```
clc;
clear;
close all;
% Dataset Path
% InputPath = './Data';
imageFolder = './Data';
imds = imageDatastore(imageFolder, 'LabelSource', 'foldernames',
'IncludeSubfolders',true);
% Find the first instance of an image for each category
leaf = find(imds.Labels == '001', 1);
figure; imshow(readimage(imds,leaf))
tbl = countEachLabel(imds)
% Determine the smallest amount of images in a category
minSetCount = min(tbl{:,2});
% Limit the number of images to reduce the time it takes
% run this example.
maxNumImages = 100;
minSetCount = min(maxNumImages,minSetCount);
% Use splitEachLabel method to trim the set.
imds = splitEachLabel(imds, minSetCount, 'randomize');
% Notice that each set now has exactly the same number of images.
countEachLabel(imds)
% Load pretrained network
net = resnet50();
% Visualize the first section of the network.
figure
plot(net)
title('First section of ResNet-50')
set(gca,'YLim',[150 170]);
% Inspect the first layer
net.Layers(1)
% Inspect the last layer
net.Layers(end)
% Number of class names for ImageNet classification task
numel(net.Layers(end).ClassNames)
[trainingSet, testSet] = splitEachLabel(imds, 0.3, 'randomize');
% Create augmentedImageDatastore from training and test sets to resize
% images in imds to the size required by the network.
imageSize = net.Layers(1).InputSize;
augmentedTrainingSet = augmentedImageDatastore(imageSize, trainingSet,
'ColorPreprocessing', 'gray2rgb');
augmentedTestSet = augmentedImageDatastore(imageSize, testSet,
'ColorPreprocessing', 'gray2rgb');
% Get the network weights for the second convolutional layer
w1 = net.Layers(2).Weights;
% Scale and resize the weights for visualization
w1 = mat2gray(w1);
w1 = imresize(w1,5);
% Display a montage of network weights. There are 96 individual sets of
% weights in the first layer.
figure
montage(w1)
title('First convolutional layer weights')
```

```
featureLayer = 'fc1000';
trainingFeatures = activations(net, augmentedTrainingSet, featureLayer, ...
   'MiniBatchSize', 32, 'OutputAs', 'columns');
% Get training labels from the trainingSet
trainingLabels = trainingSet.Labels;
% Train multiclass SVM classifier using a fast linear solver, and set
% 'ObservationsIn' to 'columns' to match the arrangement used for training
% features.
classifier = fitcecoc(trainingFeatures, trainingLabels, ...
   'Learners', 'Linear', 'Coding', 'onevsall', 'ObservationsIn', 'columns');
% Extract test features using the CNN
testFeatures = activations(net, augmentedTestSet, featureLayer, ...
   'MiniBatchSize', 32, 'OutputAs', 'columns');
% Pass CNN image features to trained classifier
predictedLabels = predict(classifier, testFeatures, 'ObservationsIn',
'columns');
% Get the known labels
testLabels = testSet.Labels;
% Tabulate the results using a confusion matrix.
confMat = confusionmat(testLabels, predictedLabels);
% Convert confusion matrix into percentage form
confMat = bsxfun(@rdivide,confMat,sum(confMat,2))
% Display the mean accuracy
mean(diag(confMat))
testImage = readimage(testSet,1);
testLabel = testSet.Labels(1)
% Create augmentedImageDatastore to automatically resize the image when
% image features are extracted using activations.
ds = augmentedImageDatastore(imageSize, testImage, 'ColorPreprocessing',
'gray2rgb');
% Extract image features using the CNN
imageFeatures = activations(net, ds, featureLayer, 'OutputAs', 'columns');
% Make a prediction using the classifier
predictedLabel = predict(classifier, imageFeatures, 'ObservationsIn',
'columns')
```