

FASE I

Inteligência Computacional 2023/2024

Licenciatura em Engenharia Informática

Dinis Meireles de Sousa Falcão / a2020130403@isec.pt

Kevin Fernando Pereira Rodrigues / a2013010749@isec.pt

ÍNDICE

ANÁLISE E RECOLHA DE DADOS	2
PROJETO	3
DEFINIÇÃO DE UM MODELO MLP	4
RESULTADOS DO TREINO	7
RESULTADOS DO TESTE	9
CONCLUSÃO	10

ANÁLISE E RECOLHA DE DADOS

Descrição do problema:

Este problema consiste em identificar o tipo de paisagem/ambiente das imagens fornecidas. Estas podem ser classificadas como: Buildings, Forests, Mountains, Glacier, Street e Sea.

Dataset:

- **24335 imagens (17 034 de treino e 7301 de teste);**
- **4055 imagens de cada classe;**
- **6 classes;**
- **150 x 150 convertidas para 28 x 28.**
- **<https://www.kaggle.com/datasets/nitishabharahti/scene-classification>**

Onde queremos chegar?

Sem termos de comparação, o nosso principal objetivo será obter o máximo de % correta na classificação das imagens que colocarmos para o teste.

PROJETO

O projeto foi desenvolvido com recurso ao MATLAB R2023b e contém os seguintes ficheiros:

- 📁 **images;**
- 📁 **resized_images;**
- 📁 **resizedteste_images;**
- 📄 **treinoRedeMLP.m;**
- 📄 **treinoRedeMLP.asv;**
- 📄 **testeRedeMLP.m;**
- 📄 **train.csv;**
- 📄 **test.csv;**

A pasta *images* contém todas as imagens utilizadas neste projeto. As pastas *resized_images* e *resizedtest_images* contêm as imagens redimensionadas (28 x 28) de treino e de teste, respetivamente. O ficheiro *treinoRedeMLP.m* contém todo o código desenvolvido para o treino da rede neuronal desenvolvida. O ficheiro *treinoRedeMLP.asv* é um ficheiro de backup do ficheiro *treinoRedeMLP.m*. O ficheiro *testeRedeMLP.m* contém todo o código desenvolvido para o teste da rede neuronal desenvolvida. Os ficheiros *train.csv* e *teste.csv* contêm as tabelas com os nomes das imagens e as respetivas classes a que pertencem.

DEFINIÇÃO DE UM MODELO MLP

De maneira a que haja uma maior clareza no entendimento deste trabalho, selecionámos alguns excertos de código importantes para efetuar uma pequena descrição.

Redimensionamento das imagens

```
label = label + 1;

nimages = length(label);

images = zeros([28, 28], nimages);

for i = 1:nimages
    image_name = DataTable.image_name{i};
    image_path = fullfile('images', image_name);
    img = imread(image_path);
    if size(img, 3) == 3
        img = rgb2gray(img);
    end
    img = imresize(img, [28, 28]);
    images(:, :, i) = img;
    %imwrite(img, fullfile('resized_images', DataTable.image_name{i}));
end
```

Inicialmente é feito um reajuste das *labels* que estão na tabela, ou seja, passam de “0 a 5” para de “1 a 6”. De seguida é criada uma matriz tridimensional para armazenar as imagens redimensionadas. Dentro do *loop*, é associado o nome da imagem da tabela à respetiva imagem que está na pasta *images*. Faz-se uma verificação de cor da imagem e, caso tenha cor, é colocada a cinzento. É feito o redimensionamento e a imagem é armazenada na matriz. Depois disso, é colocada na pasta *resized_images*.

✚ Conversão dos labels para o formato one-hot-encoding

```
num_classes = 6;
num_samples = length(label);
one_hot_labels = zeros(num_classes, num_samples);

for i = 1:num_samples
    class = label(i);
    one_hot_labels(class, i) = 1;
end
```

Criação de uma matriz de zeros e posterior atualização para refletir o formato one-hot encoding. Ou seja, a classe correspondente fica definida a 1, enquanto que as outras ficam definidas a 0.

✚ Matriz de confusão

```
%plotconfusion(one_hot_labels, classes);
classes_vector = classes;
[~, one_hot_labels_vector] = max(one_hot_labels);
C = confusionmat(one_hot_labels_vector, classes_vector);
confusionchart(C);
```

Após várias tentativas de uso da função *plotconfusion*, obtivemos alguns erros. Isto fez com que adotássemos outra estratégia. Utilização da função *confusionmat* que calcula a matriz de confusão, comparando as classes verdadeiras com as classes previstas. Por fim, a função *confusionchart* cria o gráfico da matriz de confusão.

✚ Definição da rede e treino da mesma

```
net = patternnet(10);
net.layers{end}.size = 6;

net = train(net, images_resized, one_hot_labels);
```

Criação de uma rede neuronal. Configuração do número de neurónios da camada de saída da rede para que seja igual ao número de classes existentes. Treino da rede com os dados de entrada *images_reshaped* e os targets *one_hot_labels*.

Matriz 3D para 2D

```
images_reshaped = reshape(images, [], size(images, 3));  
images_transposed = images_reshaped';  
%label_transposed = label';
```

Remodelação da matriz *images* para que as suas dimensões fiquem em apenas uma coluna, mantendo o canal de cores. É feita a transposição da matriz *images_reshaped*, ou seja, uma troca entre linhas e colunas.

Teste da rede

```
y = net(images_reshaped);  
classes_pred = vec2ind(y);
```

Teste da rede treinada com os dados de teste. Colocar os valores atribuídos na variável *classes_pred*.

RESULTADOS DO TREINO

100 neurónios

54 épocas

Accuracy: 55.90%

True Class	1	2	3	4	5	6
	1198	269	288	289	242	342
	217	1904	91	89	92	352
	200	101	1603	374	523	156
	155	140	352	1894	405	91
	211	259	531	537	1132	114
	282	329	222	140	119	1791
		Predicted Class				

200 neurónios

78 épocas

Accuracy: 59.15%

True Class	1	2	3	4	5	6
	1309	283	231	291	219	295
	152	2048	57	107	94	287
	161	106	1738	397	399	156
	141	143	355	1942	375	81
	175	227	561	523	1185	113
	270	317	191	120	131	1854
		Predicted Class				

300 neurónios

103 épocas

Accuracy: 64.37%

True Class	1	2	3	4	5	6
	1582	237	171	188	209	241
	183	2086	37	82	95	262
	152	89	1842	336	391	147
	140	121	298	2027	363	88
	174	206	406	468	1440	90
	237	285	150	114	110	1987
		Predicted Class				

400 neurónios

102 épocas

Accuracy: 63.37%

True Class	1	2	3	4	5	6
	1298	277	247	236	238	332
	202	1961	75	85	107	315
	189	101	1638	366	485	180
	168	147	324	1903	404	91
	172	240	516	483	1261	112
	275	279	183	116	129	1901
		Predicted Class				

500 neurónios

103 épocas

Accuracy: 64.78%

True Class	1	2	3	4	5	6
	1547	234	162	217	180	288
	171	2132	41	80	78	243
	147	96	1849	391	359	115
	110	138	297	2098	331	63
	166	231	418	464	1413	92
Predicted Class	1	2	3	4	5	6

750 neurónios

120 épocas

Accuracy: 63.37%

True Class	1	2	3	4	5	6
	1436	285	239	181	205	282
	167	2118	61	79	81	239
	161	119	1742	347	447	141
	147	162	371	1901	362	94
	185	263	474	454	1321	87
Predicted Class	1	2	3	4	5	6

1000 neurónios

133 épocas

Accuracy: 65.49%

True Class	1	2	3	4	5	6
	1622	242	178	180	209	197
	163	2159	56	75	114	178
	173	107	1840	291	427	119
	144	131	300	1951	459	52
	143	175	394	376	1624	72
Predicted Class	1	2	3	4	5	6

RESULTADOS DO TESTE

Para o teste da rede, decidimos utilizar a rede *patternnet* com 300 neurónios, visto que a partir da mesma os resultados se mantiveram praticamente iguais. A diferença entre esta e a de 1000 neurónios não é significativa, e demorará menos tempo a realizar o teste.

Devido à falta de classificação das imagens de teste, tivemos de ser nós a efetuar essa classificação para ver se os resultados do teste estariam corretos ou não. Sendo assim, apenas o fizemos para as 50 primeiras imagens, obtendo uma taxa de sucesso de 40.00%.

CONCLUSÃO

Ao longo deste trabalho, tivemos de enfrentar vários desafios com os quais não estávamos à espera, e para os quais ainda não estávamos preparados. Ainda assim, conseguimos ultrapassá-los e conseguir concluir esta 1ª fase, ainda que tenha sido com uma taxa de sucesso baixa, tanto para o treino como para o teste. Pretendemos, daqui para a frente, procurar as melhores soluções para aumentar estas taxas de sucesso.

FIM