Departamento de Engenharia Informática e de Sistemas

Instituto Superior de Engenharia de Coimbra

Licenciatura em Engenharia Informática

Conhecimento e Raciocínio - 2020/2021

Relatório do Trabalho Prático:

Tema 1 – Redes Neuronais

Turma Prática Nº 4

André Lopes | a2019139754@isec.pt

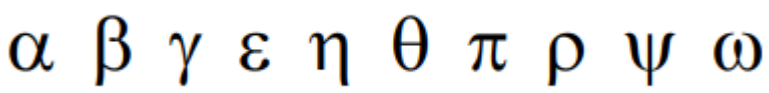
Samuel Tavares | a2019126468@isec.pt

Índice

1. Introdução

No âmbito da Unidade Curricular de Conhecimento e Raciocínio, foram propostos pelos docentes da unidade curricular, a realização de um dos seguintes temas**: Redes Neuronais** ou **Cbr e Inferência Difusa**, dos quais o primeiro foi escolhido por nós.

O objetivo deste tema consiste na implementação de uma rede neuronal capaz de classificar corretamente os seguintes 10 carateres gregos:



Para realizar este processo, foram-nos fornecidos os ficheiros de imagens a preto e branco, separadas por três pastas diferentes, que vão ser usadas nas tarefas pedidas pelo enunciado. A rede neuronal treinada deve ser capaz de identificar os caracteres que se encontram no tamanho 3024 x 3024 pixéis, que posteriormente vão ser redimensionadas.

A pasta 1 apresenta uma imagem para cada caracter grego, tendo no total 10 imagens (1x10).

A pasta 2 apresenta 10 imagens para cada caracter grego, tendo no total 100 imagens (10x10).

A pasta 1 apresenta uma imagem para cada caracter grego, tendo no total 10 imagens (4x10).

A pasta 4, contém as imagens desenhadas necessárias para a alínea d), e apresenta uma imagem para cada caracter grego, tendo no total 10 imagens (1x10).

No processo de implementação, realizamos alguns testes que estão disponíveis no ficheiro Testes.xlsx fornecido, onde podemos verificar se as redes treinadas conseguiam classificar corretamente os caracteres, e quais as arquiteturas de redes que apresentavam melhores desempenhos.

2. Considerações Iniciais

Tratamento das Imagens

Como as imagens disponibilizadas para o treino das redes neuronais apresentam uma dimensão muito alta, 3024x3024 pixéis, o que equivaleria a 9 144 576 inputs, vimos necessário fazer o seu redimensionamento para valores mais baixos, como 30x30 pixéis. Isto iria prevenir tempos e velocidades de treino muito elevadas.

Para fazer esse redimensionamento, utilizamos a função **imresize** do Matlab, onde lhe demos o input da imagem, e a escala que vai passar a ter que no caso foi 30x30. Isto para todas as pastas que usamos.

```
%PASTA 1
   p1 = zeros(30*30, 10);
    for i=1:10
        img = read(IDS1);
        imagem_reduzida = imbinarize(imresize(img,[30 30]));
        figure(1);
        imshow(imagem_reduzida);
        imagem_reduzida = imagem_reduzida(:);
        p1(:,i) = imagem_reduzida;

    end

t1 = zeros(10,10);
for i=1:10
    t1(i,i)=1;
end
```

Figura 1 - Tratamento de imagem + Inicialização dos Targets e Inputs para a pasta 1

Para além disso, demos uso da função **imbinarize** que vai ser responsável por converter a imagem em binário, ficando os pixels escuros e claros com os valores 1 e 0.

Essa informação vai ser guardada em matrizes que foram declaradas como globais, de maneira que não fosse necessário repetir o procedimento de leitura e inicialização, visto que assim seria possível aceder lhes a qualquer momento no programa.

3. Testes e Resultados

3.1. Alínea a)

Para a alínea a), foi pedida a realização de testes para um rede neuronal de uma camada escondida com 10 neurónios, e com as imagens que se encontram na Pasta1. Foram utilizadas diversas topologias, funções de treino e de ativação, de modo a verificar os diferentes desempenhos.

O número de épocas utilizado foi sempre 100. As topologias usadas foram **cascadeforwardnet** e **feedforwardnet**. As funções de ativação usadas foram a **tansig**, **purelin** e **logsig**. As função de treino testadas foram a **traingdx** (Descida gradiente com impulso e propagação linear adaptativa) e **trainrp** (Propagação resiliente) e não foi usada a segmentação.

Tendo estes parâmetros em conta, fizemos 5 testes para cada configuração e obtivemos os seguintes resultados para os melhores, piores e média de desempenhos (apenas estão ilustrados alguns exemplos):

Alínea A (5 testes para cada configuração)

Configurações	Número de camadas escondidas	Número de Neurónios	Topologia	Funções de ativação (camadas escondidas + camada de saída)	Função de Treino	Pior Desempenho	Melhor Desempenho	Média Desempenho
Conf10	1	10	feedforwardnet	tansig, purelin	**trainrp**	90%	100%	94%
Conf11	1	10	feedforwardnet	logsig, purelin	**trainrp**	100%	100%	100%
Conf12	1	10	feedforwardnet	purelin, purelin	**trainrp**	100%	100%	100%
Conf13	1	10	feedforwardnet	tansig, tansig	**trainrp**	80%	100%	92%
Conf14	1	10	feedforwardnet	logsig, tansig	**trainrp**	100%	100%	100%

*Tabela 1 - Alguns dos melhores testes realizados para a alínea a) com a função de treino **trainrp***

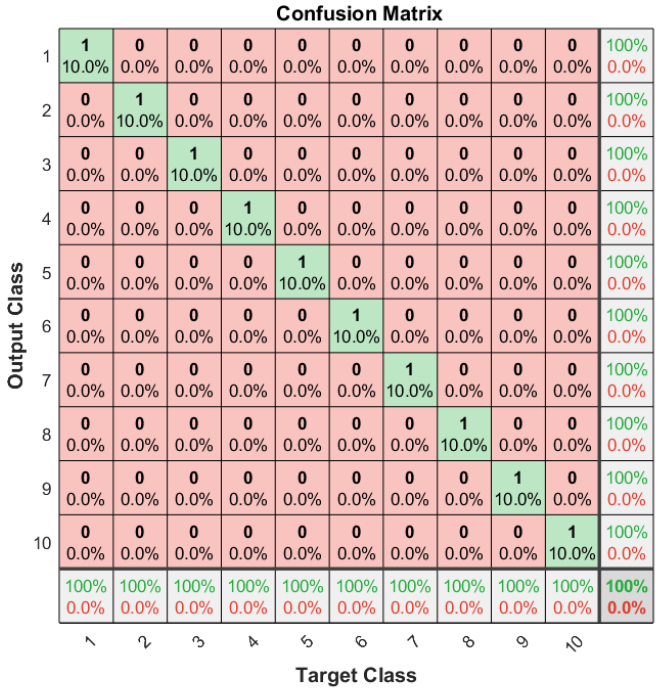


Figura 2 - Melhor Matriz de Confusão obtido para uma das configurações

Tendo em conta os resultados obtidos e que estão pormenorizados no ficheiro Excel para esta alínea, é possível concluir que com a topologia **cascadeforwardnet,** a média de desempenhos foi mais elevada. Relativamente às funções de treino, tanto com a trainrp ou traingdx, os valores não foram muito diferentes. Notamos ainda que quando a função de saída era purelin, os resultados eram elevados situando-se o desempenho normalmente nos 100%.

Alínea A (5 testes para cada configuração)

Configurações	Número de camadas escondidas	Número de Neurónios	Topologia	Funções de ativação (camadas escondidas + camada de saída)	Função de Treino	Pior Desempenho	Melhor Desempenho	Média Desempenho
Conf19	1	10	cascadeforwardnet	tansig, purelin	**traingdx**	100%	100%	100%
Conf20	1	10	cascadeforwardnet	logsig, purelin	**traingdx**	100%	100%	100%
Conf21	1	10	cascadeforwardnet	purelin, purelin	**traingdx**	100%	100%	100%
Conf22	1	10	cascadeforwardnet	tansig, tansig	**traingdx**	80%	100%	90%
Conf23	1	10	cascadeforwardnet	logsig, tansig	**traingdx**	70%	100%	84%

*Tabela 2 - Alguns dos melhores testes realizados para a alínea a) com a função de treino **traingdx***

3.2. Alínea b)

Tendo em conta os resultados da alínea a), considerámos a topologia cascadeforwardnet como sendo a melhor, visto que esta foi a que apresentou a maioria dos melhores resultados.

Agarrando nessas configurações de net, foram realizados 5 testes diferentes, sendo estes:

- Default:

Configurações	Número de camadas escondidas	Número de Neurónios	Topologia	Funções de ativação (camadas escondidas + camada de saída)	Função de Treino	Divisão dos exemplos	Pior precisão de teste	Melhor precisão de teste	Média precisão de teste	Pior precisão de global	Melhor precisão de global	Media de precisão de global
Conf1	1	10	cascadeforwardnet	tansig, purelin	**traingdx**	**dividerand = {0.7, 0.15, 0.15}**	6.6%	33%	13%	19%	67%	47%
Conf2	1	10	cascadeforwardnet	logsig, purelin	**traingdx**	**dividerand = {0.7, 0.15, 0.15}**	0%	20%	7%	30%	65%	46%
Conf3	1	10	cascadeforwardnet	purelin, purelin	**traingdx**	**dividerand = {0.7, 0.15, 0.15}**	6.6%	20%	11.96%	6%	50%	30.6%
Conf6	1	10	cascadeforwardnet	tansig, purelin	**trainrp**	**dividerand = {0.7, 0.15, 0.15}**	0%	20%	10.64%	14%	75%	38.8%
Conf7	1	10	cascadeforwardnet	logsig, purelin	**trainrp**	**dividerand = {0.7, 0.15, 0.15}**	7%	27%	15.96%	19%	61%	35.4%
Conf8	1	10	cascadeforwardnet	purelin, purelin	**trainrp**	**dividerand = {0.7, 0.15, 0.15}**	7%	13%	10.62%	18%	56%	37%

Tabela 3 – Resultados dos teste com parâmetros default

Como podemos observar na tabela, em termos de médias as funções andam bastante próximas, visto que variam entre 30.6% (purelin purelin, traingdx) e 47% (tansig purelin, traingdx). Já em relação à melhor precisão global, existe uma maior discrepância, sendo possível visualizar uma diferença de 25% de 50% (purelin purelin, traingdx) a 75% (tansig purelin, trainrp).

- 20 neurónios

Configurações	Número de camadas escondidas	Número de Neurónios	Topologia	Funções de ativação (camadas escondidas + camada de saída)	Função de Treino	Divisão dos exemplos	Pior precisão de teste	Melhor precisão de teste	Média precisão de teste	Pior precisão de global	Melhor precisão de global	Media de precisão de global
Conf9	1	20	cascadeforwardnet	tansig, purelin	**traingdx**	**dividerand = {0.7, 0.15, 0.15}**	6.6%	26%	13.28%	37%	72%	64.4%
Conf10	1	20	cascadeforwardnet	logsig, purelin	**traingdx**	**dividerand = {0.7, 0.15, 0.15}**	0%	27%	13%	19%	61%	32.8%
Conf11	1	20	cascadeforwardnet	purelin, purelin	**traingdx**	**dividerand = {0.7, 0.15, 0.15}**	6.6%	20%	11.96%	13%	31%	19%
Conf12	1	20	cascadeforwardnet	tansig, purelin	**trainrp**	**dividerand = {0.7, 0.15, 0.15}**	6.6%	20%	13.18%	13%	28%	19%
Conf13	1	20	cascadeforwardnet	logsig, purelin	**trainrp**	**dividerand = {0.7, 0.15, 0.15}**	6.6%	13.3%	9.28%	12%	61%	28.6%
Conf14	1	20	cascadeforwardnet	purelin, purelin	**trainrp**	**dividerand = {0.7, 0.15, 0.15}**	6.6%	26.6%	14.62%	14%	58%	35.4%

Tabela 4 – Resultados dos teste com 20 neurónios

Nesta tabela de dados podemos ver que as camadas tansig, purelin, quando combinadas com 20 neurónios e com a função de treino traingdx, obteve a melhor precisão global e média.

Por outro lado, quando esta é combinada com a função de treino trainrp, torna-se aquela com os piores resultados.

- 2 camadas (10 10):

Configurações	Número de camadas escondidas	Número de Neurónios	Topologia	Funções de ativação (camadas escondidas + camada de saída)	Função de Treino	Divisão dos exemplos	Pior precisão de teste	Melhor precisão de teste	Média precisão de teste	Pior precisão de global	Melhor precisão de global	Media de precisão de global
Conf15	2	10 10	cascadeforwardnet	tansig, purelin, purelin	**traingdx**	**dividerand = {0.7, 0.15, 0.15}**	13%	20%	14.64%	18%	30%	23%
Conf16	2	10 10	cascadeforwardnet	logsig, purelin, purelin	**traingdx**	**dividerand = {0.7, 0.15, 0.15}**	0%	26.6%	13.3%	13%	57%	36.2%
Conf17	2	10 10	cascadeforwardnet	purelin, purelin, purelin	**traingdx**	**dividerand = {0.7, 0.15, 0.15}**	0%	20%	9.32%	9%	16%	13%
Conf18	2	10 10	cascadeforwardnet	tansig, purelin, purelin	**trainrp**	**dividerand = {0.7, 0.15, 0.15}**	6.6%	33.3%	15.96%	12%	56%	30.4%
Conf19	2	10 10	cascadeforwardnet	logsig, purelin, purelin	**trainrp**	**dividerand = {0.7, 0.15, 0.15}**	6.6%	20%	10.62%	18%	72%	34%
Conf20	2	10 10	cascadeforwardnet	purelin, purelin, purelin	**trainrp**	**dividerand = {0.7, 0.15, 0.15}**	0%	13%	3.98%	17%	30%	26%

Tabela 5 – Resultados dos teste com 2 camadas de 10 neurónios cada

Ao expor as mesmas funções, com a adição de uma nova camada purelin de 10 neurónios, a configuração 16 obteve a melhor média com 36.2%.

Em relação à melhor precisão, esta verificou-se pela configuração 19 com 72% de precisão global com uma diferença de 15% do segundo lugar e 56% do último lugar, ocupado pela configuração 17.

- Divisão de exemplos {0.33, 0.33 0.33}:

Configurações	Número de camadas escondidas	Número de Neurónios	Topologia	Funções de ativação (camadas escondidas + camada de saída)	Função de Treino	Divisão dos exemplos	Pior precisão de teste	Melhor precisão de teste	Média precisão de teste	Pior precisão de global	Melhor precisão de global	Media de precisão de global
Conf21	1	10	cascadeforwardnet	tansig, purelin	**traingdx**	**dividerand = {0.33, 0.33, 0.33}**	3.03%	12.12%	9.09%	12%	43%	36%
Conf22	1	10	cascadeforwardnet	logsig, purelin	**traingdx**	**dividerand = {0.33, 0.33, 0.33}**	6.06%	24%	11.514%	38%	45%	41.2%
Conf23	1	10	cascadeforwardnet	purelin, purelin	**traingdx**	**dividerand = {0.33, 0.33, 0.33}**	3.03%	15.15%	9.696%	11%	16%	14%
Conf24	1	10	cascadeforwardnet	tansig, purelin	**trainrp**	**dividerand = {0.33, 0.33, 0.33}**	3.03%	18.18%	9.09%	6%	37%	22%
Conf25	1	10	cascadeforwardnet	logsig, purelin	**trainrp**	**dividerand = {0.33, 0.33, 0.33}**	0%	21.21%	12.12%	15%	46%	28.8%
Conf26	1	10	cascadeforwardnet	purelin, purelin	**trainrp**	**dividerand = {0.33, 0.33, 0.33}**	6.06%	9.09%	7.878%	14%	37%	24.2%

Tabela 6 – Resultados dos teste com divisão de exemplos (0.33,0.33,0.33)

Modificando a segmentação para {0.33, 0.33, 0.33}, nota-se que os resultados se encontram baixos, não superando os 50% de precisão global.

Conf22 possui a melhor media com 41.2%, enquanto Conf25 obtém 46% de precisão global, superando a Conf22 por apenas 1%.

Embora esta seja tenha mais firmeza que a Conf25, como se pode observar nas médias obtidas.

Conf23 mostra-se a pior configuração, tendo como melhor precisão global apenas 16%.

- Divisão de exemplos {0.9, 0.05 0.05}:

Configurações	Número de camadas escondidas	Número de Neurónios	Topologia	Funções de ativação (camadas escondidas + camada de saída)	Função de Treino	Divisão dos exemplos	Pior precisão de teste	Melhor precisão de teste	Média precisão de teste	Pior precisão de global	Melhor precisão de global	Media de precisão de global
Conf27	1	10	cascadeforwardnet	tansig, purelin	**traingdx**	**dividerand = {0.9, 0.05, 0.05}**	0%	40%	24%	16%	54%	41%
Conf28	1	10	cascadeforwardnet	logsig, purelin	**traingdx**	**dividerand = {0.9, 0.05, 0.05}**	0%	40%	16%	11%	45%	23.4%
Conf29	1	10	cascadeforwardnet	purelin, purelin	**traingdx**	**dividerand = {0.9, 0.05, 0.05}**	0%	40%	8%	9%	23%	14.8%
Conf30	1	10	cascadeforwardnet	tansig, purelin	**trainrp**	**dividerand = {0.9, 0.05, 0.05}**	0%	20%	8%	12%	39%	25.4%
Conf31	1	10	cascadeforwardnet	logsig, purelin	**trainrp**	**dividerand = {0.9, 0.05, 0.05}**	0%	0%	0%	9%	29%	17.2%
Conf32	1	10	cascadeforwardnet	purelin, purelin	**trainrp**	**dividerand = {0.9, 0.05, 0.05}**	0%	40%	16%	12%	95%	38%

Tabela 6 – Resultados dos teste com divisão de exemplos (0.9,0.05,0.05)

Alterando a segmentação para {0.9, 0.05, 0.05}, visualiza-se que a Conf27 possui a melhor media, sendo esta de 41%. No entanto Conf32 demonstra a melhor precisão global, sendo esta 95%.

A pior configuração obtida foi a Conf29 com apenas 23% de melhor precisão obtida nos testes e apenas 14.8% de média.

3.3. Alínea c)

3.3.1 Parte 1

Com a realização destes testes podemos observar que os resultados obtidos não se encontram no nível esperado, visto que ao usar um maior número de imagens por letra seria esperada um melhor resultado.

Muito pelo contrário, o resultado obtido foi bastante baixo, sendo que a melhor configuração foi a Conf1 com 17.5%, o que contrasta bastante com os 67% anteriormente obtidos. Este contraste nota-se ainda melhor na Conf8 que obteve apenas 7.5% quando tinha anteriormente obtido 95%.

Configurações	Número de camadas escondidas	Número de Neurónios	Topologia	Funções de ativação (camadas escondidas + camada de saída)	Função de Treino	Divisão dos exemplos	precisão de teste	precisão global
Conf1	1	10	cascadeforwardnet	tansig, purelin	**traingdx**	**dividerand = {0.7, 0.15, 0.15}**	20%	17.5%
Conf8	1	10	cascadeforwardnet	purelin, purelin	**trainrp**	**dividerand = {0.9, 0.05, 0.05}**	40%	7.5%

Tabela 7 – Melhores resultados da parte 1 da alínea c)

3.3.2 Parte 2

Este teste já se encontra mais previsível visto que os melhores resultados são os da própria pasta, sendo que esta obtém 77.5% de melhor precisão global na Conf6, pertencente à Pasta 3, enquanto, a maior da Pasta 1 e 2 é apenas 40% na Conf4, pertencente à Pasta 2.

Configurações	Número de camadas escondidas	Número de Neurónios	Topologia	Funções de ativação (camadas escondidas + camada de saída)	Função de Treino	Divisão dos exemplos	Pior precisão de teste	Melhor precisão de teste	Média precisão de teste	Pior precisão de global	Melhor precisão de global	Media de precisão de global	
Conf1	1	10	cascadeforwardnet	tansig, purelin	**trainrp**	**dividerand = {0.7, 0.15, 0.15}**	0%	33.3%	16.64%	10%	30%	14%	**Pasta1**
Conf2	1	10	cascadeforwardnet	purelin, purelin	**trainrp**	**dividerand = {0.9, 0.05, 0.05}**	0%	50%	20%	10%	20%	8%	
Conf3	1	10	cascadeforwardnet	tansig, purelin	**trainrp**	**dividerand = {0.7, 0.15, 0.15}**	0%	50%	13%	11%	18%	26.64%	**Pasta2**
Conf4	1	10	cascadeforwardnet	purelin, purelin	**trainrp**	**dividerand = {0.9, 0.05, 0.05}**	0%	50%	10%	0%	40%	16%	
Conf5	1	10	cascadeforwardnet	tansig, purelin	**trainrp**	**dividerand = {0.7, 0.15, 0.15}**	0%	33.3%	6.66%	17.5%	70%	34%	**Pasta3**
Conf6	1	10	cascadeforwardnet	purelin, purelin	**trainrp**	**dividerand = {0.9, 0.05, 0.05}**	0%	100%	30%	15%	77.5%	41%	

Tabela 8– Resultados da parte 2 da alínea c)

3.3.3 Parte 3

Os resultados obtidos com o treino das 3 pastas em simultâneo manteve a lógica da parte 2 da alínea C, onde os melhores resultados foram obtidos no teste da Pasta 3, mais propriamente, a Conf6 com 92.5% de melhor precisão global e 48% de média.

Configurações	Número de camadas escondidas	Número de Neurónios	Topologia	Funções de ativação (camadas escondidas + camada de saída)	Função de Treino	Divisão dos exemplos	Pior precisão de teste	Melhor precisão de teste	Média precisão de teste	Pior precisão de global	Melhor precisão de global	Media de precisão de global	
Conf1	1	10	cascadeforwardnet	tansig, purelin	**trainrp**	**dividerand = {0.7, 0.15, 0.15}**	4.34%	21.73%	12.168%	10%	60%	36%	**P1**
Conf2	1	10	cascadeforwardnet	purelin, purelin	**trainrp**	**dividerand = {0.9, 0.05, 0.05}**	0%	25%	15%	10%	30%	20%	
Conf3	1	10	cascadeforwardnet	tansig, purelin	**trainrp**	**dividerand = {0.7, 0.15, 0.15}**	8.69%	17.39%	12.17%	11%	44%	31%	**P2**
Conf4	1	10	cascadeforwardnet	purelin, purelin	**trainrp**	**dividerand = {0.9, 0.05, 0.05}**	0%	12.5%	2.5%	12%	46%	27%	
Conf5	1	10	cascadeforwardnet	tansig, purelin	**trainrp**	**dividerand = {0.7, 0.15, 0.15}**	8.69%	30.43%	14.778%	22.5%	62.5%	44%	**P3**
Conf6	1	10	cascadeforwardnet	purelin, purelin	**trainrp**	**dividerand = {0.9, 0.05, 0.05}**	13%	25%	20%	25%	92.5%	48%	

Tabela 9– Resultados da parte 3 da alínea c)

3.4. Alínea d)

Ao treinar a melhor configuração obtida na alínea C e testar a Pasta 4, com letras desenhadas durante a elaboração, obteve-se um resultado já esperado, pelo que se experienciou na parte 1 da alínea C, onde este se encontra bastante diminuto.

Conf6	1	10	cascadeforwardnet	purelin, purelin	**trainrp**	**dividerand = {0.9, 0.05, 0.05}**	13%	25%	20%	25%	92.5%	48%

Tabela 10 – Melhor resultado obtido na alínea c)

Configurações	Número de camadas escondidas	Número de Neurónios	Topologia	Funções de ativação (camadas escondidas + camada de saída)	Função de Treino	Divisão dos exemplos	precisão de teste	precisão global
Conf1	1	10	cascadeforwardnet	purelin, purelin	**trainrp**	**dividerand = {0.9, 0.05, 0.05}**	12.5%	10%

Tabela 11– Resultado da alínea d)

4. Interface Gráfica - Alínea e)

A interface gráfica foi inicialmente começada no editor de Gui do Matlab antigo. No entanto, fizemos a transição para o appDesigner, visto que este apresentava uma interface mais atualizada e uma maior facilidade na edição da mesma.

Na interface que criamos, colocamos botões, checkboxes, e texto necessários para a implementação correta do que era pedido no enunciado.

Para além disso, é possível guardar o treino de uma rede neuronal para mais tarde fazer o seu carregamento e testar com outras imagens.

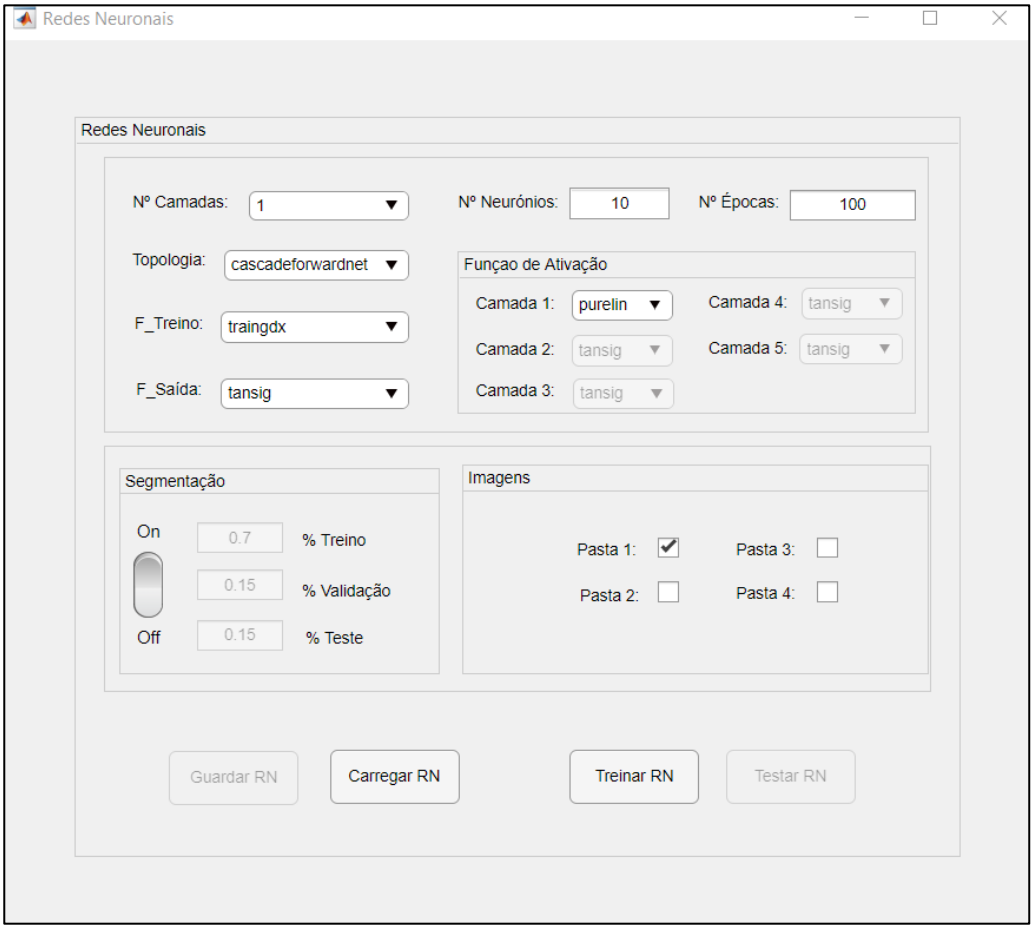


Figura 3 - Interface Gráfica da aplicação

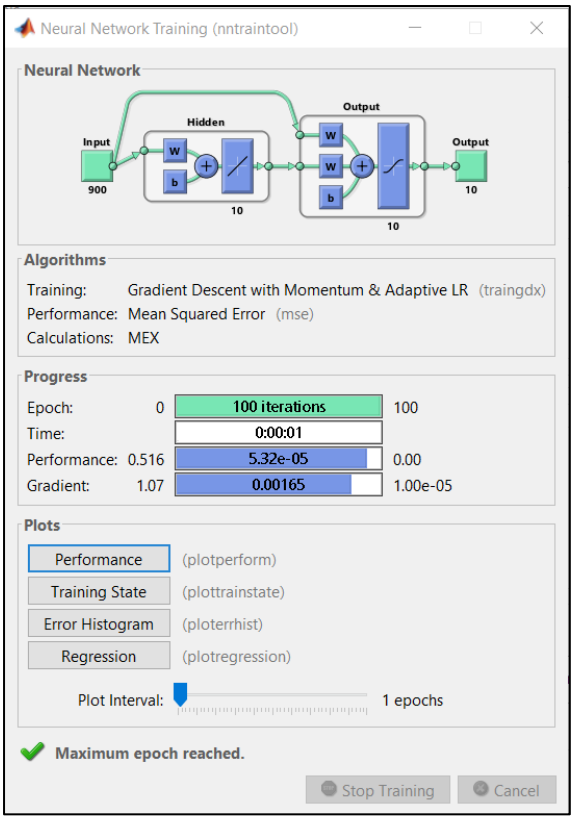


Figura 3- Treino da Rede Neuronal

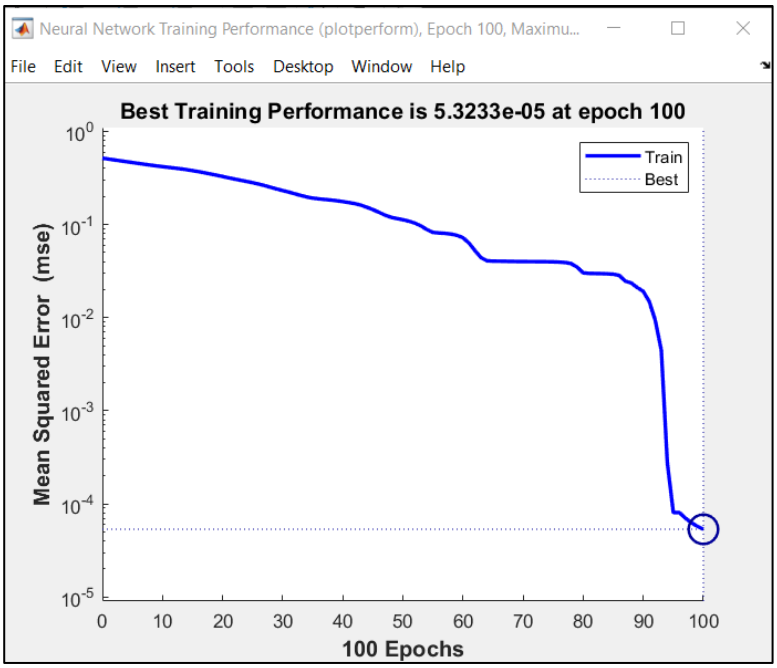


Figura 2 - Desempenho da Rede Neuronal

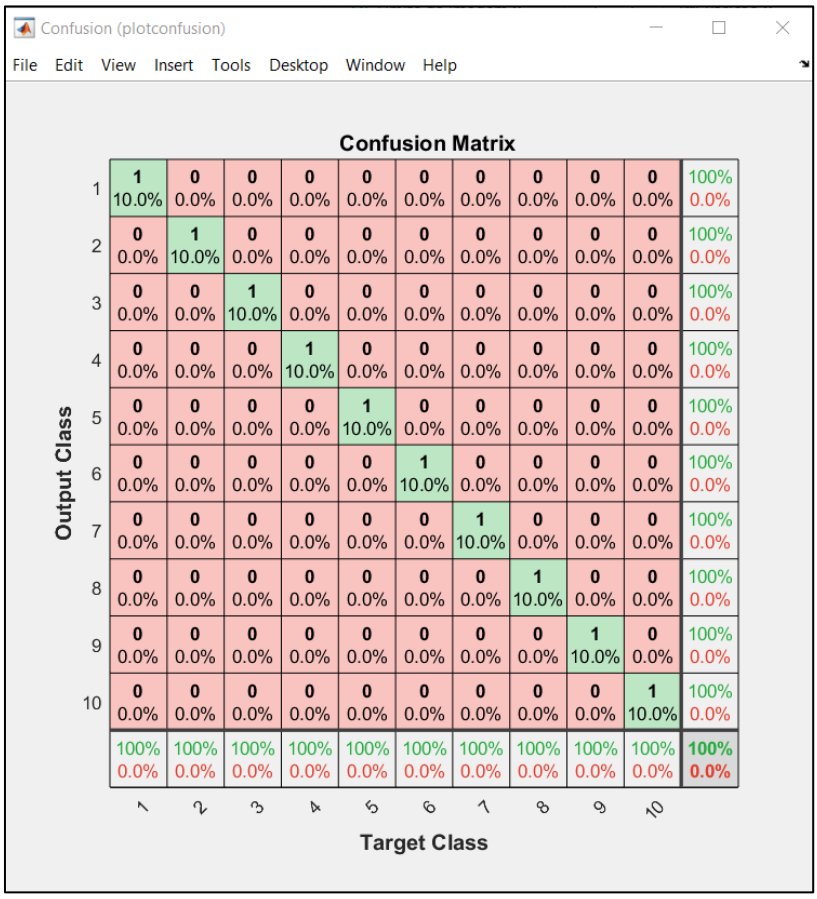


Figura 4 - Matriz de Confusão com os resultados da classificação

5. Conclusão

Com este trabalho, foi possível adquirir bastantes conceitos relativos à temática das redes neuronais e à linguagem de programação Matlab.

Verificamos que as redes neuronais têm um potencial muito grande a nível de processamento lógico. No entanto, este processamento envolve muitos recursos e por isso, quem as prepara tem de ter o cuidado de gerir recursos de modo a não afetar a qualidade de análise. O que pode prevenir estes problemas de memória pode ser a conversão de tamanhos no caso de imagens por exemplo, ou a utilização de funções de treino de baixo uso de memória.

Por norma, verificou-se que a topologia cascadeforwardnet apresentou melhores resultados para a alínea a), relativamente a feedforwardnet. Também se verificou que quando a função de saída é do tipo purelin, e a função treino do tipo, trainrp, os desempenhos foram mais altos.

Com isto, podemos concluir que as redes neuronais são idealmente desenvolvidas para ajudar a resolver problemas complexos em diversas situações da vida real ,como é o caso aqui de identificar caracteres.