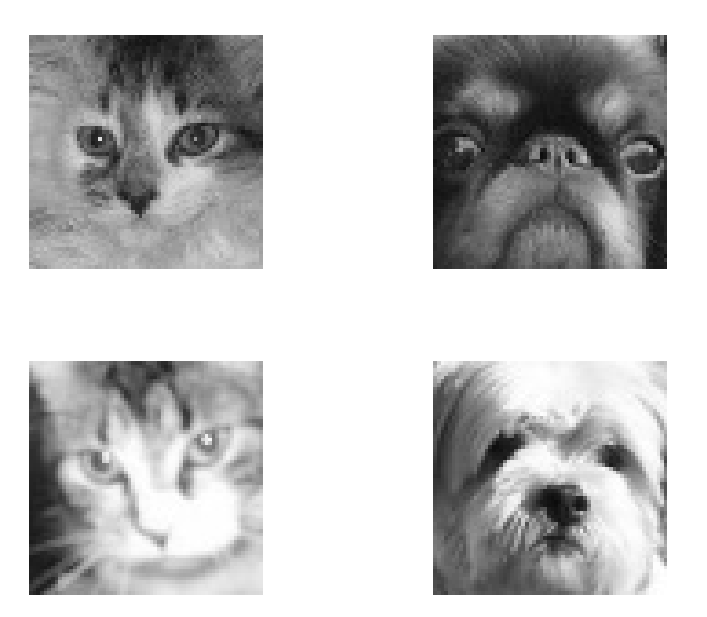
# Homework-3: Image Classification using HOG feature extraction in Machine Learning MATLAB

**Problem: Classify Cats and Dog images using the classification app in MATLAB**

**Image samples:**

****

**Dataset:**

**Kaggle data set:** Cats and Dogs faces 1000 images/class

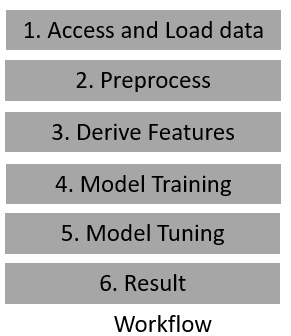
(<https://github.com/andechen/EK381-Prob-Stats-Data-Science/tree/main/catsfolder>)

**Initial Size**- Training 100 image/ class, Test 10 image/class

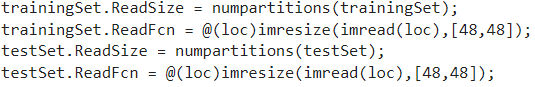
**Class/ Data Labeling:** 2, Cats and Dogs

**Data Characteristics:** Grayscale, Front facing image, 64\*64 pixels with slightly blurred.

Data variable: trainingFeatures\_HOG, trainingLabels\_HOG, trainingFeatures\_LBP, trainingLabels\_LBP

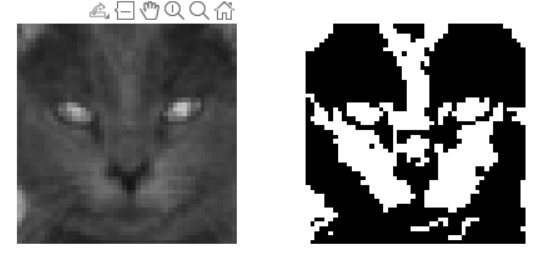
**Workflow:**

1. **Data resize**: Use ***imresize()*** to resize each image to 48\*48 pixels

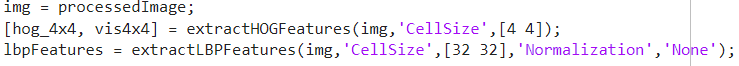
****

1. **Normalization: *Use im2gray()*** to normalize the image in greyscale and ***rescale(img, 0, 1)*** helps to get between 0 to 1.

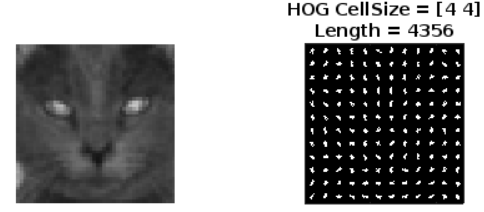
****

****

1. **Hog Feature:** Use ***extractHOGFeatures(img,'CellSize',cellSize)*** to extract hoq feature of Cell size.

****

**HOG**

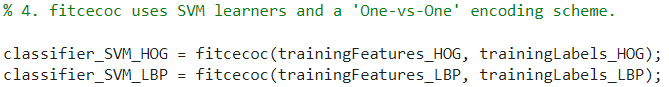
****

**LBP**

**Number of Hog features trained: 4356**

**Number of LBP features trained: 2048**

1. **Training:** Use ***fitcecoc ()*** to train the data set in the SVM classifier.



1. **Evaluate:** Use ***predict(classifier\_SVM\_HOG, testFeatures\_HOG)*** to evaluate the test set.



1. **Result:** Use ***confusionmat(testLabels\_HOG, predictedLabels\_HOG)*** to get the confusin matrix.

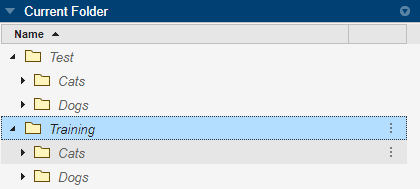


**Data Set Architecture:**

Labeled data: Cats and Dogs categorized in the Test and Train set

Train set Size: 100 images per class

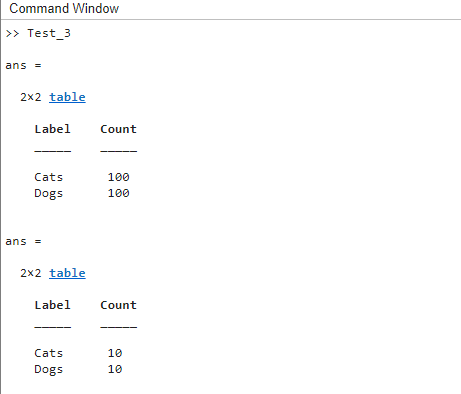
Test set size: 10 image per class



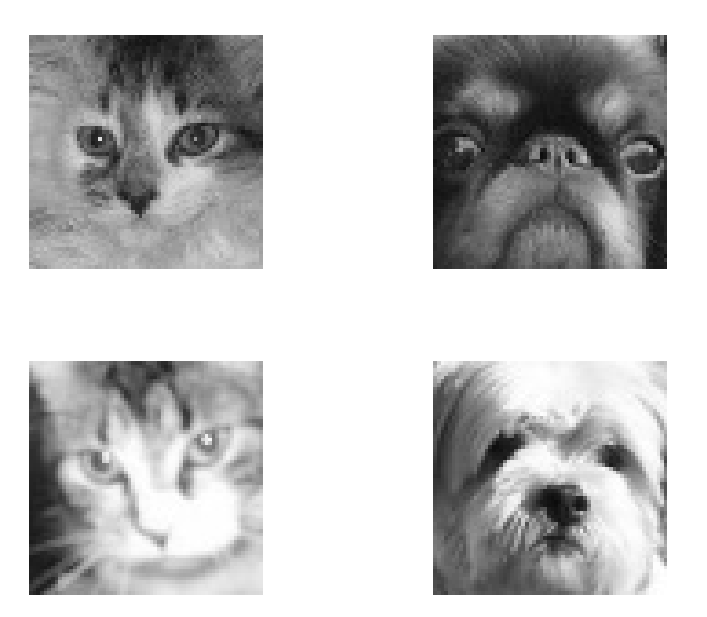
1. **MATLAB Source Code (Manual Training):**
2. % % https://www.mathworks.com/help/vision/ref/extracthogfeatures.html
3. % % https://www.mathworks.com/help/vision/ug/digit-classification-using-hog-features.html
4. % https://www.mathworks.com/help/vision/ug/digit-classification-using-hog-features.html
5. %https://github.com/andechen/EK381-Prob-Stats-Data-Science
6. %https://github.com/johnsonj561/Cat-Dog-Classification
7. %https://www.dropbox.com/s/zdeb04s5785ep5h/cats\_dogs\_starter.zip?dl=0&file\_subpath=%2Fcats\_dogs\_starter
8. %1. read data
9. Training\_folder = './Training';
10. Test\_folder = './Test';
11. trainingSet = imageDatastore(Training\_folder,'IncludeSubfolders',true,'LabelSource','foldernames');
12. testSet = imageDatastore(Test\_folder,'IncludeSubfolders',true,'LabelSource','foldernames');
13. % % 2. Resize dataset
14. trainingSet.ReadSize = numpartitions(trainingSet);
15. trainingSet.ReadFcn = @(loc)imresize(imread(loc),[48,48]);
16. testSet.ReadSize = numpartitions(testSet);
17. testSet.ReadFcn = @(loc)imresize(imread(loc),[48,48]);
18. countEachLabel(trainingSet)
19. countEachLabel(testSet)
20. figure;
21. fprintf('Training Set')
22. subplot(2,2,1);
23. imshow(trainingSet.Files{100});
24. subplot(2,2,2);
25. imshow(trainingSet.Files{200});
26. fprintf('Testing Set')
27. subplot(2,2,3);
28. imshow(testSet.Files{10});
29. subplot(2,2,4);
30. imshow(testSet.Files{20});
31. % Show pre-processing results- removing the noise and scaling manually
32. exTrainImage = readimage(trainingSet,1);
33. processedImage = imbinarize(im2gray(exTrainImage));
34. figure;
35. subplot(1,2,1)
36. imshow(exTrainImage)
37. subplot(1,2,2)
38. imshow(processedImage)
39. % % 3. Extract HOG features and HOG visualization 4\*4
40. img = processedImage;
41. [hog\_4x4, vis4x4] = extractHOGFeatures(img,'CellSize',[4 4]);
42. %lbpFeatures = extractLBPFeatures(img,'CellSize',[32 32],'Normalization','None');
43. figure;
44. subplot(2,2,1);
45. title({'Real Image'})
46. imshow(img);
47. subplot(2,2,2);
48. plot(vis4x4);
49. title({'HOG CellSize = [4 4]'; ['Length = ' num2str(length(hog\_4x4))]});
50. % % 3.1 Extract HOG features for both full train dataset
51. cellSize = [4 4];
52. FeatureSize = length(hog\_4x4);
53. [trainingFeatures\_HOG, trainingLabels\_HOG] = helperExtractHOGFeaturesFromImageSet(trainingSet, FeatureSize, cellSize);
54. size\_HOG\_train = size(trainingFeatures\_HOG)
55. %[trainingFeatures\_LBP, trainingLabels\_LBP] = helperExtractLBPFeaturesFromImageSet(trainingSet, cellSize);
56. %size\_LBP\_train = size(trainingFeatures\_LBP)
57. % 4. fitcecoc uses SVM learners and a 'One-vs-One' encoding scheme.
58. classifier\_SVM\_HOG = fitcecoc(trainingFeatures\_HOG, trainingLabels\_HOG);
59. %classifier\_SVM\_LBP = fitcecoc(trainingFeatures\_LBP, trainingLabels\_LBP);
60. % % 5. Evaluate Classifier by Testing set
61. [testFeatures\_HOG, testLabels\_HOG] = helperExtractHOGFeaturesFromImageSet(testSet, FeatureSize, cellSize);
62. %[testFeatures\_LBP, testLabels\_LBP] = helperExtractLBPFeaturesFromImageSet(testSet, cellSize);
63. predictedLabels\_HOG = predict(classifier\_SVM\_HOG, testFeatures\_HOG);
64. %predictedLabels\_LBP = predict(classifier\_SVM\_LBP, testFeatures\_LBP);
65. % 6. Tabulate the results using a confusion matrix.
66. confMat\_HOG = confusionmat(testLabels\_HOG, predictedLabels\_HOG);
67. helperDisplayConfusionMatrix(confMat\_HOG)
68. %confMat\_LBP = confusionmat(testLabels\_LBP, predictedLabels\_LBP);
69. %helperDisplayConfusionMatrix(confMat\_LBP)
70. % % 3.1 Extract HOG features from an imageDatastore.
71. function [features, setLabels] = helperExtractHOGFeaturesFromImageSet(imds, hogFeatureSize, cellSize)
72. setLabels = imds.Labels;
73. numImages = numel(imds.Files);
74. features = zeros(numImages,hogFeatureSize,'single');
75. % Process each image and extract features
76. for j = 1:numImages
77. img = readimage(imds,j);
78. % Normalize the image from 0 to 1
79. % Apply pre-processing steps
80. img = im2gray(img);
81. img = imbinarize(img);
82. img = rescale(img, 0, 1);
84. features(j, :) = extractHOGFeatures(img,'CellSize',cellSize);
85. end
86. end
87. % % 3.1 Extract LBP features from an imageDatastore.
88. function [features, setLabels] = helperExtractLBPFeaturesFromImageSet(imds, cellSize)
89. setLabels = imds.Labels;
90. numImages = numel(imds.Files);
91. features = zeros(numImages,'single');
92. % Process each image and extract features
93. for j = 1:numImages
94. img = readimage(imds,j);
95. % Normalize the image from 0 to 1
96. img = im2gray(img);
98. % Apply pre-processing steps
99. img = imbinarize(img);
100. %lbpFeatures = extractLBPFeatures(I,'CellSize',[32 32],'Normalization','None');
101. features(j, :) = extractLBPFeatures(img,'CellSize',cellSize);
102. end
103. end
104. % 6. Display the confusion matrix in a formatted table.
105. function helperDisplayConfusionMatrix(confMat)
106. % Convert confusion matrix into percentage form
107. confMat = bsxfun(@rdivide,confMat,sum(confMat,2));
108. digits = '0':'1';
109. colHeadings = arrayfun(@(x)sprintf('%d',x),0:1,'UniformOutput',false);
110. format = repmat('%-9s',1,2);
111. header = sprintf(format,'Pet |',colHeadings{:});
112. fprintf('\n%s\n%s\n',header,repmat('-',size(header)));
113. for idx = 1:numel(digits)
114. fprintf('%-9s', [digits(idx) ' |']);
115. fprintf('%-9.2f', confMat(idx,:));
116. fprintf('\n')
117. end
118. end

**Output:**

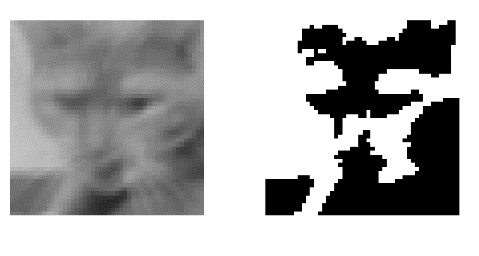
Train and Test data sets:

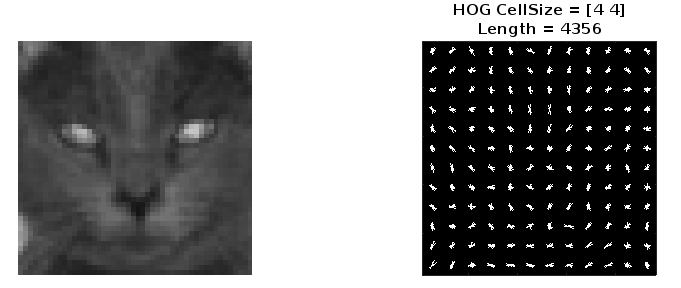


Visualize data from Train and Test set

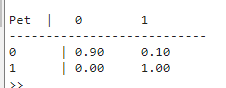


Noise removal:

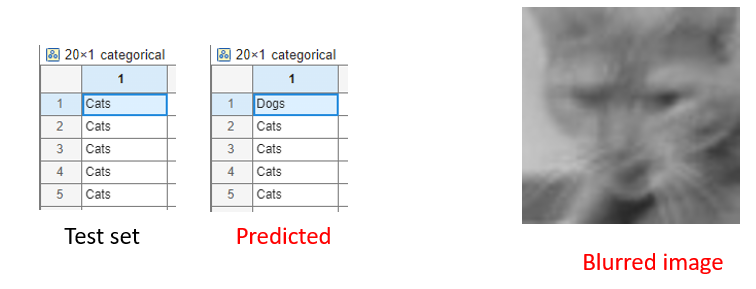
  
HOG feature:

****

Confusing Matrix using SVM manual code:

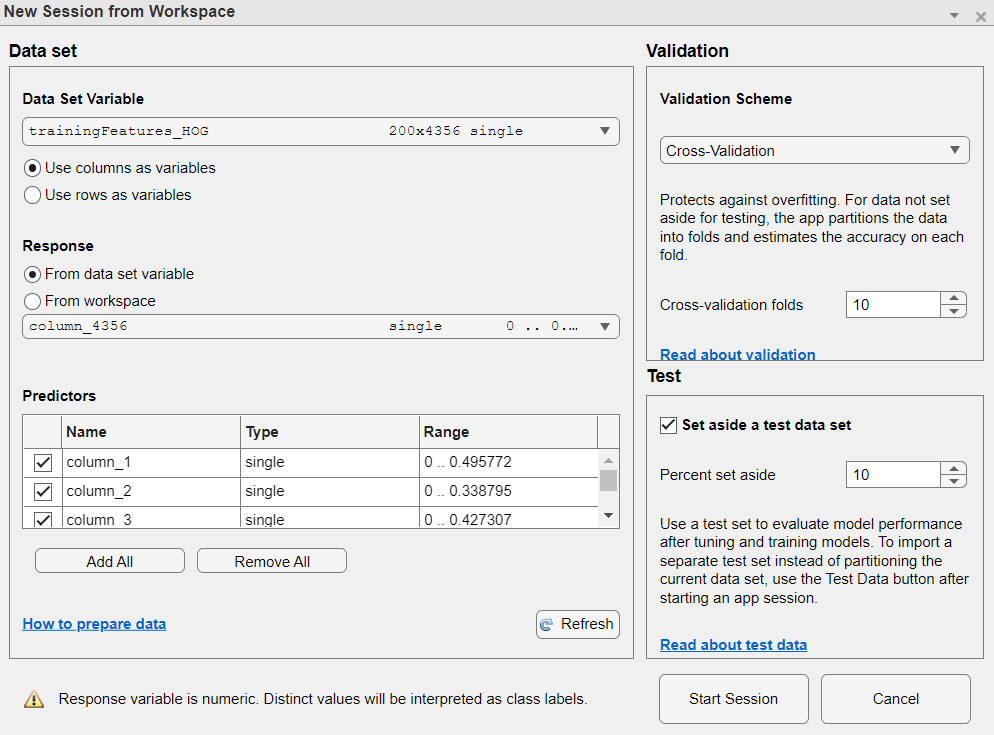


Accuracy: 90%

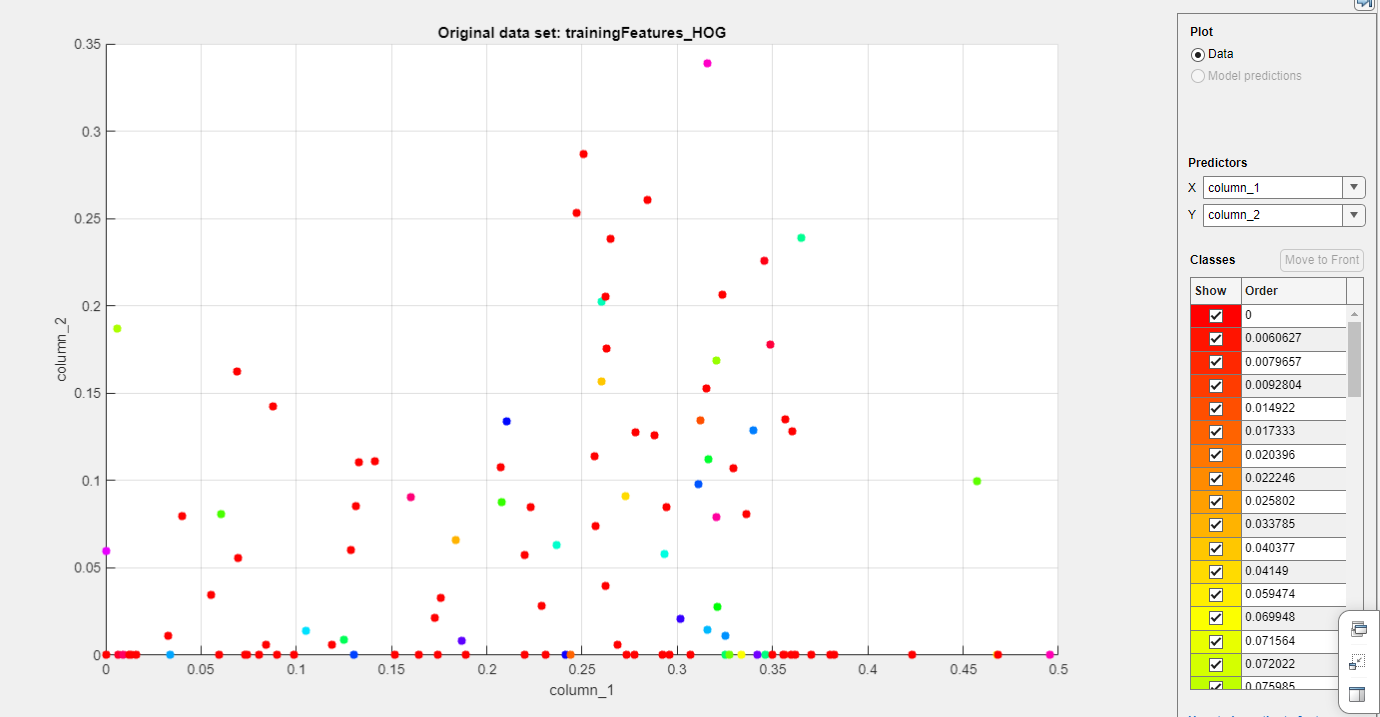


**2. With using Classification Learner App:**

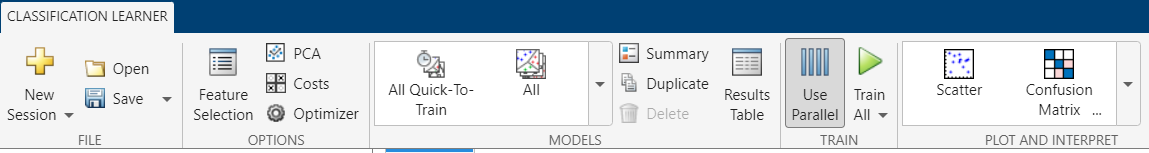
a) Load the Classification Learner App and pick **trainingFeature\_HOG, trainingFeature\_LBP** from above code as a variable and start the session. (with validation fold = 5 and set aside test data: 10 )

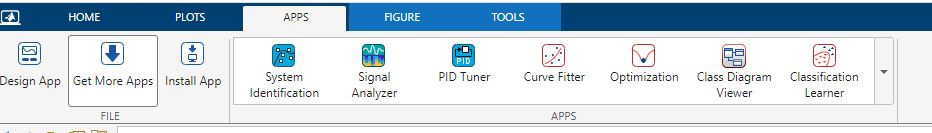
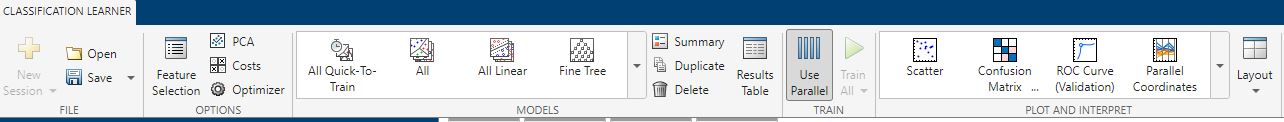


b) Scatter plot:

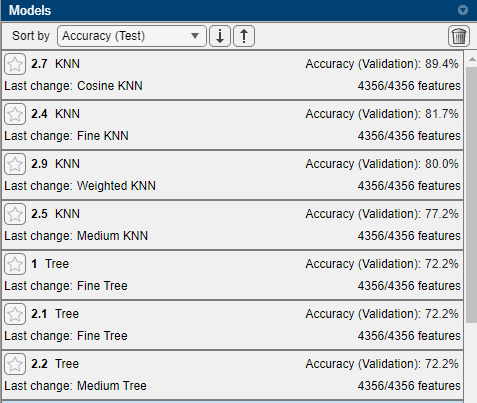


c) select **All Quick-To-Train** from classification learner appand click **Train All** button.



d) Learning Result (Confusion Matrix Accuracy): 89% maximum



Confusion matrix

