Implementação e análise do desempenho de redes neurais convolucionais (CNNs) no Reconhecimento de Faces – 2021/2*

*Nota: Projeto da disciplina Tópicos Avançados de Aprendizagem de Máquina do 2o. Semestre de 2021

Aluno: Carlos Fonseca
Programa de Pós Graduação em Eng. Elétrica
PPGEE/UFAM
Manaus/AM, Brasil
henrique_fonseca@hotmail.com

Aluno: Willian Guerreiro
Programa de Pós Graduação em Eng. Elétrica
PPGEE/UFAM
Manaus/AM, Brasil
wguerreiro31@gmail.com

***Resumo*—A crescente demanda por soluções de reconhecimento facial nos mais diversos segmentos do comércio, indústria e serviços, visando otimizar a identificação, o controle e rastreamento biométrico de usuários, tem motivado a pesquisa e o desenvolvimento de algoritmos computacionais que alcancem índices de acurácia elevados, o mais próximo possível do ideal: 100%. Neste trabalho, foram estudadas e implementadas arquiteturas de redes neurais profundas convolucionais (CNNs) e comparado seu desempenho com as redes de propagação direta utilizando o método 2D2PCA com distância euclidiana bidimensional proposto pelos autores em pesquisa anterior [5]. Utilizou-se a base de dados pública ORL composta de 400 faces (40 pessoas, com 10 imagens de cada pessoa). Dividindo-se o conjunto de dados para treinamento e teste, respectivamente em 70% e 30%, a arquitetura da rede neural CNN proposta com múltiplas camadas de convolução obteve melhores resultados do que técnica 2D2PCA para reconhecimento de faces, tendo alcançado o máximo de 99,17% de acurácia, superando os resultados alcançados pela CNN proposta por KAMENCAY et al [3].**

***Index Terms*—redes neurais convolucionais, reconhecimento facial, análise de componentes principais, distância euclidiana bidimensional.**

I. Introdução

Uma tendência na área de Inteligência Artificial tem sido a pesquisa por modelos de Visão Computacional que melhor representem as características de faces humanas de forma que uma máquina consiga reproduzir a habilidade do cérebro de aprender e, instantaneamente, reconhecer o rosto de pessoas em posições, momentos e condições diversas. Os modelos já desenvolvidos e testados, que se encontram no estado da arte, deram origem a novas tecnologias de biometria por Reconhecimento Facial, revolucionando o mercado de segurança eletrônica, controle de acessos, rastreamento de pessoas, identificação criminal, dentre outros. As técnicas utilizadas nos sistemas de reconhecimento facial dependem do tipo de aplicação. Há várias categorias de problemas, por exemplo:

1. Reconhecimento Facial 1:N : consiste em encontrar uma pessoa em um grande banco de dados de "N" faces (por exemplo, em um banco de dados policial). Esses sistemas normalmente retornam uma lista das pessoas mais prováveis no banco de dados pelo grau de similaridade. Nessas aplicações, usualmente, apenas uma imagem está disponível por pessoa e o processamento não ocorre em tempo real.

2. Reconhecimento Facial 1:N em tempo real: identificar e autorizar o acesso de pessoas específicas em um sistema de monitoramento de segurança, controle de entrada/saída, rastreamento, etc., ou permitir o acesso a um grupo de pessoas previamente registradas e negar o acesso a todos os outros. Por exemplo, acesso a um edifício, empresa, etc. Muitas imagens por pessoa estão frequentemente disponíveis para o treinamento e reconhecimento em tempo real.

3. Autenticação Facial 1:1 : esse tipo de algoritmo é utilizado para analisar a similaridade de uma face obtida em tempo real de um indivíduo pré-identificado por algum registro informado ao sistema (ex: nº de matrícula, RG, CPF, etc). O algoritmo processa e analisa as imagens armazenadas em um banco de dados daquela mesma pessoa, a fim de concluir se de fato corresponde a ela.

Neste artigo, estamos principalmente interessados no segundo caso, em que é necessário reconhecer um indivíduo em condições diversas de expressão facial, da posição do rosto, pose, etc. O diferencial está na classificação rápida e, portanto, presume-se que não haja tempo disponível para extenso pré-processamento e normalização.

Conforme exemplificado na figura 1, KAMENCAY et al [3] compararam o desempenho das três principais técnicas de reconhecimento facial: PCA (Análise de Componentes Principais), KNN (k-vizinhos mais próximos) e LBPH (Histograma de Padrões Binários Locais) com a técnica de redes neurais convolucionais (CNNs). Realizando experimentos sobre o conjunto de dados de faces ORL, com 400 imagens, concluíram que tais redes profundas, quando construídas com arquiteturas complexas, apesar do maior custo computacional, superaram

as demais técnicas tendo alcançado 98,3% de acurácia.

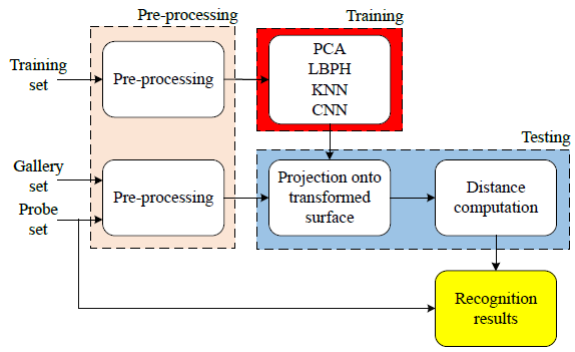

Figura 1. Exemplo de Sistema de Reconhecimento Facial [3]

Neste trabalho, implementamos diferentes arquiteturas de redes convolucionais objetivando otimizar os hiperparâmetros e comparar os resultados com a técnica 2D2PCA utilizando a distância euclidiana bidirecional apresentada em trabalho anterior [5], assim como com os resultados apresentados por KAMENCAY et al. [3].

Na seção 2 será apresentada uma breve Fundamentação Teórica sobre redes neurais profundas convolucionais. Na seção 3, a Metologia adotada para e implementação em Matlab, na seção 4 a comparação dos Resultados encontrados e, por fim, na seção 5 as Conclusões sobre a aplicação do método no problema proposto.

II. Fundamentação Teórica

As Redes neurais convolucionais (CNNs) são "redes neurais especializadas em tratar dados que possuem uma topologia matricial" [2], a exemplo de imagens. São assim denominadas pois utilizam a operação matemática de convolução no lugar de multiplicação geral da matriz em pelo menos uma de suas camadas.

Em uma rede CNN, a imagem de entrada passa por uma série de camadas, entre elas: convolução com filtros (kernels), normalização (Batch Normalization), ativação ReLU, Pooling e, por último, uma camada totalmente conectada (Fully Connected) aplicando a função Softmax para classificar um objeto, na saída da rede, com valores probabilísticos entre 0 e 1.

A figura 2 apresenta um fluxo completo de camadas de uma rede CNN utilizada para processar uma imagem:

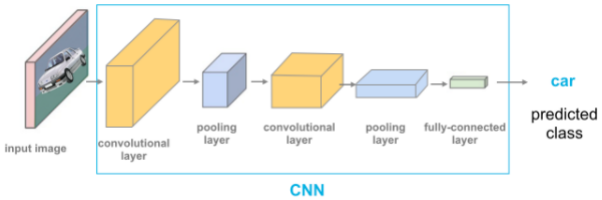

Figura 2. Estrutura de uma Rede Neural Convolucional

A. Camada de Convolução

Em aplicações de aprendizado de máquina, tais como classificação de imagens, detecção de objetos e reconhecimento facial, a entrada geralmente é uma matriz multidimensional de dados e o kernel, uma matriz multidimensional de parâmetros que são ajustados pelo algoritmo de aprendizagem. Na equação 1, apresentamos a forma discreta da convolução de uma imagem I 2D por um kernel K 2D :

$$S(i,j) = (K*I)(i,j) = \sum_m \sum_n I(i-m, j-n).K(m,n) \quad (1)$$

Em processamento digital de imagens, denomina-se de "filtragem espacial" o processo de convolução de uma imagem de entrada por um filtro (ou kernel) obtendo-se como resultado o mapa de atributos. Por exemplo, quando usado um filtro derivativo, os atributos de saída serão as bordas da imagem. Em uma CNN, a primeira camada é a de convolução que tem por objetivo extrair os atributos da imagem de entrada.

B. Camada de ativação ReLU

Para cada um dos valores dos Mapas de Atributos obtidos após a camada de convolução, aplica-se a função de ativação ReLU, $R(z) = max('threshold', z)$, a fim passar somente os valores acima do limiar ('threshold') à próxima camada.

Segundo AGGARWAL [1], constatou-se que o uso da função de ativação ReLU (Rectified Linear Unit) traz grandes vantagens sobre as funções de ativação de saturação (ex: sigmóide e tanh) em termos de velocidade e precisão.

C. Camada de Pooling

As camadas de pooling são colocadas entre as camadas de convolução a fim de reduzir a dimensão das imagens de entrada, atribuindo-se um valor máximo ou médio a cada sub região dos dados de entrada. A Figura 3 ilustra como o max pooling opera. Assim, segundo [4] o "objetivo de usar a camada de pooling é minimizar a quantidade de cálculos mantendo os dados mais importantes para a camada seguinte".

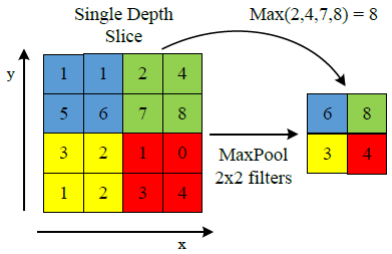

Figura 3. Max Pooling [3]

D. Camadas Totalmente Conectadas - Fully Connected

As últimas camadas de uma arquitetura de redes CNN usam camadas totalmente conectadas com o objetivo de obter a classificação. Esta camada funciona exatamente da mesma maneira que uma rede feed-forward tradicional. Mesmo que

as camadas convolucionais tenham um maior número de ativações, as camadas totalmente conectadas geralmente têm um número maior de conexões.

E. Camada de regressão SoftMax

O classificador softmax, que tem uma forte capacidade de classificação não linear, é usado na última camada da rede. Tendo em vista que as saídas da camada inteiramente conectada assumem valores reais que não estão necessariamente no intervalo entre 0 e 1, faz-se necessário a utilização de uma função capaz de reescalonar essa faixa de valores, a fim de caracterizar as classes.

III. METODOLOGIA

Nesta seção serão descritos os procedimentos metodológicos necessários à solução do problema proposto. Serão apresentadas as ferramentas de software, o método utilizado para classificação com base em redes neurais convolucionais, as topologias de redes utilizadas, bem como os parâmetros de simulação.

A. Ferramentas computacionais

A principal ferramenta utilizada neste trabalho foi o *Deep Learning Toolbox* do *software* MATLAB, que possui um conjunto de recursos voltados para o treinamento, projeto e simulação de redes neurais de múltiplas camadas. A seguir são apresentadas as principais funções utilizadas neste trabalho.

1) imageDatastore: Um recurso que possibilita gerenciar uma coleção de arquivos de imagens, sendo útil inclusive para associar os nomes das classes aos nomes das pastas onde as imagens estão armazenadas, o que facilita o processo de anotação.
2) splitEachLabel: Com este recurso é possível realizar a segmentação do banco de imagens original em subconjuntos menores, isto é, treino e teste. É possível também especificar a proporção desta divisão.
3) imageInputLayer: Definição da camada de entrada, sendo especificados seu nome e a dimensão dos dados.
4) convolution2dLayer: Definição da camada de convolução 2D, sendo especificados o número de kernels, sua ordem e o tipo de padding.
5) batchNormalizationLayer: Definição da camada de normalização.
6) reluLayer: Definição da função de ativação do tipo ReLu.
7) maxPooling2dLayer: Definição da camada de Pooling 2D, sendo especificados o pooling e o Stride, ou passo.
8) fullyConnectedLayer: Definição da camada inteiramente conectada com as anteriores, sendo especificado o número de saídas, que para este problema corresponde ao número de indivíduos, isto é, 40.
9) softmaxLayer: Penúltima camada, responsável por associar as respectivas ativações a um número que varia entre 0 e 1, significando, portanto, a probabilidade do elemento apresentado à rede pertencer à determinada classe.
10) classificationLayer: A partir da maior ativação da camada softMax, associa a saída a uma label, isto é, classifica o indivíduo que foi apresentado à rede.
11) trainingOptions: Retorna um objeto com as definições dos parâmetros de treinamento, como função de perda, conjunto de dados de validação e se o usuário deseja visualizar os gráficos de convergência.
12) trainNetwork: Através deste comando, parametrizado com os dados, as camadas e as opções de treinamento, é possível realizar o treinamento da rede de forma a obter os pesos e polarizações das respectivas camadas.
13) activations: Com este comando é possível observar quais áreas da imagem são ativadas em uma camada convolutiva. Desta maneira, é possível associar quais características estão sendo extraídas por cada kernel/canal.
14) GRADCAM: De maneira complementar à função activations, a função GRADCAM retorna quais partes da imagem são mais importantes para a classificação, isto é, quais características mais impactam na tomada de decisão.

B. Descrição do problema proposto

A base de dados "ORL" é constituída por 40 pastas, sendo cada pasta um conjunto de dados isolado contendo 10 imagens da face de uma pessoa, sob diferentes condições de iluminação e posicionamento. Cada imagem possui originalmente as dimensões 112x92 pixels.

Desta maneira, cada imagem do conjunto de treinamento é submetida à primeira camada de uma Rede Neural Convolucional. Com este modelo objetiva-se classificar corretamente um indivíduo dentre 40 de um banco de dados pré-estabelecido, sendo essa classificação realizada por meio da fotografia de sua face. O modelo será treinado com 7 faces de cada indivíduo e testado com as 3 faces remanescentes. Para este processo de aprendizado de máquina serão avaliadas a taxa de convergência, a matriz de confusão e a acurácia do modelo.

C. Implementação do modelo

Para a construção do modelo de aprendizado de máquina, foi desenvolvido um script na linguagem MATLAB, tendo o suporte do pacote de Aprendizado Profundo disponibilizado em sua biblioteca.

Inicialmente foi realizada a leitura do conjunto de dados, seguida do seu particionamento na razão 7:3, ou seja, para cada pasta 7 elementos constituem o conjunto de treino e 3 elementos constituem o conjunto de teste, totalizando 280 elementos no conjunto de treino e 120 elementos no conjunto de teste.

Após a definição dos conjuntos de treino e teste, definiu-se a arquitetura da rede a ser utilizada. Neste trabalho foram avaliadas 3 arquiteturas: uma arquitetura de porte simples (ARQ1), uma arquitetura de porte médio (ARQ2) e uma arquitetura expandida (ARQ3), isto é, com maior número de camadas.

Para a realização do fluxo de aprendizado de máquina, foram seguidos os seguintes passos:

- Definição das camadas: Nesta aplicação foram utilizadas camadas de convolução, normalização de batch, Pooling, camadas com função de ativação do tipo ReLu, camada inteiramente conectada e função softmax na última camada, para fins de classificação. Estas camadas foram reorganizadas para cada topologia de rede adotada.
- Escolha da função de otimização: Através do método "trainningOptions" foi possível especificar o método "ADAM" como otimizador da função de custo.
- Início do processo de treinamento e escolha do melhor modelo com base na acurácia ao realizar a inferência no conjunto de teste.

O diagrama exposto na figura 4 ilustra a sequência de passos necessários ao treinamento do modelo de classificação de faces baseado em redes neurais convolucionais.

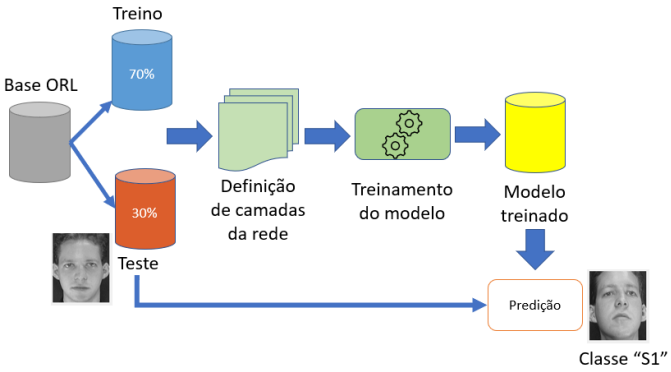

Figura 4. Sequência de passos para treinamento e depuração do modelo

D. *Arquiteturas de rede utilizadas*

Para a análise e verificação do impacto das arquiteturas de rede sobre o desempenho do modelo, foram estudadas 3 topologias de rede, com diferentes distribuições de camadas. Conforme os estudos de KAMENCAY et al [3], melhor é o desempenho do modelo quanto mais densas forem suas camadas, ainda que isto incorra em um maior custo computacional.

Para as redes exibidas na figura 5, tem-se que as abreviações BN, Conv e FC correspondem às camadas de *batch normalization*, *convolução* e *full connected*.

As redes ARQ1 e ARQ3 foram sugeridas como modelos simplificado e estendido, respectivamente. Neste trabalho propôs-se a rede ARQ2 sendo obtida como um intermediário entre ARQ1 e ARQ3, obtendo resultados de acurácia superiores a 95%.

Dentre as características da rede ARQ2, pode-se citar que diferente da rede ARQ1 que utiliza 32 kernels de convolução com as dimensões 5x5, optou-se por utilizar duas camadas de extração de características, ambas com 16 kernels de convolução, com dimensões 3x3.

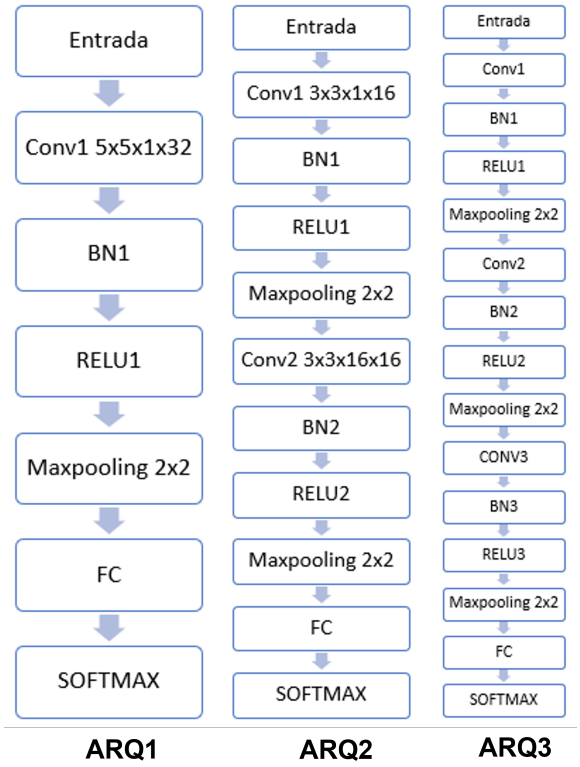

Figura 5. Arquiteturas de rede utilizadas

IV. Resultados

O processo de obtenção de resultados compreendeu os experimentos realizados com as 3 arquiteturas de redes apresentadas. Deve-se ressaltar que a seleção de elementos para o conjunto de treinamento e teste deu-se de maneira aleatória, de forma que foram adotados os resultados das rodadas de melhor desempenho do modelo.

Como ponto de partida, foi realizado um estudo a respeito da maneira como a proporção dos conjuntos de treino e teste influenciam no desempenho do modelo. E para isso, foi considerada apenas a arquitetura 3 (ARQ3). A métrica utilizada para este experimento foi a acurácia, e os resultados podem ser vistos na tabela I, indicando um melhor desempenho para a razão 70/30.

Tabela I
Relação entre base de treinamento e desempenho do modelo

Razão treino/teste	Exper. 1	Exper. 2	Exper. 3	Média	Acurácia Máxima
(50% / 50%)	88,50%	86%	81,50%	85,33%	88,50%
(60% / 40%)	90%	76,68%	83,75%	83,47%	90%
(70% / 30%)	97,50%	96,67%	99,17%	97,78%	*99,17%
(80% / 20%)	97,50%	95%	96,25%	96,25%	97,50%

*maior resultado obtido neste paper

Como parte fundamental do processo de entendimento do mecanismo de extração de características, foi utilizada a função "activations", capaz de permitir a visualização das regiões que sofreram ativações em determinada camada da rede. Como exemplo, a figura 6 ilustra o resultado da interação entre uma imagem com cada um dos 32 kernels da primeira camada convolutiva.

Figura 6. Saída de cada kernel na primeira camada convolutiva

O processo de treinamento das redes de arquiteturas ARQ1, ARQ2 e ARQ3 pôde ser acompanhado em tempo real. Os respectivos gráficos contendo a variação da acurácia e a perda ao longo do treinamento podem ser vistos nas figuras 7, 8 e 9.

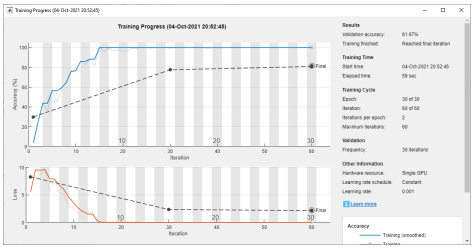

Figura 7. Curvas de treinamento para a arquitetura 1

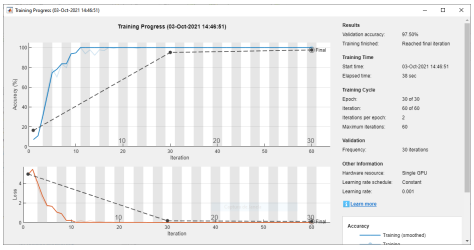

Figura 8. Curvas de treinamento para a arquitetura 2

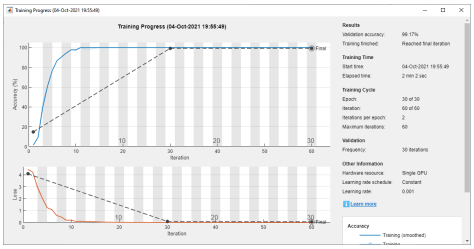

Figura 9. Curvas de treinamento para a arquitetura 3

Ao fim do treinamento, foi elaborada tabela a fim de comparar os resultados provenientes das 3 arquiteturas estudadas, bem como o método de classificação baseados em distância Euclidiana exposto por [5] e os avanços alcançados por [3].

Tabela II
COMPARAÇÃO ENTRE DIFERENTES AUTORES

Trabalho	Acurácia
2D2LDA + Distância Euclidiana [5] (FONSECA, C., GUERREIRO, W.)	98,33%
Arquitetura 1 (ARQ1)	81,67%
Arquitetura 2 (ARQ2)	97,50%
Arquitetura 3 (ARQ3)	99,17%
CNN [3] (KAMENCAY, P. et al.)	98,33%

V. CONCLUSÕES

A implementação da rede neural convolucional mostrou-se eficaz na resolução do problema apresentado. Os conceitos vistos em sala de aula se fizeram presentes no entendimento da função de cada camada na arquitetura da rede, como pôde ser notado na figura 6, que ilustra a extração de bordas e outros detalhes da imagem ainda na primeira camada. Durante o processo de modelagem do classificador foi possível depreender que o software MATLAB dispõe de recursos avançados tanto para o monitoramento do treinamento da rede, quanto para a visualização de suas camadas. De acordo com a experiência obtida e com os dados levantados pela literatura, tornou-se claro que estruturas mais densas tendem a proporcionar resultados melhores, porém introduzem maior carga computacional, influenciando inclusive no tempo de treinamento do modelo. Da análise da tabela II, é possível notar que a arquitetura 3 superou os resultados alcançados por [5] e por [3], porém deve-se observar que esta acurácia não ocorreu com frequência, o que não descarta a qualidade do treinamento, mas sugere um aumento na base e realização de validação cruzada.

REFERÊNCIAS

[1] AGGARWAL, C. "Neural Networks and Deep Learning", Springer International Publishing, 2018.

[2] GOODFELLOW, I. et al. "Deep Learning", MIT Press, 2016.

[3] KAMENCAY, P. et al. "A new method for Face Recognition using Convolutional Neural Network", Digital Image Processing and Computer Graphics, Vol 15, No.04, 2017.

[4] MOHRA, A. et al. "Deep Learning Face Detection and Recognition", Intl Journal of Electronics and Telecommunications, 2019.

[5] FONSECA, C., GUERREIRO, W. "Desempenho de uma Rede de Propagação Direta no Reconhecimento de Faces reduzidas pelos métodos 2D2PCA e 2D2LDA", Programa de pós-graduação em Engenharia Elétrica UFAM - PPGEE, 2021.

ANEXO I

```
%Projeto2: Rede Neural Convolucional
%UFAM/PPGEE/ Aprendizado de Maquina 2021-2
%Carlos Fonseca, Willian Guerreiro

% Carregamento do Dataset
imds = imageDatastore('ORL','IncludeSubfolders',true,'LabelSource','foldernames');

numTrainFiles = 7;

[imdsTrain, imdsValidation] = splitEachLabel(imds, numTrainFiles, 'randomize');

%[imdsTest, imdsValidation] = splitEachLabel(imdsValidation, numTestFiles, 'randomize');

%{
for i=1:100
    img = readimage(imds,i);
    imshow(img);
end
%}


layers = [
imageInputLayer([112 92 1],"Name","imageinput")

convolution2dLayer([5 5],32,"Name","conv1","Padding","same")
batchNormalizationLayer("Name","batchnorm1")
reluLayer("Name","relu1")
maxPooling2dLayer(2,'Stride',2)

convolution2dLayer([5 5],32,"Name","conv2","Padding","same")
batchNormalizationLayer("Name","batchnorm2")
reluLayer("Name","relu2")
maxPooling2dLayer(2,'Stride',2)

convolution2dLayer([5 5],32,"Name","conv3","Padding","same")
batchNormalizationLayer("Name","batchnorm3")
reluLayer("Name","relu3")
maxPooling2dLayer(2,'Stride',2)

fullyConnectedLayer(40,"Name","fc")  %40 Numero de classes
softmaxLayer("Name","softmax")
classificationLayer("Name","classoutput")];

options = trainingOptions('adam', 'ValidationData',imdsValidation, ...
                          'ValidationFrequency',30,'Verbose',true, ...
                          'Plots','training-progress');

op = input('Train?_(y/n)','s');

if strcmp(op,'y') > 0
    net = trainNetwork(imdsTrain,layers,options);
else
    net = net_best;
end

%Fun  o para visualizar a rede:
analyzeNetwork(net)

YPred = classify(net,imdsValidation);
YTest = imdsValidation.Labels;
accuracy = sum(YPred == YTest)/numel(YTest);


%{
for i=1:15
    img = readimage(imdsValidation,i);
    label = classify(net,img);
    scoreMap = gradCAM(net,img,label);

    figure
    imshow(img)
    hold on
    imagesc(scoreMap,'AlphaData',0.5)
    colormap jet
end

%}

%Lendo a primeira imagem do conjunto de valida  o
img_teste = readimage(imdsValidation,1);
%Obtm-se as sa das ap s passada pela camada conv1
act1 = activations(net,img_teste,'conv1');
%Obtm-se as dimens es da estrutura 112x92x32x1 [H x W x Kernels x ColorChannels]
sz = size(act1);
%Configura a estrutura no formato 112x92x1x32
%S o 32 Kernels 112x92 com 1 canal de cores
act1 = reshape(act1,[sz(1) sz(2) 1 sz(3)]);
%Exibi  o dos 32 Kernels em uma  nica grade
I = imtile(mat2gray(act1),'GridSize',[4 8]);
imshow(I);

individuo1 = readimage(imdsValidation,10);
individuo2 = readimage(imdsValidation,30);

label1 = classify(net,individuo1);

label2 = classify(net,individuo2);

scoreMap1 = gradCAM(net,individuo1,label1);

scoreMap2 = gradCAM(net,individuo2,label2);

figure
imshow(individuo1)
hold on
imagesc(scoreMap1,'AlphaData',0.5)
colormap jet
```

```
figure
imshow(individuo2)
hold on
imagesc(scoreMap2,'AlphaData',0.5)
colormap jet

C = confusionmat(YTest,YPred);
figure
confusionchart(C)

%Obten  o de pesos e polariza  es da camada N

W = net.Layers(2).Weights; %Weights of Convolution layer
B = net.Layers(2).Bias; %Biases of Convolution layer
```