KENNESAW STATE
UNIVERSITY

CS 4267
MACHINE LEARNING

PROJECT 3

DEEP LEARNING

INSTRUCTOR

Emin Mary Abraham

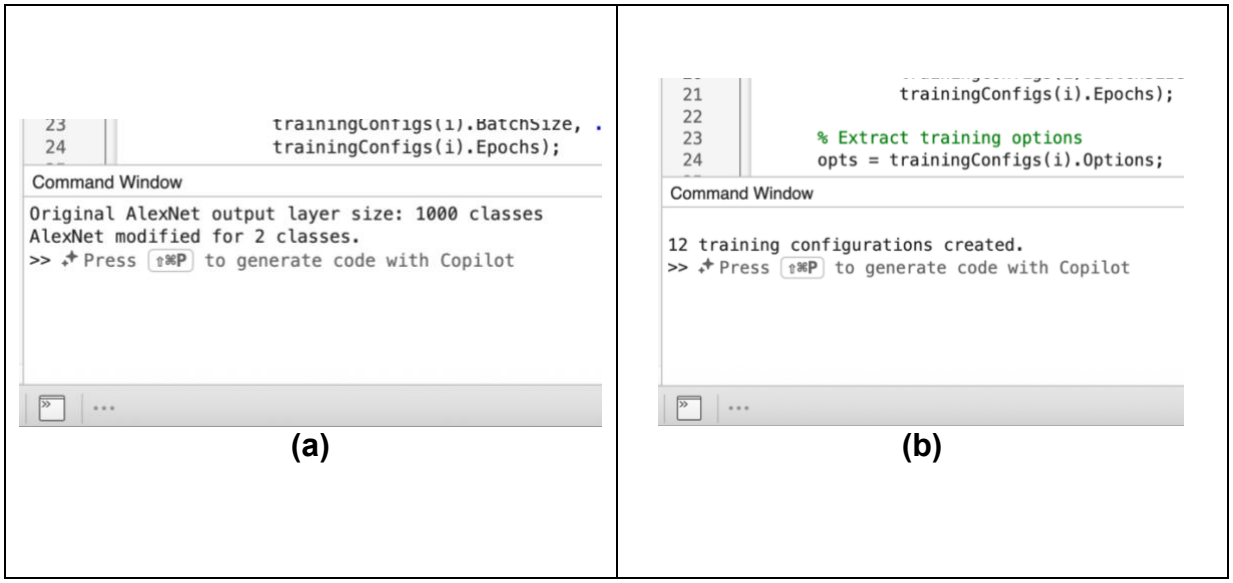
Franck DIPANDA
001049149

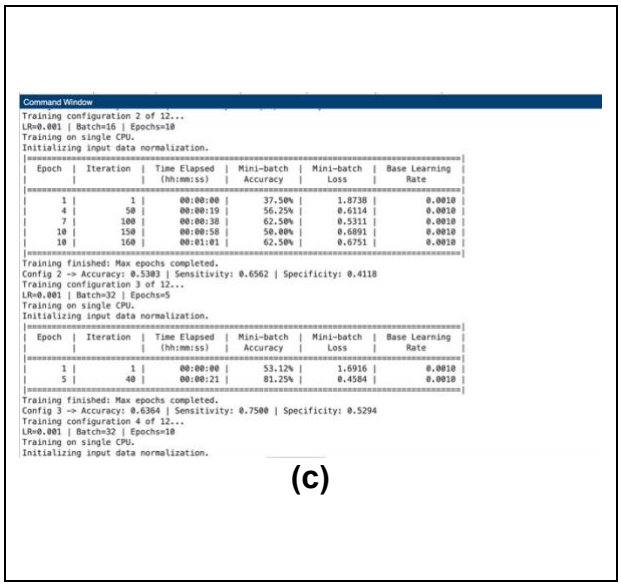
1. ABSTRACT

In this project, I used transfer learning on AlexNet to classify images into two categories: DR and NonDR. The original IDRID dataset contains five levels, however, the assignment required restructuring the dataset into a DR vs nonDR. The project involved dataset preparation, image organization, preprocessing, model modification, training with multiple hyperparameter configurations, and evaluating the final models using confusion matrices, accuracy, sensitivity, and specificity.

2. TEST RESULTS

To evaluate the performance of the trained AlexNet models, I selected the two best-performing hyperparameter configurations from the training stage: Configuration 8 and Configuration 12. Both models were tested on the same test set using the images prepared in 3.1 mentioned below. The evaluation metrics include overall accuracy, class sensitivity (DR detection rate), class specificity (NonDR detection rate), and confusion matrices.

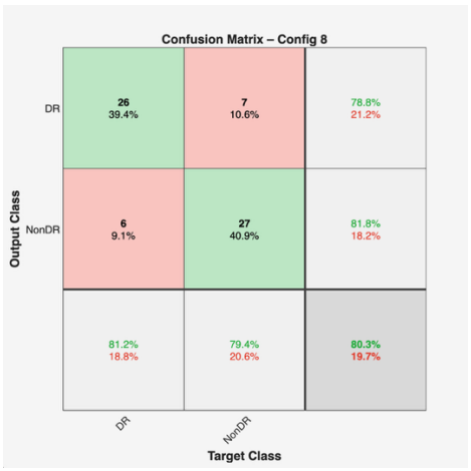




(c)

Figure 1: (a) Re-classification of AlexNet, (b) Defining hyperparameters to train models, (c) Running the training script on the training data.

Once the training was done using the training data, the two best configurations were selected from the training results and were used to classify the test data.



(a)

(b)

Figure 2: (a) Confusion matrix of hyparameter setting #8 on test data, (b) Confusion matrix of hyparameter setting #12 on test data

3. CODES

3.1 Code for the initial data preparation

```
% Name: Franck Dipanda
% Number: 001047147
% Project 3


% Prepare dataset
clear; clc;

rootFolder = '/Users/franckalexis/Documents/MATLAB/Assignment3/DS_IDRID';
trainSrc = fullfile(rootFolder, "Train");
testSrc  = fullfile(rootFolder, "Test");

baseOut = "dataset";
outTrain_DR    = fullfile(baseOut, "train", "DR");
outTrain_NonDR = fullfile(baseOut, "train", "NonDR");

outTest_DR     = fullfile(baseOut, "test", "DR");
outTest_NonDR  = fullfile(baseOut, "test", "NonDR");

% Creating output folders
folders = {outTrain_DR, outTrain_NonDR, outTest_DR, outTest_NonDR};

for i = 1:length(folders)
    if ~exist(folders{i}, "dir")
        mkdir(folders{i});
    end
end

% Processing training images
trainFiles = dir(fullfile(trainSrc, "*.jpg"));

for i = 1:numel(trainFiles)
```

```
    filename = trainFiles(i).name;
    src = fullfile(trainFiles(i).folder, filename);

    % Extract labels from file name
    parts = split(filename, "-");
    labelStr = erase(parts{end}, ".jpg");
    label = str2double(labelStr);

    % Skip labels 1 and 2
    if label == 1 || label == 2
        continue;
    end

    if label == 0
        dest = fullfile(outTrain_NonDR, filename);
    elseif label == 3 || label == 4
        dest = fullfile(outTrain_DR, filename);
    else
        fprintf("Unknown Train label %d in %s\n", label, filename);
        continue;
    end

    copyfile(src, dest);
end

% Processing Test Images
testFiles = dir(fullfile(testSrc, "*.jpg"));
for i = 1:numel(testFiles)

    filename = testFiles(i).name;
    src = fullfile(testFiles(i).folder, filename);

    % Extract label
```

```
    parts = split(filename, "-");
    labelStr = erase(parts{end}, ".jpg");
    label = str2double(labelStr);

    % Skip labels 1 and 2
    if label == 1 || label == 2
        continue;
    end

    if label == 0
        dest = fullfile(outTest_NonDR, filename);
    elseif label == 3 || label == 4
        dest = fullfile(outTest_DR, filename);
    else
        fprintf("Unknown Test label %d in %s\n", label, filename);
        continue;
    end

    copyfile(src, dest);
end

fprintf("\nDataset preparation complete\n");
```

3.2 Code to load data

```
% Name: Franck Dipanda
% Number: 001047147
% Project 3
```

```
% Load prepared data

clear; clc;

trainFolder = fullfile("dataset", "train");
testFolder  = fullfile("dataset", "test");

% AlexNet input size
inputSize = [227 227 3];

% Load training images
imdsTrain = imageDatastore(trainFolder, ...
    "IncludeSubfolders", true, ...
    "LabelSource", "foldernames");

% Load testing images
imdsTest = imageDatastore(testFolder, ...
    "IncludeSubfolders", true, ...
    "LabelSource", "foldernames");

fprintf("Training images: %d\n", numel(imdsTrain.Files));
fprintf("Testing images : %d\n", numel(imdsTest.Files));

% Resize images for AlexNet
augTrain = augmentedImageDatastore(inputSize, imdsTrain);
augTest  = augmentedImageDatastore(inputSize, imdsTest);



% Save stored data
save("Stored_data.mat", "augTrain", "augTest", "imdsTrain", "imdsTest");

fprintf("Stored data loaded\n");
```

3.3 Code to modify AlexNet for binary classification

```
% Name: Franck Dipanda
% Number: 001047147
% Project 3


% Load and modify AlexNet for DR and nonDR classification

clear; clc;

% Load pretrained AlexNet
net = alexnet;
layers = net.Layers;

% Display original architecture size for reference
fprintf("Original AlexNet output layer size: %d classes\n", numel(layers(end).Classes));

% Modify the last 3 layers:
% Replace the fullyConnectedLayer(1000) with fullyConnectedLayer(2)
% Replace softmax & classification layers accordingly

numClasses = 2;

layers(end-2) = fullyConnectedLayer(numClasses, ...
    "Name", "fc_final", ...
    "WeightLearnRateFactor", 10, ...
```

```
    "BiasLearnRateFactor", 10);

layers(end-1) = softmaxLayer("Name", "softmax");
layers(end)   = classificationLayer("Name", "classoutput");

% Save modified layers for training step
save("modifiedAlexNet.mat", "layers");

fprintf("AlexNet modified for %d classes.\n", numClasses);

% referenced Matlab guide and chatgpt to better understand transferlearning
% implementation: https://www.mathworks.com/help/deeplearning/ug/transfer-learning-using-alexnet.html
```

3.4 Code for setting up training

```
% Name: Franck Dipanda
% Number: 001047147
% Project 3


% Define Training Options

clear; clc;


% Load the modified AlexNet layers
load("modifiedAlexNet.mat", "layers");


% Hyperparameter ranges
learningRates = [1e-3, 1e-4, 1e-5];
batchSizes    = [16, 32];
```

```
epochCounts   = [5, 10];

% Create all combinations
configIndex = 1;
trainingConfigs = struct();

for lr = learningRates
    for bs = batchSizes
        for ep = epochCounts

            opts = trainingOptions("sgdm", ...
                "InitialLearnRate", lr, ...
                "MiniBatchSize", bs, ...
                "MaxEpochs", ep, ...
                "Shuffle", "every-epoch", ...
                "Verbose", true, ...
                "Plots", "none", ...
                "ExecutionEnvironment", "auto");

            trainingConfigs(configIndex).LearnRate = lr;
            trainingConfigs(configIndex).BatchSize = bs;
            trainingConfigs(configIndex).Epochs    = ep;
            trainingConfigs(configIndex).Options   = opts;

            configIndex = configIndex + 1;
        end
    end
end


save("trainingConfigs.mat", "trainingConfigs", "layers");

fprintf("\n%d training configurations created.\n", length(trainingConfigs));
```

```
% referenced Matlab guide and chatgpt to better understand transferlearning
% implementation: https://www.mathworks.com/help/deeplearning/ug/transfer-learning-using-alexnet.html
```

3.5 Code for training on train samples

```
% Name: Franck Dipanda
% Number: 001047147
% Project 3


% Train AlexNet using all training configurations

clear; clc;


% Load datastores, modified layers, and training configurations
load("Stored_data.mat", "augTrain", "augTest", "imdsTrain", "imdsTest");
load("modifiedAlexNet.mat", "layers");
load("trainingConfigs.mat", "trainingConfigs");

numConfigs = length(trainingConfigs);
results = struct();


for i = 1:numConfigs

    fprintf("Training configuration %d of %d...\n", i, numConfigs);
    fprintf("LR=%g | Batch=%d | Epochs=%d\n", ...
        trainingConfigs(i).LearnRate, ...
```

```
        trainingConfigs(i).BatchSize, ...
        trainingConfigs(i).Epochs);

% Extract training options
opts = trainingConfigs(i).Options;

% Train the network
trainedNet = trainNetwork(augTrain, layers, opts);

% Evaluate
predictedLabels = classify(trainedNet, augTest);
trueLabels = imdsTest.Labels;

accuracy = mean(predictedLabels == trueLabels);

% Confusion matrix
cm = confusionmat(trueLabels, predictedLabels);

% Sensitivity
sensitivity = cm(1,1) / sum(cm(1,:));

% Specificity
specificity = cm(2,2) / sum(cm(2,:));


% Store results
results(i).LearnRate   = trainingConfigs(i).LearnRate;
results(i).BatchSize   = trainingConfigs(i).BatchSize;
results(i).Epochs      = trainingConfigs(i).Epochs;
results(i).Accuracy    = accuracy;
results(i).Sensitivity = sensitivity;
results(i).Specificity = specificity;
results(i).ConfMat     = cm;
```

```
    % Save trained model
    modelName = sprintf("trainedModel_config_%d.mat", i);
    save(modelName, "trainedNet");

    fprintf("Config %d -> Accuracy: %.4f | Sensitivity: %.4f | Specificity: %.4f\n", ...
        i, accuracy, sensitivity, specificity);

end

save("results.mat", "results");

fprintf("\nAll models trained and evaluated.\n");
```

3.6 Code for testing on test samples

```
% Name: Franck Dipanda
% Number: 001047147
% Project 3


%  Running AlexNet on test data

clear; clc;

% Load test data
load("Stored_data.mat", "augTest", "imdsTest");
trueLabels = imdsTest.Labels;
```

```
% Load the two selected models
load("trainedModel_config_8.mat", "trainedNet");
net8 = trainedNet;

load("trainedModel_config_12.mat", "trainedNet");
net12 = trainedNet;


% Predictions for both models
pred8  = classify(net8, augTest);
pred12 = classify(net12, augTest);


% Accuracies
acc8  = mean(pred8  == trueLabels);
acc12 = mean(pred12 == trueLabels);

fprintf("\nAccuracy (Config 8):  %.4f\n", acc8);
fprintf("Accuracy (Config 12): %.4f\n", acc12);

% Confusion matrices
cm8  = confusionmat(trueLabels, pred8);
cm12 = confusionmat(trueLabels, pred12);

fprintf("\nConfusion Matrix - Config 8:\n");
disp(cm8);

fprintf("\nConfusion Matrix - Config 12:\n");
disp(cm12);

% Sensitivity and Specificity

sens8 = cm8(1,1)  / sum(cm8(1,:));
```

```
spec8 = cm8(2,2)  / sum(cm8(2,:));


sens12 = cm12(1,1) / sum(cm12(1,:));
spec12 = cm12(2,2) / sum(cm12(2,:));


fprintf("\nSensitivity / Specificity:\n");
fprintf("Config 8  - Sensitivity: %.4f | Specificity: %.4f\n", sens8,  spec8);
fprintf("Config 12 - Sensitivity: %.4f | Specificity: %.4f\n", sens12, spec12);


% Plots
figure;
plotconfusion(trueLabels, pred8);
title("Confusion Matrix – Config 8");
saveas(gcf, "Confmat_Config8.png");


figure;
plotconfusion(trueLabels, pred12);
title("Confusion Matrix – Config 12");
saveas(gcf, "Confmat_Config12.png");



fprintf("\nConfusion matrices saved.\n");
```

4. DISCUSSION

The overall process of building the DR classifier using transfer learning involved several steps. The first major decision was selecting AlexNet as the base architecture. This choice was guided by my previous use of AlexNet in the Machine Vision course I took previously. I relied on the MathWorks Transfer Learning Guide and ChatGPT to understand which layers needed to be replaced and how learning-rate factors should be set for the final fully connected layer.

A key part of the workflow was preparing the dataset correctly. All images had to be sorted into DR and NonDR folders based on the label encoded in their filenames. I also needed to resize images to match AlexNet's 227×227 input.

The model training phase involved experimenting with multiple hyperparameters. Instead of training one model, I generated 12 different configurations by varying the learning rate, batch size, and epoch count. This is a technique that I also impleted based off of my previous experience with AlexNet.

- A high learning rate (1e-3) caused instability and produced NaN losses in some runs.
- A moderate rate (1e-4) yielded the most balanced and accurate models.
- A very low learning rate (1e-5) improved sensitivity but reduced specificity.

The test results highlight these differences clearly. Config 8, which used LR = 1e-4, achieved the best overall balance with an accuracy of 80.30%, sensitivity of 81.25%, and specificity of 79.41%. Config 12, while slightly lower in accuracy (77.27%), produced the highest DR sensitivity (90.63%), making it valuable in scenarios where detecting positive DR cases is more important than reducing false positives. This is similar to the behavior seen in the kNN assignment, where specific hyperparameter choices created imbalance between stability and misclassification rates.

Acknowledgement:
I would like to thank **Dr Mahmut Karakaya** for providing the template for project report. I also wish to acknowledge the resources I used, which assisted my understanding of CNNs and their implementation:
MathWorks. (2024). Transfer Learning Using AlexNet. MATLAB Deep Learning Toolbox Documentation.
https://www.mathworks.com/help/deeplearning/ug/transfer-learning-using-alexnet.html

"Machine Vision CS 4732 – Project 4: Deep Learning for Classification)", published by Dr. Mahmut Karakaya