



# LICENCIATURA EM ENGENHARIA INFORMÁTICA 2023/2024

Inteligência Computacional

Garbage Classification

Bruno Oliveira - 2019136478  
Micael Eid - 2019112744

## Contents

Descrição do caso de estudo e objetivos do problema.....	3
Dataset (Garbage Classification).....	3
Descrição da implementação dos algoritmos.....	4
Scripts Auxiliares.....	4
Treino.....	5
Validação e Teste.....	7
Análise de Resultados.....	8
Conclusões.....	10
Referências .....	11

# Descrição do caso de estudo e objetivos do problema

## Dataset (Garbage Classification)

Problema: Problema de classificação de 5 diferentes tipos de lixo para reciclagem (Papel, Vidro, Cartão, Plástico, Metal).

“O desenvolvimento que procura satisfazer as necessidades da geração atual, sem comprometer a capacidade das gerações futuras de satisfazerem as suas próprias necessidades, significa possibilitar que as pessoas, agora e no futuro, atinjam um nível satisfatório de desenvolvimento social, econômico e de realização humana e cultural, fazendo, ao mesmo tempo, um uso razoável dos recursos da terra e preservando as espécies e os habitats naturais.” - Relatório Brundtland

Este dataset é composto por:

- Imagens RGB.
- Tamanho das imagens são aproximadamente 200x200.
- Classes: 10 (metal, glass, biological, paper, battery, trash, cardboard, shoes, clothes, and plastic).
- 803 Downloads feitos.
- +15.000 dados para utilizar.

Devido ao tema *Classificação Sustentável*, nesta meta optámos por seleccionar 5 das 10 classes presentes no Dataset (Garbage Classification):

- Metal
- Papel
- Vidro
- Plástico
- Cartão

Devido a erros(relacionados a memória) encontrados ao carregar o dataset para o Matlab, decidimos que iríamos seleccionar 1300 imagens de cada uma das 5 classes, tornando-o assim um Dataset balanceado.

Apagámos todas as imagens que não estivessem no formato .jpg (ex. .png, .jpeg), tal como as imagens CMYK, utilizando apenas exemplos RGB.

Efetuámos um resize das imagens de 200x200 para 28x28.

Dividimos o Dataset da seguinte maneira:

- 70% para treino
- 15% para teste
- 15% para validação

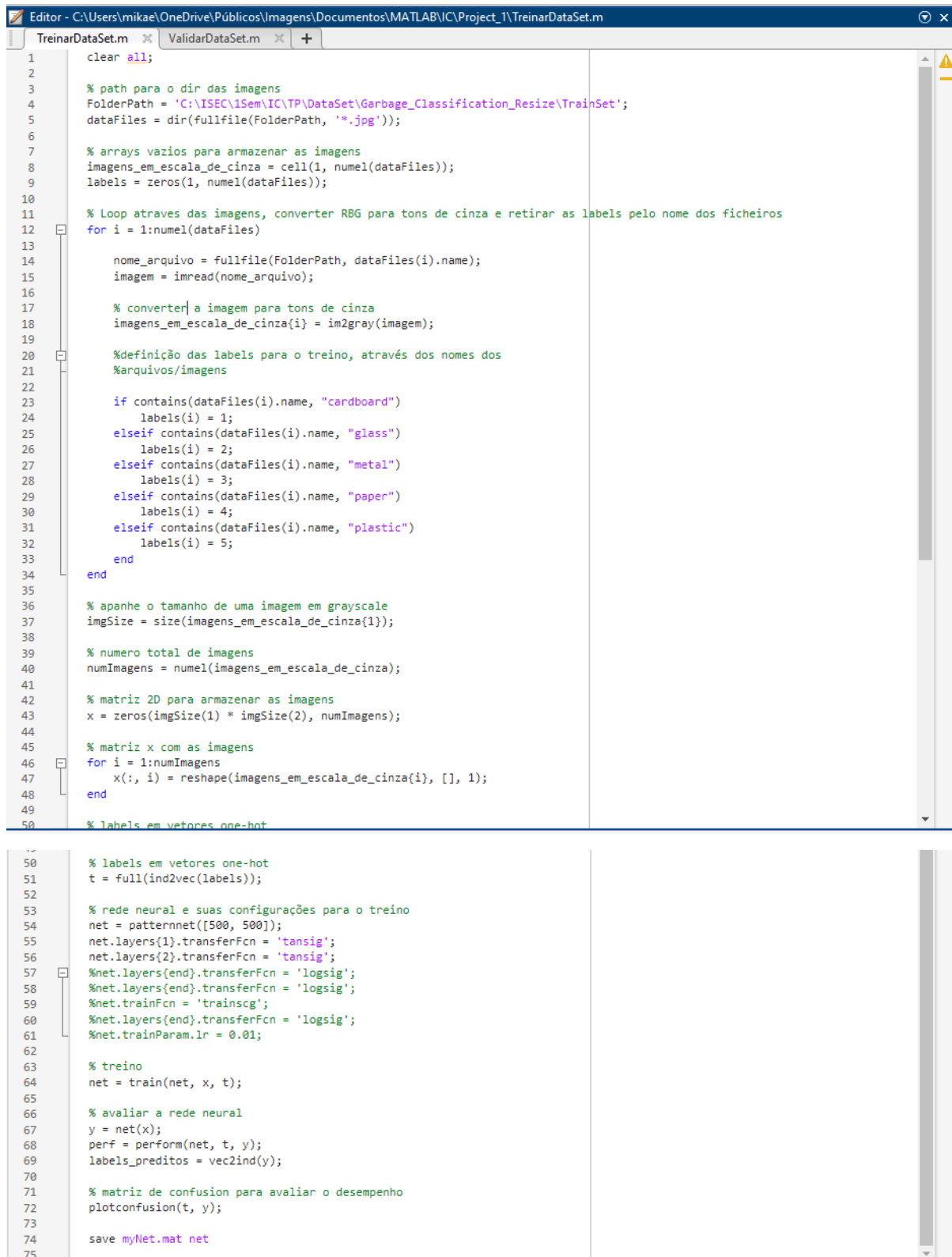
# Descrição da implementação dos algoritmos

## Scripts Auxiliares

Utilizámos vários scripts para nos facilitar no processo de organização do trabalho e futuro código escrito.

- Resizelmng.m – Script utilizado para redimensionar as imagens de 200x200 para 28x28.
- Deletelmng.m – Elimina as imagens (.jpeg e .png).
- DeleteCMYK.m – Elimina as imagens CMYK presents no Dataset.
- DividirDataSet.m – Divide o Dataset.

## Treino



```
1 clear all;
2
3 % path para o dir das imagens
4 FolderPath = 'C:\ISEC\15em\IC\TP\DataSet\Garbage_Classification_Resize\TrainSet';
5 dataFiles = dir(fullfile(FolderPath, '*.jpg'));
6
7 % arrays vazios para armazenar as imagens
8 imagens_em_escala_de_cinza = cell(1, numel(dataFiles));
9 labels = zeros(1, numel(dataFiles));
10
11 % Loop atraves das imagens, converter RGB para tons de cinza e retirar as labels pelo nome dos ficheiros
12 for i = 1:numel(dataFiles)
13
14     nome_arquivo = fullfile(FolderPath, dataFiles(i).name);
15     imagem = imread(nome_arquivo);
16
17     % converter a imagem para tons de cinza
18     imagens_em_escala_de_cinza{i} = im2gray(imagem);
19
20     %definição das labels para o treino, através dos nomes dos
21     %arquivos/imagens
22
23     if contains(dataFiles(i).name, "cardboard")
24         labels(i) = 1;
25     elseif contains(dataFiles(i).name, "glass")
26         labels(i) = 2;
27     elseif contains(dataFiles(i).name, "metal")
28         labels(i) = 3;
29     elseif contains(dataFiles(i).name, "paper")
30         labels(i) = 4;
31     elseif contains(dataFiles(i).name, "plastic")
32         labels(i) = 5;
33     end
34 end
35
36 % apanhe o tamanho de uma imagem em grayscale
37 imgSize = size(imagens_em_escala_de_cinza{1});
38
39 % numero total de imagens
40 numImagens = numel(imagens_em_escala_de_cinza);
41
42 % matriz 2D para armazenar as imagens
43 x = zeros(imgSize(1) * imgSize(2), numImagens);
44
45 % matriz x com as imagens
46 for i = 1:numImagens
47     x(:, i) = reshape(imagens_em_escala_de_cinza{i}, [], 1);
48 end
49
50 % labels em vetores one-hot
51 t = full(ind2vec(labels));
52
53 % rede neural e suas configurações para o treino
54 net = patternnet([500, 500]);
55 net.layers{1}.transferFcn = 'tansig';
56 net.layers{2}.transferFcn = 'tansig';
57 %net.layers{end}.transferFcn = 'logsig';
58 %net.layers{end}.transferFcn = 'logsig';
59 %net.trainFcn = 'trainscg';
60 %net.layers{end}.transferFcn = 'logsig';
61 %net.trainParam.lr = 0.01;
62
63 % treino
64 net = train(net, x, t);
65
66 % avaliar a rede neural
67 y = net(x);
68 perf = perform(net, t, y);
69 labels_preditos = vec2ind(y);
70
71 % matriz de confusion para avaliar o desempenho
72 plotconfusion(t, y);
73
74 save myNet.mat net
75
```

Figure 1,6 TreinarDataSet.m

No código optámos por fazer um loop que iria converter as imagens de RGB para imagens com um tom cinzento, retirando também os labels segundo o nome dos ficheiros.

Dividimos as imagens mediante os 5 labels diferentes (metal, papel, vidro, cartão e plástico) atribuindo um número correspondente (ordem alfabética Inglesa).

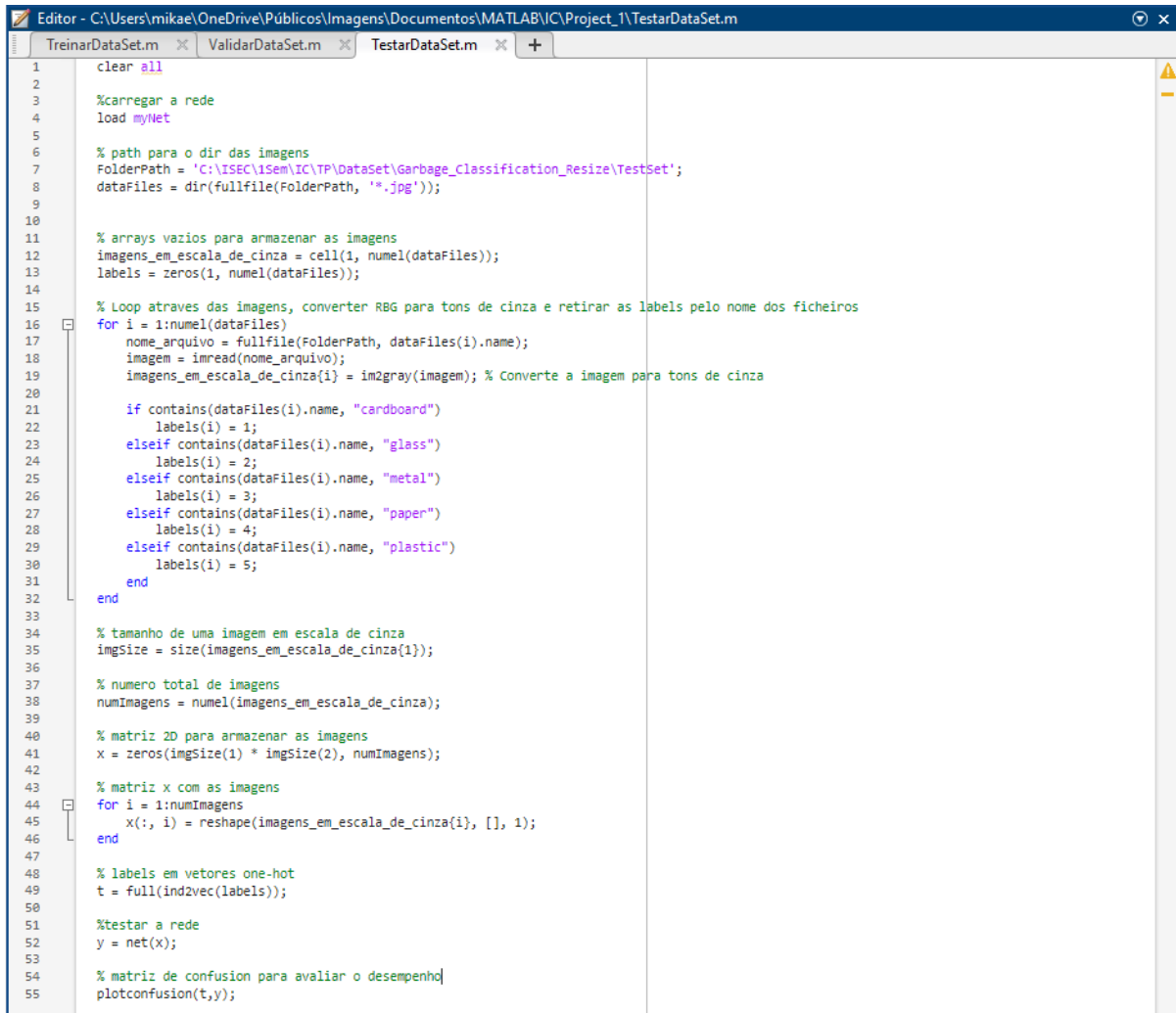
1. Cartão
2. Vidro
3. Metal
4. Papel
5. Plástico

Fizemos um loop para percorrer todas as imagens adicionando-as a uma matriz 2D designada por matriz x.

Finalmente realizámos o treino da rede neuronal com 2 hidden layers e 500 neurónios cada.

Posteriormente iremos demonstrar mais treinos e seus resultados.

## Validação e Teste



```
1 clear all
2
3 %carregar a rede
4 load myNet
5
6 % path para o dir das imagens
7 FolderPath = 'C:\ISEC\1Sem\IC\TP\DataSet\Garbage_Classification_Resize\TestSet';
8 dataFiles = dir(fullfile(FolderPath, '*.jpg'));
9
10
11 % arrays vazios para armazenar as imagens
12 imagens_em_escalade_cinza = cell(1, numel(dataFiles));
13 labels = zeros(1, numel(dataFiles));
14
15 % Loop atraves das imagens, converter RGB para tons de cinza e retirar as labels pelo nome dos ficheiros
16 for i = 1:numel(dataFiles)
17     nome_arquivo = fullfile(FolderPath, dataFiles(i).name);
18     imagem = imread(nome_arquivo);
19     imagens_em_escalade_cinza{i} = im2gray(imagem); % Converte a imagem para tons de cinza
20
21     if contains(dataFiles(i).name, "cardboard")
22         labels(i) = 1;
23     elseif contains(dataFiles(i).name, "glass")
24         labels(i) = 2;
25     elseif contains(dataFiles(i).name, "metal")
26         labels(i) = 3;
27     elseif contains(dataFiles(i).name, "paper")
28         labels(i) = 4;
29     elseif contains(dataFiles(i).name, "plastic")
30         labels(i) = 5;
31     end
32 end
33
34 % tamanho de uma imagem em escala de cinza
35 imgSize = size(imagens_em_escalade_cinza{1});
36
37 % numero total de imagens
38 numImagens = numel(imagens_em_escalade_cinza);
39
40 % matriz 2D para armazenar as imagens
41 x = zeros(imgSize(1) * imgSize(2), numImagens);
42
43 % matriz x com as imagens
44 for i = 1:numImagens
45     x(:, i) = reshape(imagens_em_escalade_cinza{i}, [], 1);
46 end
47
48 % labels em vetores one-hot
49 t = full(ind2vec(labels));
50
51 %testar a rede
52 y = net(x);
53
54 % matriz de confusion para avaliar o desempenho
55 plotconfusion(t,y);
```

Figure 2 TestarDataSet.m e ValidarDataSet.m

Devido à divisão das imagens do Dataset tivemos que atribuir os labels a cada uma das imagens, convertendo também de RGB para imagens com tom cinzento (como foi feito no TreinoDataSet.m).

## Análise de Resultados

Realizámos vários tipos de testes com diferentes funções, camadas e número de neurónios, de maneira a obter resultados distintos, verificando o melhor.

Estes testes tiveram recurso a Código dado nas aulas práticas.

### 1) 1 camada com 500 neurónios, 'trainfcg'

Ver anexo *config1.jpg*

- Accuracy = 47.18%
- Sensibilidade = 46.15%
- Especificidade = 42.02%
- F-measure = 0.46
- AUC = 0.76

### 2) 1 camadas com 1000 neurónios, 'trainfcg'

Ver anexo *config2.jpg*

- Accuracy = 46.87%
- Sensibilidade = 47.69%
- Especificidade = 45.64%
- F-measure = 0.46
- AUC = 0.74

### 3) 2 camadas com 100 neurónios, 'trainfcg'

Ver anexo *config3.jpg*

- Accuracy = 50.67%
- Sensibilidade = 47.69%
- Especificidade = 48.72%
- F-measure = 0.45
- AUC = 0.76

### 4) 2 camadas com 250 neurónios, 'trainfcg'

Ver anexo *config4.jpg*

- Accuracy 48.51%
- Sensibilidade = 48.21%
- Especificidade = 49.23%
- F-measure = 0.50
- AUC = 0.78



5) 3 camadas com 500 neurónios, 'trainfcg'

Ver anexo *config5.jpg*

- Accuracy 49.03%
- Sensibilidade = 45.64%
- Especificidade = 51.79%
- F-measure = 0.48
- AUC = 0.77

6) 4 camadas com 100 neurónios, 'trainfcg'

Ver anexo *config6.jpg*

- Accuracy = 50.15%
- Sensibilidade = 53.85%
- Especificidade = 51.79%
- F-measure = 0.51
- AUC = 0.78

7) 4 camadas com 200 200 100 50 neurónios, 'traincgp' e adicionado o valor de regularização *net.performParam.regularization* = 0.1 (Overfitting)

Ver anexo *config7.jpg*

- Accuracy = 45.44%
- Sensibilidade = 45.13%
- Especificidade = 50.77%
- F-measure = 0.45
- AUC = 0.77

8) 4 camadas com 500 500 250 100 neurónios, 'traincgp' e adicionado o valor de regularização *net.performParam.regularization* = 0.2 (Overfitting)

Ver anexo *config8.jpg*

- Accuracy = 52.82%
- Sensibilidade = 46.67%
- Especificidade = 52.31%
- F-measure = 0.49
- AUC = 0.79

Podemos verificar que o teste que obteve melhor resultado foi o teste 8 (accuracy de 52.82%), sendo que, o pior, foi o teste 7 (accuracy de 45.44%)

Os resultados obtidos não foram satisfatórios para o que estávamos à espera.

## Conclusões

Com a realização desta meta foi-nos possível adquirir novos conhecimentos sobre Matlab e a sua utilização para testes e treino, problemas de classificação com recurso a dataset retirados online e criação de scripts para facilitar o processo de escrita e organização de Código.

Tivemos dificuldades com o tamanho do dataset e com a utilização das imagens RGB.

Analizando os resultados obtidos, conseguimos verificar que estão baixos relativamente à Accuracy pretendida (acima de 90%).

N meta 2 pretendemos melhorar o código com recurso à linguagem Python, aumentando os parâmetros pretendidos de Accuracy, Sensibilidade, Especificidade, F-Measure e AUC, tanto no teste como em treino.

## Referências

Todas as imagens referidas serão enviadas em anexo no ficheiro .zip.

DataSet:

<https://www.kaggle.com/datasets/sumn2u/garbage-classification-v2/data>

Funções Matlab:

<https://www.mathworks.com/help/deeplearning/ref/fitnet.html>

Neural Network:

<https://www.mathworks.com/help/deeplearning/ug/choose-a-multilayer-neural-network-training-function.html>

ChatGPT:

<https://chat.openai.com/>

Livro de Apoio:

[https://www.nrigroupindia.com/e-book/Introduction%20to%20Machine%20Learning%20with%20Python%20\(%20PDFDrive.com%20\)-min.pdf](https://www.nrigroupindia.com/e-book/Introduction%20to%20Machine%20Learning%20with%20Python%20(%20PDFDrive.com%20)-min.pdf)