MATLAB code:

clc;

clear;

close all;

%Load Train Labels from Excel

trainFile = 'C:\Users\pkola\Downloads\AIML\_FA2\labels\_train.xlsx';

trainTable = readtable(trainFile);

% Define Paths

trainFolder = 'C:\Users\pkola\Downloads\AIML\_FA2\train\_images\train\_images';

testFolder = 'C:\Users\pkola\Downloads\AIML\_FA2\test\_images\test\_images';

trainPaths = fullfile(trainFolder, trainTable.file\_name);

trainLabels = categorical(trainTable.class\_id);

%% 2️⃣ Create Train & Test ImageDatastore

trainImgs = imageDatastore(trainPaths, 'Labels', trainLabels);

testImgs = imageDatastore(testFolder);

% Display Train Label Distribution

disp('Train Label Distribution:');

countEachLabel(trainImgs);

%Data Augmentation & Preprocessing

inputSize = [128, 128];

% ✅ Convert grayscale images to RGB (ensures uniform 3rd dimension)

augTrainImgs = augmentedImageDatastore(inputSize, trainImgs, 'ColorPreprocessing', 'gray2rgb');

augTestImgs = augmentedImageDatastore(inputSize, testImgs, 'ColorPreprocessing', 'gray2rgb');

%% 4️⃣ Define CNN-LSTM Model

layers = [

imageInputLayer([128 128 3], 'Name', 'input')

convolution2dLayer(3,16,'Padding','same', 'Name', 'conv1')

batchNormalizationLayer('Name', 'batchnorm1')

reluLayer('Name', 'relu1')

maxPooling2dLayer(2,'Stride',2, 'Name', 'maxpool1')

convolution2dLayer(3,32,'Padding','same', 'Name', 'conv2')

batchNormalizationLayer('Name', 'batchnorm2')

reluLayer('Name', 'relu2')

maxPooling2dLayer(2,'Stride',2, 'Name', 'maxpool2')

convolution2dLayer(3,64,'Padding','same', 'Name', 'conv3')

batchNormalizationLayer('Name', 'batchnorm3')

reluLayer('Name', 'relu3')

maxPooling2dLayer(2,'Stride',2, 'Name', 'maxpool3')

flattenLayer('Name', 'flatten')

lstmLayer(100, 'OutputMode', 'last', 'Name', 'lstm1')

fullyConnectedLayer(3, 'Name', 'fc') % 3 classes: Normal, Bacterial, Viral

softmaxLayer('Name', 'softmax')

classificationLayer('Name', 'output')];

%% 5️⃣ Training Options

options = trainingOptions('adam', ...

'MaxEpochs', 30, ...

'MiniBatchSize', 32, ...

'ExecutionEnvironment', 'gpu', ... % Use 'cpu' if GPU is unavailable

'Plots', 'training-progress');

%% 6️⃣ Train the Model

trainedNet = trainNetwork(augTrainImgs, layers, options);

%% 7️⃣ Compute Training Accuracy & Confusion Matrix

trainPreds = classify(trainedNet, augTrainImgs);

trainAccuracy = mean(trainPreds == trainImgs.Labels) \* 100;

disp(['Training Accuracy: ', num2str(trainAccuracy), '%']);

% Confusion Matrix for Training Set

classNames = ["No Disease", "Bacterial Pneumonia", "Viral Pneumonia"];

confMat = confusionmat(trainImgs.Labels, trainPreds);

figure;

confusionchart(confMat, classNames);

title('Confusion Matrix - Training Set');

%% 8️⃣ Classification Report (Precision, Recall, F1-Score)

precision = diag(confMat) ./ sum(confMat, 2); % TP / (TP + FN)

recall = diag(confMat) ./ sum(confMat, 1)'; % TP / (TP + FP)

f1Score = 2 \* (precision .\* recall) ./ (precision + recall);

% Handle NaN values (if division by zero occurs)

precision(isnan(precision)) = 0;

recall(isnan(recall)) = 0;

f1Score(isnan(f1Score)) = 0;

% Display Classification Report

reportTable = table(classNames', precision, recall, f1Score, ...

'VariableNames', {'Class', 'Precision', 'Recall', 'F1\_Score'});

disp('Classification Report:');

disp(reportTable);

%% 9️⃣ Predict on Test Images

YPred = classify(trainedNet, augTestImgs);

% Save predictions to Excel

predTable = table(testImgs.Files, YPred, 'VariableNames', {'Image', 'PredictedLabel'});

writetable(predTable, 'TestPredictions.xlsx');

disp('Test predictions saved to TestPredictions.xlsx');

%% 🔟 Save Trained Model

save('Pneumonia\_Classifier.mat', 'trainedNet');

%% 🔟+2️⃣ Predict & Display a Random Test Image

randomIdx = randi(numel(testImgs.Files));

testImg = readimage(testImgs, randomIdx);

% Resize & Convert to RGB

testImg = imresize(testImg, [128, 128]);

if size(testImg, 3) == 1

testImg = cat(3, testImg, testImg, testImg);

end

testImg = single(testImg);

% Predict Class

predictedLabel = classify(trainedNet, testImg);

predictedClassName = classNames(str2double(char(predictedLabel)) + 1);

% Display Image with Prediction

figure;

imshow(testImg);

title(['Predicted: ', char(predictedClassName)], 'FontSize', 14);

disp(['Tested Image: ', testImgs.Files{randomIdx}]);