```
% clear workspace and suspend all active gpu operations
clear variables;
close all;
existing_GUIs = findall(0);
if length(existing_GUIs) > 1
    delete(existing_GUIs);
end
clc;
```

```
% from hog_svm
folder = "CUB_200_2011_Subset20classes";
trainingImageNames = readtable(fullfile(folder, "train.txt"),'ReadVariableNames', false);
trainingImageNames.Properties.VariableNames = {'index', 'imageName'};

validationImageNames = readtable(fullfile(folder, "validate.txt"),'ReadVariableNames', false);
validationImageNames.Properties.VariableNames = {'index', 'imageName'};

testImageNames = readtable(fullfile(folder, "test.txt"),'ReadVariableNames', false);
testImageNames.Properties.VariableNames = {'index', 'imageName'};
```

Read class info from the relevant text files

```
classNames = readtable(fullfile(folder, "classes.txt"),'ReadVariableNames', false);
classNames.Properties.VariableNames = {'index', 'className'};

imageClassLabels = readtable(fullfile(folder, "image_class_labels.txt"),'ReadVariableNames', fa
imageClassLabels.Properties.VariableNames = {'index', 'classLabel'};
```

Create lists of image names for training, validation and test subsets.

To be precise, we create an array of strings containing the full file path and file names for each data partition.

```
folder = "CUB_200_2011_Subset20classes/";
trainingImageList = strings(height(trainingImageNames), 1);
for iI = 1:height(trainingImageNames)
    trainingImageList(iI) = string(fullfile(folder, "images/", ...
        string(cell2mat(trainingImageNames.imageName(iI)))));
end

validationImageList = strings(height(validationImageNames), 1);
for iI = 1:height(validationImageNames)
    validationImageList(iI) = string(folder + "images/" + ...
        string(cell2mat(validationImageNames.imageName(iI))));
end

testImageList = strings(height(testImageNames), 1);
for iI = 1:height(testImageNames)
    testImageList(iI) = string(folder + "images/" + ...
```

```
        string(cell2mat(testImageNames.imageName(iI))));
end
```

Create image datastores for training, validation and test subsets

```
trainingImageDS = imageDatastore(trainingImageList, 'labelSource', 'foldernames', ...
    'FileExtensions', {'.jpg'});
trainingImageDS.ReadFcn = @readImagesIntoDatastore;

validationImageDS = imageDatastore(validationImageList, 'labelSource', 'foldernames', ...
    'FileExtensions', {'.jpg'});
validationImageDS.ReadFcn = @readImagesIntoDatastore;

testImageDS = imageDatastore(testImageList, 'labelSource', 'foldernames', ...
    'FileExtensions', {'.jpg'});
testImageDS.ReadFcn = @readImagesIntoDatastore;
```

The images all have different spatial resolutions (width x height), so

need to resize them to the same size. (Experiment with different sizes!)

```
% target_size = [100, 100];
target_size = [224, 224];
% target_size = [227, 227];
trainingImageDS_Resized = transform(trainingImageDS, @(x) imresize(x,targetSize));
validationImageDS_Resized = transform(validationImageDS, @(x) imresize(x,targetSize));
testImageDS_Resized = transform(testImageDS, @(x) imresize(x,targetSize));

% Combine transformed datastores and labels
labelsTraining = arrayDatastore(trainingImageDS.Labels);
cdsTraining = combine(trainingImageDS_Resized, labelsTraining);
labelsValidation = arrayDatastore(validationImageDS.Labels);
cdsValidation = combine(validationImageDS_Resized, labelsValidation);
labelsTest = arrayDatastore(testImageDS.Labels);
cdsTest = combine(testImageDS_Resized, labelsTest);
```

```
folder = "CUB_200_2011_Subset20classes/";
imgFolder = folder + "images/";
imgTxtFolder = folder + "images.txt";
numDir = 22;
```

```
allImageDS = imageDatastore(imgFolder, 'IncludeSubfolders', true,'LabelSource','foldernames');
```

Split dataset into five folds (=partitions) for fivefold cross-validation.

Split dataset into 5 x 20% - Note, splitEachLabel splits the datastore into N+1 new datastores, so by specifying 0.2 four times, we will end up with five 20% partitions.

```
[fold1DS, fold2DS, fold3DS, fold4DS, fold5DS] = splitEachLabel(allImageDS, 0.2, 0.2, 0.2, 0.2);
```

```
% Number of folds is five in this experiment
folds = 5;
```

```
trainedNetwork = resnet50;
lgraph = layerGraph(trainedNetwork);
deltafc1000 = fullyConnectedLayer(20,'Name','dfc1000');
deltaClassificationfc1000 = classificationLayer('Name', 'dcfc1000', 'Classes', 'auto');
lgraph = replaceLayer(lgraph,'fc1000',deltafc1000);
lgraph = replaceLayer(lgraph,'ClassificationLayer_fc1000',deltaClassificationfc1000);
```

Checking if a GPU is available and clearing any old data from it

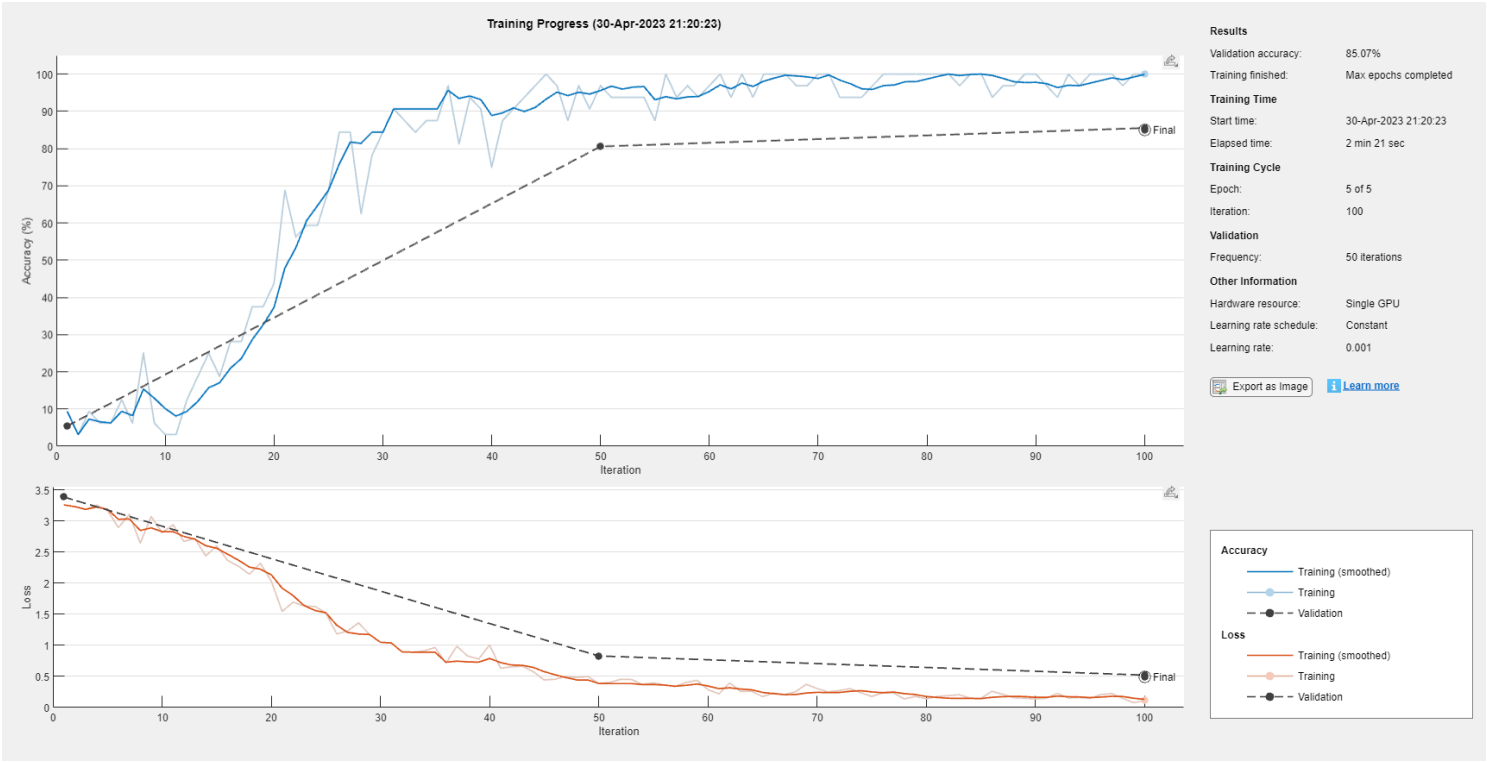
```
if (gpuDeviceCount() > 0)
    disp('Found GPU:');
    disp(gpuDeviceTable);
    gpu_device = gpuDevice(1);
    reset(gpu_device);  % Clear previous values that might still be on the GPU
end
```

```
Found GPU:
    Index                    Name                    ComputeCapability    DeviceAvailable    DeviceSelected
    _____    ____________________________________    _________________    _______________    ______________

      1      "NVIDIA GeForce RTX 3050 Ti Laptop GPU"        "8.6"              true               true
```

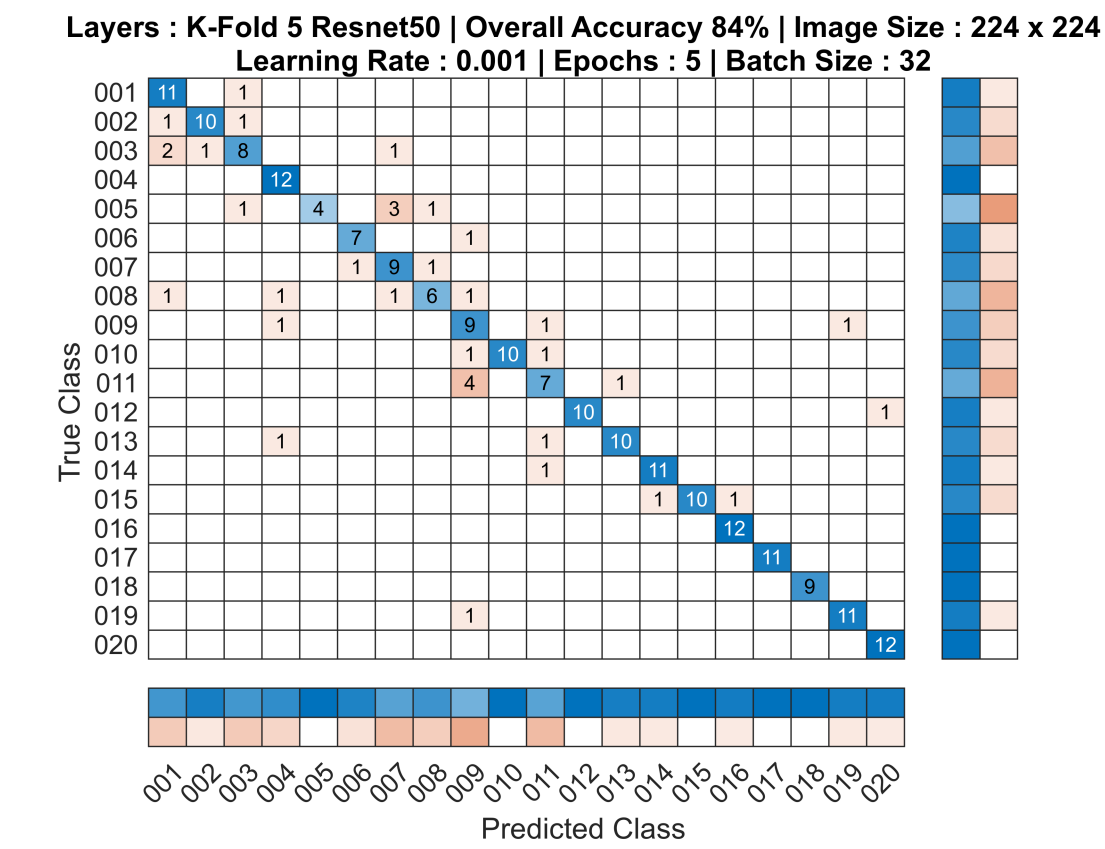
Training a multi-class SVM

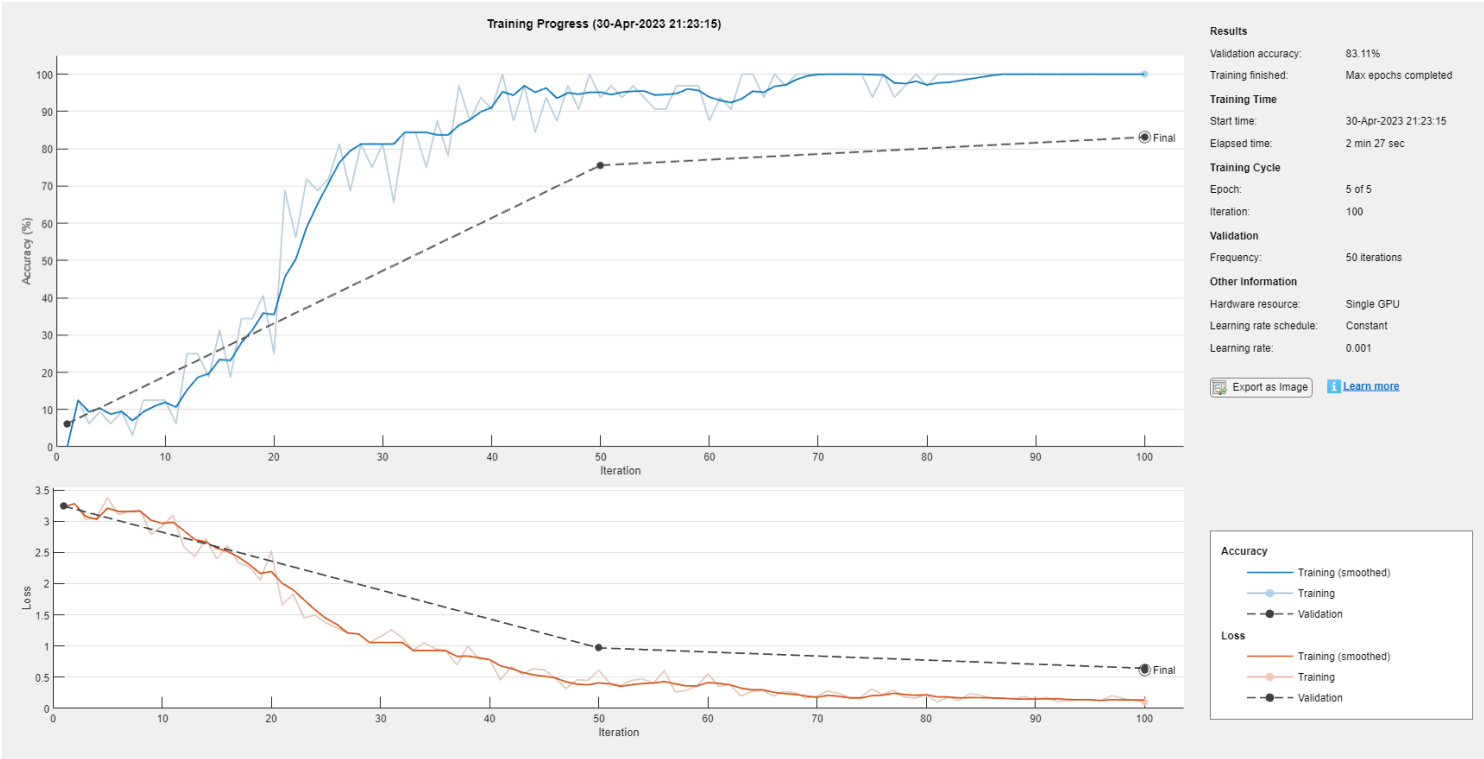
```
learning_rate = 0.001;
batch = 32;
epochs = 5;
accuracy_overall = 0.0;

for i = 1:folds
    [cdsTraining, cdsValidation, cdsTest, trainingImageDS, validationImageDS, testImageDS] = ..
    return_fold_for_cross_validation(i, fold1DS, fold2DS, fold3DS, fold4DS, fold5DS, folder, in

    % Set the training options
    options = trainingOptions('sgdm', ...
        'InitialLearnRate', learning_rate, ...
        'MiniBatchSize', batch, ...
        'MaxEpochs', epochs, ...
        'Verbose', false, ...
        'Shuffle', 'every-epoch', ...
        'VerboseFrequency', 1, ...
        'ValidationData', cdsValidation, ...
        'Plots','training-progress');

    simpleCNN = trainNetwork(cdsTraining, lgraph, options);
```

```
    YPred = classify(simpleCNN, cdsTest);
    YTest = testImageDS.Labels;

    % overall accuracy
    accuracy = sum(YPred == YTest)/numel(YTest);
    disp("Overall Accuracy for Run "+ string(i)+" is: " + accuracy);

    % Show confusion matrix in figure
    [matrix, order] = confusionmat(YTest, YPred);
    figure(i);
    confusion_matrix = confusionchart(matrix, order, ...
        'ColumnSummary','column-normalized', ...
        'RowSummary','row-normalized');
    title({"Layers : K-Fold 5 Resnet50 | Overall Accuracy " + string(round(accuracy*100, 1)) +
        " | Image Size : " + target_size(1) + " x " + target_size(1); ...
        "Learning Rate : " + learning_rate + " | Epochs : " + epochs + " | Batch Size : " + bat
    accuracy_overall = accuracy_overall+accuracy;
end
```

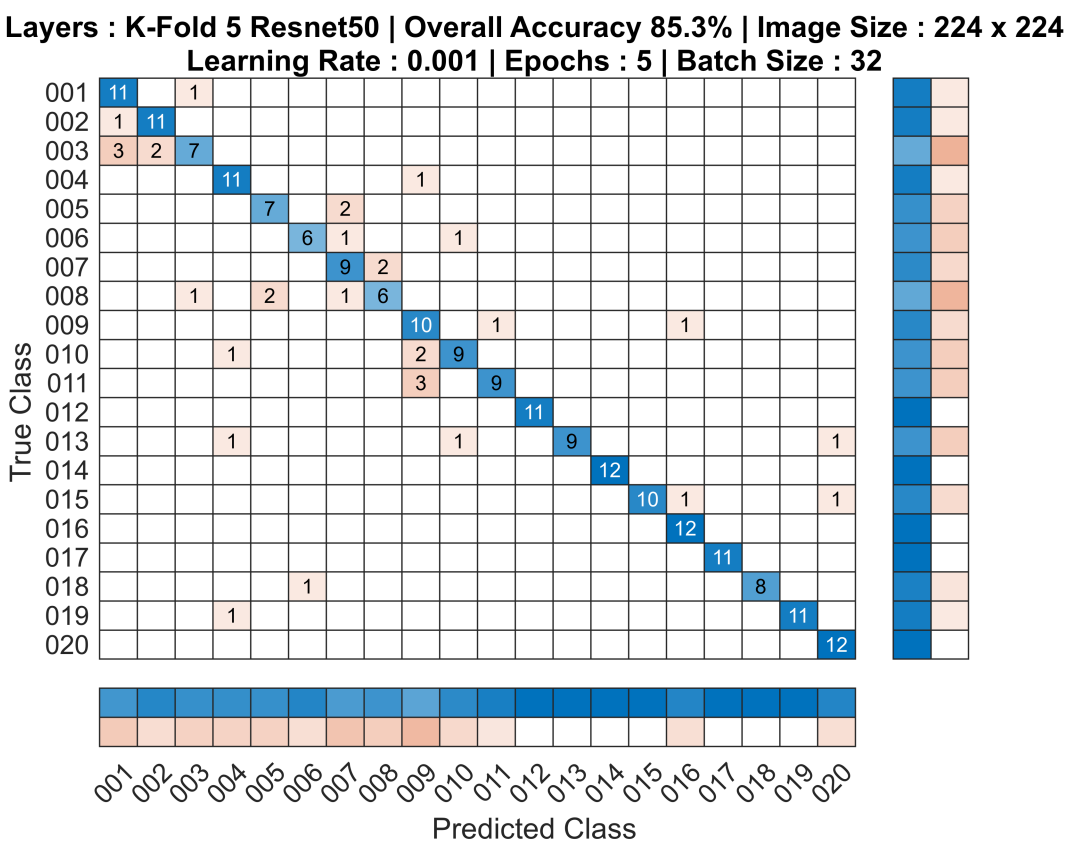


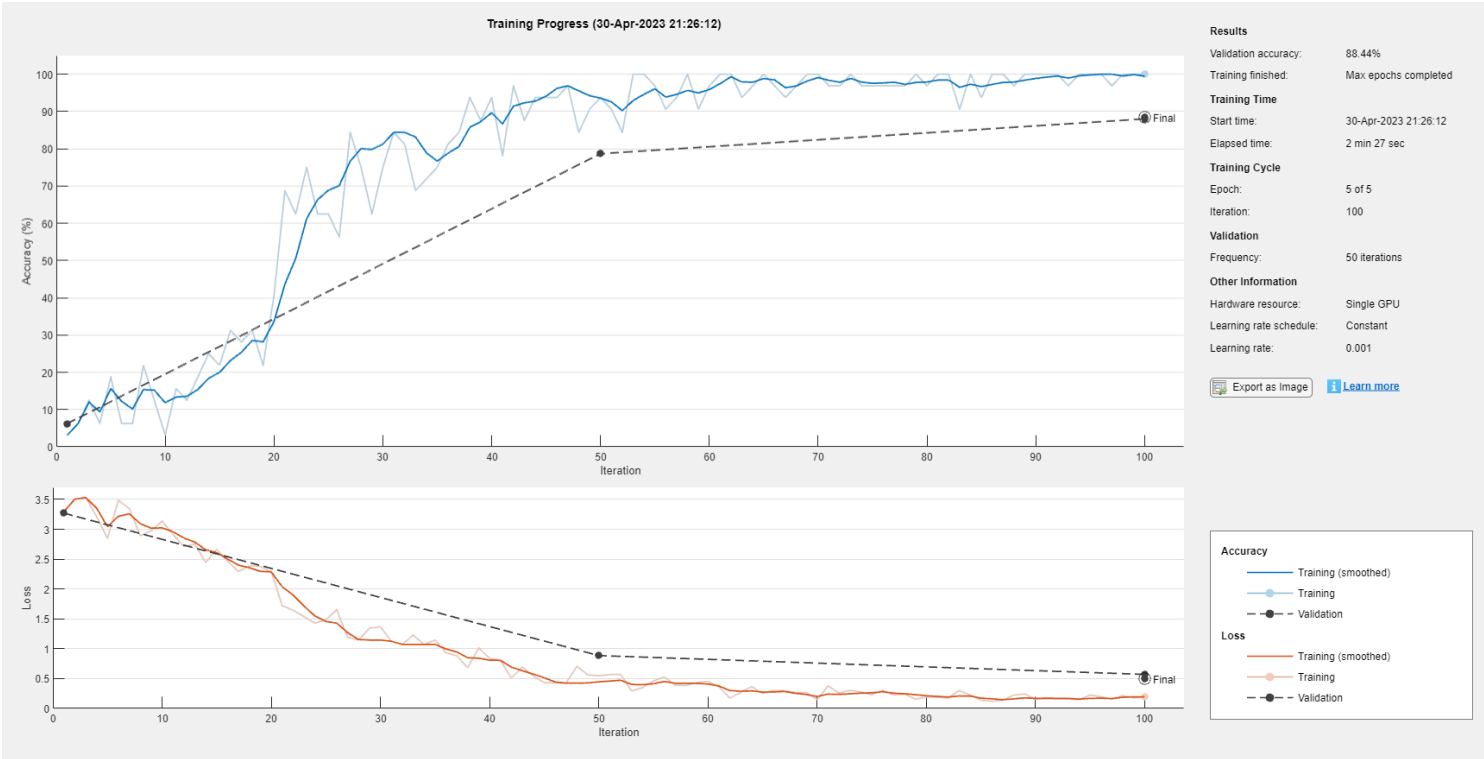

```
Overall Accuracy for Run 1 is: 0.84
```





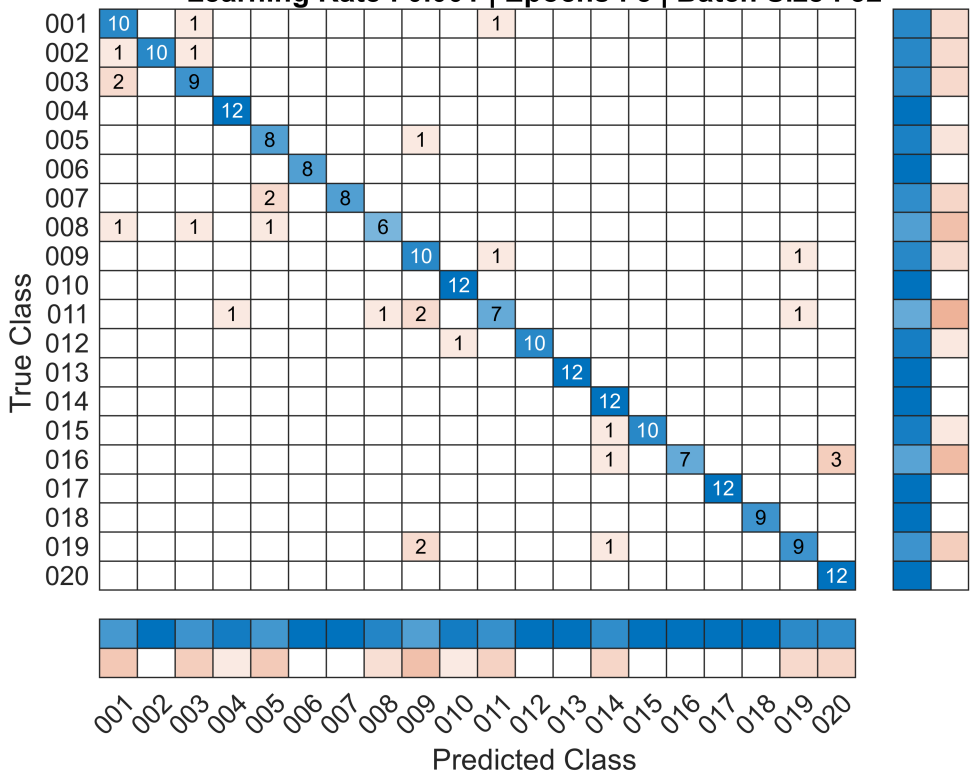
Overall Accuracy for Run 2 is: 0.85333

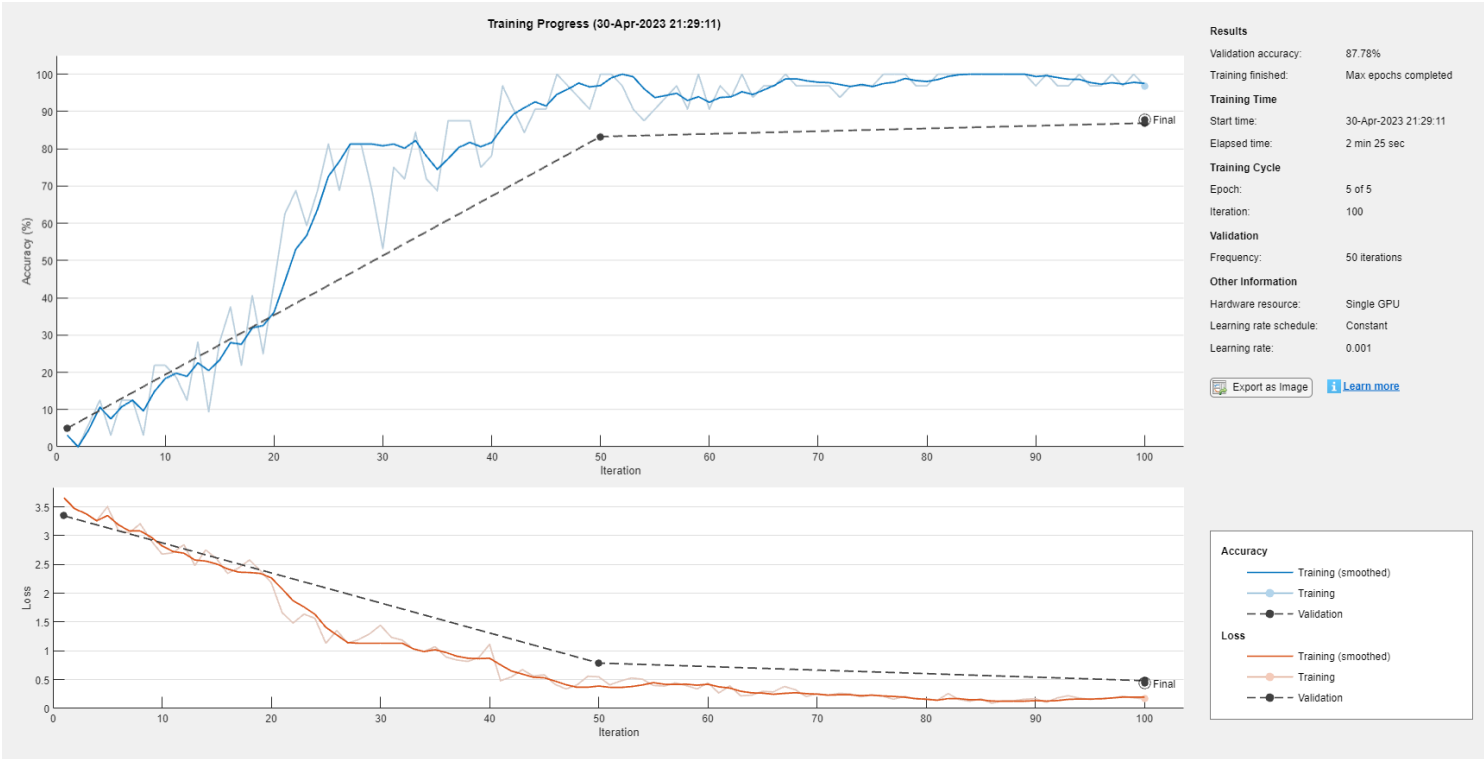




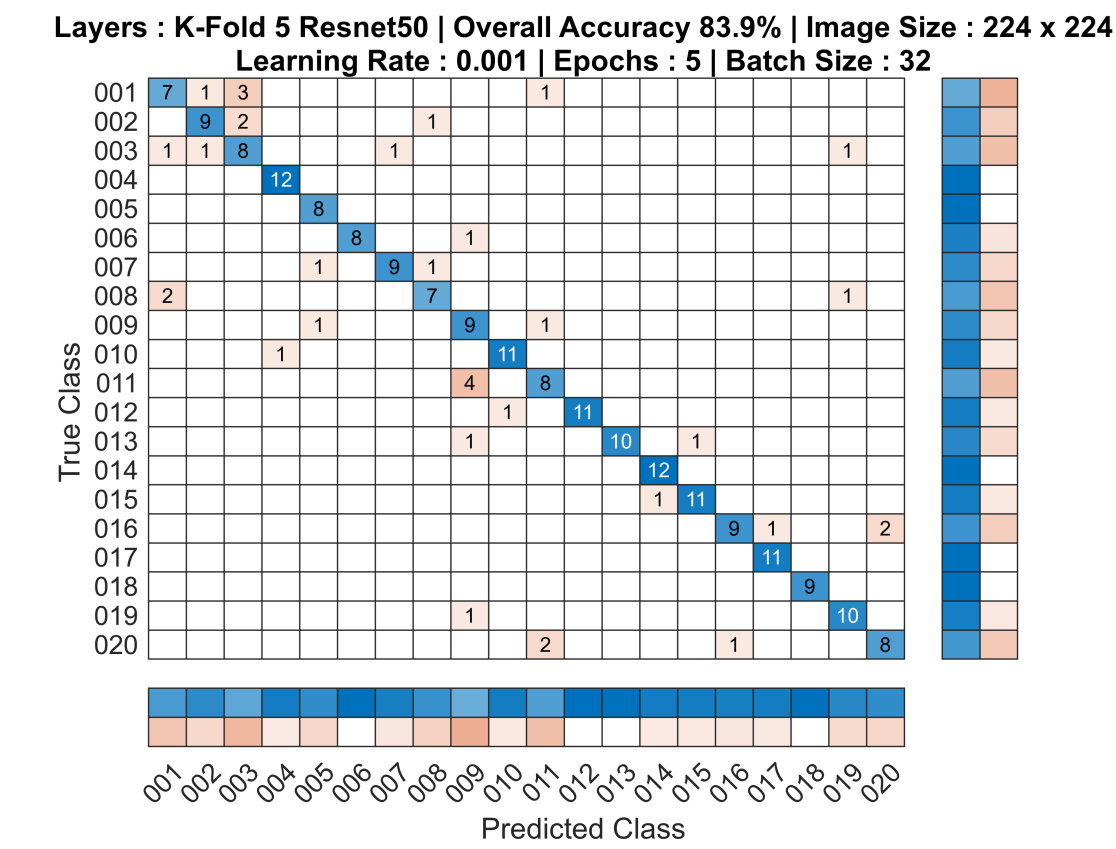
Overall Accuracy for Run 3 is: 0.8733

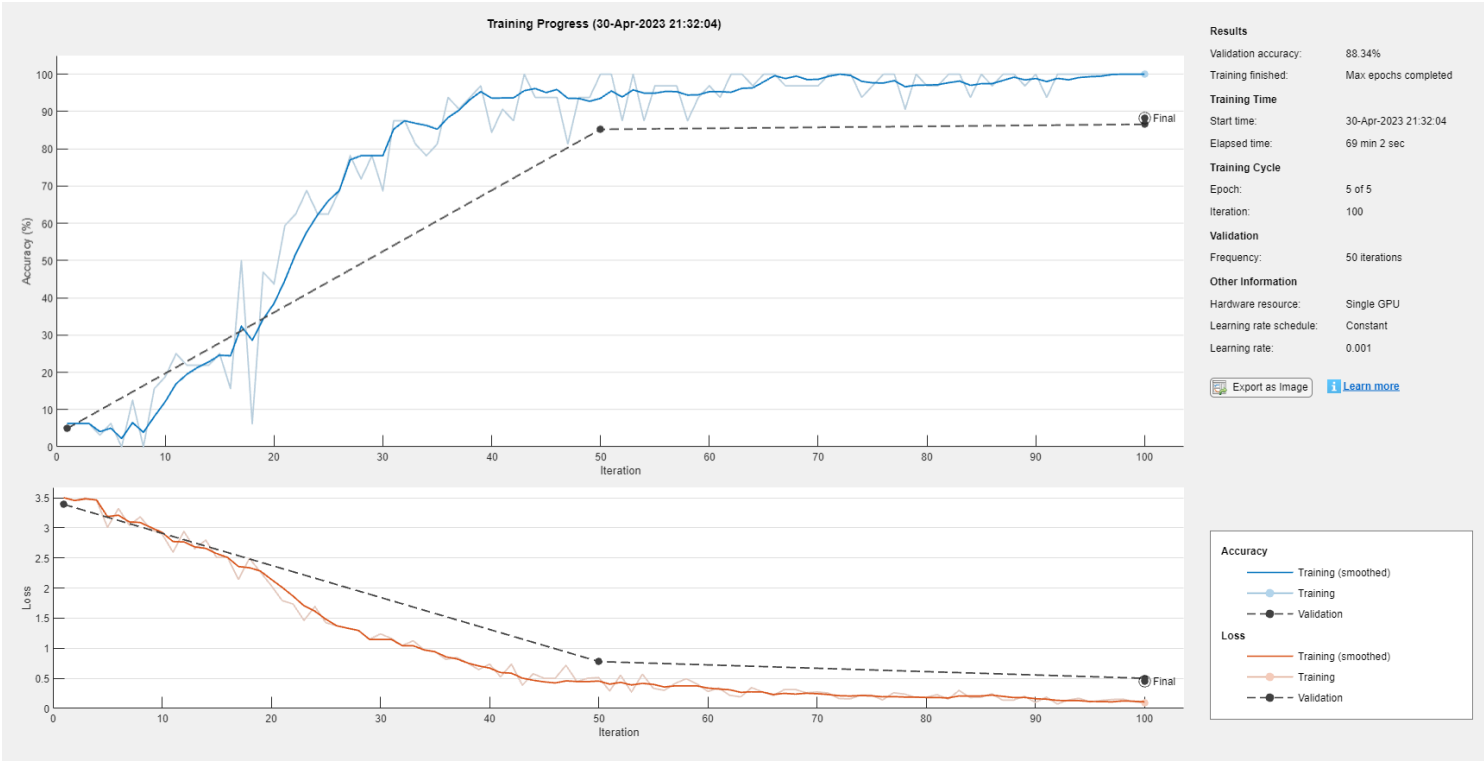
Layers : K-Fold 5 Resnet50 | Overall Accuracy 87.3% | Image Size : 224 x 224
Learning Rate : 0.001 | Epochs : 5 | Batch Size : 32



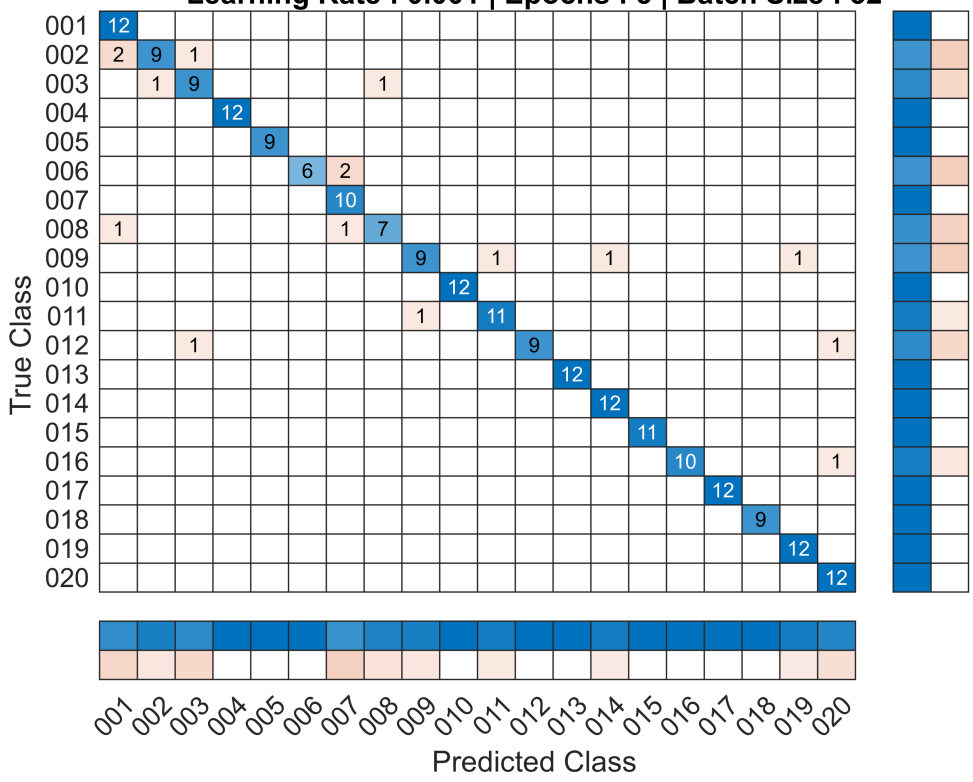


Overall Accuracy for Run 4 is: 0.83857





Overall Accuracy for Run 5 is: 0.9276




```
disp("Average accuracy of five folds is "+ string(accuracy_overall/folds))
```

```
Average accuracy of five folds is 0.86656
```