1 2 9 0

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D COIMBRA

Deparamento de Informática

Aprendizagem Computacional

Engenharia Biomédica

OCR – Optical Character Recognition

Grupo 4, Prática Laboratorial 1

Mariana Montenegro 2019245964

Pedro Ferreira 2019170165

16 de outubro 2022

Table of Contents

List of Figures

List of Tables

1 Introdução

O presente trabalho tem como objetivo utilizar a tecnologia OCR (Optical Character Recognition) para o desenvolvimento de redes neuronais que permitam classificar os algarismos 1,2,3,4,5,6,7,8,9,0, desenhados manualmente.

Com este propósito, foram criadas diferentes arquiteturas para as redes neuronais. Primeiramente, foi desenvolvida uma arquitetura constituída por um filtro (memória associativa ou perceptron binário) e por um classificador de uma camada. Seguidamente, foi desenvolvida uma arquitetura constituída apenas por um classificador, com uma ou duas camadas. Foram consideradas diferentes funções de ativação: hardlim (binária), purelin (linear) e logsig (sigmóide). Foi ainda testada uma camada com função de ativação softmax (sigmóide) para depois ser possível fazer uma comparação com as restantes.

2 Data set

Para criar o dataset utilizado para treinar e testar o classificador, foi utilizada a função fornecida mpaper.m. Foram desenhados à mão 1000 caracteres (P1 a P20), por ambos os membros do grupo, de forma ordenada (1 a 0), e agrupados numa única matriz (P: 256x1000) que serviu para treinar as redes neuronais. Ou seja, como cada quadrado que contém um dígito escrito manualmente mede 16 px por 16 px que ao unir-se numa matriz coluna (cada coluna do quadrado seguida por outra) forma uma matriz 256x1, e como temos 1000 valores (50 em cada matriz P) formam então a matriz P_total de 256x1000.

Cada neurónio em cada matriz 16x16 de cada representação de um número escrito manualmente foi associada a um valor entre 0 e 1, chamado ativação (a), sendo 1 quando o neurónio branco e representa o número, e 0 quando é preto e não faz parte da representação (figura 1).

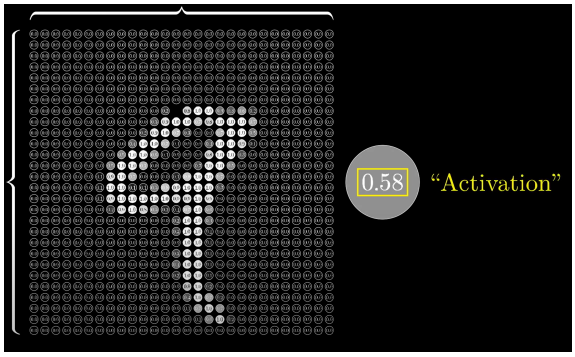


Figure 1: Esquema do valor de ativação de uma matriz representativa do número nove.

Source: (youtube) Deep learning, capítulo 1; 3Blue1Brown

De forma a prevenir overffiting dos dados, este conjunto foi dividido aleatoriamente, ficando o conjunto de treino com 85% dos caracteres e o conjunto de validação com os restantes 15%.

Para avaliar o desempenho da rede, foram ainda criados 2 conjuntos de teste, cada um escrito por um membro do grupo (P_teste1 e P_teste2, 256x50).

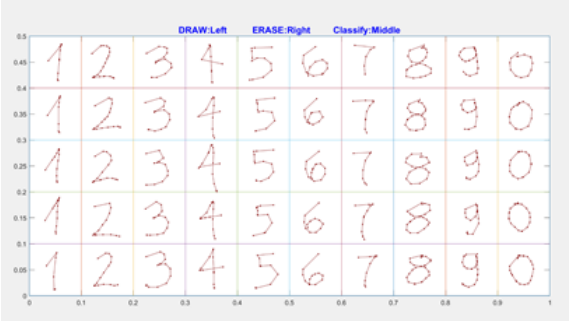


Figure 2: Exemplo de números do dataset desenhados utilizando o mpaper.m.

Source: Matlab, mpaper.m function

3 Rede Neuronal

Neste trabalho foram criados classificadores com duas arquiteturas diferentes:

• classificador com filtro (memória associativa ou perceptron binário). A matriz P criada pela função dataset_creator é submetida a filtros que pretendem melhorar à priori os parâmetros de cada neurónio para depois ser classificados com mais sucesso.

• apenas o classificador (uma camada, duas camadas e com a função softmax).

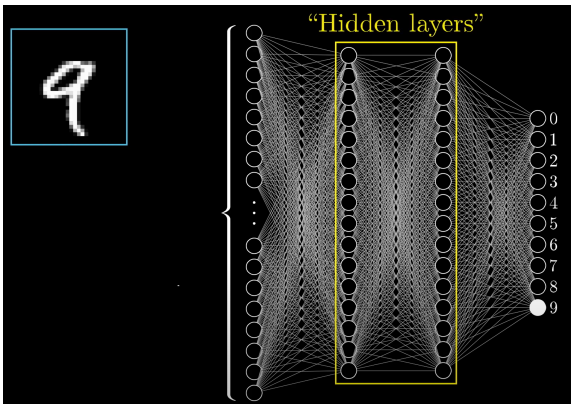


Figure 3: Rede Neuronal com duas camadas escondidas e output de 10 neurónios representativos de cada algarismo de 0-9.

Source: (youtube) Deep learning, capítulo 1; 3Blue1Brown

É necessário treinar a network de forma a obter e guardar os melhores parâmetros possíveis de weights e bias. Assim, cria-se uma rede neuronal através do comando net=network e treina-se (comando train)a mesma através da configuração de parâmetros obtidos pela comparação dos resultados da rede com a matriz Target gerada da função PerfcetArial.mat (no caso do perceptron) ou as funções T de treino e teste que representam matrizes diagonais.

Foram também usadas diferentes funções de ativação na classificação: hardlim (binária), purelin (linear) e logsig (sigmóide).

A funções purelin e logsig não são binárias e, desta forma, podem apresentar como resultado qualquer valor real entre $[-\infty, +\infty]$ $e[0, 1], respetivamente, sendonecessáriofazerpós-processamento.$

Para tal, recorremos a um método heurístico que transforma o valor mais alto em 1 e todos os outros em 0.

Estudamos também a função softmax que é um tipo de função sigmóide útil em problemas de classificação. Esta função transforma as saídas para cada classe em valores entre 0 e 1 e divide-os pela soma das saídas, dando assim probabilidade de a entrada estar numa determinada classe. Esta função não evita a necessidade de se fazer uma heurística a posteriori. Só faz sentido utilizar esta função quando não há um filtro aplicado sobre os dados.

Os parâmetros utilizados nos classificadores foram os seguintes:

- Learning rate = 0.5;
- Numero máximo de epochs = 1000;
- Goal = 1e-6;
- Critério = Sum squared error (sse).

Todas as redes foram guardadas e identificadas de acordo com a arquitetura e função de ativação selecionadas, de modo a poderem ser testadas através da interface.

Relativamente aos nomes das redes neuronais, são do tipo:

$$\text{Arquitetura} + \_ + \text{Função de ativação} + \_ + \text{trainedNet.mat} \quad (1)$$

ou

$$\text{Número de camadas} + \text{L} + \_ + \text{Função de ativação} + \_ + \text{trainedNet.mat} \quad (2)$$

ou

$$\text{Filtro} + \_ + \text{Função de ativação} + \_ + \text{trainedNet.mat} \quad (3)$$

4 Interface

Foi criada uma interface GUI (Graphical User Interface) utilizando o App Designer do Matlab que permite ao utilizador selecionar a arquitetura pretendida e filtro.

Nesta interface (GUI.mlapp), o utilizador pode selecionar a arquitetura pretendida bem como a função/funções de ativação que pretende aplicar. De seguida, deve selecionar entre "TEST", caso em que, através do mpaper.m, o utilizador pode desenhar os seus próprios algarismos e será devolvida uma grelha com os algarismos computorizados consoante a classificação feita pela rede neuronal, e "TRAIN AND TEST", caso em que a rede neuronal selecionada será testada com os nossos conjuntos de teste, P_teste1 e P_teste2, e serão apresentadas tanto a curva ROC como a matriz confusão associadas a cada conjunto.

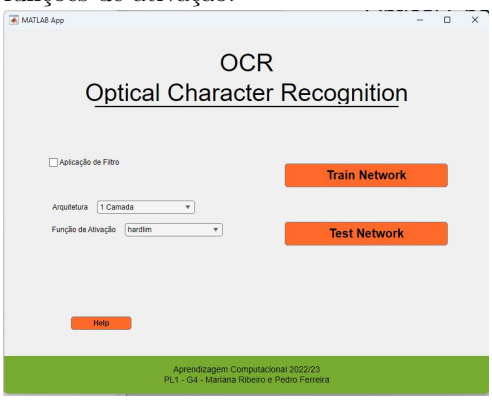
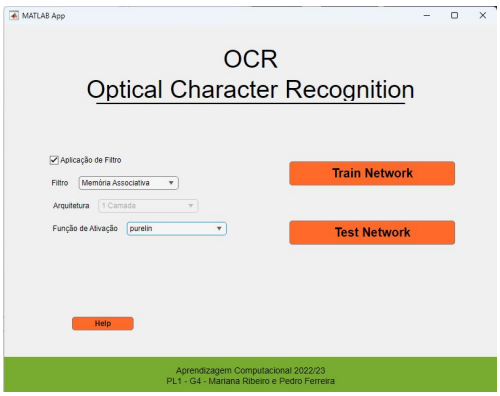
Com dropdown para a escolha de camada e funções de ativação.	Com iteração na escolha de filtro.
	

Table 1: Interface com iteração entre Treino e Teste, e as várias arquiteturas e funções de ativação.

5 Tabela Síntese

Tabela Síntese	
Função	Descrição
dataset_creation.m	Função utilizada para criar o dataset com as matrizes, isto é, criar a matriz P com todos os dados de treino, a matriz T a partir da PerfectArial, as matrizes de teste P_teste1 e P_teste2 e os targets de treino T_train e teste T_test.
train_n.m	Função com o algoritmo de treino das redes neuronais.
myclassify.m	Função chamada pela ocr_fun que carrega a rede neuronal treinada e testa os dados introduzidos na grelha.
AppOCR.mlapp	Contém as funções que permitem o treino, teste e validação dos inputs.
grafica.m	Função fornecida para mostrar até 3 dígitos dados como inputs.
mpaper.m	Função fornecida que permite o desenho dos dígitos manualmente e a gravação dos mesmos num ficheiro com extensão .mat.
ocr_fun.m	Função fornecida que permite mostrar os resultados da classificação na grelha.
showim.m	Função fornecida para mostrar um número arbitrário de dígitos dados como inputs.

Table 2: Tabela das funções criadas e fornecidas.

6 Resultados

Os valores destas tabelas correspondem à accuracy do grupo de treino e dos dois grupos de teste, para cada função de ativação, e foram obtidos através da matriz confusão.

Os melhores resultados das matrizes de confusão são apresentados nas tabelas 3 e 7, mas de notar que os resultados são iterativos e variam em cada novo ciclo de treino.

6.1 Memória Associativa + Classificador

	Hardlim	Purelin	Logsig
treino	14.5%	57.3%	27.9%
teste 1	12%	44%	28%
teste 2	12%	54%	20%

Table 3: Resultados do classificador com o filtro Memória Associativa e as três funções de ativação possíveis diferentes.

6.2 Perceptron binário + Classificador

	Hardlim	Purelin	Logsig
treino	100%	100%	90%
teste 1	100%	100%	90%
teste 2	100%	100%	90%

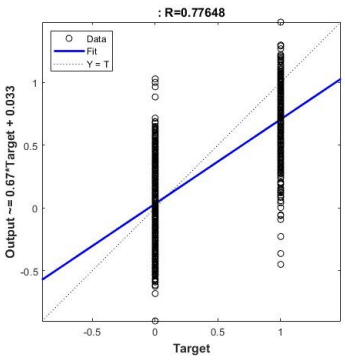
Table 4: Resultados do classificador com o filtro Perceptron Binário e as três funções de ativação possíveis diferentes.

6.3 Classificador

	Hardlim	Purelin	Logsig
treino	93.6%	92.5%	67.8%
teste 1	90%	92%	66%
teste 2	88%	80%	60%

Table 5: Resultado do Classificador com 1 Camada.

Regressão do classificador com 1 camada e função de ativação Purelin.



Evolução da Performance do classificador com 1 camada com função de ativação Purelin.

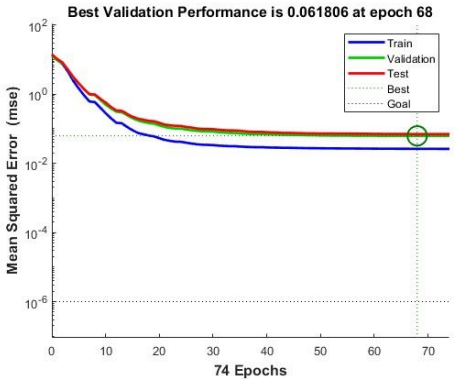


Table 6: Regressão obtida através da função "plotregression". Performance obtida através da função "plotperform". Ambas os plots estão presentes na função train_net.mat.

6.4 Classificador + Softmax

	Purelin	Logsig
treino	92.5%	67.8%
teste 1	92%	66%
teste 2	80%	60%

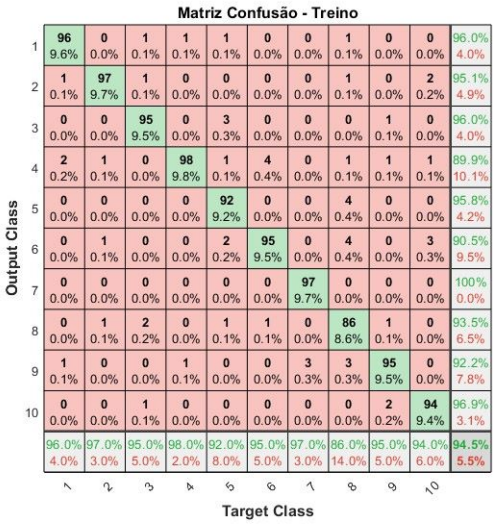
Table 7: Resultados com as Funções de Ativação para a Arquitetura de Classificador com Softmax e as duas funções de ativação possíveis diferentes.

6.5 Calssificador com 2 Camadas

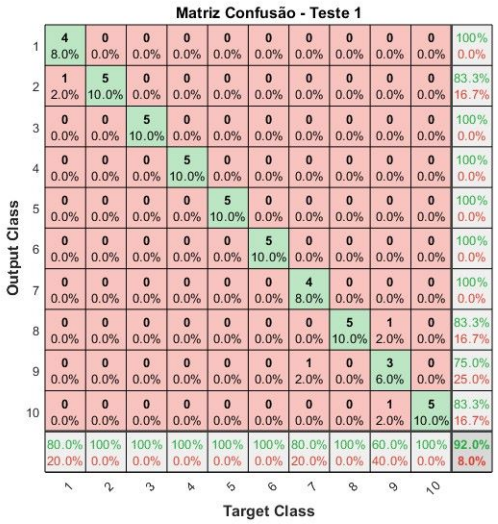
	Purelin e Logsig	Purelin e Purelin	Purelin e Logsig	Logsig e Logsig
treino	94.5%	86.1%	94.4%	94.8%
teste 1	92%	80%	94%	94%
teste 2	90%	70%	88%	92%

Table 8: Resultados com as combinações as Funções de Ativação Purelin e Logsig.

Matriz Confusão Treino



Matriz Confusão Teste 1



Matriz Confusão Teste 2

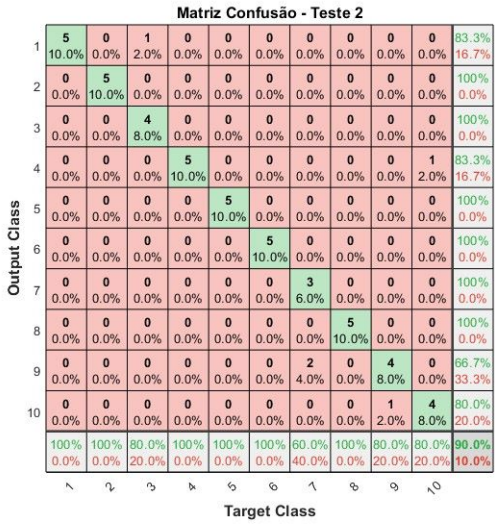


Table 9: Matrizes Confusão obtidas através do código presente na interface AppOCR.mlapp.

Exemplo de Classificação:

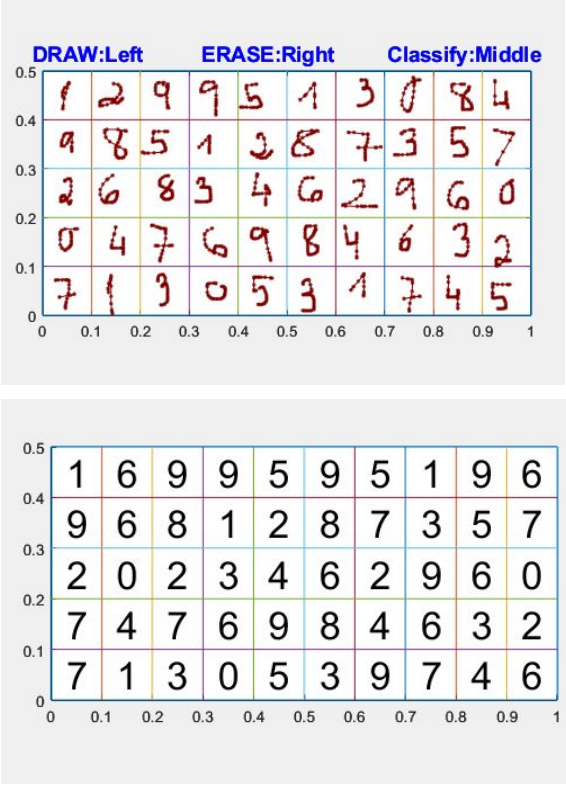


Table 10: Matrizes Confusão obtidas através do código presente na interface AppOCR.mlapp.

7 Conclusão

Observando os resulttados obtidos concluimos que accuracy tende sempre a ser superior no grupo de treino tal como é esperado. Este acontecimento ocorre quando a base de dados é adquirida por um conjunto pequeno de indivíduos e não generaliza bem a população.

Se observarmos os resultados de todos os classificadores testados, percebemos que o uso de um filtro BP ou Softmax antes da rede neuronal melhora a accuracy do classificador, tal como era esperado, sendo que o classificador que apresentou um melhor desempenho de todos, foi o que tem o Perceptron Binário como filtro com a função de ativação Purelin e Hardlim (100%). Quando se gerou novas matrizes pelo utilizador os resultados deste classificador continuaram a ser bastante bons e a estar de acordo com a accuracy esperada.

Verificou-se ainda que, na globalidade das redes neuronais, a accuracy para o grupo de teste 1 era superior ao grupo de teste 2, uma vez que o primeiro foi desenhado por pessoas que participaram na elaboração do grupo de treino, este resultado está de acordo com o esperado. Apesar do nosso dataset ter sido elaborado por diferentes pessoas e ter 1000 algarismos, para diminuir tal discrepância seria necessária uma base de dados ainda maior e mais diversificada.

Concluindo, apesar das dificuldades encontradas, o objetivo do trabalho foi amplamente atingido uma vez que encontramos mais que um classificador capaz de identificar corretamente a maioria dos inputs numéricos desenhados.

8 Bibliografia

[1]. Dourado, A. 2020. Assignment 1 – Optical Character Recognition – Guião do trabalho.

[2]. Dourado, A., 2020. Chapter 4 - Neurons Networks - Slides das aulas de Computação Neuronal e Sistemas Difusos.

[3]. Beale, M., Hagan, M. and Demuth, H., 2022. Deep Learning Toolbox - Getting Started Guide. MathWorks. Available at: https://www.mathworks.com/help/pdf_doc/deeplearning/nnet_gs.pdf.

[4]. Beale, M., Hagan, M. and Demuth, H., 2022. Deep Learning Toolbox - User's Guide. MathWorks. Available at: https://www.mathworks.com/help/pdf_doc/deeplearning/nnet_ug.pdf.