Department of Electrical and Computer Engineering
Rutgers University – College of Engineering
New Brunswick, NJ

Course: 14:332:472
Robotic & Computer Vision Final Project
12/14/2016
Eun-Sol Kim
133000680

Introduction

The object recognition in computer vision is one of the process to identify the particular objects in the images or the video clips. It is well applicate in real life such as the google glass. The object recognition is done by training the system with the particular object's features with various images of certain object. The system needs to be able to match those feature vectors with input image to detect which objects are presented in the image. The object recognition can be done by varies algorithm; however, in this project, we were to explore two particular algorithms: Bag of Words and Convolutional Neural Network.

The Bag of Words (BOW) is an object classification method that identify the testing image with the stored-train data in the system. The method is very simple; the certain number of data can be trained into the system so that when the testing data is inputted to the system, the system can classify it by generalize the input. There can be ratio between number of training and testing data such that 3:7 or 5:5 of testing and training data, respectively. The more the training data, the better the precision of output. For this project, I chose to do 5:5 split just to see how effective the classification works. The convolutional neural network is one of the type of deep learning. It is modelled based on the human brain structured. The system is trained to be able to differentiate between all the images its given and figure out the unique features by looking for low-level features then building up to more abstract concepts through the layers.

Method: Bag of Words

The object recognition can be done with such classification called Bag-of-Words (BOW). The BOW can be divided into four major steps:

<u>First step: Scale-invariant feature transform (SIFT)</u>

The SIFT is an algorithm that detects N interest points for each of images and described them in N number of 128x1 vectors called descriptors. The number of interest points for each of images varies depends on 'good points' – points with good characteristic around the local point of pixel such as corners and edges. The SIFT detects local interest points by Gaussian pyramid where it finding extrema in blurred scale around the point's neighbors in Difference of Gaussian images. Since there are different number of descriptors, there might be different weights for each interest points in each images; therefore, it is better to normalize the weights for each of interest points to

be same (can be done with finding same number of interest points or actually normalizing). For this project, I chose to find the same number of interest points, 100, for each points to normalize it, where I left with 128x100 descriptors for 50 of my data.

Second step: Visual Dictionary

The descriptors for each images can be concatenated to make the visual dictionary. The visual dictionary is the clusters of descriptors that describe the feature points of images. Just like the word dictionary contains the word and the its description, the number of cluster can be thought of as number of words in the dictionary and cluster of descriptors can be thought of as descriptions of that word. The clustering can be done by algorithm called K-means clustering which partitions the descriptors in K clusters by give one centroid (nearest mean) for each descriptors. For my project, I chose 300 and 200 as K; thereby, I result in 128x300 and 128x200, respectively.

Third step: Histogram of training data

Thirdly, having the visual words, the descriptors of each image can be classified as visual words. This can be done by using k-nearest neighbors algorithm, the classification method that finds the nearest distances in points of descriptors (dimension: Kx1 for each images) with visual words (dimension: 128xK) and assign the descriptors with closest visual word. As the result, each descriptor will have assigned closest visual words (dimension: Kx1 for each images). Each image's representation is the histogram of the each assigned descriptors, where sum of frequency is exactly K. For my project I results in 300x1 and 200x1 for each of 50 images.

Fourth step: classify the testing data.

Following the step one and three for the test image, the histogram, image representation, can be created. With that histogram, the test image can be classified by finding the distances between testing and all of trained image histograms and assign the testing histogram with closest distance of trained one; thereby the result will have dimension Kx1 for each image that are numbered between 1 to number of classes.

Method: Convolutional Neural Network

The whole network of CNN contains multiple layers. To get an idea how the CNN multiple layer works, the classic CNN architecture is structured liked this: input -> conv->ReLU -> conv->ReLU->Pool->ReLU->Conv->ReLU->Fully connected.

Layer:

- First Layer: Convolutional layer (conv)
    - The first layer of CNN is the convolutional (conv) layer, where the input of this layer is either the original image or input from the previous layer, both in matrix form. The output, often called activation or feature map, is the result of convolution of input and the filter. The purpose of this layer is to identify the low level features, such as straight edges, simple colors and curves. The filter will contain the numbers, called weights, in an array form. As the filter convolves in the image region, it will detect availability of certain feature by performing a convolution with the region. If the feature exists, the output of the region will be high number; if not, the output will be the low number. The multiple layers of convolutions will eventually combine each filter that will results in higher level features, such as semicircle. As the network goes deeper, the filters will have larger receptive field that will be more responsive to larger region of pixel shape.
- Second Layer: Rectified Linear Unit (ReLU) layer
    - The Rectified Linear Unit (ReLU) layer introduces the non-linearity to a system that have been computed linear operation during convolution layer. This layer makes training a lot faster without making distinguishable difference to the accuracy.
- Third Layer: max pooling layer (pool)
    - The max pooling layer down-samples the input of this layer. It takes the filter and a stride, which controls the filter to convolve around partitioned region of the input; therefore, the filter is applied to the input region based on stride. This layer outputs the maximum number in every partitioned sub-region that the filter convolves around.
- Fourth Layer: fully-connected layer (fc)
    - The fully connected layer takes an input from the preceding layer and outputs the N-dimensional vector where N is the number of classes. Each number in the vector represents the probability of a certain class. This layer looks at the input and detect the feature that correlates the most to a particular class.

Train

All of four layers are designed as it can be train the classes. Back-propagation is the method to training data by computing the derivative without using large memory data. It contains four different steps:

- First and Second Step: forward pass and loss function
  - The forward pass takes a training image and pass it through the whole network. Then the loss function uses the training data that contains an image and the label. In this step, the loss, error, is calculated using this formula:

    $$E_{total} = L = \frac{1}{2}\sum(target - output)^2.$$

- Third Step: Backward pass
  - The loss can be minimized by backward pass which calculates the slope in every direction such that the output of the network is scalar quantity; however, the derivation will not be the scalar quantity. This is when the Jacobian matrix comes in play, which makes possible of derivative of tensors (multidimensional array). The Jacobian matrix writes each derivative term in series in a matrix. By reference the figure 1, f represents the loss function of each tensor layer and x are the weights of particular layers.

$$\frac{d\,\mathrm{vec}\,f}{d(\mathrm{vec}\,\mathbf{x})^\top} = \begin{bmatrix} \frac{\partial y_{111}}{\partial x_{111}} & \frac{\partial y_{111}}{\partial x_{211}} & \cdots & \frac{\partial y_{111}}{\partial x_{H11}} & \frac{\partial y_{111}}{\partial x_{121}} & \cdots & \frac{\partial y_{111}}{\partial x_{HWC}} \\ \frac{\partial y_{211}}{\partial x_{111}} & \frac{\partial y_{211}}{\partial x_{211}} & \cdots & \frac{\partial y_{211}}{\partial x_{H11}} & \frac{\partial y_{211}}{\partial x_{121}} & \cdots & \frac{\partial y_{211}}{\partial x_{HWC}} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ \frac{\partial y_{H'11}}{\partial x_{111}} & \frac{\partial y_{H'11}}{\partial x_{211}} & \cdots & \frac{\partial y_{H'11}}{\partial x_{H11}} & \frac{\partial y_{H'11}}{\partial x_{121}} & \cdots & \frac{\partial y_{H'11}}{\partial x_{HWC}} \\ \frac{\partial y_{121}}{\partial x_{111}} & \frac{\partial y_{121}}{\partial x_{211}} & \cdots & \frac{\partial y_{121}}{\partial x_{H11}} & \frac{\partial y_{121}}{\partial x_{121}} & \cdots & \frac{\partial y_{121}}{\partial x_{HWC}} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ \frac{\partial y_{H'W'C'}}{\partial x_{111}} & \frac{\partial y_{H'W'C'}}{\partial x_{211}} & \cdots & \frac{\partial y_{H'W'C'}}{\partial x_{H11}} & \frac{\partial y_{H'W'C'}}{\partial x_{121}} & \cdots & \frac{\partial y_{H'W'C'}}{\partial x_{HWC}} \end{bmatrix}$$

Figure 1: Equation of Jacobian Matrix

- Fourth Step: Weight Update
  - Finally, weights are updated such that they change in direction of the gradient. Repetition of back-propagation will train the system with all of training images.

Sometimes, there are overfitting situation where the recognition rate for training data is 0.99 but for testing data is 0.60. This over-fitting means that the algorithm has not been learned; therefore, this can be avoiding by dropout regularization which drops a random set of activation map in that layer by setting them to 0 during the forward pass.

Method: Confusion Matrix:

The confusion matrix is a table that can be used to describe the performance of a classification on the set of data. The confusion matrix is used to evaluate the result of the object recognition. The confusion matrix computes the accuracy, precision, sensitivity, specificity and F-Score based on the variables True Positive, True Negative, False Positive and False negative. To

compute this, I downloaded the helpful code that is called confusionmatStats from the MATLAB resources. According to comment of that source code, these are following definitions.

Definition in terms of class:

Considering that we are evaluating for one particular class A.

- True Positive (TP): input that predicted to be A and labelled as A.
- True Negative (TN): input that predicted to be other class and labelled as other class.
- False Positive (FP): input that predicted to be A but labelled as other class.
- False Negative (FN): input that predicted to be other class but labelled as A.

Evaluation of overall confusion matrix, these terms are used:

- Accuracy: overall, how often is the classifier correct for particular class
- Precision: out of number of image(s) labelled as a particular class, how many were correctly labelled
- Sensitivity: out of number of image(s) predicted to be a particular class, how many were correctly labelled
- Specificity: out of number of image(s) predicted to be different class, how many were actually labelled as different class?
- F-Score: weighted average of true positive rate and precision

Additional to variables and terms, confusion matrix was used to help to visualize the performance. Each row of confusion matrix represents the label of each class, in my project it will be 1 to 10. The columns of confusion matrix represent the predicted label of each class. The values contained in the matrix represents the frequency of the output for that particular class for each of input class. For example, if the first row is [1 3 5 0 0 0 1 0 0 0], for input that was labelled as class 1, the system detected only one image as class 1, three images as class 2 and so on. Therefore, diagonal of the confusion matrix will represent the True Positive. Typically, confusion matrix dimension will be number of class by number of class.

Results

The 10 classes I used to train and test image are as follows for Bag of Words

- Class 1: Alligator Lizard
- Class 2: Brain Coral
- Class 3: Doberman Pinscher
- Class 4: Flamingo
- Class 5: Golden Retriever
- Class 6: Kimono
- Class 7: Kite
- Class 8: Old English Sheepdog, bobtail
- Class 9: Peacock
- Class 10: Siberian Husky

Bag of Words (BOW): K = 300

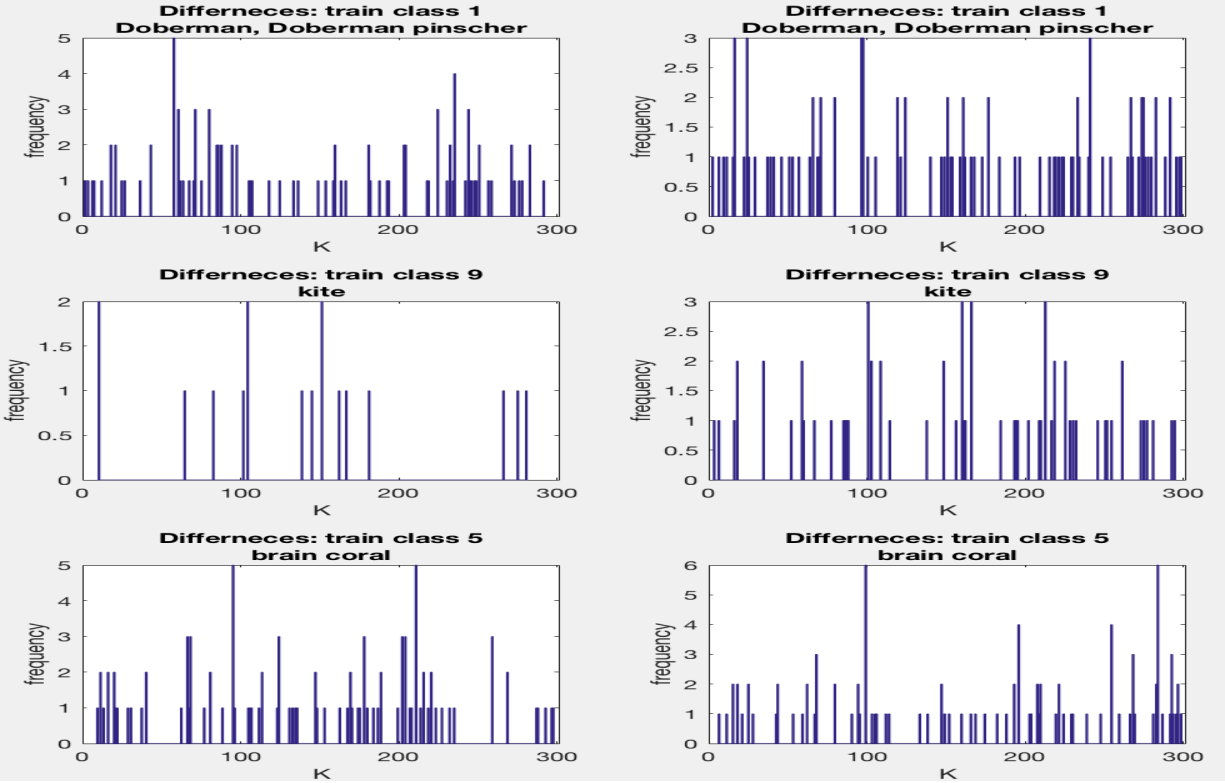

Figure 2: K=300: Similarity and Difference of histogram of classes

Referring to Figure 2, each column histogram corresponds to same class and row corresponds to the different class. As you can see, there are quiet number of pick points in Class 1 but less detection on class 9. This is due to large number of feature points that did not detect anything close to the centroids. For Brain Coral, the maximum points and modes are very similar to each other.

Confusion Matrix:

0	0	0	0	0	4	2	0	4	0
0	0	0	0	0	9	1	0	0	0
1	0	0	0	0	4	1	0	4	0
0	0	0	0	0	5	0	0	5	0
0	0	0	0	0	6	0	0	4	0
0	0	0	0	0	3	0	0	7	0
0	0	0	0	0	5	0	0	5	0
0	0	0	0	0	3	1	0	6	0
0	0	0	0	0	1	0	0	9	0
0	0	0	0	0	5	0	0	5	0

Accuracy:	Sensitivity:	Specificity:	Precision:	Fscore:
0.8900	0	0.9889	0	0
0.9000	0	1.0000	NaN	0
0.9000	0	1.0000	NaN	0
0.9000	0	1.0000	NaN	0
0.9000	0	1.0000	NaN	0
0.5100	0.3000	0.5333	0.0667	0.1091
0.8500	0	0.9444	0	0
0.9000	0	1.0000	NaN	0
0.5900	0.9000	0.5556	0.1837	0.3051
0.9000	0	1.0000	NaN	0

- The system detected class 9 (Peacock) the most and then class 5 (Golden Retriever). The only classes that contains non-zero value for sensitivity, precision and F-Score are class 9 and 5. The reason for this is accuracy calculate average accuracy and specificity calculates the percentage of images that are not predicted as other classes and labelled as other classes. The images supposed to predicted as other class were labelled as the class 9; therefore, reduces the TN but increases FP.
- NaN in Precision represents that there were any of TP nor the TN, in other words, there were no images that were labelled as that specific classes. The value 1 in specificity correlates to Precision to be NaN.
- Zero in Sensitivity, Precision and F-Score represents that there were no correctly labelled for that particular class.
- Overall, the performance of the BOW was poor because it was able to predict only two classes correctly.

Bag of Words (BOW): K = 200

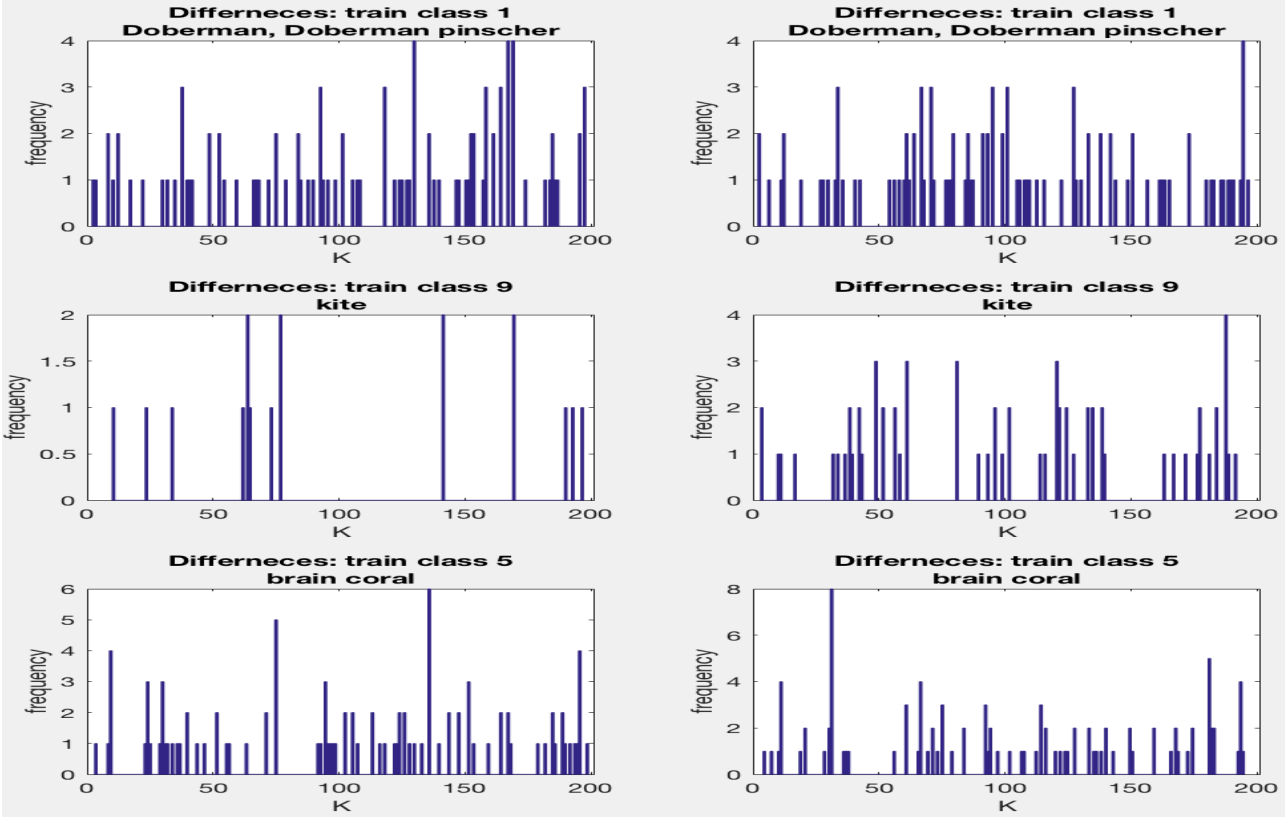

Figure 3: K=200: Similarity and Difference of histogram of classes

Comparing to the histogram for K=300, Figure 3 histograms varies significantly within same classes. Their peak values are larger than for the histogram K=300, but does not have much peaks. This can reveal that there were a lot of generalizing such that important features that represents the images can be other label. Because there is less length of visual vocabulary, it will need larger generalization of each interest point to match to the visual words.

Confusion Matrix:

1	0	1	0	0	0	3	0	5	0
0	3	2	0	0	0	0	0	5	0
0	1	1	0	0	1	2	0	5	0
0	0	0	0	0	0	1	0	9	0
0	0	0	0	0	2	3	0	5	0
0	0	0	0	0	2	1	0	7	0
0	0	0	0	0	2	1	0	7	0
0	0	0	0	0	2	1	0	7	0
0	0	0	0	0	1	0	0	9	0
0	0	0	0	0	1	1	0	8	0

Accuracy:	Sensitivity:	Specificity:	Precision:	F-Score:
0.9100	0.1000	1.0000	1.0000	0.1818
0.9200	0.3000	0.9889	0.7500	0.4286
0.8800	0.1000	0.9667	0.2500	0.1429
0.9000	0	1.0000	NaN	0
0.9000	0	1.0000	NaN	0
0.8300	0.2000	0.9000	0.1818	0.1905
0.7900	0.1000	0.8667	0.0769	0.0870
0.9000	0	1.0000	NaN	0
0.4100	0.9000	0.3556	0.1343	0.2338
0.9000	0	1.0000	NaN	0

- The class 9 was definitely precisely detected but also, detected as the most. Unlike the other confusion matrix (K=300), this confusion matrix shows multiple well-detected classes besides the class 9: class 6 and 7.
- The Precision and Specificity of class 1 are both 1. Also, the Precision for class 2 is relatively higher than any of other classes. The reason for this is there were none/relatively small images that were predicted as the other classes but labelled as class 1 or 2. This represents overfitting: class 1 or 2 are trained with their own training data but did not generalize it. The precision, not only the proves the accurately labelled, but also proves that how poorly the system learned that class.
- Overall, when K = 200, the object recognition performance is worse than when K = 300. It over-fits the data and gives output with less certainty, F-Score.
-

Convolutional Neural Network (CNN): imagenet-vgg-f.mat

The 10 classes I used to train and test image are as follows for CNN:

- Class 22: Kite
- Class 85: Peacock
- Class 131: Flamingo
- Class 230: Old English Sheepdog
- Class 45: Alligator Lizard
- Class 110: Brain Coral
- Class 208: Golden Retriever
- Class 237: Doberman Pinscher
- Class 251: Siberian Husky
- Class 615: Kimono
- Class 1001: others

Figure 4: CNN Outcome picture

As referring to figure 4, there are high probability of that the label is correct label than Bag Of Words.

Confusion Matrix:

0	0	0	0	0	0	0	0	0	0	20
0	15	0	0	0	0	0	0	0	0	5
0	0	18	0	0	0	0	0	0	0	2
0	0	0	19	0	0	0	0	0	0	1
0	0	0	0	18	0	0	0	0	0	2
0	0	0	0	0	12	0	0	0	0	8
0	0	0	0	0	0	19	0	0	0	1
0	0	0	0	0	0	0	17	0	0	3
0	0	0	0	0	0	0	0	4	0	16
0	0	0	0	0	0	0	0	0	16	4
0	0	0	0	0	0	0	0	0	0	0

Accuracy:	Sensitivity:	Specificity:	Precision:	F-Score:
0.9000	0	1.0000	NaN	0
0.9750	0.7500	1.0000	1	0.8571
0.9900	0.9000	1.0000	1	0.9474
0.9950	0.9500	1.0000	1	0.9744
0.9900	0.9000	1.0000	1	0.9474
0.9600	0.6000	1.0000	1	0.7500
0.9950	0.9500	1.0000	1	0.9744
0.9850	0.8500	1.0000	1	0.9189
0.9200	0.2000	1.0000	1	0.3333
0.9800	0.8000	1.0000	1	0.8889
0.6900	NaN	0.6900	0	0

For the confusion matrix, even though there are 10 classes total, it is dimensioned as 11 by 11. The reason for this is since the system uses the only 10 classes out of 1000 classes, it were to detect other classes as one of the label; therefore, I created another class called 1001 that represents the 'other'. Therefore, dimension is one dimension larger than what it was expected.

- Note that sensitivity for class 22 is 0. The reason for any of images were labelled as class 22. This implies that the system has not learned the features of Kite, which is over-fitting. If one of the image that were used to train the Kite were to inputted the system, the system might have detected that image as the kite.
- Overall, the confusion matrix implies that the system performs well because since each class contains twenty images, most of inputs were classified as it should be.

Convolutional Neural Network (CNN): imagenet-matconvnet-vgg-verydeep-16.mat

Confusion Matrix:

0	0	0	0	0	0	0	0	0	0	20
0	17	0	0	0	0	0	0	0	0	3
0	0	19	0	0	0	0	0	0	0	1
0	0	0	20	0	0	0	0	0	0	0
0	0	0	0	16	0	0	0	0	0	4
0	0	0	0	0	17	0	0	0	0	3
0	0	0	0	0	0	18	0	0	0	2
0	0	0	0	0	0	0	18	0	0	2
0	0	0	0	0	0	0	0	6	0	14
0	0	0	0	0	0	0	0	0	19	1
0	0	0	0	0	0	0	0	0	0	0

Accuracy:	Sensitivity:	Specificity:	Precision:	F-Score:

0.9000	0	1.0000	NaN	0
0.9850	0.8500	1.0000	1	0.9189
0.9950	0.9500	1.0000	1	0.9744
1.0000	1.0000	1.0000	1	1.0000
0.9800	0.8000	1.0000	1	0.8889
0.9850	0.8500	1.0000	1	0.9189
0.9900	0.9000	1.0000	1	0.9474
0.9900	0.9000	1.0000	1	0.9474
0.9300	0.3000	1.0000	1	0.4615
0.9950	0.9500	1.0000	1	0.9744
0.7500	NaN	0.7500	0	0

- Comparing the confusion matrix with previous confusion matrix, the system performed better with deep layer of CNN than previous one. Even though there are same output for precision, you can see that the F-Score is much better than the previous one.
- As you reference the figure 5, you will see how deep does the second CNN compared to the first one.

net1= imagenet-vgg-f.mat.
└ 21 layers

layer	name	type	
1	conv1	conv	1st conv layer
2	relu1	relu	
3	norm1	lrn	
4	pool1	pool	
5	conv2	conv	2nd conv layer
6	relu2	relu	
7	norm2	lrn	
8	pool2	pool	
9	conv3	conv	3rd conv layer
10	relu3	relu	
11	conv4	conv	4th conv layer.
12	relu4	relu	
13	conv5	conv	5th conv layer.
14	relu5	relu	
15	pool5	pool	
16	fc6	conv	1st fully-connect layer
17	relu6	relu	
18	fc7	conv	2nd fully connected layer
19	relu7	relu	
20	fc8	conv	3rd fully connected layer
21	prob	softmax	

net2= imagenet-matconvnet-vgg-verydeep-16.mat
└ 37 layers

layer	name	type	
1	conv1_1	conv	1st conv layer (2 convs)
2	relu1_1	relu	
3	conv1_2	conv	
4	relu1_2	relu	
5	pool1	pool	
6	conv2_1	conv	2nd conv layer (2 convs)
7	relu2_1	relu	
8	conv2_2	conv	
9	relu2_2	relu	
10	pool2	pool	
11	conv3_1	conv	3rd conv layer (3 convs)
12	relu3_1	relu	
13	conv3_2	conv	
14	relu3_2	relu	
15	conv3_3	conv	
16	relu3_3	relu	
17	pool3	pool	
18	conv4_1	conv	4th conv layer (3 convs)
19	relu4_1	relu	
20	conv4_2	conv	
21	relu4_2	relu	
22	conv4_3	conv	
23	relu4_3	relu	
24	pool4	pool	
25	conv5_1	conv	5th conv layer. (3rd convs)
26	relu5_1	relu	
27	conv5_2	conv	
28	relu5_2	relu	
29	conv5_3	conv	
30	relu5_3	relu	
31	pool5	pool	
32	fc6	conv	1st fully conv layer
33	relu6	relu	
34	fc7	conv	2nd fully conv layer
35	relu7	relu	
36	fc8	conv	3rd fully conv layer
37	prob	softmax	

Figure 5: analysis of layer of each dataset

Conclusion

As I computed the confusion matrix and see the outcome of same classes with same images between BOW and CNN, I learned that CNN methods works much better. Not only there are much of train data that trained the system, but also, the method of detecting the features from the images are much efficient to classify the object. Within the CNN, as there are more layers of CNN, as it goes deeper, the more and better the system gets trained and classify the images. BOW can be much better with large splits such as 1:1000, test to training, respectively; however, it will take large amount of time to compute those features and will not as much efficient as CNN.