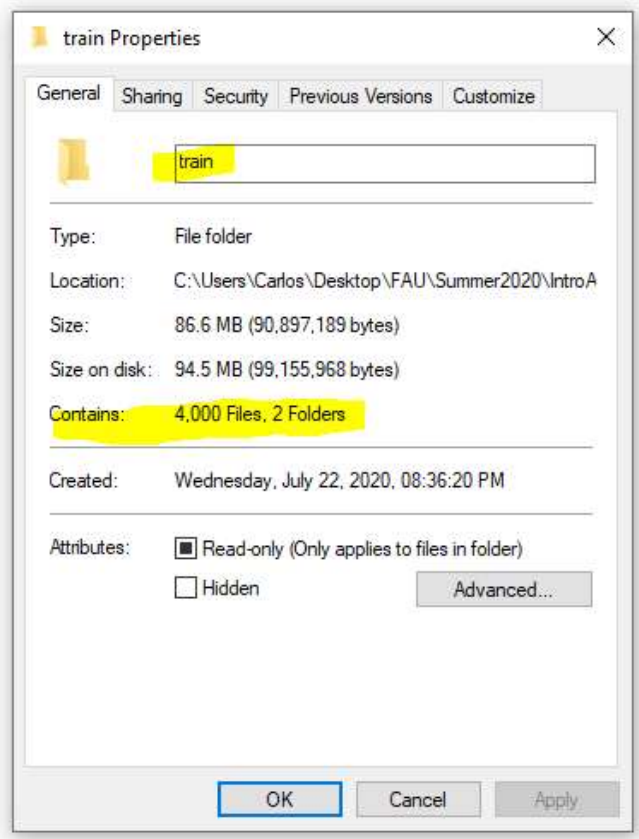
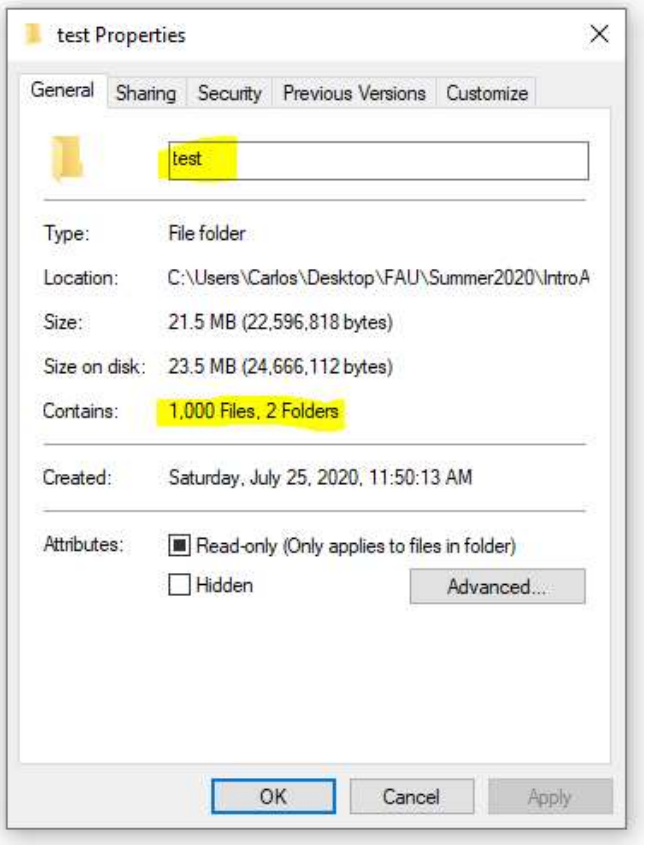
NasNetLarge Network 4000 pictures data set split 80% training and 20% Validation and 1000 pictures to test

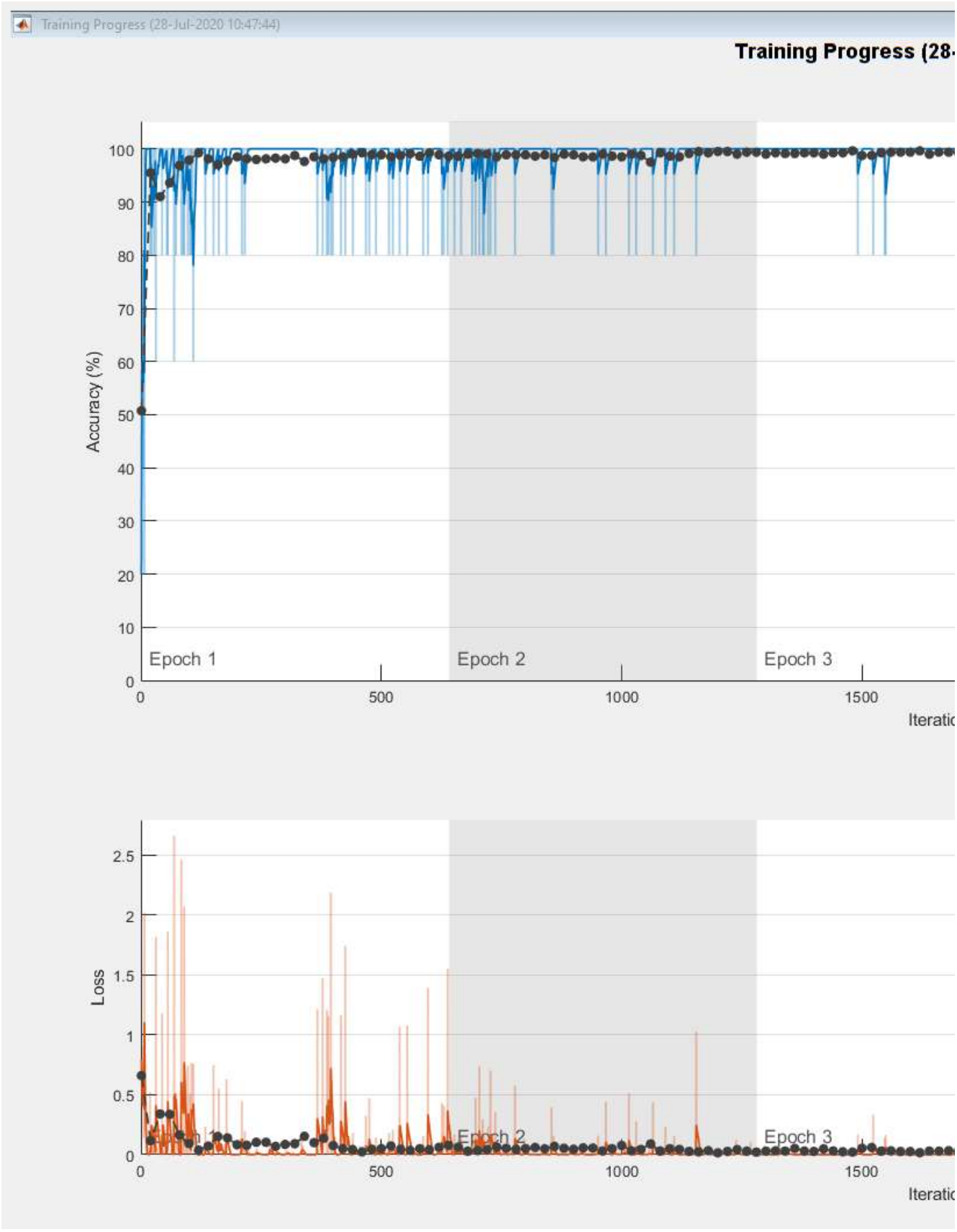
```
% get NasNetLarge CNN
model = nasnetlarge;
```






```
% set the path to our data folders
dataFolder = './data/train';
testFolder = './data/test';

% create the variable that will hold our categories
categories = {'cat', 'dog'};

% create our image databases
imds = imageDatastore(fullfile(dataFolder, categories), 'LabelSource', 'foldernames');
imdsTest = imageDatastore(fullfile(testFolder, categories),'LabelSource', 'foldernames');

% split the imds database into a training and validation set, giving 80% to the
% training set and the remaining 20% to the validation set
[trainingSet, validationSet] = splitEachLabel(imds, 0.8, 'randomized');

% get input image size
inputSize = model.Layers(1).InputSize;

% Augmenting the training, validation, and imdsTest image data stores
augimdsTrain = augmentedImageDatastore(inputSize(1:2),trainingSet,'ColorPreprocessing','gray2rgb');
augimdsValidation = augmentedImageDatastore(inputSize(1:2),validationSet,'ColorPreprocessing','gray2rgb');
augimdsTest = augmentedImageDatastore(inputSize(1:2),imdsTest, 'ColorPreprocessing','gray2rgb');

% Layer transfer
```

```
LayerGraph = layerGraph(model);
numClasses = 2;

newFullyConnectedLayer = fullyConnectedLayer(numClasses, ...
    'Name','new_fc', ...
    'WeightLearnRateFactor',20, ...
    'BiasLearnRateFactor',20);

newClassLayer = classificationLayer('Name','new_classoutput');

fclName = LayerGraph.Layers(end-2).Name;
classLayerName = LayerGraph.Layers(end).Name;

LayerGraph = replaceLayer(LayerGraph,fclName,newFullyConnectedLayer);
LayerGraph = replaceLayer(LayerGraph,classLayerName ,newClassLayer);

% set training options for model
options = trainingOptions('sgdm', ...
    'MiniBatchSize',5, ...
    'MaxEpochs',5, ...
    'InitialLearnRate',1e-3, ...
    'Shuffle','every-epoch', ...
    'ValidationData',augimdsValidation, ...
    'ValidationFrequency',20, ...
    'Verbose',false, ...
    'ExecutionEnvironment','auto', ...
    'Plots','training-progress', ...
    'CheckpointPath','./Checkpoints');

% train the new network
modelTransfer = trainNetwork(augimdsTrain,LayerGraph,options);
```




```
% saving workspace
save('./WorkSpace/AITrainedWorkSpace');

% classify the images in our validation and test folders
[ValPred, scores] = classify(modelTransfer,augimdsValidation);
[ValPredTest, scoresTest] = classify(modelTransfer, augimdsTest);
```

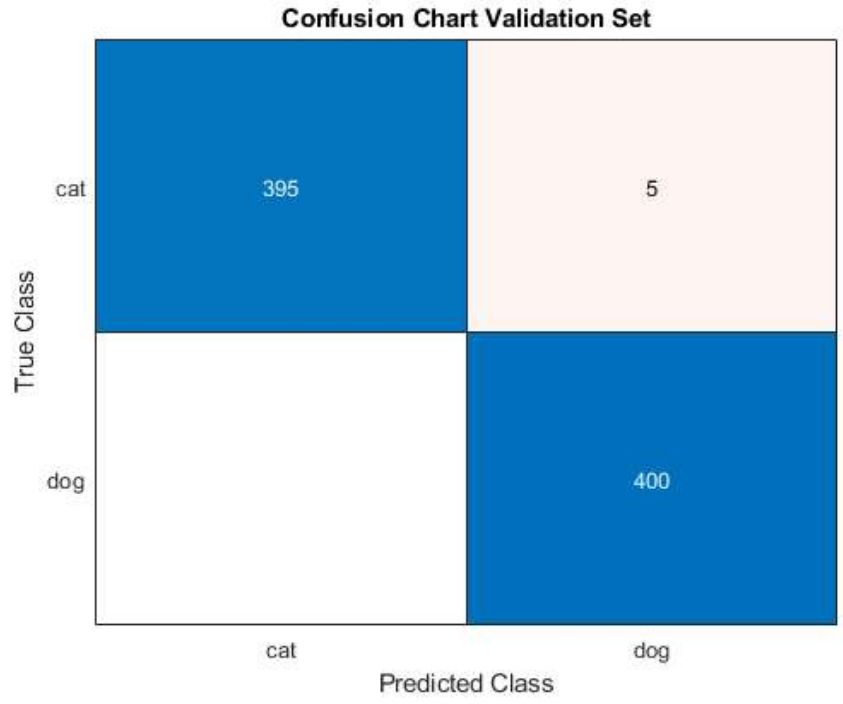
```
% obtain the right classification of image base on folder labels
ValidationActual = validationSet.Labels;
ValidationActualTest = imdsTest.Labels;

% validation accuracy for validation set
accuracy = mean(ValPred == ValidationActual);
fprintf("The validation accuracy for validation set is: %.2f %%\n", accuracy * 100);
```

```
The validation accuracy for validation set is: 99.38 %
```

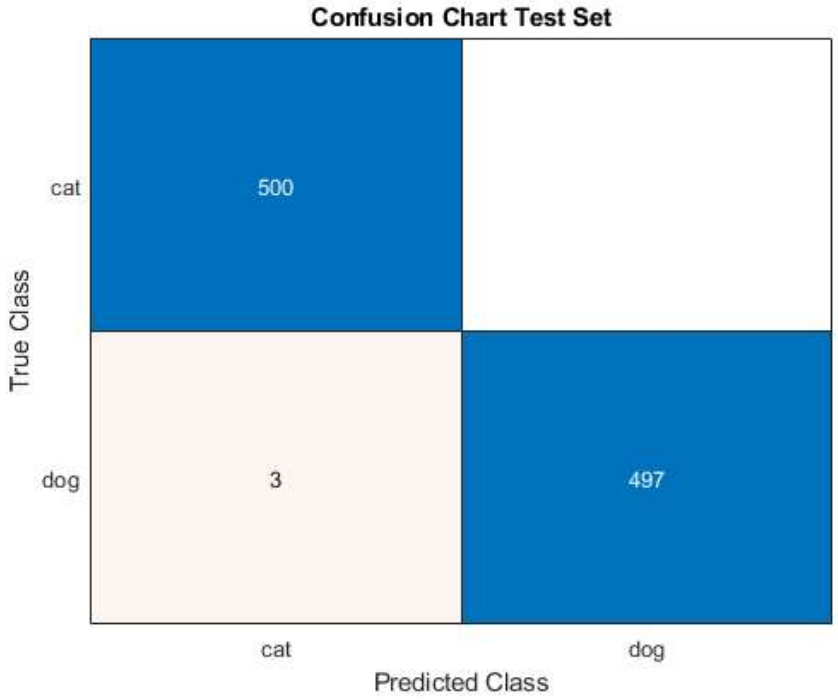
```
% validation accuracy for test set
accuracyTest = mean(ValPredTest == ValidationActualTest);
fprintf("The validation accuracy for Test set is: %.2f %%\n", accuracyTest * 100);
```

```
The validation accuracy for Test set is: 99.70 %
```

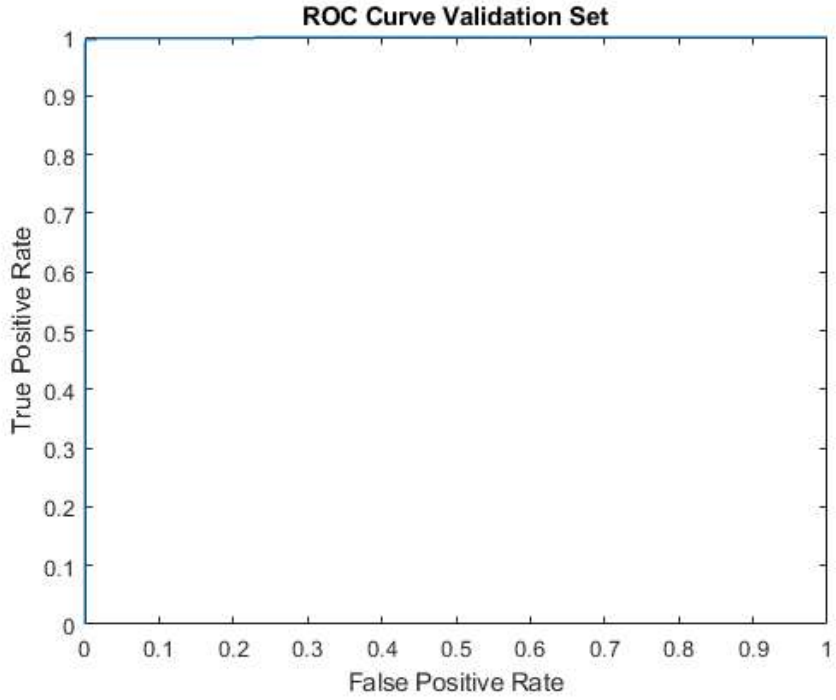
```
% separate scores for each category
scoresCat = scores(:,1);
scoresDog = scores(:,2);
scoresCatTest = scoresTest(:,1);
scoresDogTest = scoresTest(:,2);

% confusion chart validation set
confusionchart(ValidationActual, ValPred)
title('Confusion Chart Validation Set');
```



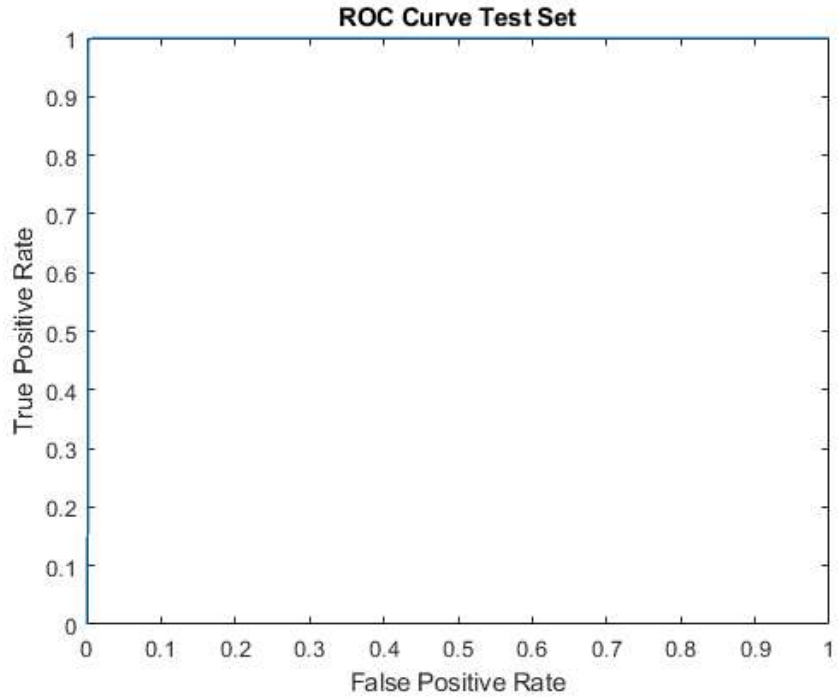

```
% confusion chart test set
confusionchart(ValidationActualTest,ValPredTest);
title('Confusion Chart Test Set');
```



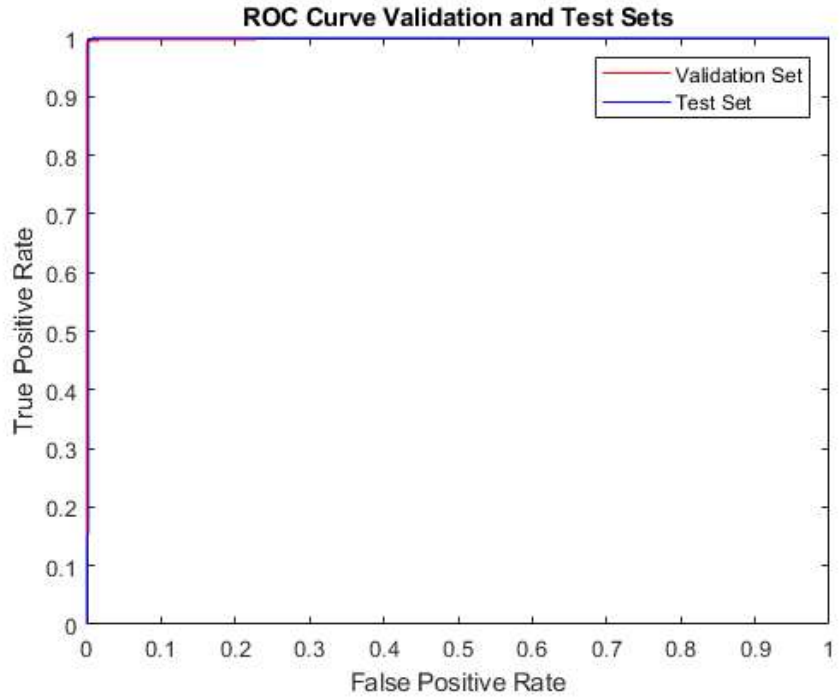

```
%roc curve provided by professor in validation set
[FPRValidation,TPRValidation] = perfcurve(double(ValidationActual),scores(:,1),1);
plot(FPRValidation,TPRValidation);
title('ROC Curve Validation Set');
xlabel('False Positive Rate');
ylabel('True Positive Rate');
```



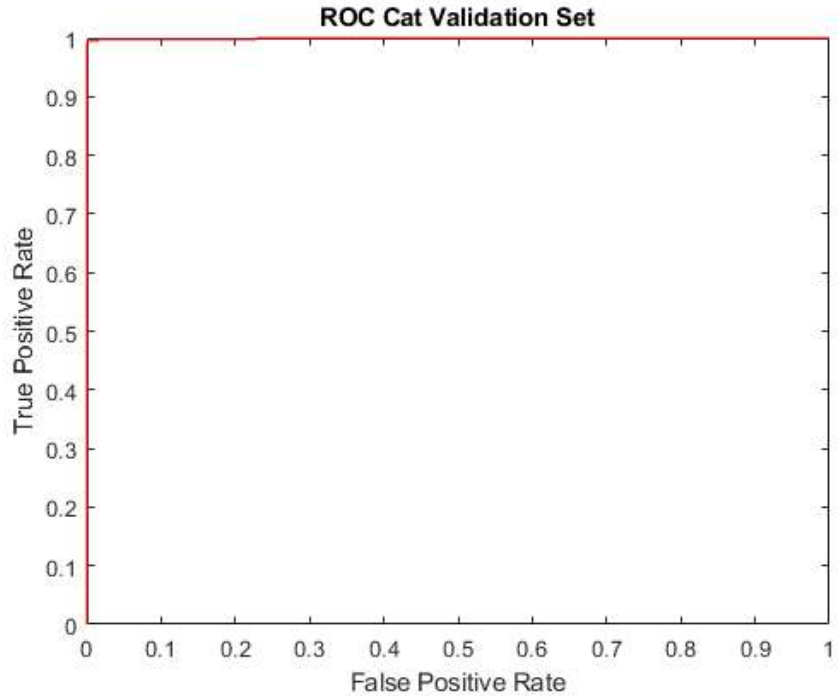

```
%roc curve provided by professor in test set
[FPRTest,TPRTest] = perfcurve(double(ValidationActualTest),scoresTest(:,1),1);
plot(FPRTest, TPRTest);
title('ROC Curve Test Set');
xlabel('False Positive Rate');
ylabel('True Positive Rate');
```



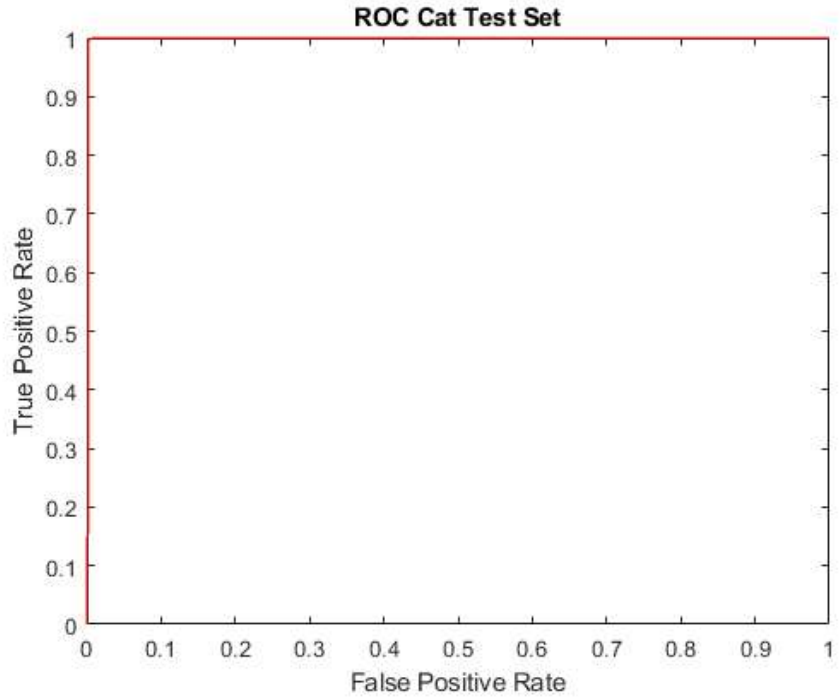

```
%roc curve provided by professor in validations and test sets
plot(FPRValidation,TPRValidation,'r-',FPRTest,TPRTest,'b-')
title('ROC Curve Validation and Test Sets');
xlabel('False Positive Rate');
ylabel('True Positive Rate');
legend('Validation Set','Test Set');
```



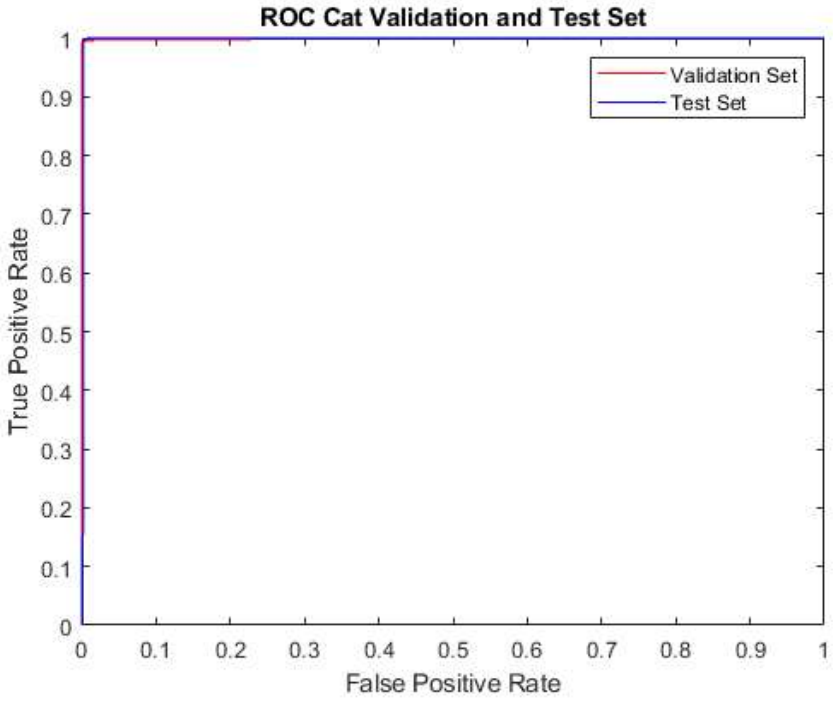

```
% roc graph for cat from Validation Set
[FPRCat,TPRCat] = perfcurve(ValidationActual,scoresCat,'cat');
plot(FPRCat,TPRCat, 'r-');
title('ROC Cat Validation Set');
xlabel('False Positive Rate');
ylabel('True Positive Rate');
```



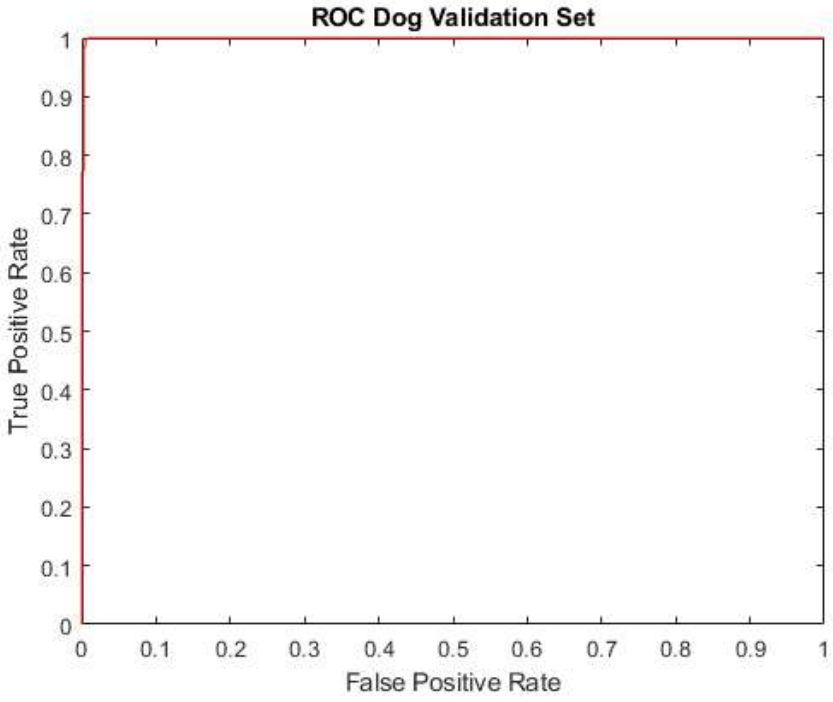

```
% roc graph for cat from Test Set
[FPRCatTest,TPRCatTest] = perfcurve(ValidationActualTest,scoresCatTest,'cat');
plot(FPRCatTest,TPRCatTest, 'r-');
title('ROC Cat Test Set');
xlabel('False Positive Rate');
ylabel('True Positive Rate');
```



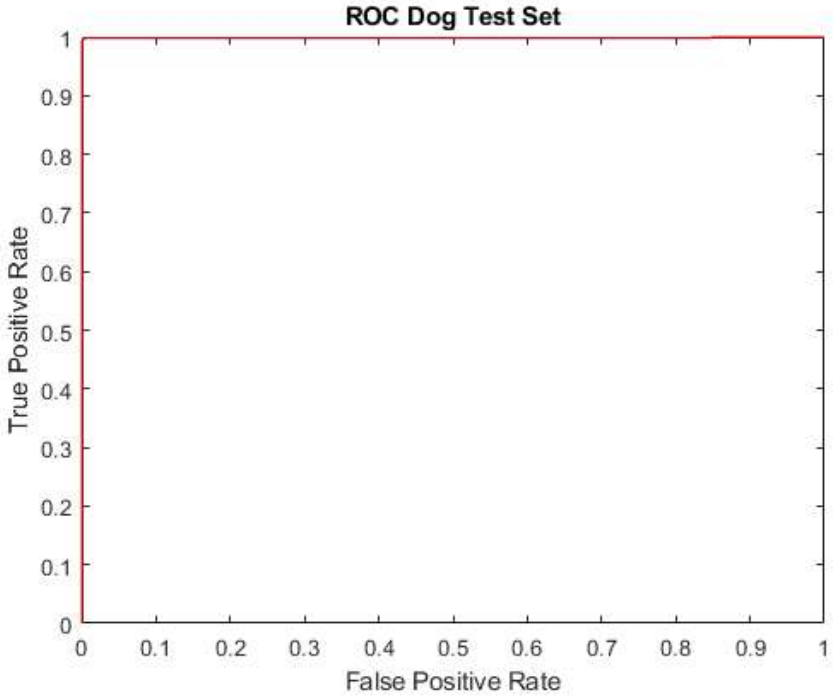

```
% roc graph for cat from Validation and test Set
plot(FPRCat, TPRCat, 'r-', FPRCatTest, TPRCatTest, 'b-');
title('ROC Cat Validation and Test Set');
xlabel('False Positive Rate');
ylabel('True Positive Rate');
legend('Validation Set', 'Test Set');
```



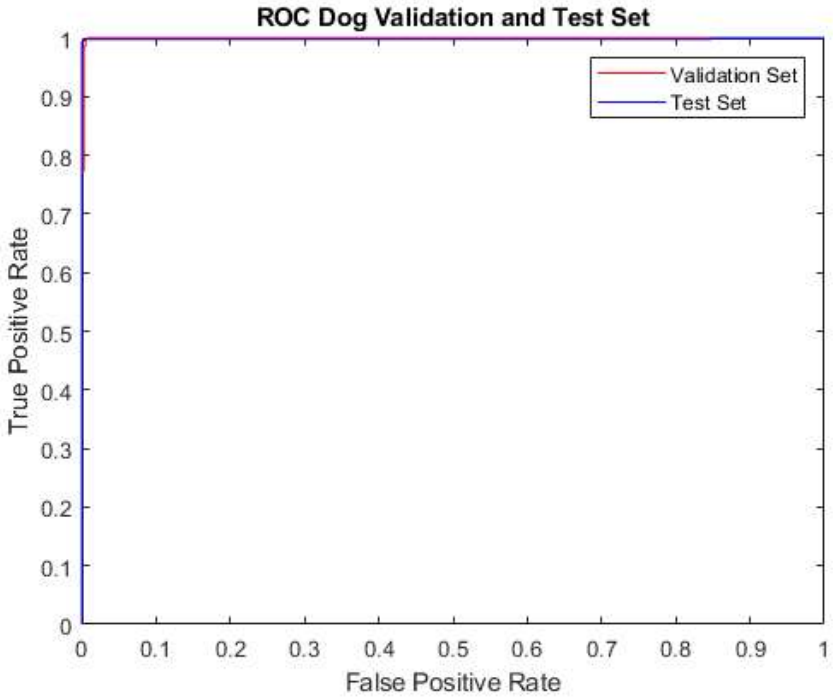

```
% roc graph for dog from Validation Set
[FPRDog,TPRDog] = perfcurve(ValidationActual,scoresDog,'dog');
plot(FPRDog,TPRDog, 'r-');
title('ROC Dog Validation Set');
xlabel('False Positive Rate');
ylabel('True Positive Rate');
```




```
% roc graph for dog from test set
[FPRDogTest,TPRDogTest] = perfcurve(ValidationActualTest,scoresDogTest,'dog');
plot(FPRDogTest,TPRDogTest, 'r-');
title('ROC Dog Test Set');
xlabel('False Positive Rate');
ylabel('True Positive Rate');
```




```
% roc graph for dog from Validation and test Set
plot(FPRDog, TPRDog, 'r-', FPRDogTest, TPRDogTest, 'b-');
title('ROC Dog Validation and Test Set');
xlabel('False Positive Rate');
ylabel('True Positive Rate');
legend('Validation Set', 'Test Set');

%testing random image 1
newImage1 = './dog.jpg'; % any dog image should do!
img1 = readAndPreprocessImage(newImage1);
YPred1 = predict(modelTransfer,img1);
[confidence1,idx1] = max(YPred1);
label1 = categories{idx1};
% Display test image and assigned label
figure
imshow(img1)
```




```
title(string(label1) + ", " + num2str(100*confidence1) + "%");
```


```
%testing random image 2
newImage2 = './cat.jpg'; % any cat image should do!
img2 = readAndPreprocessImage(newImage2);
YPred2 = predict(modelTransfer,img2);
[confidence2,idx2] = max(YPred2);
label2 = categories{idx2};
% Display test image and assigned label
figure
imshow(img2)
title(string(label2) + ", " + num2str(100*confidence2) + "%");
```

cat, 99.9968%

```
%testing random image 3
newImage3 = './doge.jpg';
img3 = readAndPreprocessImage(newImage3);
YPred3 = predict(modelTransfer,img3);
[confidence3,idx3] = max(YPred3);
label3 = categories{idx3};
% Display test image and assigned label
figure
imshow(img3)
title(string(label3) + ", " + num2str(100*confidence3) + "%");
```

dog, 99.9476%