Report Homework # 1

Deep Learning for Computer Vision

Name: Divya Jain

Student Id: R07943158

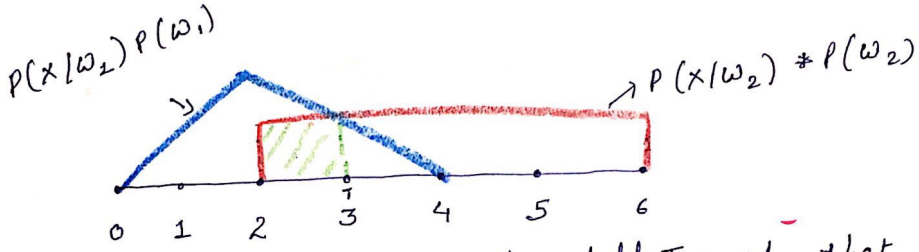
This Report presents the solutions of homework 1 (DLCV). Problem 1 has been solved on paper and then attached in report for clarification.
Problem 2 and Problem 3 requires coding and hence python coding is used by me to solve these two problems.
For problem 2 and 3, appropriate tables, figures and charts are attached. Also some critical parts of code have been written in report. Python code is indicated infront of them and they are mentioned in bold letters.

Problem 1: Bayesian Data Theory

1.) $P(x/\omega_1) = \begin{cases} \frac{1}{4}x, & x \in [0,2) \\ -\frac{1}{4}x+1, & x \in [2,4] \end{cases}$

$P(x/\omega_2)$ = uniform over (2,6)

$P(\omega_2) = \frac{4}{9}$ $\quad P(\omega_1) = 1 - P(\omega_2) = 1 - \frac{4}{9} = \frac{5}{9}$



If need to choose threshold T such that

If $x < T$, then output is ω_1

If $x > T$, then output is ω_2

$P_e = \min \{ P(\omega_1/x), P(\omega_2/x) \}$

$= \int_2^T P(x/\omega_2) P(\omega_2)\, dx + \int_T^4 P(x/\omega_1) P(\omega_1)\, dx.$

$= \int_2^T \frac{1}{4} \times \frac{4}{9} + \int_T^4 \left(\frac{-1}{4}x+1\right)\frac{5}{9}\, dx.$

$= \left[\frac{T-2}{9}\right] + \frac{5}{9}\left[\frac{-x^2}{8} + x\right]_T^4$

solving this ~~guess~~ and choosing Threshold $T = 16/5$, we get

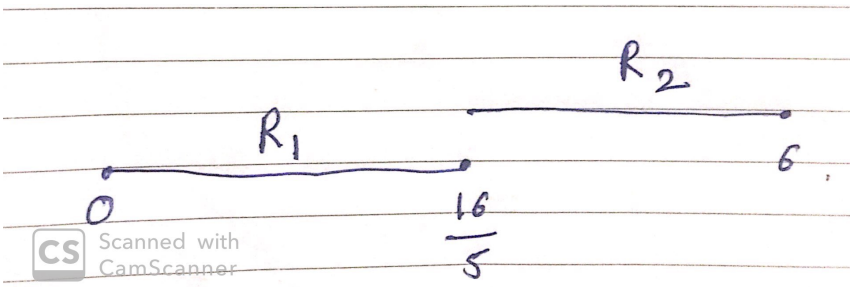
$P_e = \frac{8}{45}$

Scanned with CamScanner

We get Probability of Error as 8/45

Decision Regions

R1 = {0 < x <16/5}, Choose w1
R2 = { 16/5 < x < 6}, Choose w2

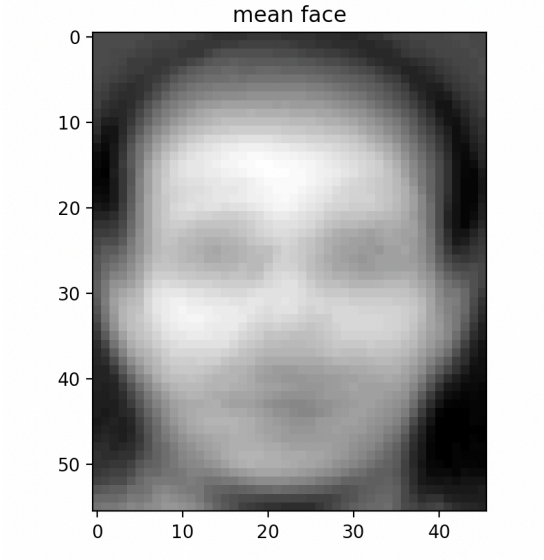


Problem 2 : Principal Component Analysis

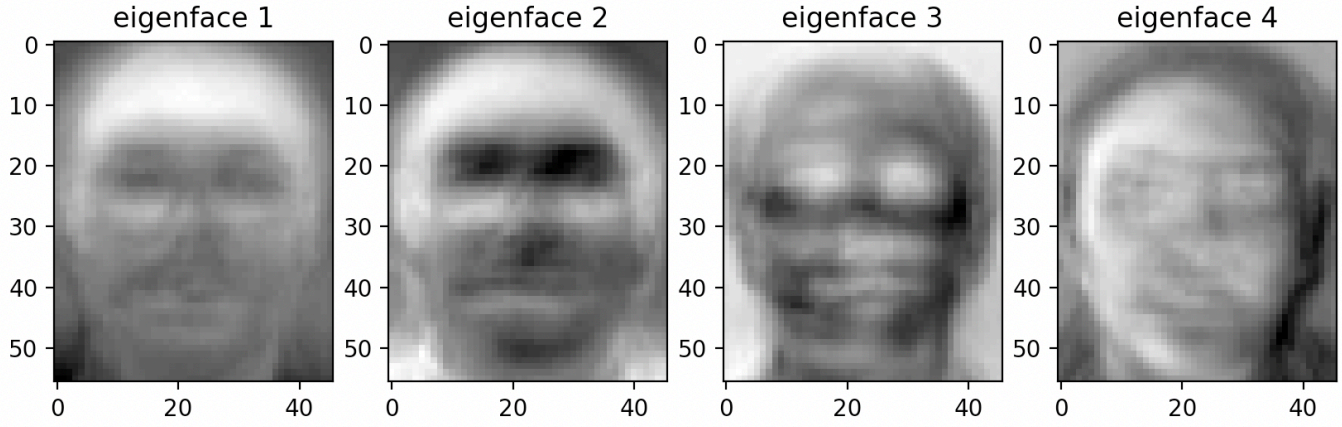
1. Perform PCA on the training set. Plot the mean face and the first four eigenfaces.

We use python code to calculate the mean face. Using mean function from numpy library help us calculate the mean face.

mean_face = train_data.mean(axis =0) **//python code**

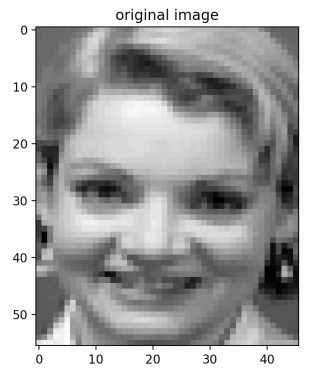


The first four eigen faces
Eigen Faces are calculated using the algorithm PCA.
We import PCA from sklearn.decomposition.

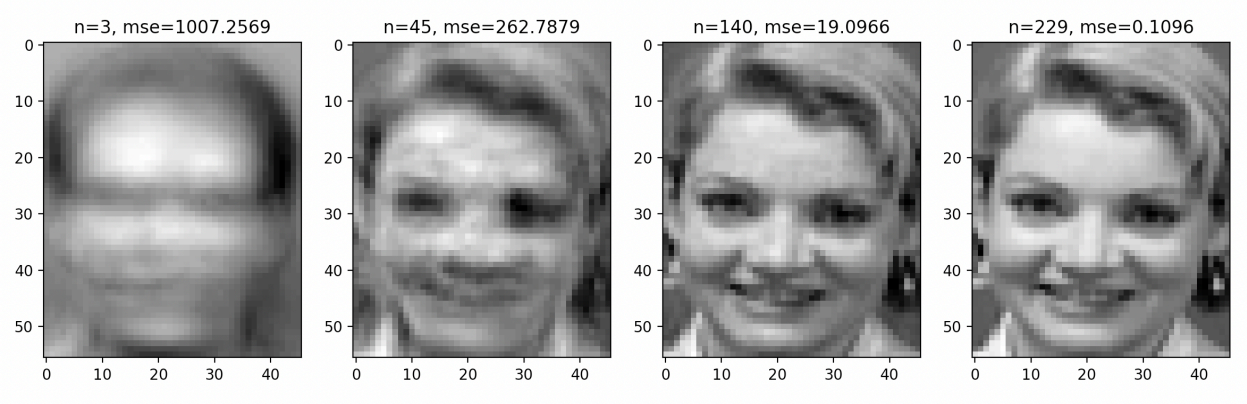


2. Take **person**$_1$**image**$_1$, and project it onto the PCA eigenspace you obtained above. Reconstruct this image using the first n = 3, 45, 140, 229 eigenfaces. Plot the four reconstructed images.
3. For each of the four images you obtained in 2., compute the mean squared error (MSE) between the reconstructed image and the original image. Record the corresponding MSE values in your report.

This is the image1 of person 1. This can be referred as the original image. Now we use the eigenspace already obtained to reconstruct the image.



Here n denotes the number of Eigen Faces.
After reconstructing the images, we compute mean square error loss.
The Mean Square Error (mse) decreases as we increases the number of eigenfaces.



No. Of Eigen Faces (n)	3	45	140	229
Mean Square Error (MSE)	1007.2569	262.7879	19.0966	0.1096

Table 1: Using the first n eigenfaces to reconstruct the original images and the corresponding mean square error (MSE)

4. Now, apply the k-nearest neighbors algorithm to classify the testing set images. First, you need to determine the best k and n values by 3-fold cross-validation. For simplicity, the choices for such hyper-parameters are k = {1, 3, 5} and n = {3, 45, 140}. Show the cross- validation results and explain your choice for (k, n).

KNN = KNeighborsClassifier() //python code

We set our hyper parameters with. K = {1,3,5} and n = {3, 45, 140} to obtain the following score.

k\n	3	45	140
1	0.70416667	0.92916667	0.92916667
3	0.61666667	0.85833333	0.85833333
5	0.52083333	0.79166667	0.75416667

Table 2: Accuracy for different choices of k and n as stated in problem

We notice that accuracy rate is same for (k,n) = (1,45) and (k,n) = (1,140)

We choose k = 1 and n = 45 for lower dimension.

The recognition rate is 95.625 %

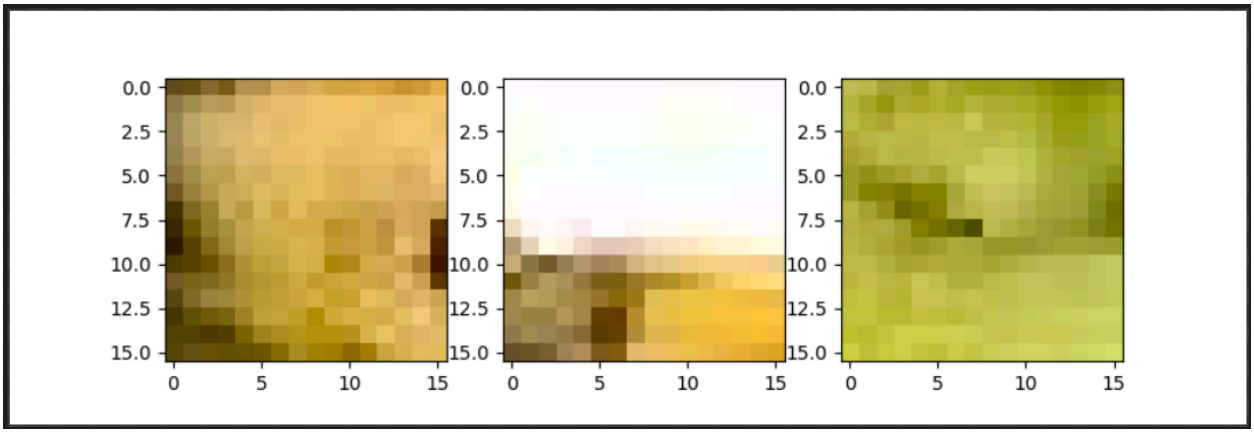
Problem 3: Visual Bag-of-Words (40%)

1. Divide up each image in both X_train and X_test into a grid of (16,16,3) imagepatches. 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.

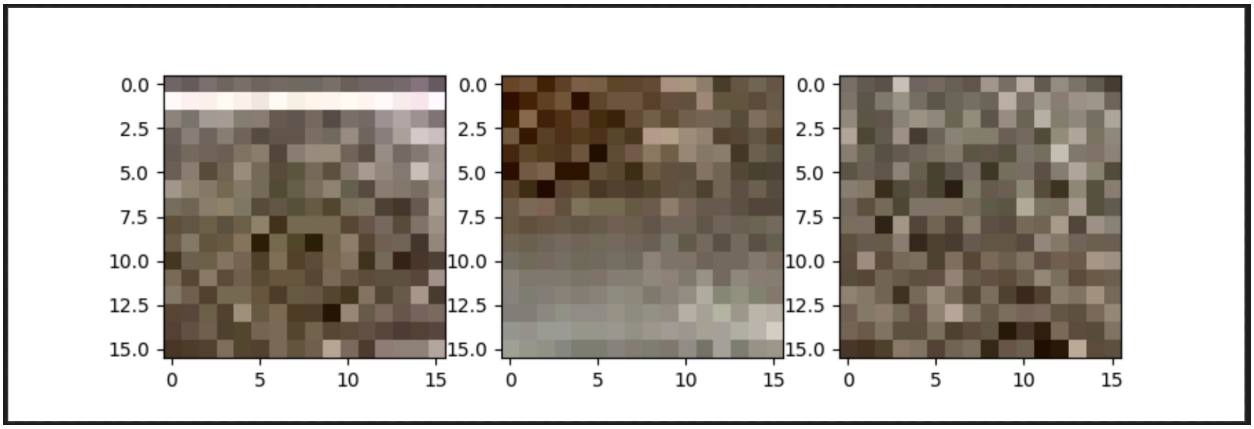
Three random patches for different class

I take first image of every class.

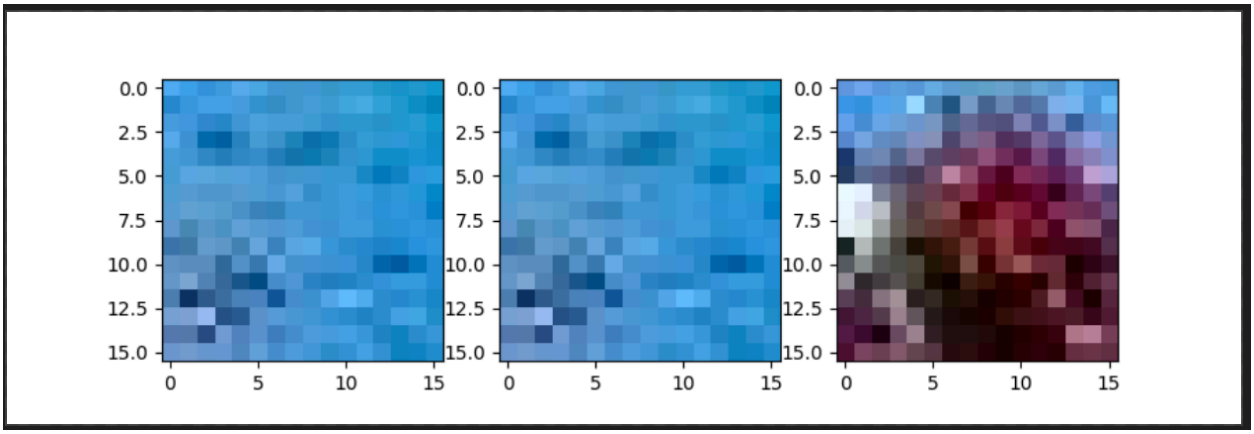
Class 1 : Banana



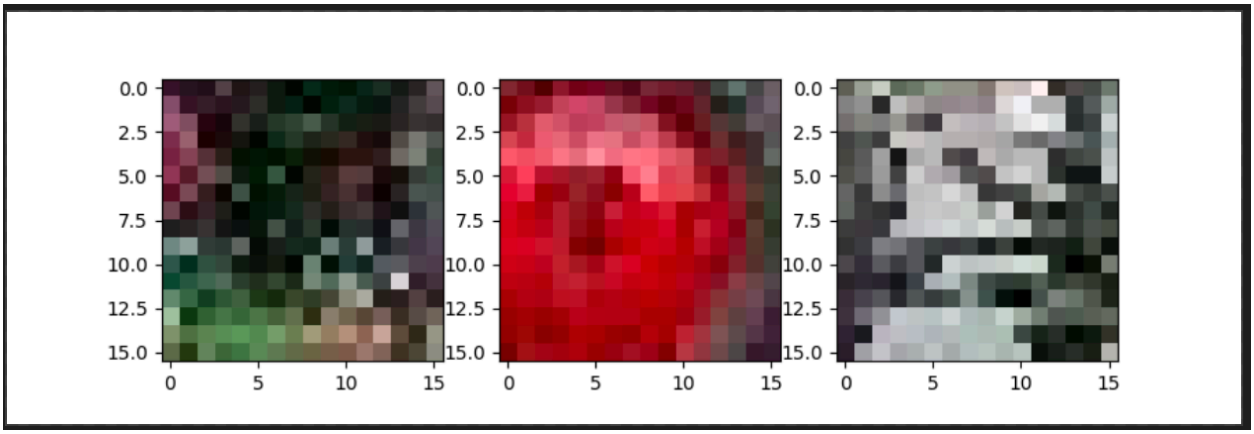
Class 2: Fountain



Class 3: Reef



Class 4: Tractor



We are able to classify two classes - banana and reef. May be their color help them in their classification. Classification of other two classes seems a bit difficult.

2. We choose k-means to divide the training patches into 15 clusters. We have maximum number of iterations as 5000.

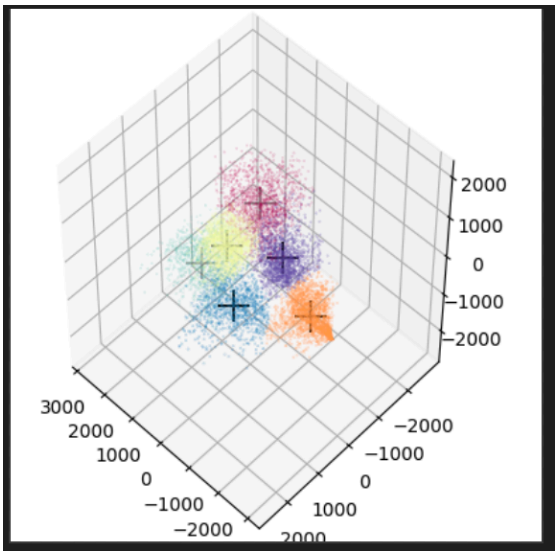
We import Kmeans from sklearn.cluster.

Kmeans = KMeans(n_clusters = 15, max_iter = 5000) **//python code**

Then we construct the 3-dimensional PCA subspace from the training features. Randomly select 6 clusters from the above results

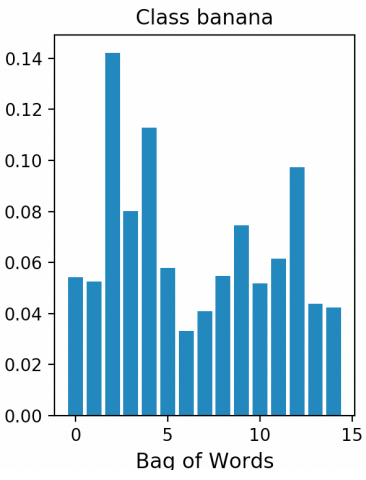
Pca = PCA(n_components =3) **//python code**

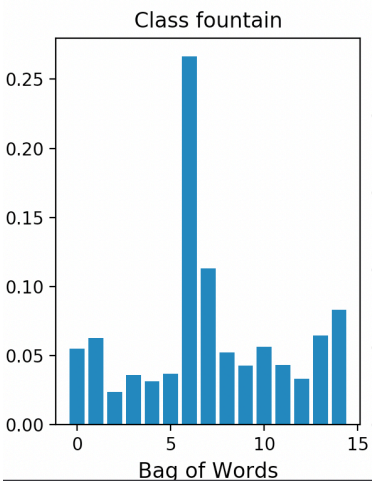
Selected_cluster = np.random.choice(15, size =6, replace = False) **//python code**

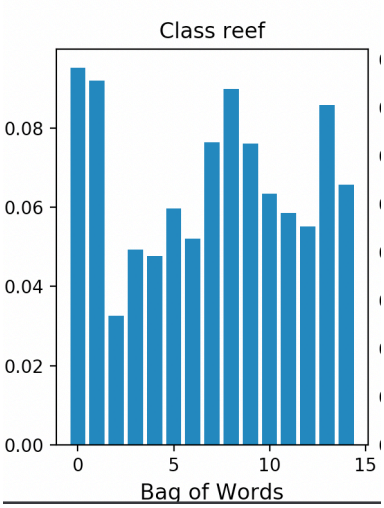


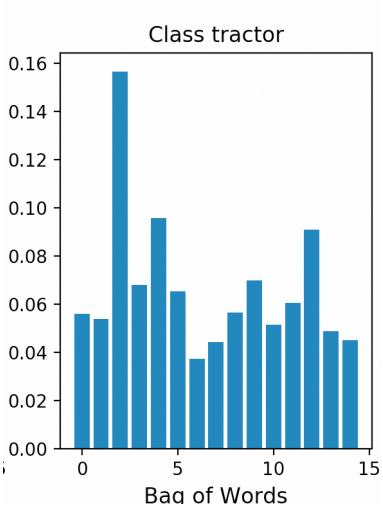
Plot showing the visual words (i.e., centroids) and their associated features (i.e., patches) in this PCA subspace.

3. We choose one image (second image) from each category and visualize its BoW using histogram plot.









4. Adopt the k-nearest neighbors classifier (k-NN) to perform classification on X_test using the above BoW features and we choose k = 5 for simplicity.

The 15 Centroids of k-means clustering are BoW

We use KNeighborClassifier on test images and we choose k =5.

Class	Classication Accuracy
Banana	0.51
Fountain	0.37
Reef	0.55
Tractor	0.46
Average Accuracy	0.47