Extract Image Features Using Pretrained Network

This example shows how to extract learned image features from a pretrained convolutional neural network and use those features to train an image classifier. Feature extraction is the easiest and fastest way to use the representational power of pretrained deep networks. For example, you can train a support vector machine (SVM) using `fitcecoc` (Statistics and Machine Learning Toolbox™) on the extracted features. Because feature extraction only requires a single pass through the data, it is a good starting point if you do not have a GPU to accelerate network training with.

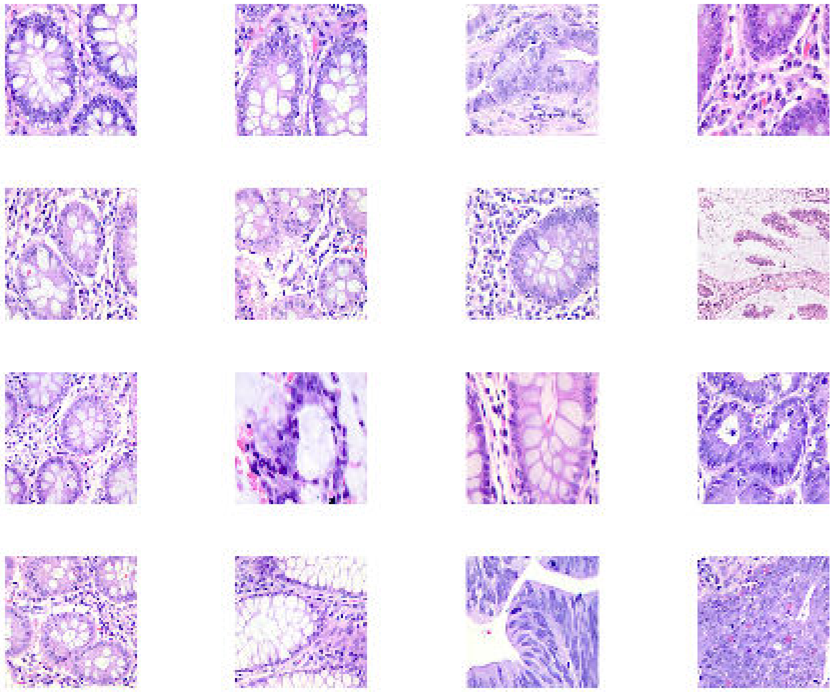
Load Data

Unzip and load the sample images as an image datastore. `imageDatastore` automatically labels the images based on folder names and stores the data as an `ImageDatastore` object. An image datastore lets you store large image data, including data that does not fit in memory. Split the data into 70% training and 30% test data.

```
imdsTrain = imageDatastore('C:\Users\manju\Downloads\complete dataset','IncludeSubfolders',true
%[imdsTrain,imdsTest] = splitEachLabel(imds,0.8,'randomized');
imdsCRAG = imageDatastore('C:\Users\manju\Downloads\crag_class\test','IncludeSubfolders',true,
imdsHosC = imageDatastore('C:\Users\manju\Downloads\Hosc_test_conf','IncludeSubfolders',true,'L
imdsTestA = imageDatastore('C:\Users\manju\Downloads\dataset\test_A','IncludeSubfolders',true,
imdsTestB = imageDatastore('C:\Users\manju\Downloads\dataset\test_B','IncludeSubfolders',true,
imdsTrain1 = imageDatastore('C:\Users\manju\Downloads\lc25000d','IncludeSubfolders',true,'Label
[imdsTrain2,imdsLC25000] = splitEachLabel(imdsTrain1,0.3,'randomized');
```

There are now 55 training images and 20 validation images in this very small data set. Display some sample images.

```
numTrainImages = numel(imdsTrain.Labels);
idx = randperm(numTrainImages,16);
figure
for i = 1:16
    subplot(4,4,i)
    I = readimage(imdsTrain,idx(i));
    imshow(I)
end
```



Load Pretrained Network

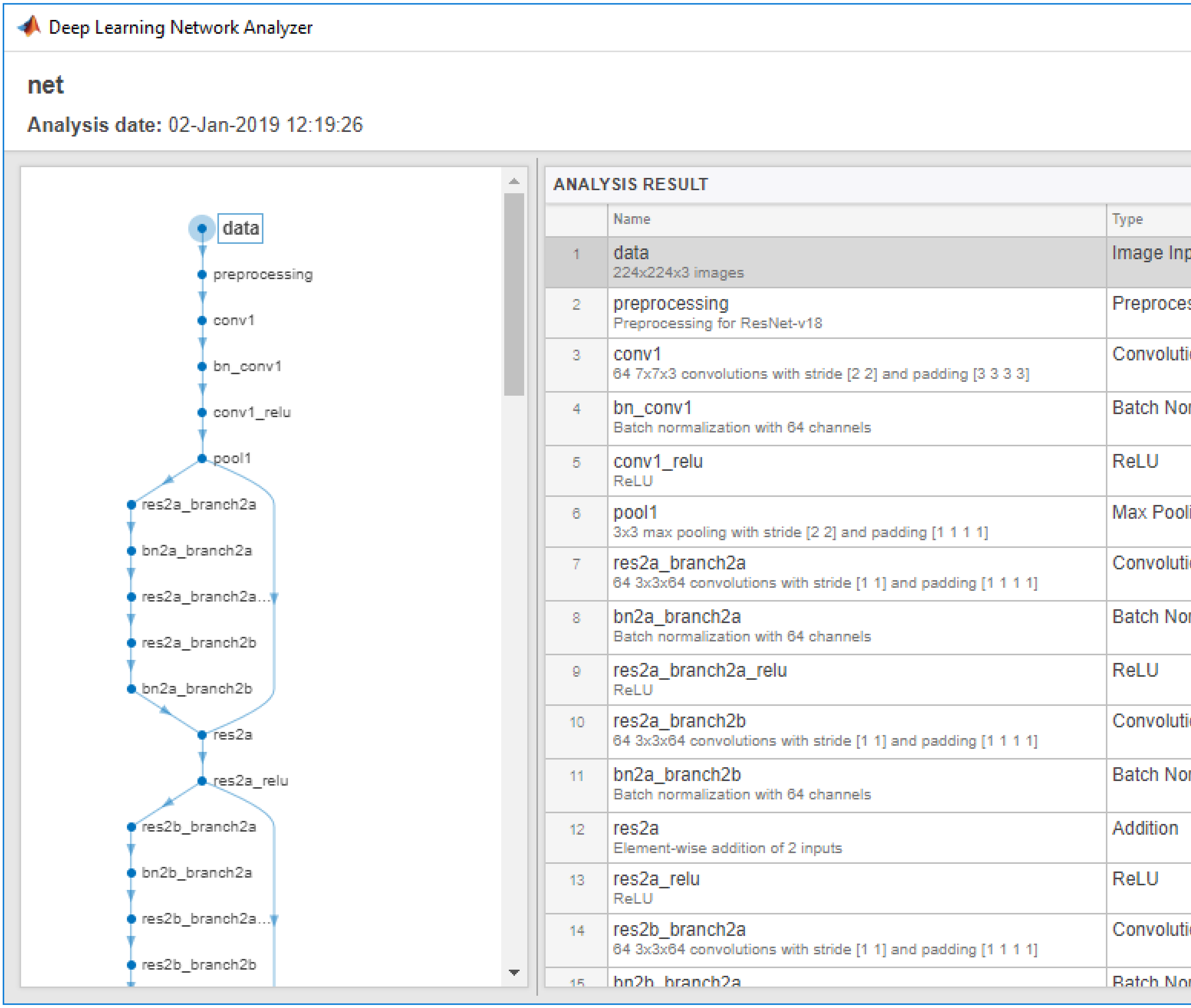
Load a pretrained ResNet-18 network. If the Deep Learning Toolbox Model *for ResNet-18 Network* support package is not installed, then the software provides a download link. ResNet-18 is trained on more than a million images and can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the model has learned rich feature representations for a wide range of images.

```
net = inceptionv3
```

```
net = 
  DAGNetwork with properties:

         Layers: [315×1 nnet.cnn.layer.Layer]
    Connections: [349×2 table]
     InputNames: {'input_1'}
    OutputNames: {'ClassificationLayer_predictions'}
```

Analyze the network architecture. The first layer, the image input layer, requires input images of size 224-by-224-by-3, where 3 is the number of color channels.

```
inputSize = net.Layers(1).InputSize;
analyzeNetwork(net)
```



Extract Image Features

The network requires input images of size 224-by-224-by-3, but the images in the image datastores have different sizes. To automatically resize the training and test images before they are input to the network, create augmented image datastores, specify the desired image size, and use these datastores as input arguments to `activations`.

```
augimdsTrain = augmentedImageDatastore(inputSize(1:2),imdsTrain);
augimdsCRAG = augmentedImageDatastore(inputSize(1:2),imdsCRAG);
augimdsHosC = augmentedImageDatastore(inputSize(1:2),imdsHosC);
augimdsTestA = augmentedImageDatastore(inputSize(1:2),imdsTestA);
augimdsTestB = augmentedImageDatastore(inputSize(1:2),imdsTestB);
```

The network constructs a hierarchical representation of input images. Deeper layers contain higher-level features, constructed using the lower-level features of earlier layers. To get the feature representations of the training and test images, use `activations` on the global pooling layer, `'pool5'`, at the end of the network. The global pooling layer pools the input features over all spatial locations, giving 512 features in total.

```
layer = 'avg_pool';
featuresTrain = activations(net,augimdsTrain,layer,'OutputAs','rows');
whos featuresTrain
```

```
  Name                  Size                  Bytes  Class     Attributes

  featuresTrain      10258x2048            84033536  single
```

```
featuresTestA = activations(net,augimdsTestA,layer,'OutputAs','rows');
featuresTestB = activations(net,augimdsTestB,layer,'OutputAs','rows');
featuresCRAG = activations(net,augimdsCRAG,layer,'OutputAs','rows');
featuresHosC = activations(net,augimdsHosC,layer,'OutputAs','rows');
```

```
augimdslc25000 = augmentedImageDatastore(inputSize(1:2),imdsLC25000);
featureslc25000 = activations(net,augimdslc25000,layer,'OutputAs','rows');
YTestlc25000 = imdsLC25000.Labels;
```

Extract the class labels from the training and test data.

```
YTrain = imdsTrain.Labels;
YTestA = imdsTestA.Labels;
YTestB = imdsTestB.Labels;
YTestCRAG = imdsCRAG.Labels;
YTestHosC = imdsHosC.Labels;
```

Fit Image Classifier

Use the features extracted from the training images as predictor variables and fit a multiclass support vector machine (SVM) using `fitcecoc` (Statistics and Machine Learning Toolbox).

```
 start = tic;
   % [net,info] = trainNetwork(augimdsTrain,lgraph,options);

classifier = fitcecoc(featuresTrain,YTrain);
  elapsed = toc(start)
```

```
elapsed = 2.7761
```

Classify Test Images

Classify the test images using the trained SVM model using the features extracted from the test images.

```
start = tic;
[YPredA,score_testa] = predict(classifier,featuresTestA);
elapsed = toc(start)
```

```
elapsed = 0.0394
```

```
score_testa
```

```
score_testa = 60×2 single matrix
         0   -2.2423
         0   -2.2671
         0   -1.0231
         0   -1.7612
         0   -1.5713
   -0.7306   -0.2694
         0   -1.0635
         0   -1.3055
         0   -2.1314
         0   -1.9921
      ⋮
```

```
classifier.ClassNames
```

```
ans = 2×1 categorical
0
1
```

```
start = tic;
YPredB = predict(classifier,featuresTestB);
elapsedB = toc(start)
```

```
elapsedB = 0.0072
```

```
start = tic;
YPredCRAG = predict(classifier,featuresCRAG);
elapsedCRAG = toc(start)
```

```
elapsedCRAG = 0.0054
```

```
start = tic;
YPredHosC = predict(classifier,featuresHosC);
elapsedHosC = toc(start)
```

```
elapsedHosC = 0.0061
```

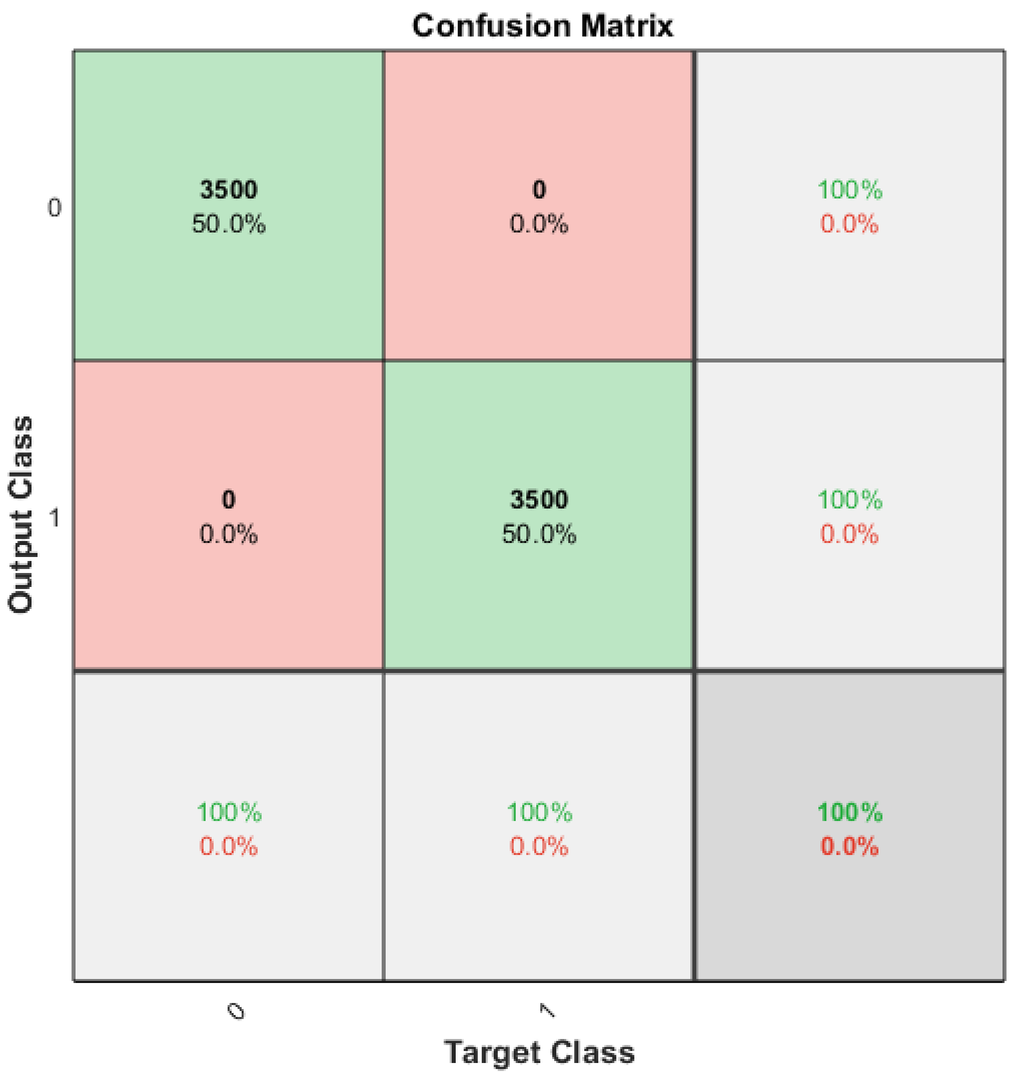
```
start = tic;
YPredlc25000 = predict(classifier,featureslc25000);
elapsedLC25000 = toc(start)
```

```
elapsedLC25000 = 0.0392
```

```
accuracy = mean(YPredlc25000 == YTestlc25000)
```

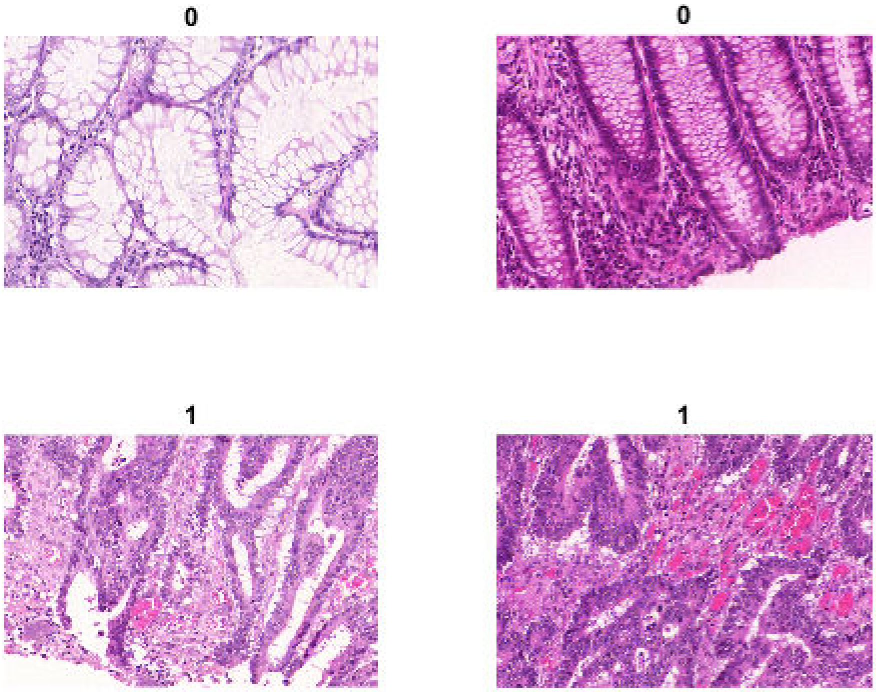
```
accuracy = 1
```

```
plotconfusion(YTestlc25000,YPredlc25000)
```



Display four sample test images with their predicted labels.

```
idx = [1 7 40 60];
figure
for i = 1:numel(idx)
    subplot(2,2,i)
    I = readimage(imdsTestA,idx(i));
    label = YPredA(idx(i));
    imshow(I)
    title(char(label))
    %title(string(label) + ", " + num2str(5*probs(idx(i))) + "%");
end
```

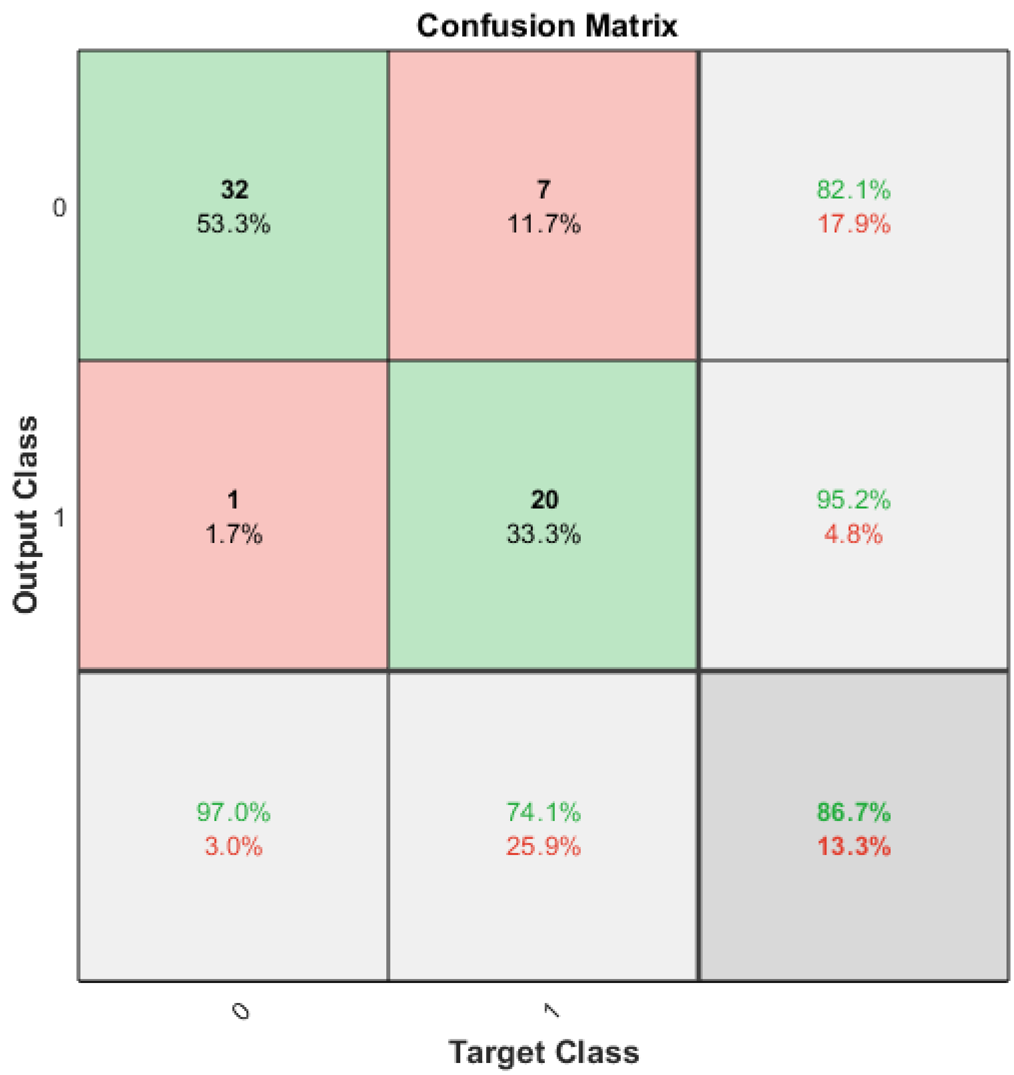


Calculate the classification accuracy on the test set. Accuracy is the fraction of labels that the network predicts correctly.

```
accuracyA = mean(YPredA == YTestA)
```

```
accuracyA = 0.8667
```

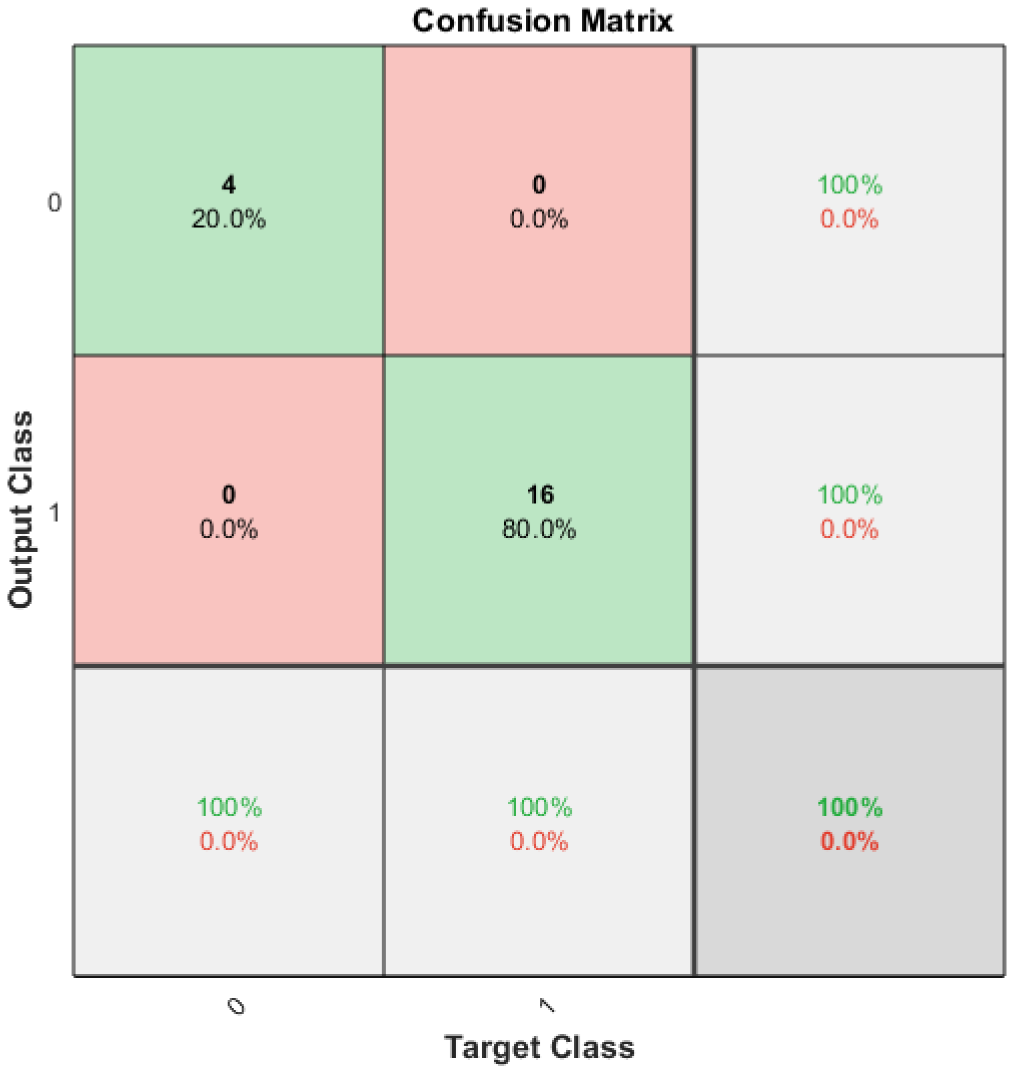
```
plotconfusion(YTestA,YPredA)
```



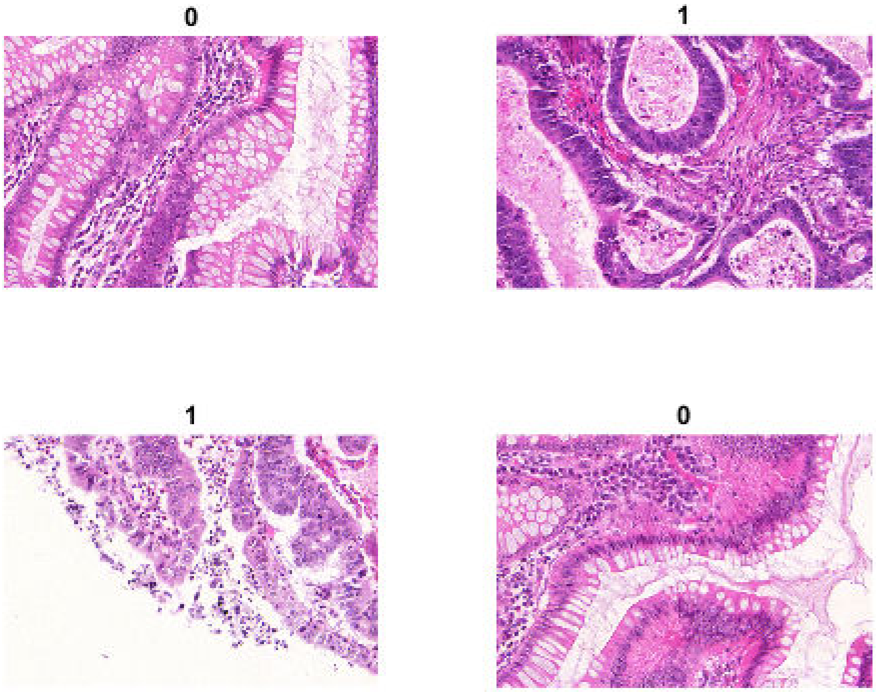

```
accuracyB = mean(YPredB == YTestB)
```

```
accuracyB = 1
```

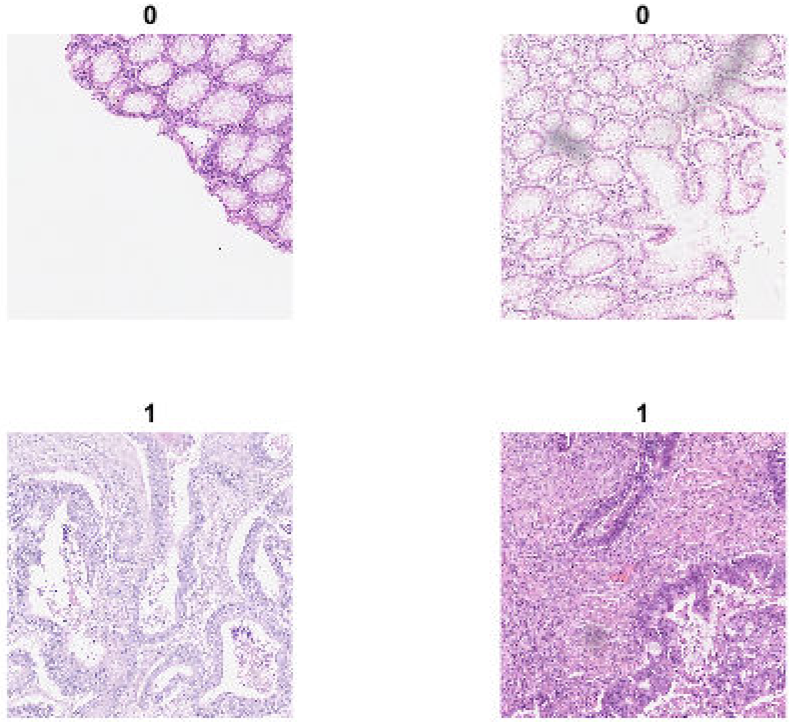
```
plotconfusion(YTestB,YPredB)
```




```
idx = [1 17 10 2];
figure
for i = 1:numel(idx)
    subplot(2,2,i)
    I = readimage(imdsTestB,idx(i));
    label = YPredB(idx(i));
    imshow(I)
    title(char(label))
    %title(string(label) + ", " + num2str(5*probs(idx(i))) + "%");
end
```



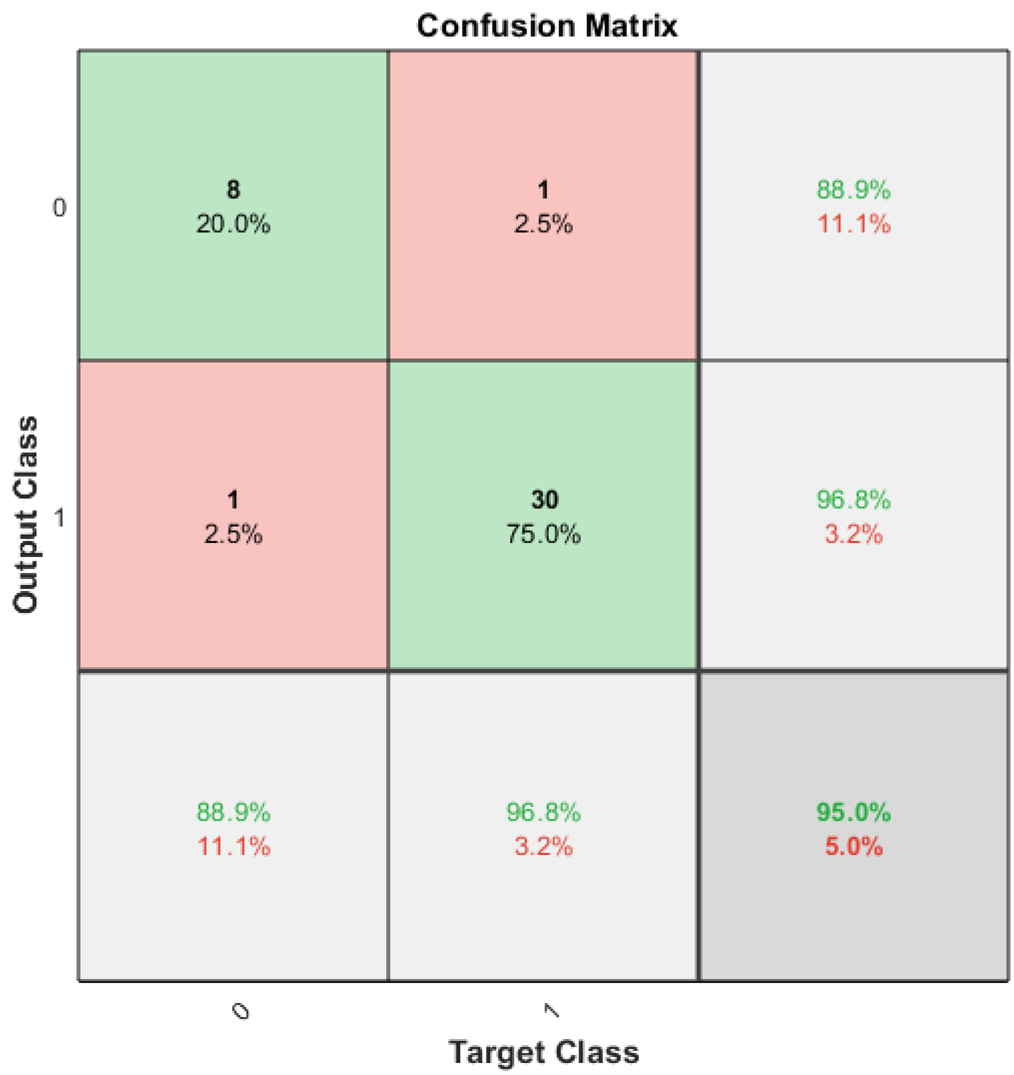

```
idx = [1 7 40 26];
figure
for i = 1:numel(idx)
    subplot(2,2,i)
    I = readimage(imdsCRAG,idx(i));
    label = YPredCRAG(idx(i));
    imshow(I)
    title(char(label))
    %title(string(label) + ", " + num2str(5*probs(idx(i))) + "%");
end
```



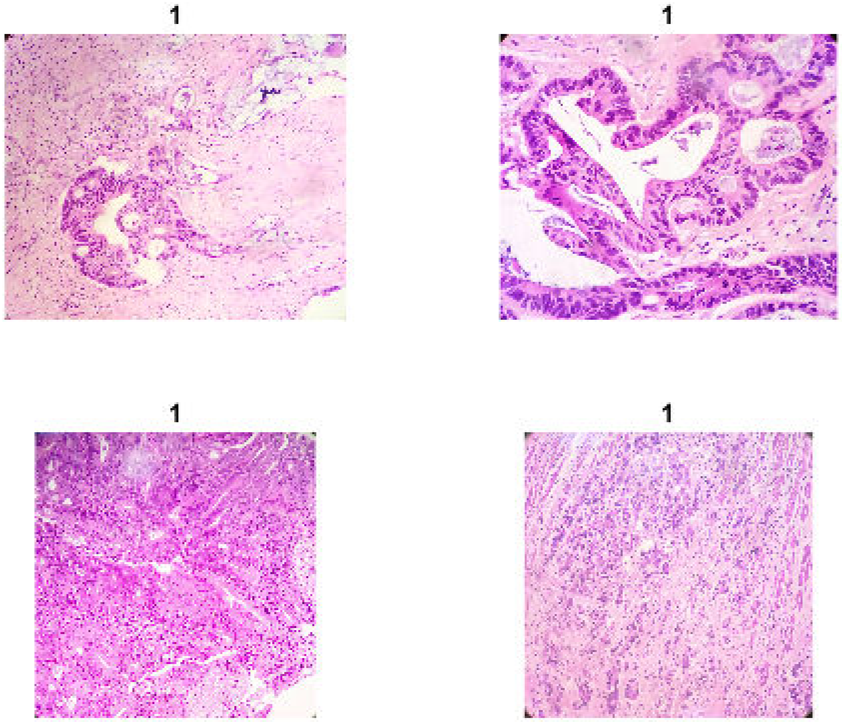

```
accuracyCRAG = mean(YPredCRAG == YTestCRAG)
```

```
accuracyCRAG = 0.9500
```

```
plotconfusion(YTestCRAG,YPredCRAG)
```



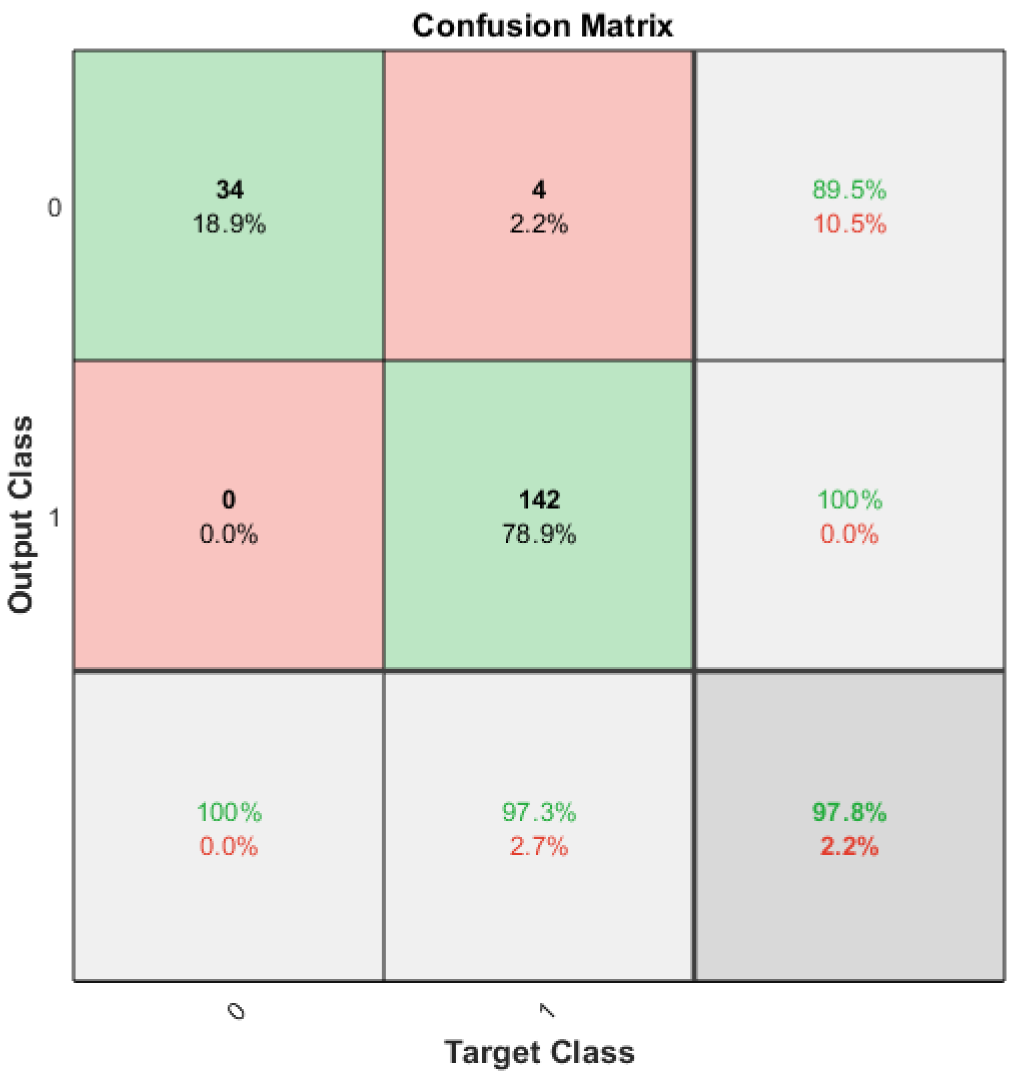

```
idx = [35 47 41 53];
figure
for i = 1:numel(idx)
    subplot(2,2,i)
    I = readimage(imdsHosC,idx(i));
    label = YPredHosC(idx(i));
    imshow(I)
    title(char(label))
    %title(string(label) + ", " + num2str(5*probs(idx(i))) + "%");
end
```




```
accuracyHosC = mean(YPredHosC == YTestHosC)
```

```
accuracyHosC = 0.9778
```

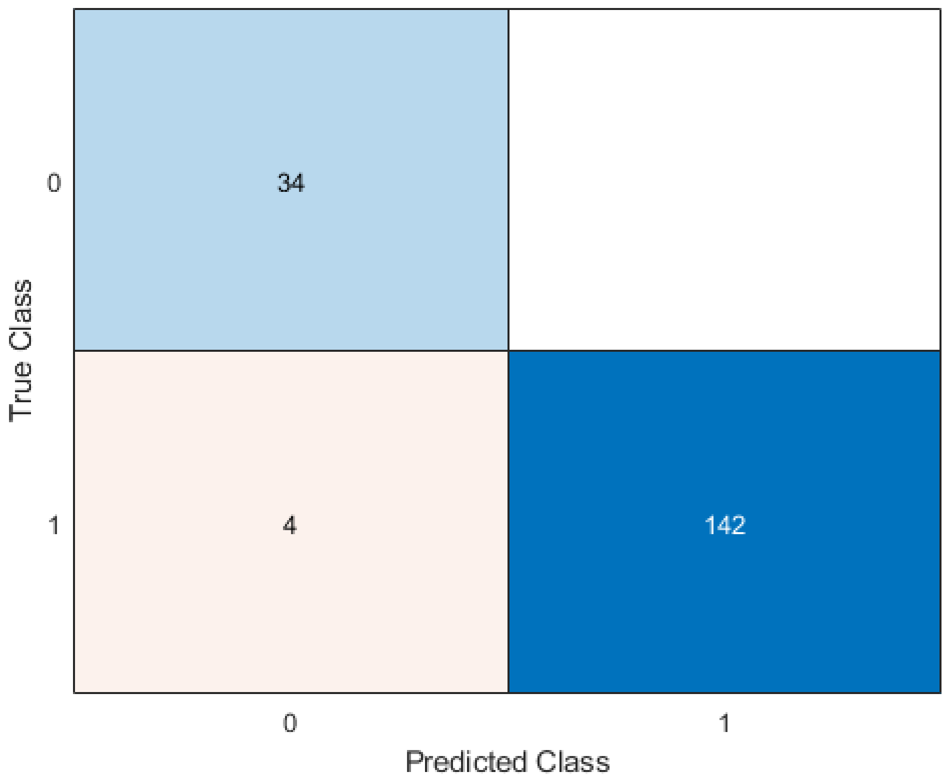
```
plotconfusion(YTestHosC,YPredHosC)
```



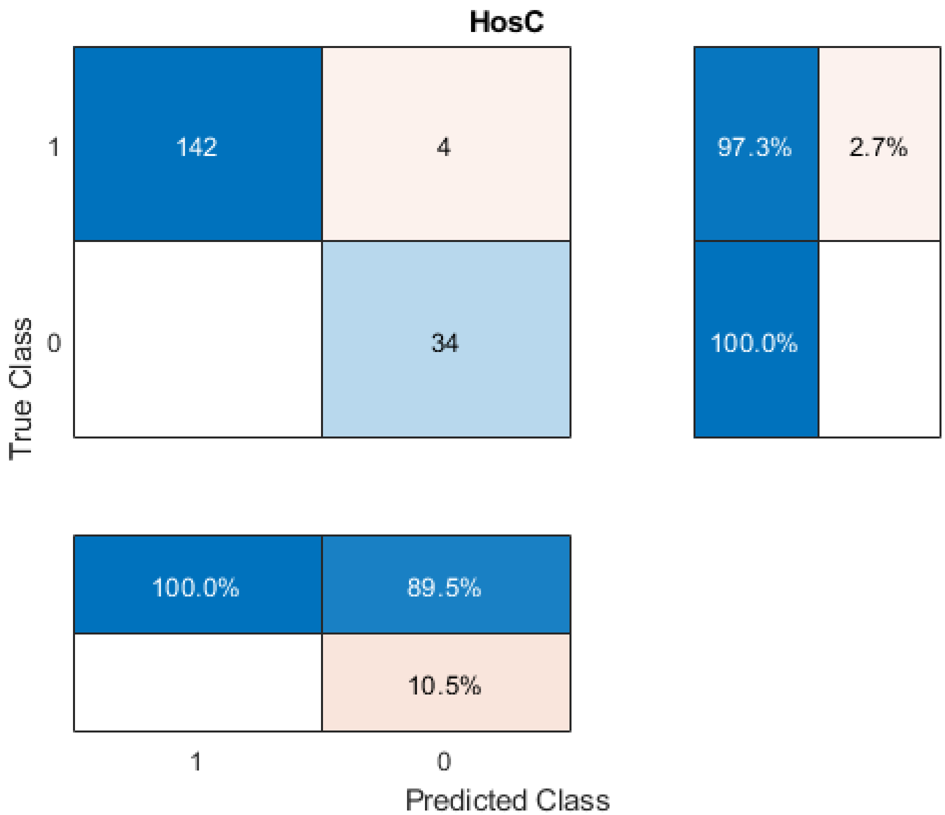

```
[m,order] = confusionmat(YTestHosC,YPredHosC)
```

```
m = 2×2
    34     0
     4   142
order = 2×1 categorical
0
1
```

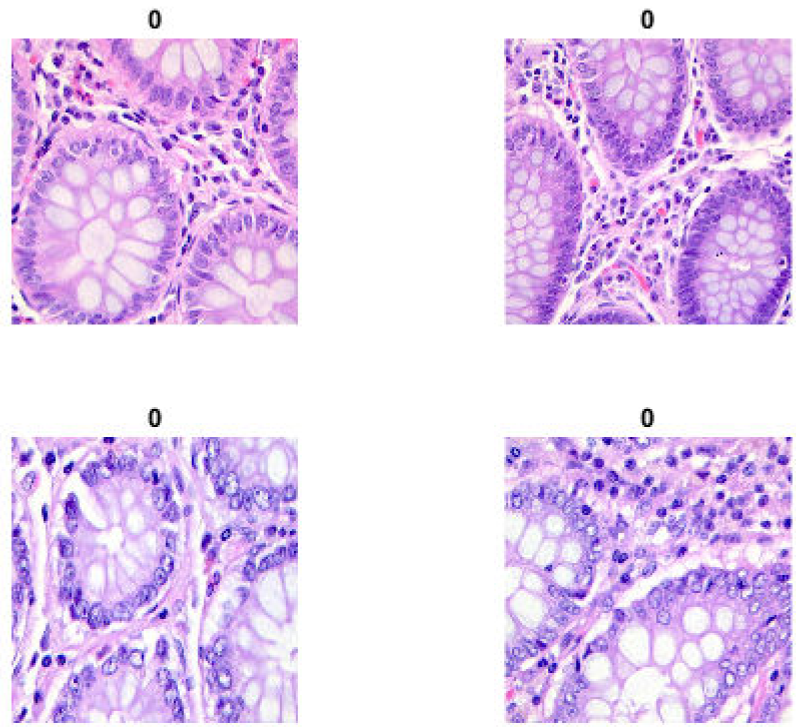
```
figure
cm = confusionchart(m,order);
```




```
figure
cm = confusionchart(YTestHosC,YPredHosC, ...
    'Title','HosC', ...
    'RowSummary','row-normalized', ...
    'ColumnSummary','column-normalized');
% class wise recall
cm.Normalization = 'row-normalized';
sortClasses(cm,'descending-diagonal');
cm.Normalization = 'absolute';
% class wise precision
cm.Normalization = 'column-normalized';
sortClasses(cm,'descending-diagonal');
cm.Normalization = 'absolute';
```



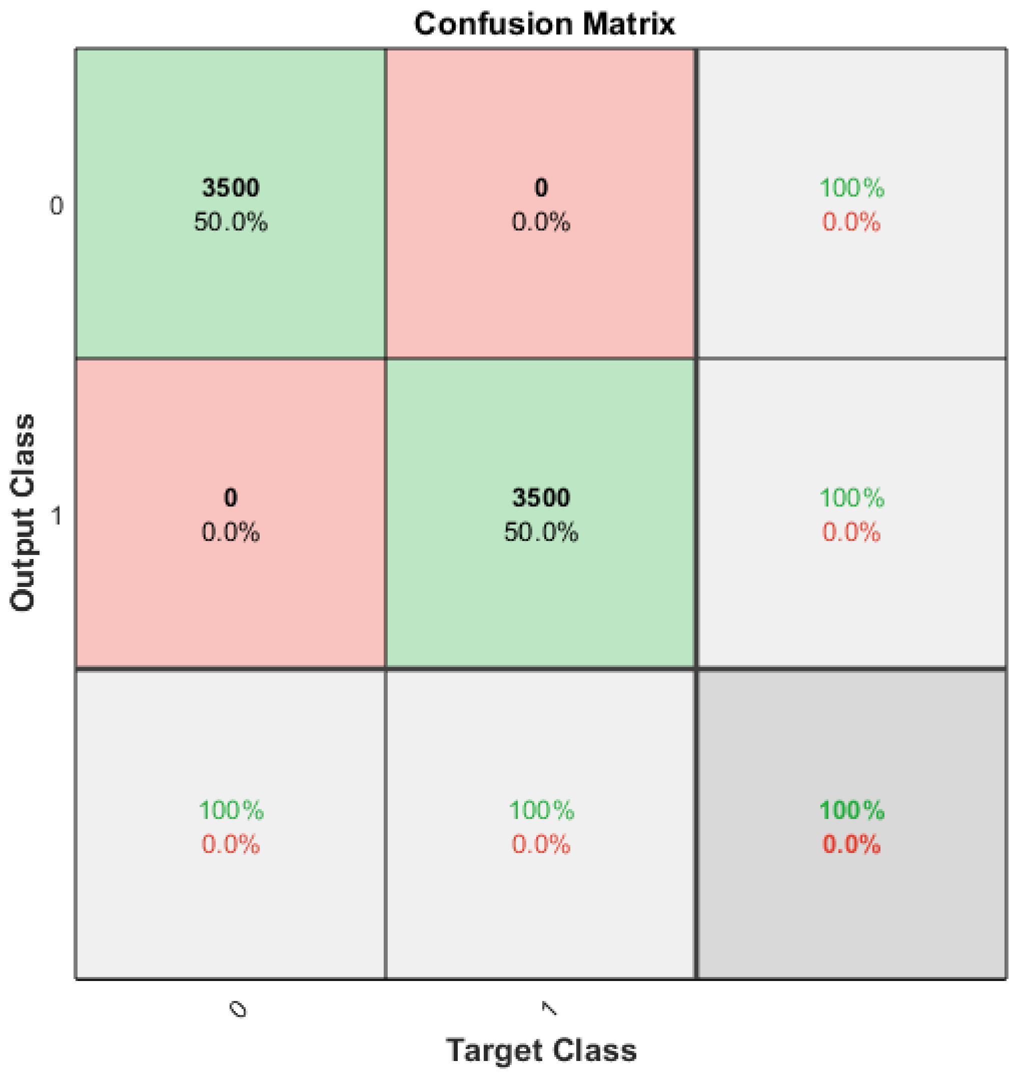

```
idx = [2500 1070 405 600];
figure
for i = 1:numel(idx)
    subplot(2,2,i)
    I = readimage(imdsLC25000,idx(i));
    label = YPredlc25000(idx(i));
    imshow(I)
    title(char(label))
    %title(string(label) + ", " + num2str(5*probs(idx(i))) + "%");
end
```




```
accuracylc25000 = mean(YPredlc25000 == YTestlc25000)
```

```
accuracylc25000 = 1
```

```
plotconfusion(YTestlc25000,YPredlc25000)
```



Both trained SVMs have high accuracies. If the accuracy is not high enough using feature extraction, then try transfer learning instead. For an example, see Train Deep Learning Network to Classify New Images. For a list and comparison of the pretrained networks, see Pretrained Deep Neural Networks.

```
X = imread("C:\Users\manju\Downloads\crag_class\test\1\test_4.png");
inputSize = net.Layers(1).InputSize(1:2);
X = imresize(X,inputSize);
```

Classify the image to get the class label.

```
label = classify(net,X)
```

```
label = categorical
```

```
brain coral
```

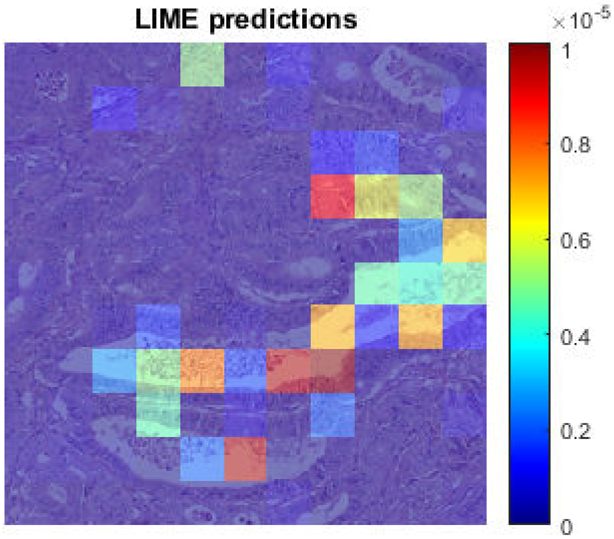
Compute the map of the feature importance and also obtain the map of the features and the feature importance. Set the image segmentation method to `'grid'`, the number of features to `64`, and the number of synthetic images to `3072`.

```
[scoreMap,featureMap,featureImportance]  = imageLIME(net,X,label,'Segmentation','grid','NumFeat
```

```
scoreMap = 299×299
10^-4 ×
    0.0010    0.0010    0.0010    0.0010    0.0010    0.0010    0.0010    0.0010 ···
    0.0010    0.0010    0.0010    0.0010    0.0010    0.0010    0.0010    0.0010
    0.0010    0.0010    0.0010    0.0010    0.0010    0.0010    0.0010    0.0010
    0.0010    0.0010    0.0010    0.0010    0.0010    0.0010    0.0010    0.0010
    0.0010    0.0010    0.0010    0.0010    0.0010    0.0010    0.0010    0.0010
    0.0010    0.0010    0.0010    0.0010    0.0010    0.0010    0.0010    0.0010
    0.0010    0.0010    0.0010    0.0010    0.0010    0.0010    0.0010    0.0010
    0.0010    0.0010    0.0010    0.0010    0.0010    0.0010    0.0010    0.0010
    0.0010    0.0010    0.0010    0.0010    0.0010    0.0010    0.0010    0.0010
    0.0010    0.0010    0.0010    0.0010    0.0010    0.0010    0.0010    0.0010
      ⋮
featureMap = 299×299
     1     1     1     1     1     1     1     1     1     1     1     1     1 ···
     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1     1     1     1
      ⋮
featureImportance = 121×1
10^-4 ×
    0.0010
    0.0004
    0.0001
    0.0007
    0.0500
    0.0001
    0.0089
    0.0027
    0.0009
    0.0000
      ⋮
```

Plot the result over the original image with transparency to see which areas of the image affect the classification score.

```
figure
imshow(X)
hold on
imagesc(scoreMap,'AlphaData',0.5)
colormap jet
colorbar
title(sprintf("LIME predictions"))
```



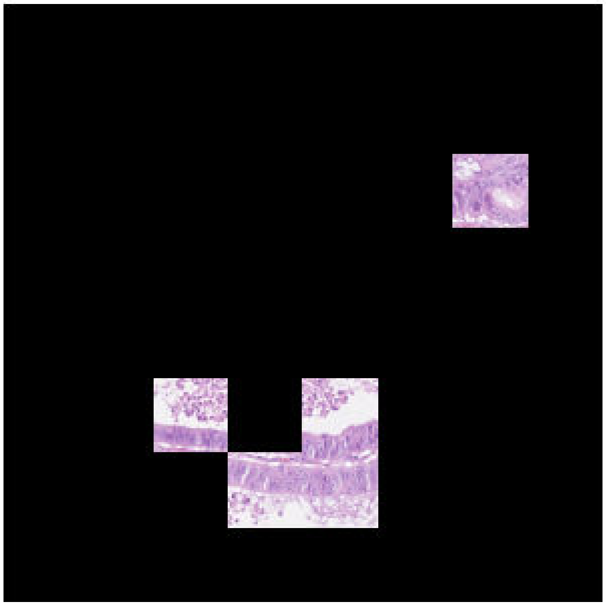
Use the feature importance to find the indices of the most important five features.

```
numTopFeatures = 20;
[~,idx] = maxk(featureImportance,numTopFeatures)
```

```
idx = 5×1
    43
    23
    53
    52
    45
```

Use the map of the features to mask out the image so only the most important five features are visible. Display the masked image.

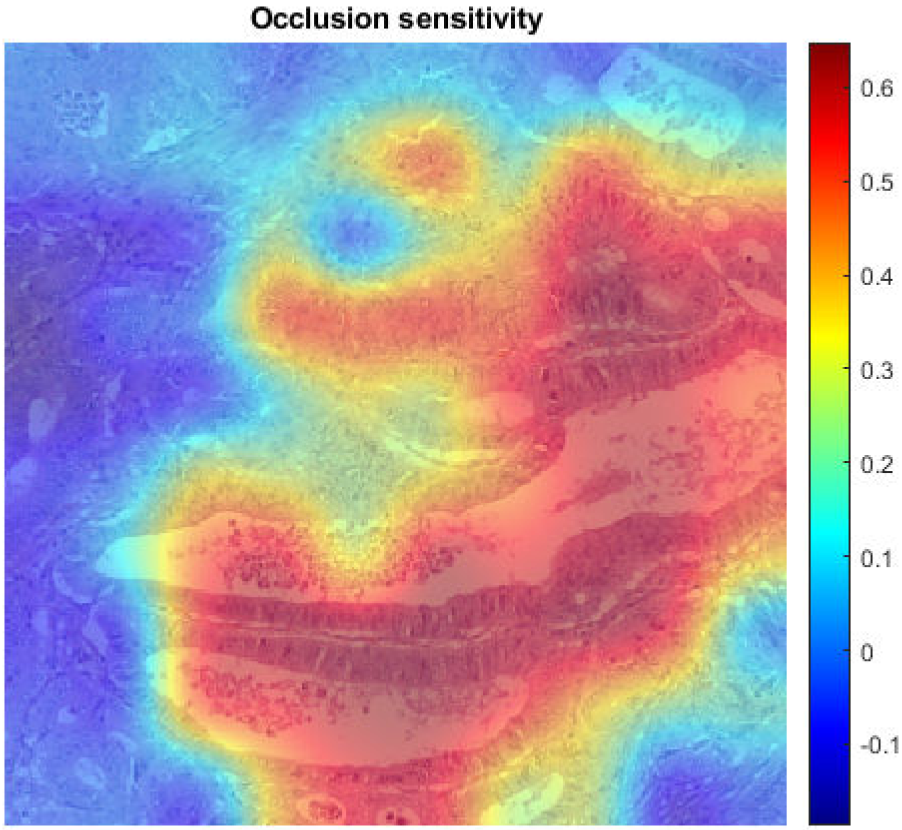
```
mask = ismember(featureMap,idx);
maskedImg = uint8(mask).*X;
figure
imshow(maskedImg);
```



```
map = occlusionSensitivity(net,X,label)
```

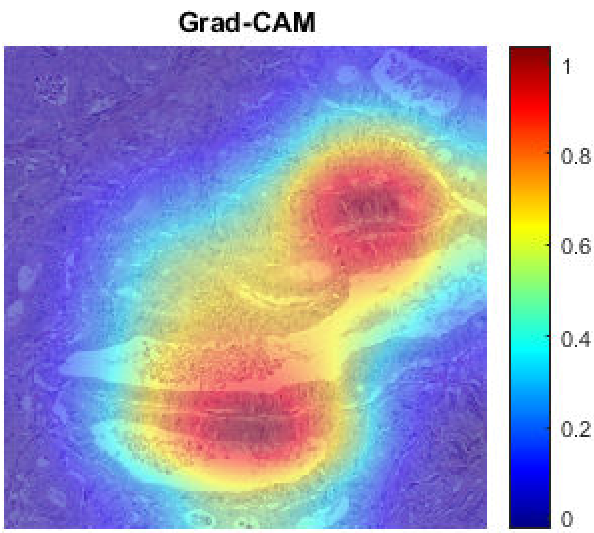
```
map = 299×299 single matrix
    0.0399    0.0399    0.0399    0.0399    0.0398    0.0398    0.0397    0.0396 ···
    0.0399    0.0399    0.0398    0.0398    0.0398    0.0397    0.0397    0.0396
    0.0398    0.0398    0.0398    0.0397    0.0397    0.0396    0.0396    0.0395
    0.0397    0.0397    0.0396    0.0396    0.0396    0.0395    0.0395    0.0394
    0.0395    0.0395    0.0395    0.0394    0.0394    0.0393    0.0393    0.0392
    0.0393    0.0393    0.0392    0.0392    0.0392    0.0391    0.0391    0.0390
    0.0390    0.0390    0.0390    0.0389    0.0389    0.0389    0.0388    0.0388
    0.0387    0.0387    0.0387    0.0386    0.0386    0.0386    0.0385    0.0385
    0.0383    0.0383    0.0383    0.0383    0.0383    0.0382    0.0382    0.0381
    0.0379    0.0379    0.0379    0.0379    0.0379    0.0378    0.0378    0.0377
      ⋮
```

```
imshow(X,'InitialMagnification', 150)
hold on
imagesc(map,'AlphaData',0.5)
colormap jet
colorbar
title(sprintf("Occlusion sensitivity"))
```




```
[classfn,score] = classify(net,X);
imshow(X);
title(sprintf("%s (%.2f)", classfn, score(classfn)));


%GoogLeNet correctly classifies the image as a golden retriever. But why? What characteristics

%Grad-CAM Explains Why
%The Grad-CAM technique utilizes the gradients of the classification score with respect to the

%The gradCAM function computes the importance map by taking the derivative of the reduction lay

%Compute the Grad-CAM map.

map = gradCAM(net,X,classfn);
%Show the Grad-CAM map on top of the image by using an 'AlphaData' value of 0.5. The 'jet' colo

imshow(X);
hold on;
imagesc(map,'AlphaData',0.5);
```

```
colormap jet
hold off;
title("Grad-CAM");
```




```
%Clearly, the upper face and ear of the dog have the greatest impact on the classification.

%For a different approach to investigating the reasons for deep network classifications, see oc
```