Problem 3: Visual Bag-of-Words (40%)

A **bag-of-words model** (BoW) can be applied to image classification, by treating image features as words. In computer vision, a bag of visual words is a vector of occurrence counts of a vocabulary of local image *features*. In this problem, you will implement a basic image-based BoW model for an image dataset with 4 categories, where we use small image patches as features.

The folder `p3_data` contains images of 4 categories (classes) and 500 RGB images for each category, all of size $(64, 64, 3)$ pixels.

First, split the dataset into two subsets (i.e., training and testing sets). The first subset contains the first 375 images of each category, while the second subset contains the remaining images. Thus, a total of $375 \times 4 = 1500$ images are in the training set, and $125 \times 4 = 500$ images in the testing set. We will denote them as `X_train` and `X_test`, respectively.

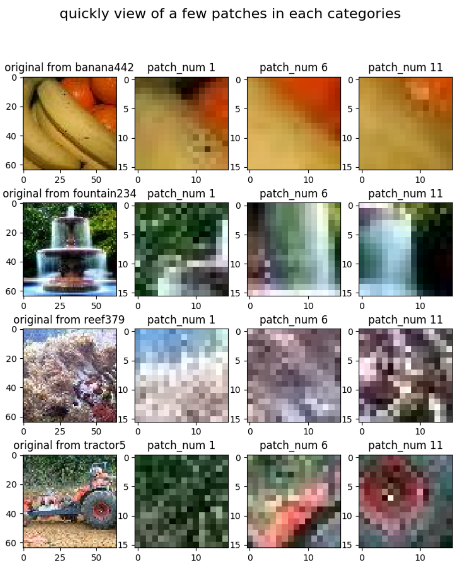
1. (5%) Divide up each image in both `X_train` and `X_test` into a grid of $(16, 16, 3)$ image patches. Since each image is of size $(64, 64, 3)$, this will result in 16 different patches of size $(16, 16, 3)$ for each image. You can imagine the patches as puzzle pieces which together would reconstitute the whole image. Pick 4 images (one from each category) randomly and plot 3 such patches from each image you choose. Describe whether you are able to classify an image by seeing just a few patches and write why.

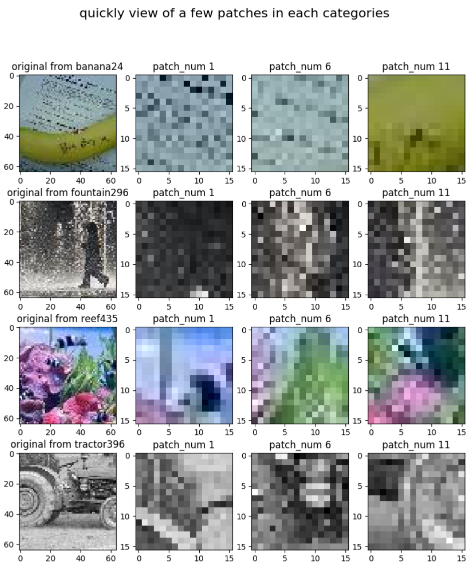
Ans of problem3.1:

Naming of the patches in one image:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

We plot the 1, 6, 7 patches in each images, and show ay below:





I think that in some cases, if the main feature that helping we to recognize the category to an image is on the patches we choose, it is possible to classify an image by seeing just three patches. E.g. banana442(黃色，而且有看起來像果蒂的咖啡色), fountain234(有流水的淺藍色), reef379 and reef435(粉紅色、綠色和藍色，可能為珊瑚，海草和海的顏色)，tractor005 and tractor396(有圓圓看起來像輪胎的東西)。However, in some cases we would not that lucky to see any useful feature in those three patches. E.g. banana024(前

兩個 patches 完全直截到背景和主角無關), fountain(黑白又切成 patches 無法認出來那些白點是水花)。所以結論是能不能靠幾個 patches 就認出來，那就要看你能不能剛好取到有重點(feature)的 patches 來看。

2. (15%) Flatten the patches into 1D vectors and store them as variables `X_train_patches` and `X_test_patches`. Since each patch is of size $16 \times 16 \times 3 = 768$ where each image contains 16 such patches, `X_train_patches` and `X_test_patches` should be of size $(24000, 768)$ and $(8000, 768)$, respectively.

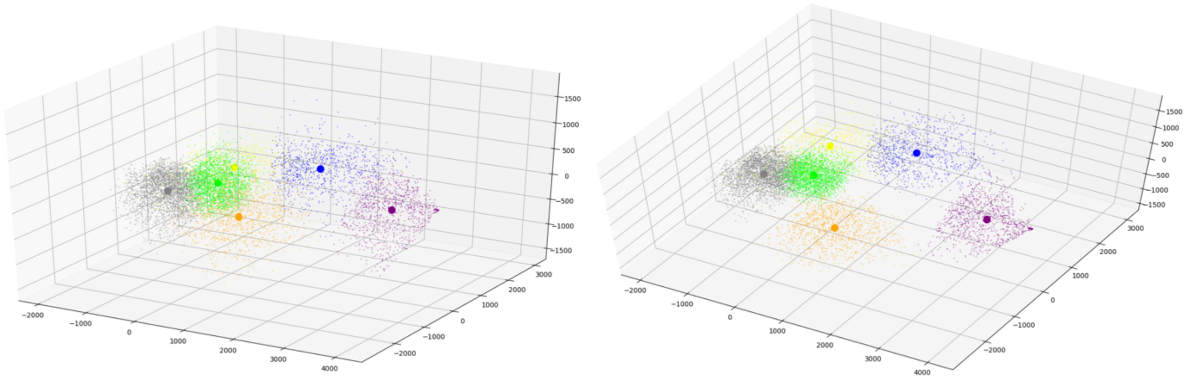
   Use the k-means algorithm to divide the training patches (features) into C clusters. You may choose $C = 15$ and maximum number of iterations $= 5000$ for simplicity. The centroid of each cluster then indicates a visual word.

   Construct the 3-dimensional PCA subspace from the training features. Randomly select 6 clusters from the above results. Plot the visual words (i.e., centroids) and their associated features (i.e., patches) in this PCA subspace. Use the same color for features belonging to the same cluster in your visualization.

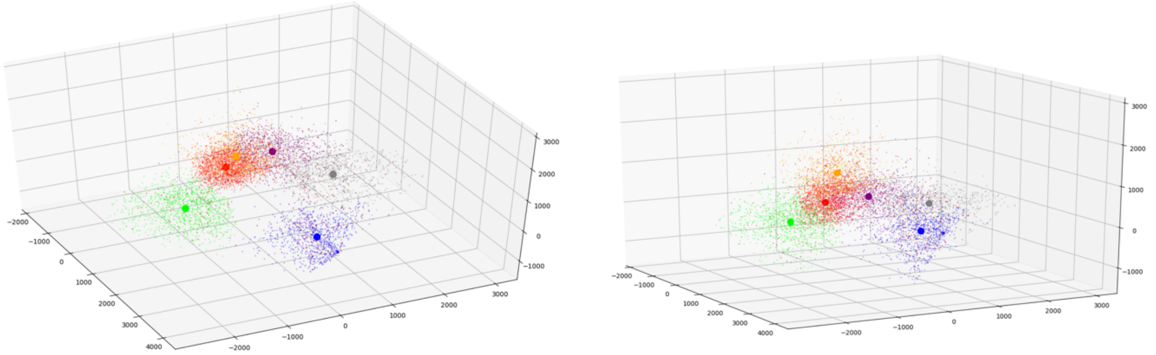
Ans of problem3.2:

共分為 15 個 clusters from cluster0~cluster15

下圖為 cluster3~cluster8 的 visualization 與其中心(3~8 對應到灰藍紫黃綠橘)



下圖為 cluster4~cluster9 的 visualization 與其中心(4~9 對應到灰藍紫紅綠橘)

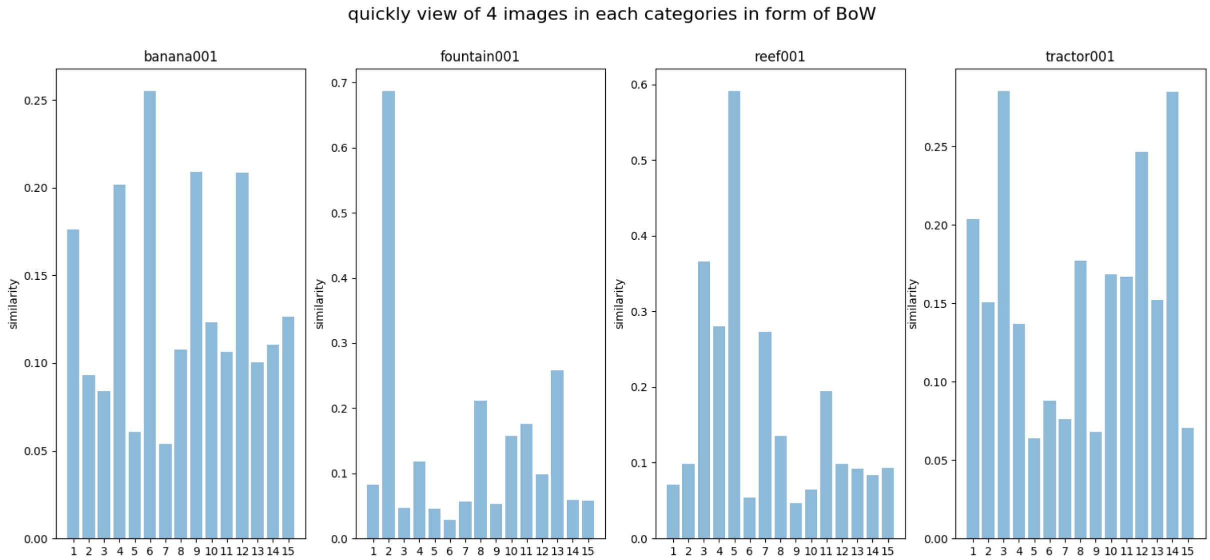


3. (10%) With the derived dictionary of visual words, you can now represent each image as BoW features. We will adopt the **soft-max** strategy when encoding image patches as detailed below.

   Take an image with 4 patches and a learned dictionary with 3 visual words ($C = 3$) for example. Table 1 lists the Euclidean distance between the patches f_i (with $i = 1, 2, 3, 4$) and the centroids c_j (with $j = 1, 2, 3$). For each patch, the reciprocal of its distance to each centroid would be normalized (i.e., the reciprocal of each entry in a row in Table 1 sums to unity, as shown in Table 2). Each attribute in the BoW is then determined by the maximum value of the soft-encoded features in that dimension (i.e., **max-pooling**). For example, the BoW of the patches in Table 1 would be $[0.55\, 0.27\, 0.55]$ by taking the maximum value of each column.

   Now, compute the BoW of training images in `X_train`, resulting in a matrix of size $(1500, C)$. Choose one image from each category and visualize its BoW using histogram plot.

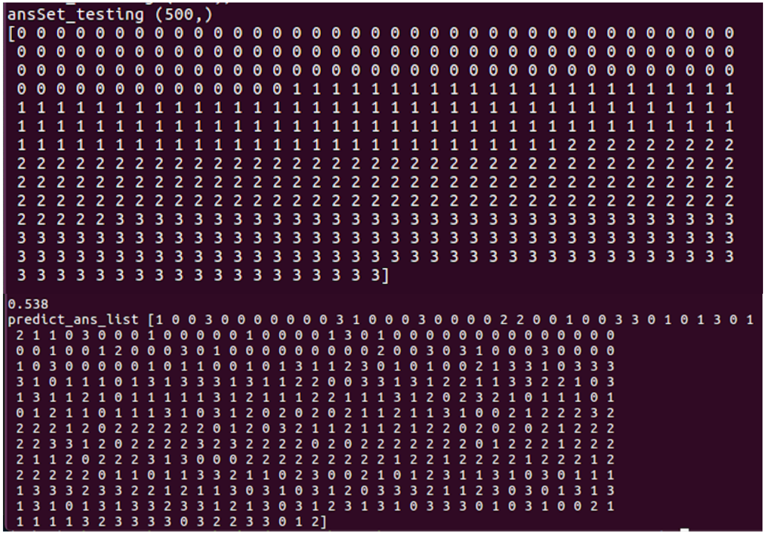
Ans of problem3.3:



4. (10%) Adopt the k-nearest neighbors classifier (k-NN) to perform classification on `X_test` using the above BoW features (you may choose $k = 5$ for simplicity). Report the classification accuracy.

Ans of problem3.4:

Accuracy= 0.538



```
ansSet_testing (500,)
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 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 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3]

0.538
predict_ans_list [1 0 0 3 0 0 0 0 0 0 0 3 1 0 0 0 3 0 0 0 0 2 2 0 0 1 0 0 3 3 0 1 0 1 3 0 1
 2 1 1 0 3 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 3 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 1 0 0 1 2 0 0 0 3 0 1 0 0 0 0 0 0 0 0 0 2 0 0 3 0 3 1 0 0 0 3 0 0 0 0
 1 0 3 0 0 0 0 0 1 0 1 1 0 0 1 0 1 3 1 1 2 3 0 1 0 1 0 0 2 1 3 3 1 0 3 3 3
 3 1 0 1 1 1 0 1 3 1 3 3 3 1 3 1 1 2 2 0 0 3 3 1 3 1 2 2 1 1 3 3 2 2 1 0 3
 1 3 1 1 2 1 0 1 1 1 1 1 3 1 2 1 1 1 2 2 1 1 1 3 1 2 0 2 3 2 1 0 1 1 1 0 1
 0 1 2 1 1 0 1 1 1 3 1 0 3 1 2 0 2 0 2 0 2 1 1 2 1 1 3 1 0 0 2 1 2 2 2 3 2
 2 2 2 1 2 0 2 2 2 2 2 2 0 1 2 0 3 2 1 1 2 1 1 2 1 2 2 0 2 0 2 0 2 1 2 2 2
 2 2 3 3 1 2 0 2 2 2 2 3 2 3 2 2 2 2 0 2 0 2 2 2 2 2 2 2 0 1 2 2 2 1 2 2 2
 2 1 1 2 0 2 2 2 3 1 3 0 0 0 2 2 2 2 2 2 2 2 2 1 2 2 1 2 2 2 2 1 2 2 2 1 2
 2 2 2 2 2 0 1 1 0 1 1 3 3 2 1 1 0 2 3 0 0 2 1 0 1 2 3 1 1 3 1 0 3 0 1 1 1
 1 3 3 3 2 3 3 2 2 1 2 1 1 3 0 3 1 0 3 1 2 0 3 3 3 2 1 1 2 3 0 3 0 1 3 1 3
 1 3 1 0 1 3 1 3 3 2 3 3 1 2 1 3 0 3 1 2 3 1 3 1 0 3 3 3 0 1 0 3 1 0 0 2 1
 1 1 1 1 3 2 3 3 3 3 0 3 2 2 3 3 0 1 2]
```

My source code of problem 2 and problem3: https://github.com/shannon112/DLCVizsla