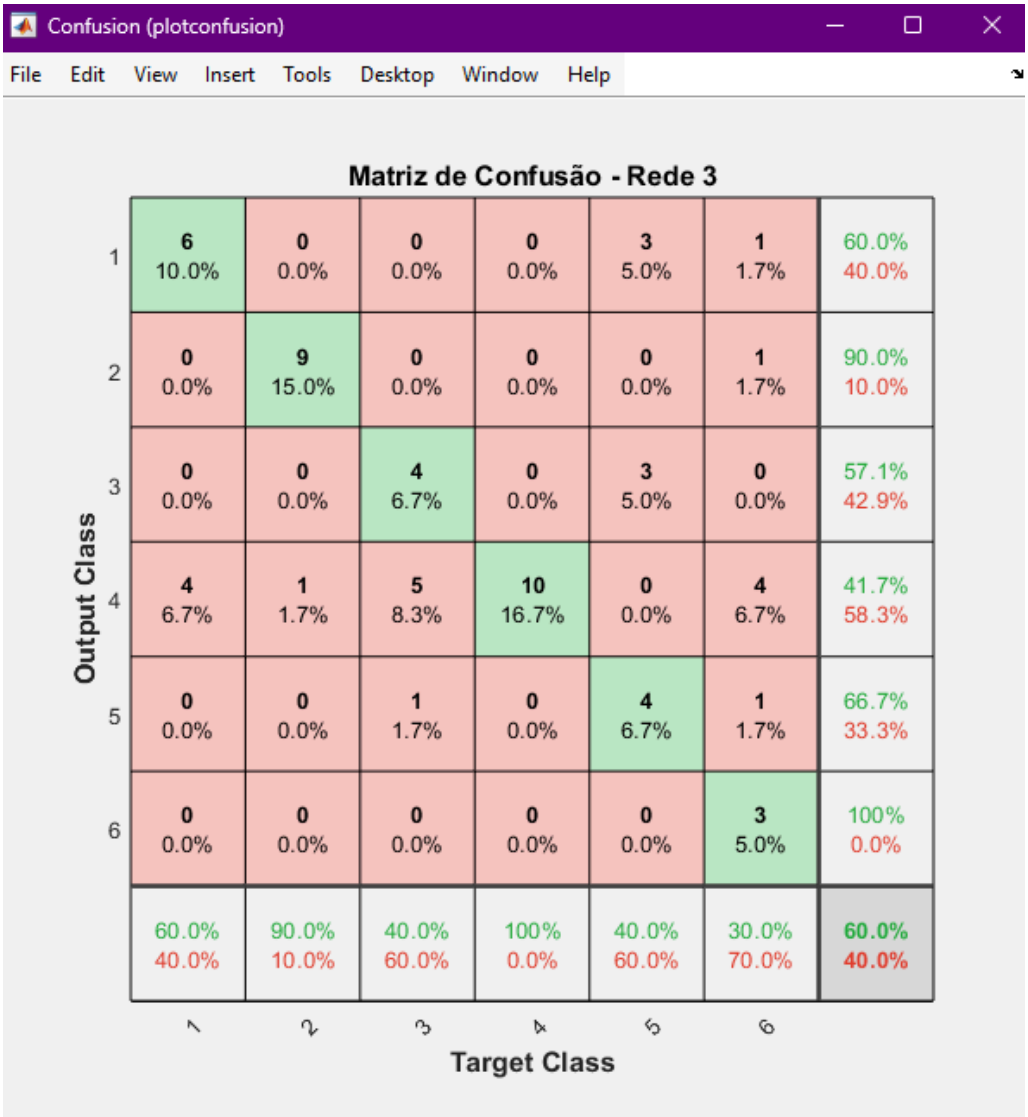
Obteve-se uma precisão de teste de 63.33%, que foi inferior à precisão de teste de 68.33% da alínea b).

Com a rede: *alineaB_diferentTopo_Conf1.mat* e a seguinte matriz de confusão:



Obteve-se uma precisão de teste de 76.67%, que foi inferior à precisão de teste de 86.67% da alínea b).

Com a rede: *alineaB_diferentFuncAtiv_Conf3.mat* e a seguinte matriz de confusão:



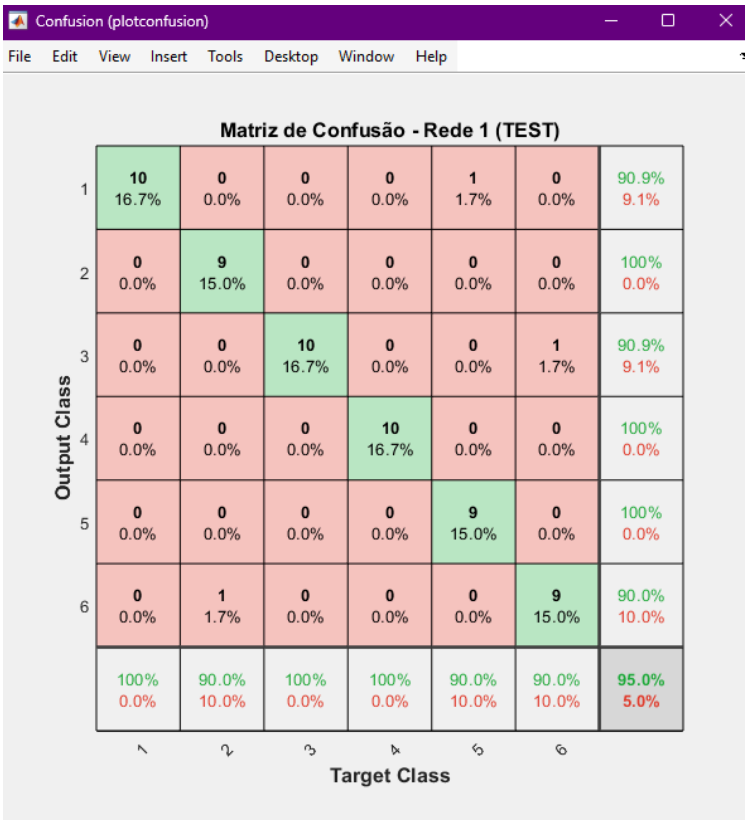
Obteve-se uma precisão de teste de 60.00%, que foi inferior à precisão de teste de 77.78% da alínea b).

Na **alínea ii)**, o objetivo é treinar e testar novamente as melhores redes identificadas na alínea b), desta vez utilizando exclusivamente as imagens contidas na pasta 'test'.

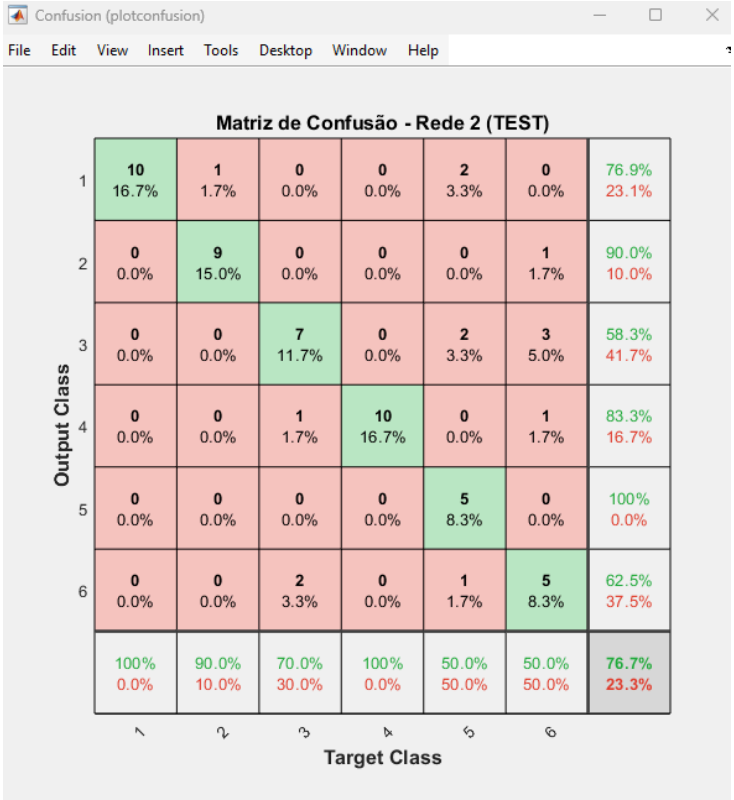
Após o processo de treino, as redes devem ser utilizadas para classificar as imagens presentes em cada uma das seguintes pastas: 'start', 'train' e 'test'. Em seguida, deve-se registar as precisões de teste obtidas na classificação de cada

conjunto, permitindo assim uma análise comparativa do desempenho das redes em diferentes subconjuntos de dados.

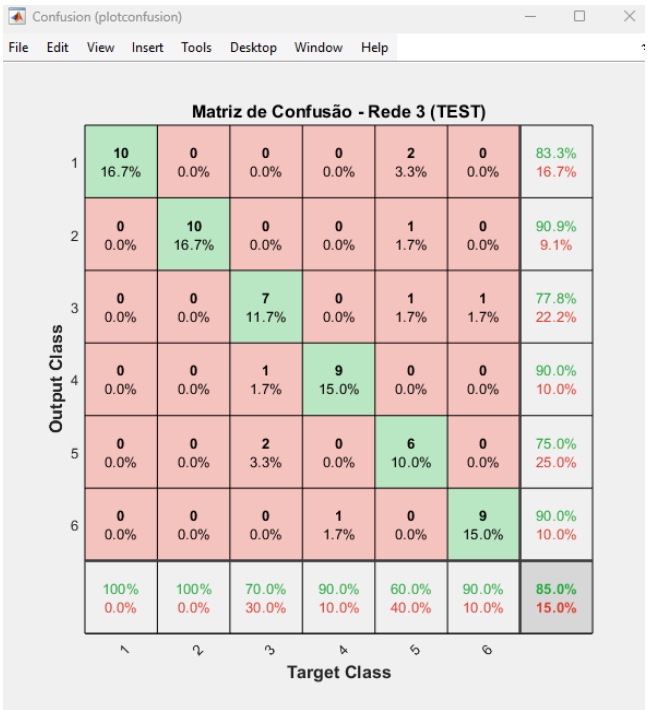
A **Rede 1**, carregada a partir do ficheiro *alineaB_diferentRatios_Conf3.mat*, obteve uma precisão de 53.33% para as imagens da pasta start, 69.33% para a pasta train e 95.00% para a pasta test, destacando-se com um desempenho muito elevado no conjunto de teste, embora com desempenhos mais baixos nos restantes conjuntos.



A **Rede 2**, proveniente de *alineaB_diferentTopo_Conf1.mat*, apresentou uma precisão de 73.33% na pasta start, 93.67% na train e 76.67% na test, mostrando um desempenho equilibrado com melhor generalização sobre os dados de treino.



Já a **Rede 3**, carregada de *alineaB_diferentFuncAtiv_Conf3.mat*, alcançou 66.67% na start, 82.33% na train e 85.00% na test, revelando um comportamento intermédio entre as duas anteriores.



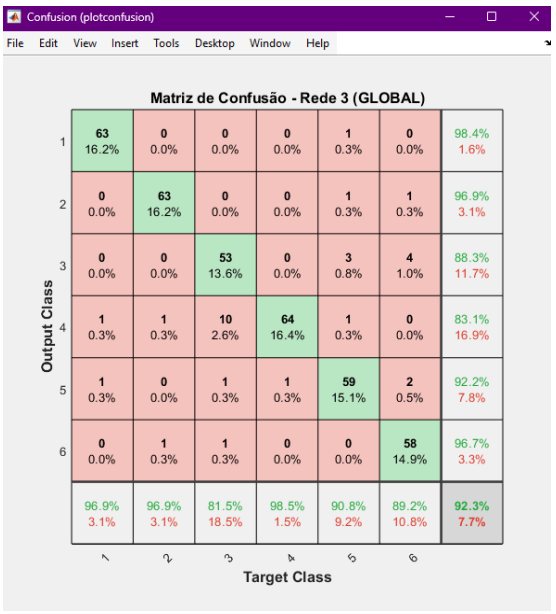
A análise das matrizes de confusão guardadas para cada rede permitiu verificar que a **Rede 1** teve a melhor precisão de acerto na pasta test, indicando uma adaptação mais eficaz aos dados com que foi treinada, enquanto apresentou menor capacidade de generalização.

A **Rede 2** demonstrou uma elevada precisão na pasta train e um desempenho relativamente equilibrado nos outros conjuntos, sugerindo maior robustez.

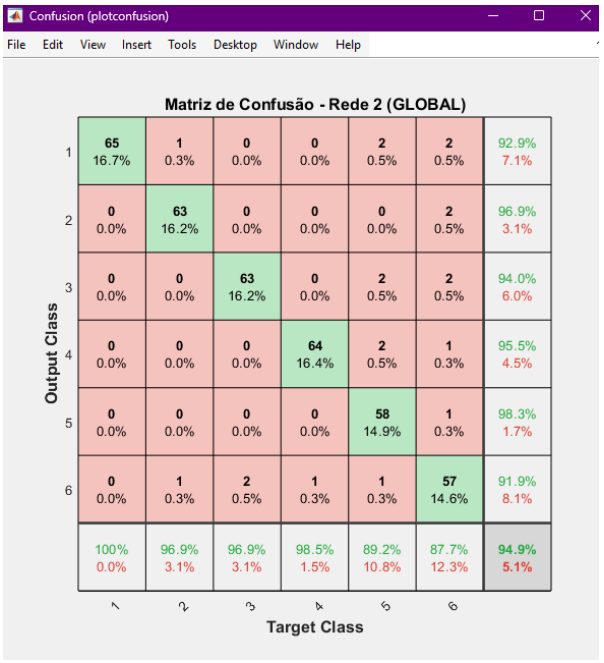
Na alínea iii), foi realizado o treino e teste das melhores redes identificadas na alínea b), agora utilizando a totalidade das imagens disponíveis, ou seja, combinando os conjuntos das pastas start, train e test.

A Rede 1, guardada com o nome "rede_top_92_172919.mat", revelou um desempenho bastante equilibrado e robusto. Com uma precisão global de 92.31%, destacou-se pela sua consistência nos diferentes conjuntos de dados: alcançou 93.33% de precisão na pasta *start*, 92.33% na pasta *train* e 91.67% na pasta *test*.

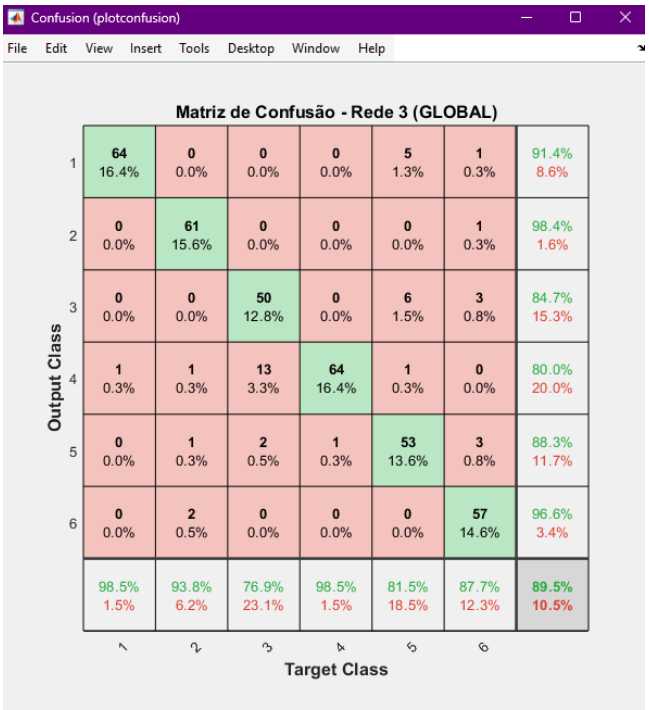
Estes resultados demonstram que a rede foi capaz de aprender eficazmente os padrões das diferentes formas geométricas, mantendo uma forte capacidade de generalização. O seu desempenho consistente nos três conjuntos reforça a fiabilidade da rede e confirma-a como uma das melhores soluções obtidas ao longo deste estudo.



A **Rede 2**, armazenada como *rede_top_85_171052.mat*, destacou-se com a melhor precisão global de 94.87%, alcançando 90.00% na start, 97.33% na train e 85.00% na test, demonstrando forte capacidade de aprendizagem e generalização.



Já a **Rede 3**, salva como *rede_top_83_171120.mat*, obteve uma precisão global de 89.49%, com 73.33% na start, 92.33% na train e 83.33% na test, mantendo um desempenho estável e próximo ao da Rede 2, embora ligeiramente inferior.



A análise dos resultados mostra que a Rede 2 apresenta o melhor compromisso entre precisão e capacidade de generalização, destacando-se com a melhor precisão global entre as três. Esta rede demonstrou um excelente desempenho na pasta *train* (97.33%) e manteve resultados consistentes nas pastas *start* e *test*, o que indica uma forte capacidade de aprendizagem sem comprometer a generalização.

Já a Rede 3, apesar de apresentar uma precisão global ligeiramente inferior, manteve um desempenho estável nos diferentes conjuntos, sendo também uma solução fiável. Estes dados evidenciam a importância de uma configuração equilibrada da rede e da utilização de um conjunto de dados diversificado durante o treino, garantindo um desempenho sólido em diferentes contextos de classificação.

6. Alínea D)

Nesta alínea, pretende-se avaliar a capacidade de generalização das redes neuronais treinadas através da classificação de imagens desenhadas manualmente que estão na pasta *draw*. Para isso, foram criadas 5 imagens de cada categoria (círculo, papagaio, paralelogramo, quadrado, trapézio e triângulo), com características visuais semelhantes às utilizadas no treino da rede. Estas imagens foram posteriormente convertidas para matrizes binárias, de forma a poderem ser processadas pelas redes. Não desenvolvemos nenhum programa para carregar estas imagens e classificá-las recorrendo às melhores redes obtidas na alínea c iv), usamos a aplicação gráfica para tal.

O objetivo é verificar se as redes conseguem reconhecer corretamente formas novas, desenhadas fora do conjunto original de dados, analisando os resultados obtidos e retirando conclusões sobre o desempenho da rede em dados reais. Usamos apenas a seguinte rede: *rede_top_92_172919.mat*.

Circle:

- **Imagem 0:** Acertou (Confiança: 59.87%).
- **Imagem 1:** Não acertou, previu uma classe de trapézio (Confiança: 57.79%).
- **Imagem 2:** Não acertou, previu uma classe de triângulo (Confiança: 36.33%).

- **Imagem 3:** Não acertou, previu uma classe de trapézio (Confiança: 94.11%).
- **Imagem 4:** Não acertou, previu uma classe de trapézio (Confiança: 84.13%).

Kite:

- **Imagem 0:** Acertou (Confiança: 63.69%).
- **Imagem 1:** Não acertou, previu uma classe de círculo (Confiança: 36.29%)
- **Imagem 2:** Não acertou, previu uma classe de triângulo (Confiança: 83.57%)
- **Imagem 3:** Não acertou, previu uma classe de triângulo (Confiança: 78.30%)
- **Imagem 4:** Acertou (Confiança: 87.89%).

Parallelogram:

- **Imagem 0:** Não acertou, previu que era um trapézio (Confiança: 81.18%).
- **Imagem 1:** Não acertou, previu que era um trapézio (Confiança: 71.40%).
- **Imagem 2:** Não acertou, previu que era um trapézio (Confiança: 97.04%).
- **Imagem 3:** Não acertou, previu que era um trapézio (Confiança: 60.58%).
- **Imagem 4:** Não acertou, previu que era um trapézio (Confiança: 62.52%).

Square:

- **Imagem 0:** Não acertou, previu que era um trapezoid (Confiança: 91.49%)
- **Imagem 1:** Não acertou, previu que era um circle (Confiança: 87.58%)
- **Imagem 2:** Não acertou, previu que era um triangle (Confiança: 46.98%)
- **Imagem 3:** Não acertou, previu que era um circle (Confiança: 69.52%)
- **Imagem 4:** Não acertou, previu que era um circle (Confiança: 57.44%)

Trapezoid:

- **Imagem 0:** Acertou, previu que era um trapezoid (Confiança: 91.49%)
- **Imagem 1:** Acertou, previu que era um trapezoid (Confiança: 51.70%)
- **Imagem 2:** Acertou, previu que era um trapezoid (Confiança: 100%)
- **Imagem 3:** Acertou, previu que era um trapezoid (Confiança: 100%)
- **Imagem 4:** Acertou, previu que era um trapezoid (Confiança: 75.91%)

Triangle:

- **Imagem 0:** Não acertou, previu que era um parallelogram (Confiança: 72.17%)
- **Imagem 1:** Não acertou, previu que era um parallelogram (Confiança: 96.92%)
- **Imagem 2:** Não acertou, previu que era um kite (Confiança: 85.32%)
- **Imagem 3:** Não acertou, previu que era um trapezoid (Confiança: 64.82%)
- **Imagem 4:** Acertou, previu que era um triangle (Confiança: 98.54%)

Verificou-se que a rede revelou uma tendência acentuada para classificar as imagens desenhadas manualmente como pertencentes à classe trapézio, mesmo quando estas correspondiam a outras formas geométricas. Este comportamento pode indicar que as imagens desenhadas têm semelhanças visuais com a classe "trapézio" ou que a rede é mais sensível aos padrões dessa classe.

7. Alínea E)

Nesta alínea, foi solicitado o desenvolvimento de uma aplicação gráfica em MATLAB que permita ao utilizador interagir com redes neuronais de forma simples e intuitiva, reunindo as funcionalidades exploradas nas alíneas anteriores.

A aplicação deve permitir configurar a topologia da rede, selecionar funções de treino e de ativação, treinar redes neuronais, bem como carregar e guardar redes previamente treinadas. Além disso, o utilizador deve poder aplicar uma rede a conjuntos de dados predefinidos, desenhar manualmente uma nova forma ou carregar uma imagem com a figura já desenhada, e classificar essa figura utilizando uma rede previamente treinada. Por fim, é possível visualizar os resultados da classificação e, se necessário, gerar e guardar ficheiros com os resultados obtidos.

Um dos outputs para servir de exemplo é:

Iniciando treino com 5 repetições e os seguintes parametros:
- Topologia: 10
- Número de Épocas: 20
- Função de Treino: trainlm
- Função de Ativação: tansig
- Parametros de Divisão: 0.70 0.15 0.15
- Pasta de Imagens: Start

Repetição 1/5
Global: 86.67% | Teste: 60.00%
Repetição 2/5
Global: 70.00% | Teste: 40.00%
Repetição 3/5

Global: 83.33% | Teste: 40.00%
Repetição 4/5
Global: 86.67% | Teste: 40.00%
Repetição 5/5
Global: 80.00% | Teste: 20.00%

Médias (5 repetições):
Global: 81.33% (±6.91)
Teste: 40.00% (±14.14)
Melhor: 1 (Teste: 60.00%)

8. Experiências feitas por nós

Aqui pretende-se analisar se a alteração do número de épocas influencia os resultados obtidos pela rede, e de que forma essa alteração impacta positiva ou negativamente o seu desempenho.

Mudar número de épocas - Train									
Conf1	1	10	tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	20	87.38% (±8.13)	66.44% (±10.06)	72.88%
Conf2	1	10	tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	40	88.67% (±4.58)	65.76% (±7.71)	71.19%
Conf3	1	10	tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	60	89.93% (±1.48)	68.00% (±5.12)	73.33%
Conf4	1	10	tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	100	90.60% (±1.75)	64.00% (±6.36)	73.33%

Como era de esperar, a partir das 20 épocas não se observaram melhorias significativas no desempenho da rede, uma vez que o processo de treino é frequentemente interrompido antes desse limite devido ao critério de *early stopping* com base na validação.

De seguida, pretende-se analisar de que forma a alteração da pasta de imagens utilizada para treino influencia os resultados obtidos pela rede, avaliando se essa mudança tem um impacto positivo ou negativo no seu desempenho.

Mudar a pasta de imagens para treino									
Conf1 - Start	1	10	tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	20	73.33% (±12.69)	32.00% (±17.89)	60.00%
Conf2 - Test	1	10	tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	20	85.00% (±2.64)	55.56% (±11.11)	66.67%
Conf3 - Train	1	10	tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	20	89.40% (±1.42)	63.11% (±3.37)	69.67%
Conf4 - All	1	10	tansig, purelin	trainlm	dividerand = {0.7, 0.15, 0.15}	20	89.03% (±2.47)	68.47% (±7.91)	79.66%

Como era expectável, o uso de pastas com um maior número de imagens por classe resultou em melhores desempenhos da rede. No entanto, ao passar da pasta *Train* para o conjunto completo (*All*), não se verificou uma melhoria significativa nos resultados. Este comportamento poderá estar relacionado com o número reduzido de camadas escondidas, limitando a capacidade da rede para tirar pleno proveito do aumento do volume de dados.

9. Conclusão

Em suma, o trabalho desenvolvido permitiu compreender em profundidade o funcionamento das redes neuronais feedforward e os diversos fatores que influenciam o seu desempenho na tarefa de classificação de formas geométricas.

Através das várias alíneas, foi possível observar como diferentes parâmetros como a topologia da rede, as funções de ativação, os algoritmos de treino e os rácios de divisão dos dados, afetam diretamente a capacidade de generalização e a precisão das redes.

Verificou-se que redes com estruturas mais simples, especialmente quando o número de exemplos por classe é reduzido, tendem a apresentar melhor desempenho, evitando o sobreajuste. Por outro lado, com conjuntos de dados maiores, o uso de redes mais complexas pode ser vantajoso, desde que haja um equilíbrio adequado entre os conjuntos de treino, validação e teste. A função de treino *trainlm* destacou-se de forma consistente, evidenciando melhor desempenho, embora à custa de maior tempo de processamento.

As redes testadas nas diferentes combinações de pastas (start, train, test) demonstraram variações relevantes na capacidade de generalização. Notou-se que, embora algumas redes tenham tido um excelente desempenho nos dados de treino, nem sempre conseguiram manter esse nível nos dados de teste, o que reforça a importância da validação cruzada e da análise das matrizes de confusão.

Através da comparação entre as melhores redes, foi possível identificar quais apresentam melhor equilíbrio entre desempenho e robustez. Conclui-se que o sucesso de uma rede não depende apenas da sua configuração, mas também da distribuição e diversidade dos dados com que é treinada. Este projeto revelou-se essencial para consolidar os conhecimentos teóricos sobre redes neuronais e para adquirir experiência prática no seu desenvolvimento e análise no contexto do MATLAB.