PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100 Ft. Road, BSK III Stage, Bengaluru – 560 085

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Course Title: Image Processing and Data Visualization Using MATLAB		
Course code: UE19CS257B		
Semester : 4th sem	Branch: CSE	Team Id: 31
SRN: PES1UG19CS019	Name: Abhishek Aditya B S	
SRN: PES1UG19CS567	Name: Vinay P	
SRN: PES1UG19CS571	Name: Vishal R	

PROJECT REPORT

Problem Statement:

Classify Brain MRI scan volumes into chronological age of participants using image processing and deep learning in MATLAB.

Objectives:

- Exploring brain MRI dataset
- Preparing the dataset for deep learning
- Training a model to classify the brain scans
- Evaluating the model.

Description:

We have used brain MRI dataset that is publicly available for our project. We will use this to classify the brain volumes into 3 categories, Ages 3-5, Ages 7-12, Adults. Firstly, we have preprocessed our dataset using many methods available in the Image Processing Toolbox. Pre-processing steps include applying skull stripping, random rotation and image resizing. We then use this augmented dataset for classification. We have used ResNet-18 to extract features from the brain volumes and use the extracted features to train a classification model. The ResNet-18 model is taken from the Deep Learning Toolbox provided by MATLAB. After training the classification model, we will use it to evaluate its predictions on random images from the dataset.

New Concept Learnt (Explanation) :

- Working with image datasets and applying image augmentation like skull stripping and random image rotations and image resizing.
- Learnt about Neural Networks and how they work and also how we can use them for our project.
- Using transfer learning to extract features from images (like edges and some high dimensional features) and use the feature maps for training our classifier.
- Model evaluation.

Learning Outcome:

We have learnt how to perform image classification using MATLAB. We have also learnt how to work with image datasets and augmenting the dataset so that it can be used for feature extraction and model training using feature maps. We also learnt visualizing the images present in the dataset.

Code:

```
mriRootDataFolder = 'ds000228-1.1.0-subset';

mriDataFolder = fullfile(mriRootDataFolder,
'derivatives', 'preprocessed_data');

vol=niftiread(fullfile(mriDataFolder,"sub-pixar001","sub-
pixar001_normed_anat.nii.gz"));

mask=niftiread(fullfile(mriDataFolder,"sub-pixar001","sub
-pixar001_analysis_mask.nii.gz"));

vol = int16(vol) .* int16(mask); % Apply skull-stripping

numSlices = size(vol,3);

imshow(vol(:,:,round(numSlices/2)));

volshow(vol(:,:,1:round(numSlices/2)),'CameraViewAngle',7.5);

participantFilename = fullfile(mriRootDataFolder,
'participants.tsv');

participantData=readtable(participantFilename,'FileType',
'delimitedtext','VariableNamingRule','preserve');
```

```
head(participantData) % view the first entries in the
participant table to illustrate the original dataset
contents

participantData.AgeClass(participantData.Age >= 3 &
participantData.Age < 6) = categorical("Ages3-5");

participantData.AgeClass(participantData.Age >= 7 &
participantData.Age < 13) = categorical("Ages7-12");

participantData.AgeClass(participantData.Age >= 18) =
categorical("Adults");

summary(participantData.AgeClass)

exemplars = prepare2DImageDataset(mriDataFolder);

ageClasses =string(categories(participantData.AgeClass));

figure('Position',[10 10 500 150]);

tiledlayout(1,numel(ageClasses),'Padding','none','TileSpa
cing','none');

for ii=1:length(ageClasses)
    nexttile;
    imshow(mat2gray(exemplars{ii}));
    title(ageClasses((ii)));
end

classifierDataFolder=['2DImageSet_'datestr(datevec(now),30)];

applySkullStripping = true;
applyAugmentation = true;

prepare2DImageDataset(mriDataFolder,classifierDataFolder,
applyAugmentation, applySkullStripping);

mriImgds=imageDatastore(classifierDataFolder,'IncludeSubf
olders',true,'LabelSource','foldernames','FileExtensions'
,'.png');

mriImgdsRand = shuffle(mriImgds);

mriImgdsExamples = splitEachLabel(mriImgdsRand,int16(1));
```

```
% Select the firstmost image from the just-shuffled
datastore for each age class
numLabels = length(mriImgdsExamples.Labels);

for ii=numLabels:-1:1
    [imgArray{ii},infoArray(ii)] =
readimage(mriImgdsExamples,ii);
end

figure('Position',[10 10 500 150]);
tiledlayout(1,3,'Padding','none','TileSpacing','none');

for selCount=1:length(ageClasses)
    % Find the random-selected image in imageArray which
pertains to the next age class for display
    idx = find(ageClasses(selCount) ==
[infoArray.Label]);

    % If image is a flipped image (from the offline data
augmentation), then flip it back, so all images have the
same orientation
    [~,fname] = fileparts(infoArray(idx).Filename);
    if startsWith(fname,'image2')
        img = imrotate(imgArray{idx},-180);
    else
        img = imgArray{idx};
    end

    % Show the random selected image for the next age class
    nexttile;
    imshow(mat2gray(img));
    title(infoArray(idx).Label);
end

[trainImgs,testImgs] =
splitEachLabel(mriImgds,0.85,'randomized'); % Reserve 15
percent of overall dataset for testing

[trainImgs, valImgs] =
splitEachLabel(trainImgs,0.8,'randomized'); % Reserve 20
percent of dataset available for training for online
validation

netName = 'resnet18';
net = resnet18();
```

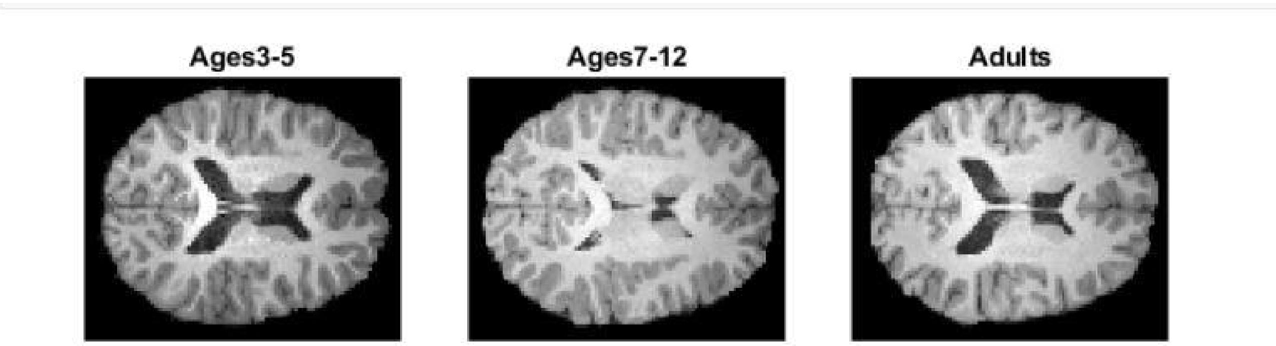
```
netInputSize = net.Layers(1).InputSize;
inputImageSize = netInputSize(1:2);

img = imread(fullfile(classifierDataFolder, 'Adults',
'image_033.png'));
imgLabel = classify(net, imresize(img, inputImageSize));
% resize 2D brain image to match network's input image
size

figure('Position',[10 10 200 200]);
imshow(img);
title([netName ' prediction: ' char(imgLabel)]);

lgraph = layerGraph(net);
numClasses = numel(categories(mriImgds.Labels));

newLearnableLayer = fullyConnectedLayer(numClasses, ...
    'Name','new_fc', ...
    'WeightLearnRateFactor',10, ...
    'BiasLearnRateFactor',10);
lgraph = replaceLayer(lgraph,'fc1000',newLearnableLayer);

newClassLayer=classificationLayer('Name','new_classoutput');
lgraph=replaceLayer(lgraph,'ClassificationLayer_predictio
ns',newClassLayer);

figure("Position",[10 10 900 600])
subplot(1,2,1)
plot(layerGraph(net))
xlim ([0 4]);ylim([0 8])
title('Final Layers of ResNet-18')
subplot(1,2,2)
plot(lgraph)
xlim ([0 4]);ylim([0 8])
title('Final Layers of the Modified Network')

net = resnet18();
imageSize = net.Layers(1).InputSize(1:2);

imageAugmenter=imageDataAugmenter('RandRotation',[-30,30]);
% Use randomized rotation for further data augmentation

datastore_train=augmentedImageDatastore(imageSize,trainIm
gs,'DataAugmentation',imageAugmenter);
```

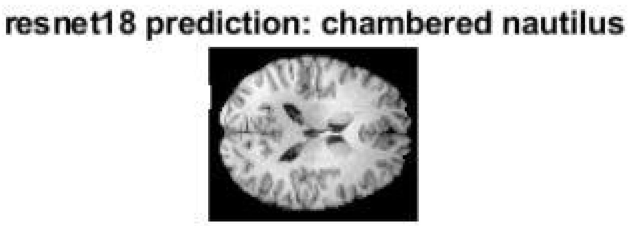
```
datastore_validate=augmentedImageDatastore(imageSize,valI
mgs);
datastore_test=augmentedImageDatastore(imageSize,testImgs);

disp(table({'Train';'Validate';'Test'},[datastore_train.N
umObservations;datastore_validate.NumObservations;datasto
re_test.NumObservations],'VariableNames',{'Datastore','Im
age Count'}))

trainOpts.initLearnRate   = 0.001;
% 10x reduction in initial learning rate
trainOpts.valFrequency    = 4;
trainOpts.miniBatchSize=floor(numel(datastore_train.Files
)/trainOpts.valFrequency);
trainOpts.maxEpochs       = 15;

options = trainingOptions('sgdm', ...
    'MiniBatchSize',trainOpts.miniBatchSize, ...
    'MaxEpochs',trainOpts.maxEpochs, ...
    'InitialLearnRate',trainOpts.initLearnRate, ...
    'Shuffle','every-epoch', ... % this handles the case
where the mini-batch size doesn't evenly divide the
number of training images
    'ValidationData',datastore_validate, ... % source of
validation data to evaluate learning during training
    'ValidationFrequency',trainOpts.valFrequency, ...
    'Verbose',false, ...
    'Plots','training-progress'); % display a plot of
progress during training

if canUseGPU()
    gpudev = gpuDevice; % Use the default GPU device, if
there's more than one
    reset(gpudev);
end

[mriNet,~] = trainNetwork(datastore_train, lgraph,
options);

[test_preds,test_scores] =
classify(mriNet,datastore_test);

accuracy = mean(test_preds == testImgs.Labels)

figure;
cm = confusionchart(testImgs.Labels,test_preds);
```

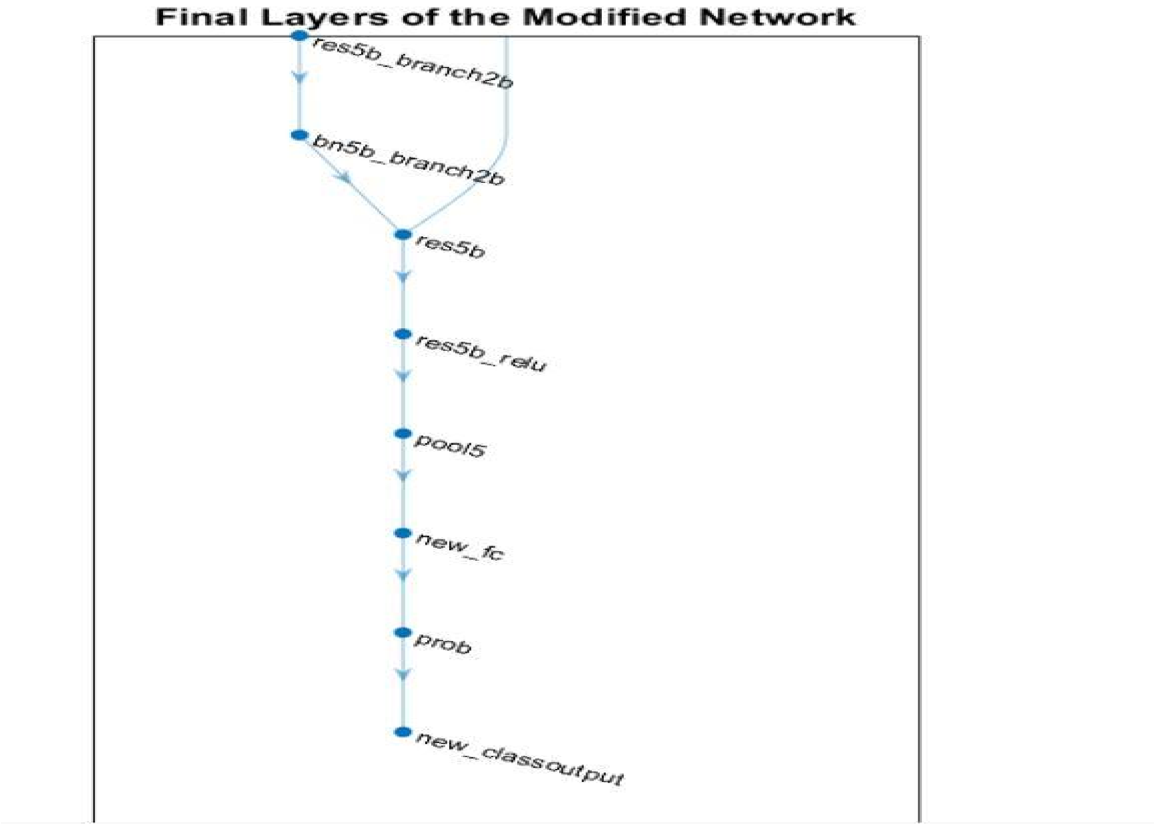
```
sortClasses(cm,["Ages3-5","Ages7-12","Adults"])
cm.Title = ['Confusion Matrix for the Network - Accuracy
: ' num2str(100*round(accuracy,4)) '%'];

viewOcclusionSensitivityMaps("Ages3-5",3, mriNet,
testImgs, test_preds, test_scores);

ageClass = categorical("Adults");
numSelections = 3;
viewOcclusionSensitivityMaps(ageClass,numSelections,
mriNet, testImgs, test_preds, test_scores);
```

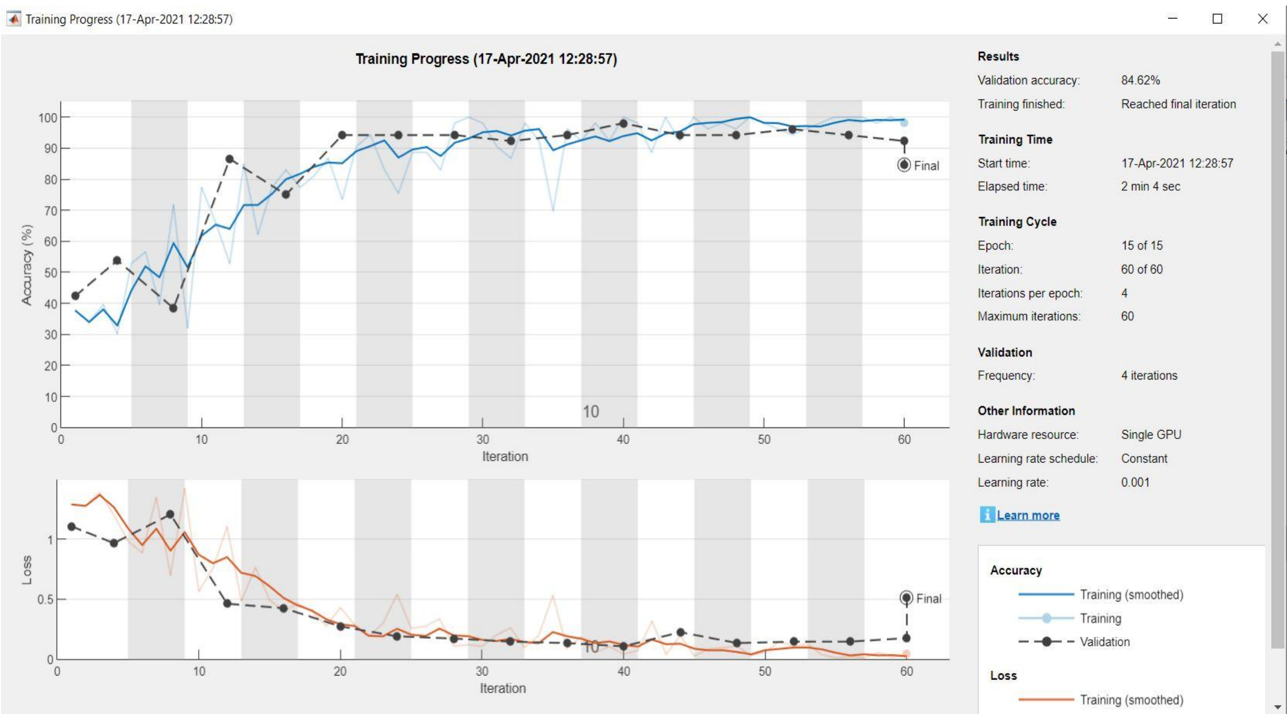
Output Screenshots

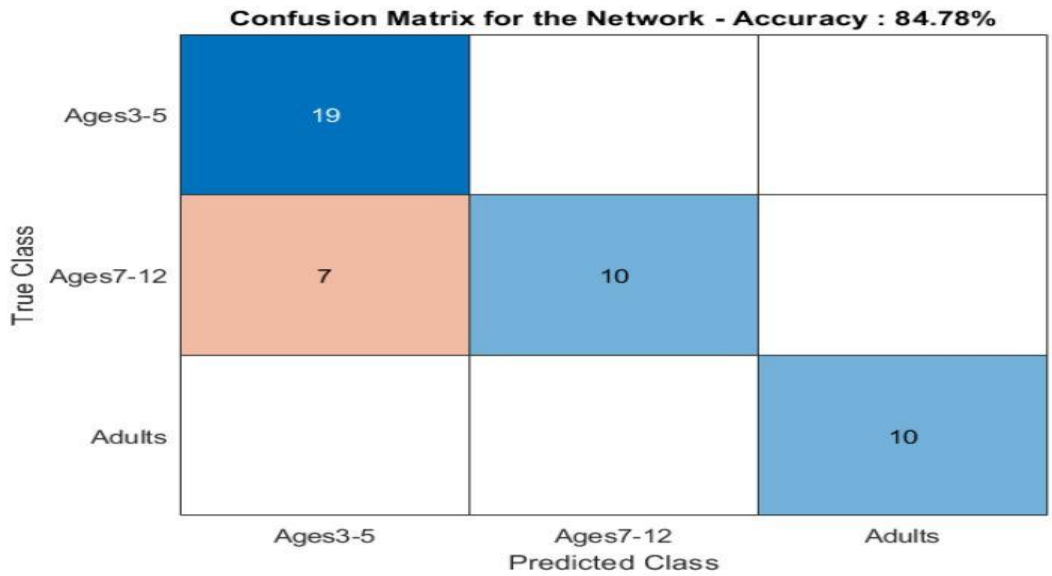
3D Brain volume MRI

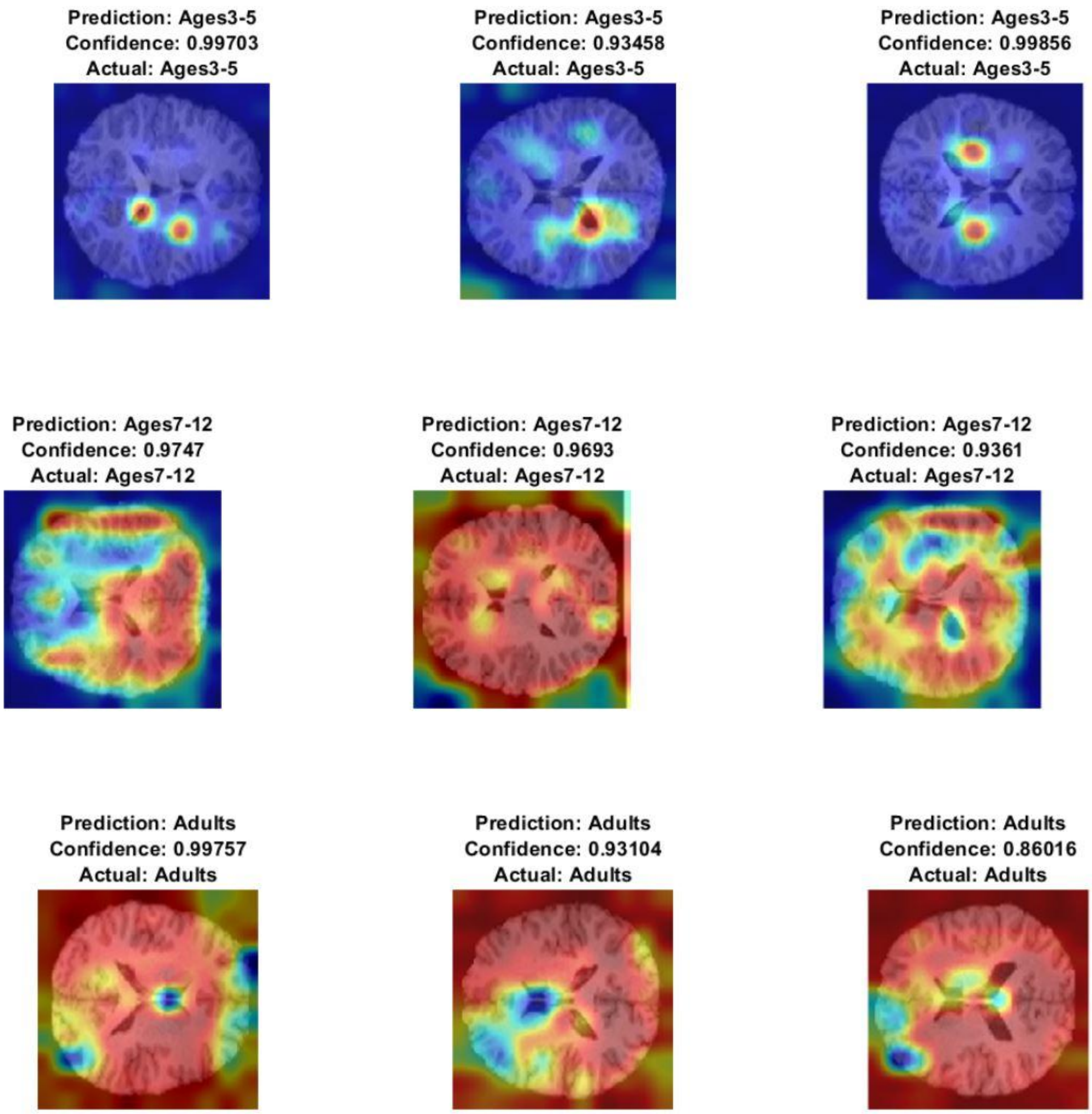





Final Layers of the Modified Network
res5b_branch2b
bn5b_branch2b
res5b
res5b_relu
pool5
new_fc
prob
new_classoutput


Training Progress (17-Apr-2021 12:28:57)
Training Progress (17-Apr-2021 12:28:57)
Accuracy (%)
Iteration
Final
Loss
Iteration
Final
Results
Validation accuracy: 84.62%
Training finished: Reached final iteration
Training Time
Start time: 17-Apr-2021 12:28:57
Elapsed time: 2 min 4 sec
Training Cycle
Epoch: 15 of 15
Iteration: 60 of 60
Iterations per epoch: 4
Maximum iterations: 60
Validation
Frequency: 4 iterations
Other Information
Hardware resource: Single GPU
Learning rate schedule: Constant
Learning rate: 0.001
Learn more
Accuracy
Training (smoothed)
Training
Validation
Loss
Training (smoothed)





Name and Signature of the Faculty