RELATÓRIO

Trabalho Prático

Conhecimento e Raciocínio

Licenciatura em Engenharia Informática

O trabalho prático sobre Redes Neuronais consiste em treinar um algoritmo para classificar corretamente imagens de 6 formas geométricas.

Trabalho realizado por:

Tomás Gomes Silva - 2020143845

Rafael Gerardo Couto - 2019142454

Índice

Alínea A

Conversão das imagens para matrizes binárias

De modo a converter as várias imagens para matrizes binárias criámos uma função chamada process_images que recebe como argumento a pasta onde as imagens se encontram (a pesquisa é efetuada dentro das subpastas também) e retorna uma matriz com as várias matrizes binárias das imagens concatenadas (input) e a quantidade de imagens lidas (size).

Para cada imagem encontrada é efetuado o seguinte processo:

- A imagem é lida utilizando a função imread
- É efetuado um redimensionamento à imagem de 224 píxeis para 32 píxeis (quadrados) com recurso à função imresize
- A imagem redimensionada é então convertida para uma matriz binária com o imbinarize
- A matriz é transposta de modo a que os dados fiquem em colunas.
- A matriz binária da imagem é concatenada à matriz input

```
function [input, size] = process_images(folder)

directory = dir(folder);
size = length(directory);

for i = 1 : length(directory)
    % Read image
    img = imread(append(directory(i).folder, "/", directory(i).name));
    img = img(:,:,1);

    % Rezise image from 224px to 32px
    img = imresize(img, [32, 32]);

    % Convert image in binary matrix
    img = imbinarize(img);

    % Transforma a matriz numa coluna
    img = img(:);

    % Adiciona a coluna aos dados
    input(:, i) = img;
end

end
```

Criação e parametrização da rede neuronal

O enunciado pedia-nos que criássemos uma rede neuronal do tipo feedforward com 10 neurónios. Para que isso fosse possível utilizámos a função feedforwardnet(10).
Demos também a possibilidade ao utilizador de escolher outras arquiteturas de redes neuronais como as arquiteturas Pattern e Fitting.
Existem vários parâmetros que podem ser configurados numa rede neuronal, como por exemplo: a função de ativação (transferFcn), a função de treino (trainFcn), a função de divisão (divideFcn) e as iterações (epochs).

```
% Função de Ativação
act_func = "hardlim";


% Função de Treino
trn_func = "trainlm";


% Função de Divisão
dvd_func = "dividerand";


% Épocas (iterações)
epochs = 1000;


net.layers{1}.transferFcn = act_func;
net.trainFcn = trn_func;
net.divideFcn = dvd_func;
net.trainParam.epochs = epochs;
```

Treino e simulação da rede neuronal

Depois de termos a rede neuronal configurada a gosto podemos passar à fase de treino da mesma. Nesta altura fornecemos a matriz binária que gerámos anteriormente e que representa todas as imagens existentes e fornecemos também uma matriz alvo (target). O objetivo da rede neuronal é aproximar-se o máximo possível do target.

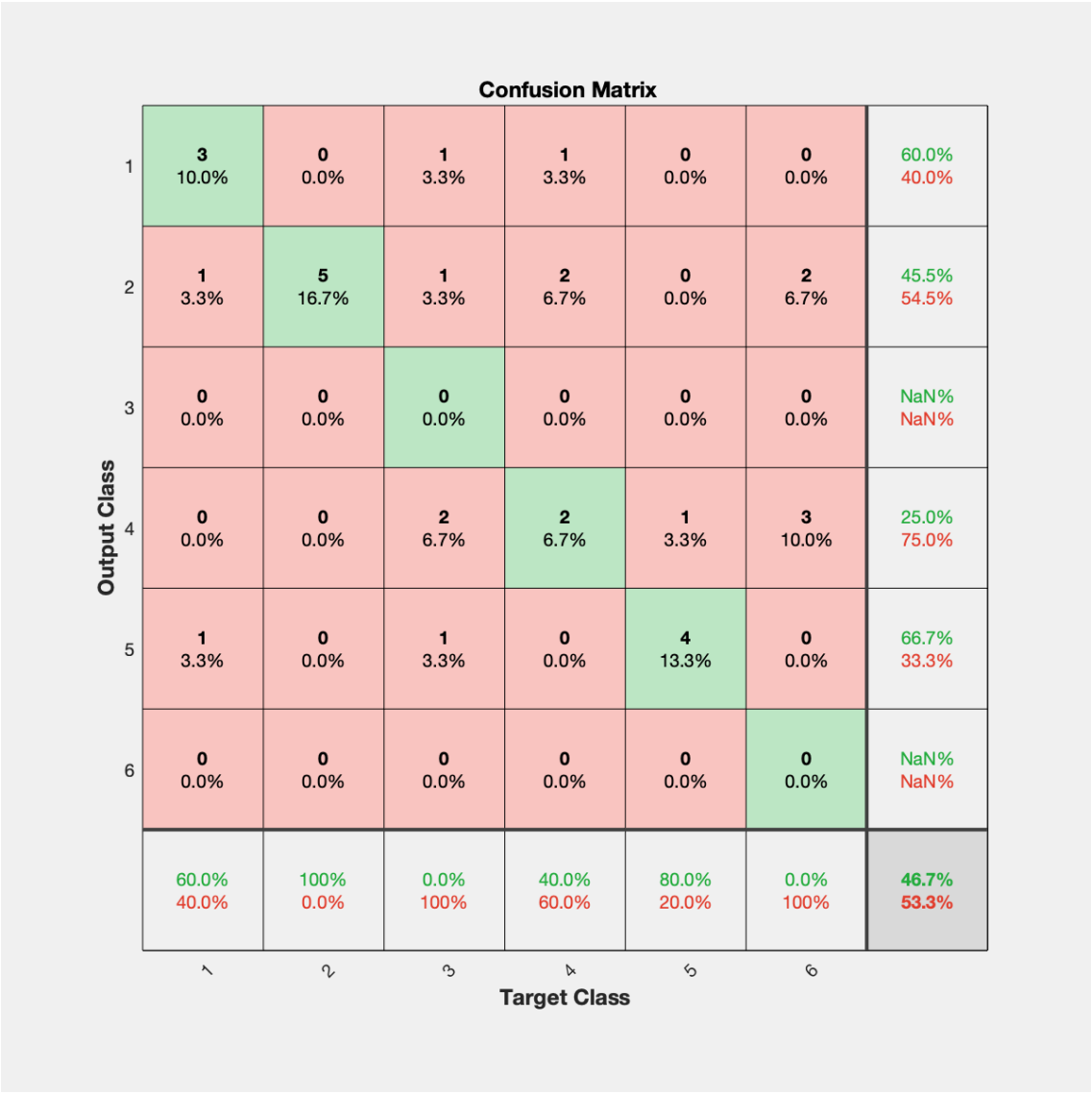
```
net = train(net, input, target);
```

Com a nossa rede treinada chega à altura de simular a rede. Fazemos isto 5 vezes para podermos observar o quão precisa é a rede neuronal que treinámos.
A função sim recebe como argumentos a rede treinada e a matriz de input com as imagens convertidas em binário. O retorno desta função é o output da rede neuronal que corresponde à matriz aproximada ao target.
Para gerarmos valores de desempenho utilizamos um ciclo em que sempre que um valor da matriz de output corresponda ao valor da matriz target adicionamos um valor ao r.

```
nSim = 5;
accuracyFinal = 0;


for i=1 : nSim
    out = sim(net, input);
    plotconfusion(target, out);
    r = 0;
    for j=1 : size(out, 2)
        [a, b] = max(out(:,i));
        [c, d] = max(target(:,i));
        if(b == d)
            r = r + 1;
        end
    end
    accuracy = r/size(out,2)*100;
    fprintf('Precisão do teste (%d): %.3f\n', i, accuracy);
    accuracyFinal = accuracyFinal + accuracy;
end
```

A matriz de confusão mostra-nos visualmente o desempenho da simulação da rede neuronal.



Alínea B

Segmentação do Dataset

Era pedido no enunciado para segmentar o dataset pelas várias fases do modelo (70% para treino, 15% para validação e 15% para teste).
Para que isto fosse possível bastou-nos configurar os vários parâmetros da estrutura divideParam da variável da rede neuronal (net).

```
net.divideParam.trainRatio = 0.70;
net.divideParam.valRatio = 0.15;
net.divideParam.testRatio = 0.15;
```

Precisão total e de teste

Ao treinar a rede com os parâmetros abaixo apresentados atingimos uma média global dos testes de 75.56% de desempenho. Consideramos que este valor é bastante satisfatório mas mesmo assim, nos próximos passos, vamos parametrizar a rede de maneira diferente de modo a obter um desempenho melhor.

```
net.layers{1}.transferFcn = "logsig";
net.trainFcn = "trainlm";
net.divideFcn = "dividerand";
```

```
Média global final dos [Exemplos] depois de 1 testes: 92.667;
Média global final dos [Testes] depois de 1 testes: 75.556;
```

Melhor rede neuronal

A rede neuronal com melhor desempenho obteve 98,33% de precisão global e 75% de precisão nos testes, demorando 1m40s a ser treinada.

Os parâmetros da rede são os seguintes:

- **Número de camadas escondidas:** 1
- **Número de neurónios:** 10
- **Funções de ativação:** tansig, purelin
- **Função de treino:** tansig
- **Divisão dos exemplos:** dividerand (0.95, 0.025, 0.025)

Alínea C

Sem re-treinar a rede

Ao usarmos um dataset diferente do que foi usado para treinar a rede, e sem re-treinar a mesma, obtivemos uma precisão global de 70%.

```
% Resultado sem re-treinar a rede:
70.000%
```

A re-treinar a rede apenas com as imagens da pasta "test"

Re-treinar a rede com as imagens da pasta "test", em teoria, deverá apresentar uma maior precisão pois existem 50 imagens de cada forma geométrica e maiores datasets levam a uma maior precisão normalmente.

```
folder = "imagens/test/*/*.png";
load("best_nn.mat", "net");
[net, tr] = train(net, input, target);

% Precisão total nos exemplos: 96.667%
```

Testar a rede com as imagens da pasta "start"

Ao testar a rede apenas com as imagens da pasta "start" obtivemos uma precisão global de 60%.

```
% Resultado a testar a rede re-treinada, só com as imagens de "start":
60.000%
```

Testar a rede com as imagens da pasta "train"

Ao testar a rede apenas com as imagens da pasta "train" obtivemos uma precisão global de 84.333%.

```
% Resultado a testar a rede re-treinada, só com as imagens de "train":
84.333%
```

Testar a rede com as imagens da pasta "test"

Ao testar a rede apenas com as imagens da pasta "test" obtivemos uma precisão global de 96.667%.

```
% Resultado a testar a rede re-treinada, só com as imagens de "test":
96.667%
```

A re-treinar a rede com os todas as imagens

Como previsto no tópico anterior, era suposto termos precisões mais elevadas graças ao maior número de imagens e, como esperado, foi o que se verificou.
Neste tópico treinámos a rede com todas as imagens fornecidas e testámos a rede para cada pasta separadamente, mas, estranhamente obtivemos uma precisão menor.

```
% Precisão total nos exemplos: 61.520%
```

Testar a rede com as imagens da pasta "start"

Ao testar a rede apenas com as imagens da pasta "start" obtivemos uma precisão global de 26.667%.

```
% Resultado a testar a rede re-treinada com o dataset todo, só com as
imagens de "start":
26.667%
```

Testar a rede com as imagens da pasta "train"

Ao testar a rede apenas com as imagens da pasta "train" obtivemos uma precisão global de 33.000%.

```
% Resultado a testar a rede re-treinada com o dataset todo, só com as
imagens de "train":
33.000%
```

Testar a rede com as imagens da pasta "test"

Ao testar a rede apenas com as imagens da pasta "test" obtivemos uma precisão global de 25%.

```
% Resultado a testar a rede re-treinada com o dataset todo, só com as
imagens de "test":
25.000%
```

Observação: achamos bastante estranho a precisão global ter diminuído apesar de termos aumentado o dataset.

Alínea D

Nesta alínea era-nos pedido para desenharmos formas geométricas semelhantes às fornecidas e testar a melhor rede obtida na alínea anterior.
Desenhámos 3 imagens para cada forma geométrica, eis alguns exemplos:

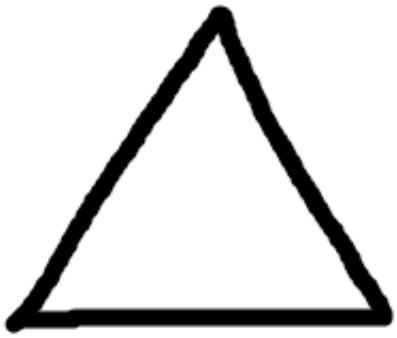
A melhor rede obtida foi a rede re-treinada apenas com as imagens da pasta “test”, rede esta que obteve uma precisão global de 96.667%.

```
folder = "imagens/manuais/*/*.png";
image =
"imagens/manuais/triangle/triangle-
manual-0.png";
shapeLabel = ["Círculo" "Papagaio"
"Paralelogramo" "Quadrado" "Trapézio"
"Triângulo"];
shapeLabel = shapeLabel.';
input = process_images(folder);
image = process_images(image);

load("best_net_c.mat", "net");

out = sim(net, image);
[M, i] = max(out);

fprintf('A imagem introduzida é
considerada um %s com uma precisão de
%.2f% ', upper(shapeLabel(i)), M * 100);
```

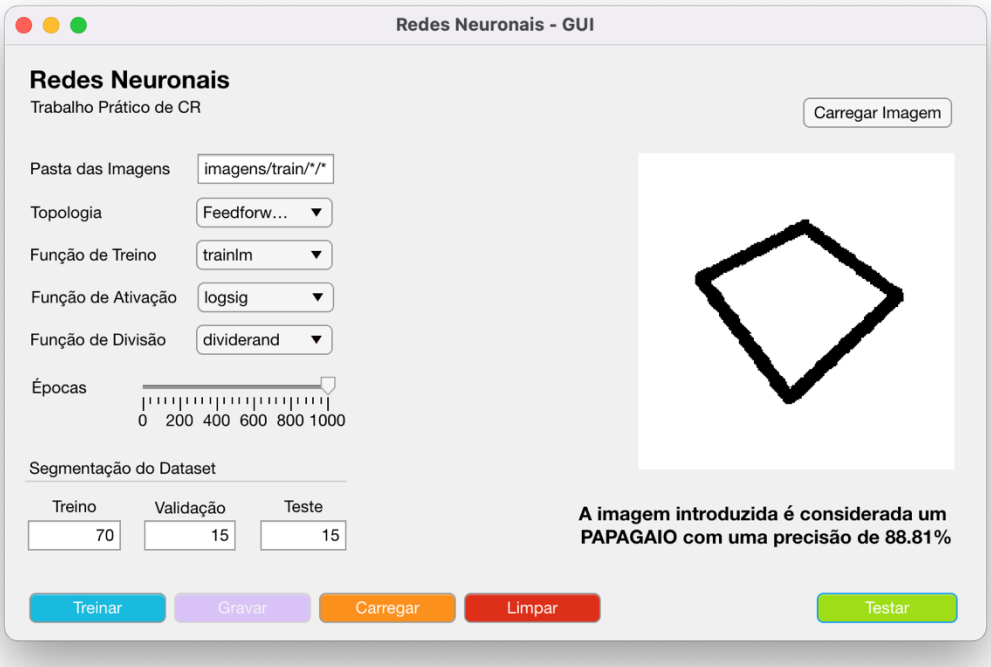


Dando a imagem em cima representada à melhor rede neuronal da alínea C, a mesma concluiu que a figura se tratava de um TRIÂNGULO com uma precisão de 68.59%.

```
> A imagem introduzida é considerada um TRINÂNGULO com uma precisão de
68.59%
```

Alínea E

Para permitir que qualquer utilizador interagisse com o nosso programa de forma fácil e intuitiva, criámos uma aplicação com interface gráfica (GUI) no MATLAB.



A aplicação conta com várias funcionalidades tais como:

- Configurar a rede neuronal quanto à sua topologia, função de treino, função de ativação, função de divisão, épocas e segmentação do dataset
- Treinar a rede neuronal
- Gravar uma rede neuronal previamente treinada
- Carregar uma rede neuronal previamente treinada
- Carregar uma imagem e testar a rede neuronal de forma a classificar a forma geométrica

Treinar a rede neuronal

Após termos utilizado os dropdowns e os sliders para configurarmos a rede neuronal a gosto, basta clicar no botão **Treinar** para começar a treinar a rede neuronal.
Ao clicar neste botão é chamada a função TreinarBotaoPushed que começa por processar as imagens da pasta de imagens indicada previamente (transforma as imagens em matrizes binárias) e seguidamente configura a rede neuronal utilizando a função loadNet.

```
function TreinarButtonPushed(app, event)
     folder = app.PastadasImagensEditField.Value;
     [input, tamanho] = process_images(folder);
     target = gen_target(tamanho);
     app.Target = target;

     type = app.TopologiaDropDown.Value;
     transferFcn = app.FunodeAtivaoDropDown.Value;
     trainFcn = app.FunodeTreinoDropDown.Value;
     divideFcn = app.FunodeDivisoDropDown.Value;
     epochs = app.pocasSlider.Value;

     net = loadNet(type, transferFcn, trainFcn, divideFcn, epochs);
     app.TreinarButton.Enable = 'off';
     app.CarregarButton.Enable = 'off';
     app.GravarButton.Enable = 'off';
     app.TestarButton.Enable = 'off';
     [net, tr] =  train(net, input, target);
     app.Net = net;
     app.TreinarButton.Enable = 'on';
     app.CarregarButton.Enable = 'on';
     app.GravarButton.Enable = 'on';
     app.TestarButton.Enable = 'on';
end


function net = loadNet(type, transferFcn, trainFcn, divideFcn, epochs)

switch(type)
    case "Feedforward (10 neur√≥nios)"
        net = feedforwardnet;
    case "Pattern (10 neur√≥nios)"
        net = pattern;
    case "Fitting (10 neur√≥nios)"
        net = fitting;
end

net.userdata = type;
net.layers{1}.transferFcn = transferFcn;
net.trainFcn = trainFcn;
net.divideFcn = divideFcn;

net.trainParam.epochs = epochs;

end
```

Carregar uma rede neuronal

Para carregar uma rede neuronal previamente treinada criámos a função CarregarButtonPushed que é chamada sempre que o botão **Carregar** é premido.

```
function CarregarButtonPushed(app, event)
      [file,path] = uigetfile('*.mat');
      if(path ~= 0)
            load(append(path, "/", file), "net");
             app.Net = net;
       end

       switch(app.Net.userdata)
            case "Feedforward (10 neurónios)"
                  app.TopologiaDropDown.Value = "Feedforward";
            case "Pattern (10 neurónios)"
                  app.TopologiaDropDown.Value = "Pattern";
            case "Fitting (10 neurónios)"
                  app.TopologiaDropDown.Value = "Fitting";
            otherwise
                  app.TopologiaDropDown.Value = "Feedforward";
                  msgbox('O programa não reconhece essa arquitetura
                  de rede neuronal', 'Error', 'Error');
                  app.Net = [];
                  return;
      end

       app.FunodeAtivaoDropDown.Value =
       app.Net.layers{1}.transferFcn;
       app.FunodeTreinoDropDown.Value = app.Net.trainFcn;
       app.FunodeDivisoDropDown.Value = app.Net.divideFcn;
       app.pocasSlider.Value = app.Net.trainParam.epochs;
       app.TestarButton.Enable = 'on';
end
```

Esta função pede ao utilizador para selecionar um ficheiro MATLAB que contenha a variável net que corresponde à rede neuronal treinada. Seguidamente, a rede neuronal é carregada para a variável global Net e os valores dos sliders e dos dropdowns são atualizados com os valores correspondentes da rede carregada.

Gravar rede neuronal

A funcionalidade de gravar a rede neuronal treinada fez-se com relativa facilidade. Bastou-nos perguntar ao utilizador onde é que queria guardar o ficheiro que continha a rede neuronal e após serem feitas algumas verificações (como por exemplo: "existe alguma rede neuronal treinada de momento?" ou "o caminho especificado é válido?") e se tudo se verificar é utilizada a função save do MATLAB para guardar a rede.
A função que efetua todo este trabalho é chamada quando o utilizador clica no botão **Guardar**.

```
function GravarButtonPushed(app, event)
      if(~isempty(app.Net))
            file, path] = uiputfile({'*.mat'}, "Guardar rede
      neuronal", "net.mat");
            if(path ~= 0)
                  net = app.Net;
                  save(append(path, "/", file), 'net');
            end
      else
            msgbox('Não existe nenhuma rede para ser guardada',
            'Error','error');
      end
end
```

Carregar imagem para análise

O processo de carregar uma imagem para análise pela rede neuronal é feito utilizando a função do MATLAB uigetfile que permite ao utilizador escolher um ficheiro (neste caso uma imagem). Esta função retorna o caminho e o nome do ficheiro selecionados e com esta informação basta processarmos a imagem para a transformarmos numa matriz binária que vai ser utilizada como input na simulação da rede.

```
function CarregarImagemButtonPushed(app, event)
      [file, path] = uigetfile('*.png');
      if(path ~= 0)
            app.Imagem = imread(append(path, "/", file));
            app.Imagem = app.Imagem(:,:,1);
            imshow(app.Imagem, 'parent', app.UIAxes);
            app.Imagem = imresize(app.Imagem, [32 32]);
            app.Imagem = imbinarize(app.Imagem);
            app.Imagem = app.Imagem(:);
      end
end
```

Testar a rede neuronal com uma imagem

Finalmente, para testarmos a rede neuronal, basta clicar no botão **Testar**. Para que possamos correr o teste é necessário que exista uma rede carregada e treinada e é necessário também que o utilizador tenha escolhido uma imagem para o programa classificar.

Após clicar no botão Testar, é chamada a função TestarButtonPushed que, após efetuar as verificações acima mencionadas, classifica a imagem utilizando a função de simulação, sim, que tem como parâmetros a rede neuronal treinada e a imagem objetivo.

```
function TestarButtonPushed(app, event)
      if(isempty(app.Net))
            msgbox('Não existe nenhuma rede neuronal ativa',
            'Error', 'Error');
      elseif(isempty(app.Imagem))
            msgbox('Não foi carregada nenhuma imagem', 'Error',
            'Error');
      else
            out = sim(app.Net, app.Imagem);
            [M, i] = max(out);
            app.Descricao.Text = sprintf("A imagem introduzida é
            considerada um %s com uma precisão de %2.2f%%",
            upper(app.Figuras(i)), (M * 100));
            app.Out = out;
      end
end
```