Brain Tumor Detection Using Deep Learning (ResNet-18 on MRI Images)

Code:

1):

```
%% Brain Tumor Classification using ResNet-18
% Author: Vijayagiri Shanthipriya
% Description: Trains a ResNet-18 model to classify brain tumors
% (glioma, meningioma, notumor, pituitary)

clc; clear; close all;

%% STEP 1: Set dataset folder path
dataDir ='C:\Users\shanthipriya vijayagiri\OneDrive\Desktop\Matlabproject\Braintumordetection';
if ~isfolder(dataDir)
   error('Data folder not found. Please check path.');
end

%% STEP 2: Create image datastore
imds = imageDatastore('C:\Users\shanthipriya
vijayagiri \OneDrive\Desktop\Matlabproject\', ...
   'IncludeSubfolders', true, ...
   'LabelSource', 'foldernames');

%% STEP 3: Display label distribution
tbl = countEachLabel(imds);
disp('Label counts:');
```

```
disp(tbl);

%% STEP 4: Check and fix grayscale images
inputSize = [224 224 3]; % ResNet-18 input size

imds.ReadFcn = @(filename) readAndFix(filename, inputSize);

function I = readAndFix(filename, inputSize)
  % Read image
  I = imread(filename);
  % Convert grayscale to RGB
  if size(I,3) == 1
    I = cat(3, I, I, I);
  end
  % Resize
  I = imresize(I, inputSize(1:2));
  % Convert to uint8 if needed
  if ~isa(I, 'uint8')
    I = im2uint8(I);
  end
end

%% STEP 5: Split dataset into training and validation
[imdsTrain, imdsVal] = splitEachLabel(imds, 0.8, 'randomized');
```

```
%% STEP 6: Load pretrained ResNet-18
net = resnet18;

%% STEP 7: Modify final layers for 4 classes
numClasses = numel(categories(imds.Labels));
lgraph = layerGraph(net);

newLayers = [
  fullyConnectedLayer(numClasses, 'Name','fc_new')
  softmaxLayer('Name','softmax_new')
  classificationLayer('Name','classoutput')];

lgraph = replaceLayer(lgraph,'fc1000',newLayers(1));
lgraph = replaceLayer(lgraph,'prob',newLayers(2));
lgraph =
replaceLayer(lgraph,'ClassificationLayer_predictions',newLayers(3));

%% STEP 8: Create augmented image datastores
augimdsTrain = augmentedImageDatastore(inputSize(1:2), imdsTrain,
'ColorPreprocessing','gray2rgb');
augimdsVal  = augmentedImageDatastore(inputSize(1:2), imdsVal,
'ColorPreprocessing','gray2rgb');

%% STEP G: Define training options
options = trainingOptions('sgdm', ...
  'MiniBatchSize', 8, ...
```

```
    'MaxEpochs', 8, ...
    'InitialLearnRate', 1e-4, ...
    'Shuffle', 'every-epoch', ...
    'ValidationData', augimdsVal, ...
    'ValidationFrequency', 30, ...
    'Verbose', true, ...
    'Plots', 'training-progress');

%% STEP 10: Train the network
trainedNet = trainNetwork(augimdsTrain, lgraph, options);

%% STEP 11: Save trained model
modelSaveFile = fullfile(pwd, 'trainedResNet18_brain.mat');
save(modelSaveFile, 'trainedNet');

%% STEP 12: Evaluate performance
YPred = classify(trainedNet, augimdsVal);
YValidation = imdsVal.Labels;

accuracy = mean(YPred == YValidation);
fprintf('\nValidation accuracy: %.2f%%\n', accuracy * 100);

%% STEP 13: Confusion matrix
figure;
plotconfusion(YValidation, YPred);
```

```
title('Confusion Matrix - Brain Tumor Classification');
2):
clc;
clear;
close all;

% Step 1: Load trained network
modelFile = 'trainedResNet18_brain.mat';
if isfile(modelFile)
    load(modelFile, 'trainedNet');
    disp(' Trained model loaded successfully.');
else
    error('+ Trained model file not found!');
end

% Step 2: Select image
[filename, pathname] = uigetfile({'*.jpg;*.png;*.jpeg'}, 'Select a Brain
MRI Image');
if isequal(filename,0)
    disp('No image selected.');
    return;
end
imgPath = fullfile(pathname, filename);
img = imread(imgPath);
figure, imshow(img), title('Input Brain MRI Image');
```

```
% Step 3: Resize image to match network input
img_resized = imresize(img, [224 224]);

% Step 4: Predict using trained model
[predictedLabel, scores] = classify(trainedNet, img_resized);
disp(['🩺 Predicted Tumor Type: ', char(predictedLabel)]);

% Step 5: Convert to grayscale for area detection
grayImg = rgb2gray(img);

% Step 6: Apply segmentation to highlight tumor region
bw = imbinarize(grayImg, 'adaptive');
bw = imfill(bw, 'holes');
bw = bwareaopen(bw, 80); % remove small noise

% Step 7: Calculate tumor area
tumorArea = bwarea(bw);
disp(['📏 Tumor Area (in pixels): ', num2str(tumorArea)]);

% Step 8: Display segmented tumor region
figure, imshow(img), title('Detected Tumor Region');
hold on;
visboundaries(bw, 'Color', 'r', 'LineWidth', 0.8);
hold off;
```

```
% Step G: Detect tumor presence clearly
thresholdArea = 500; % you can adjust this value
if tumorArea > thresholdArea
   msg = sprintf('🚨 Tumor Detected!\nType: %s\nArea: %.2f pixels', string(predictedLabel), tumorArea);
else
   msg = '✅ No Tumor Detected. Brain appears normal.';
end

disp(msg);
msgbox(msg, 'Tumor Detection Result');

disp('✅ Detection completed successfully.');
```