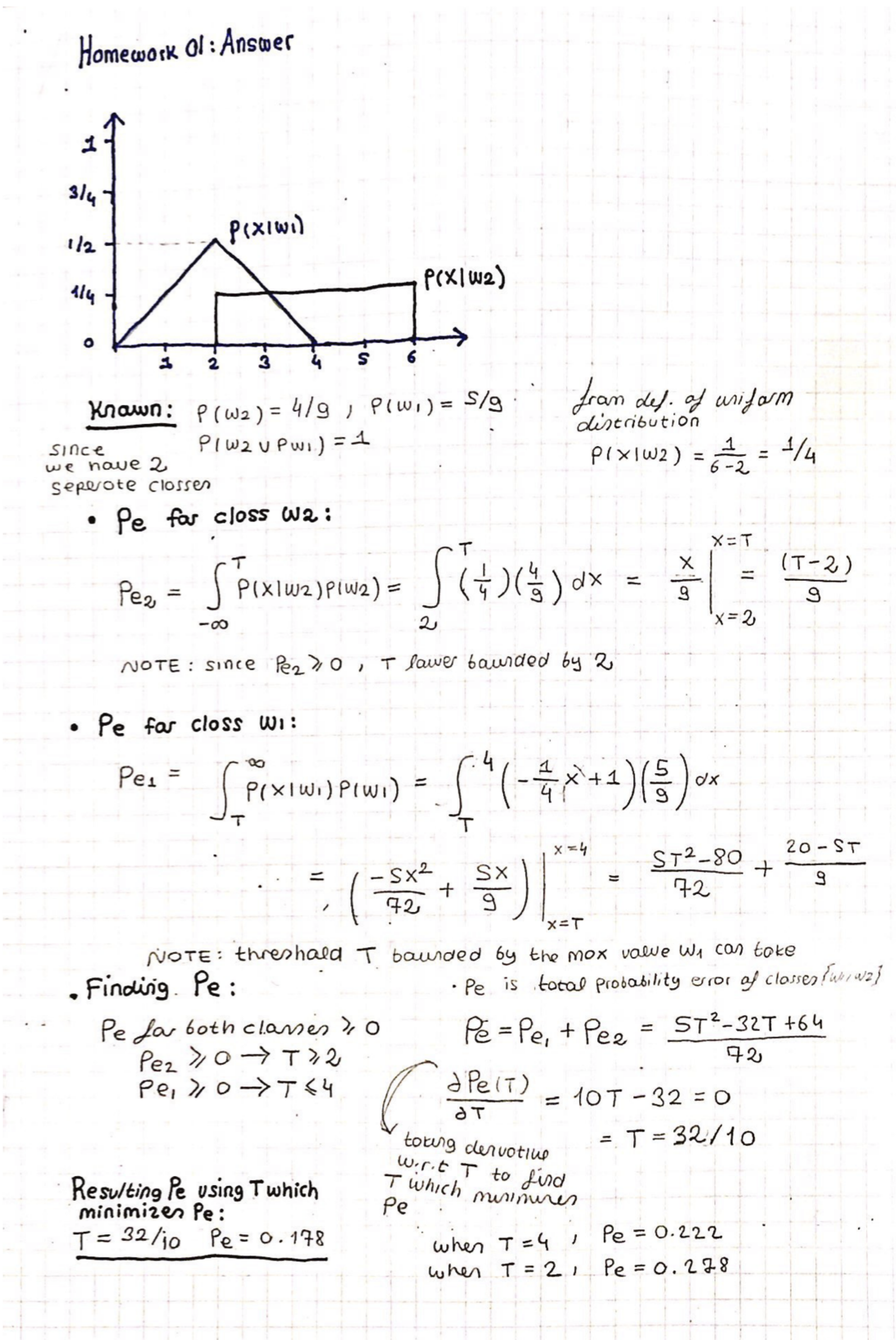
Course: Deep Learning for Computer Vision 7 October 2019
Name: Zeynep Cankara
Student Id: T08902113

Homework 01

Problem 1:



Homework 01: Answer

Known: $P(w_2) = 4/9$, $P(w_1) = 5/9$

since we have 2 seperate classes $P(w_2 \cup P w_1) = 1$

from def. of uniform distribution $P(x|w_2) = \frac{1}{6-2} = 1/4$

- **Pe for class w_2:**

$$Pe_2 = \int_{-\infty}^{T} P(x|w_2)P(w_2) = \int_{2}^{T} \left(\frac{1}{4}\right)\left(\frac{4}{9}\right) dx = \frac{x}{9}\Bigg|_{x=2}^{x=T} = \frac{(T-2)}{9}$$

NOTE: since $Pe_2 \geq 0$, T lower bounded by 2

- **Pe for class w_1:**

$$Pe_1 = \int_{T}^{\infty} P(x|w_1)P(w_1) = \int_{T}^{4} \left(-\frac{1}{4}x+1\right)\left(\frac{5}{9}\right) dx$$

$$= \left(\frac{-5x^2}{72} + \frac{5x}{9}\right)\Bigg|_{x=T}^{x=4} = \frac{5T^2-80}{72} + \frac{20-5T}{9}$$

NOTE: threshold T bounded by the max value w_1 can take

- **Finding Pe:**

· Pe is total probability error of classes $\{w_1, w_2\}$

Pe for both classes ≥ 0
$Pe_2 \geq 0 \rightarrow T \geq 2$
$Pe_1 \geq 0 \rightarrow T \leq 4$

$$Pe = Pe_1 + Pe_2 = \frac{5T^2-32T+64}{72}$$

$$\frac{\partial Pe(T)}{\partial T} = 10T - 32 = 0$$

$$= T = 32/10$$

taking derivative w.r.t T to find T which minimizes Pe

Resulting Pe using T which minimizes Pe:
$T = 32/10$ $Pe = 0.178$

when $T = 4$, $Pe = 0.222$
when $T = 2$, $Pe = 0.278$

Figure 1: Handwritten answer of the problem 1

Problem 2:

Part 2.1:

```python
# Calculate mean of the features
mean_face = np.mean(x_train, axis=0)
mean_face = np.reshape(mean_face, image_shape)

# Obtain eigenfaces via Linear dimension reduction
# Use Singular Value Decomposition of the data to project it to a lower dim space.
N = x_train.shape[0]
pca = PCA(n_components = N-1)
# Normalize training images via subtracting the mean image
pca_fit = pca.fit(np.subtract(x_train,mean_face.reshape(-1)))

# Fit the model onto the data
eigenface1 = pca_fit.components_[0].reshape(image_shape)
eigenface2 = pca_fit.components_[1].reshape(image_shape)
eigenface3 = pca_fit.components_[2].reshape(image_shape)
eigenface4 = pca_fit.components_[3].reshape(image_shape)
```

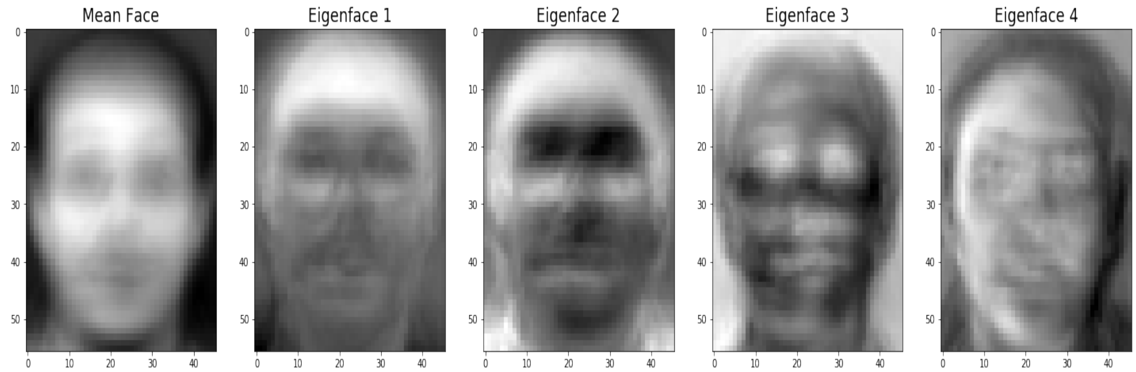


Figure 2: Plot of the mean face and first 4 eigenfaces

Part 2.2 and Part 2.3:

```python
# Normalize the image by subtracting the mean face from original face
# Project the data onto the PCA space.
pca_orig_image = pca_fit.transform(np.subtract(original_face_flat,
                                                mean_face.reshape(-1)))

# Use PCA space dimensions 3, 45, 140 and 229 to reconstruct images
for i in (3, 45, 140, 229):
    img_pca = np.dot(pca_orig_image[0,:i], pca_fit.components_[:i])
                                            + mean_face.reshape(-1)
    # Mean squared error between original and the reconstructed image
    mse = mean_squared_error(img_pca.reshape((1, img_pca.shape[0])),
                                                original_face_flat)
    img_pca = img_pca.reshape(original_face.shape)
    plt.subplot(idx)
```

```python
    plt.title("n=%s, mse=%.5f" % (i, mse))
    plt.imshow(img_pca, cmap='gray')
    idx += 1
```

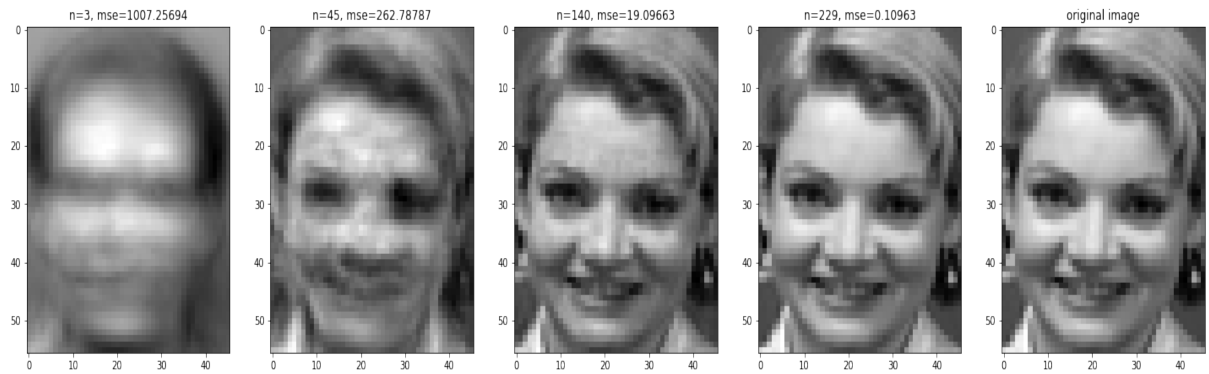


Figure 3: Reconstructed images using dimensions N equals to 3, 45, 140 and 229

Part 2.4:

```python
# Normalize and project the taining data onto the PCA subspace
x_train_normalized = pca_fit.transform(np.subtract(x_train, mean_face.reshape(-1)))
y_train = np.array(y_train)

# Configure k-Nearest Neighbours Model with k equals to 1, 3 and 5
params = {'n_neighbors':[1,3,5]}
kNN = KNeighborsClassifier()
# Perform 3 fold cross validation
cv = GridSearchCV(kNN, params, cv = 3)

df = dict()
for i in (3, 45, 140):
    cv.fit(x_train_normalized[:, :i], y_train)
    # Show cross validation results in terms of mean test score
    df['n = '+str(i)] = np.array(cv.cv_results_['mean_test_score'])
```

N	k = 1	k = 3	k = 5
3	0.691667	0.595833	0.525000
45	0.941667	0.841667	0.750000
140	0.954167	0.829167	0.745833

Part 2.5:

I decided to choose $N = 45$ and $k = 1$, even though it performs worse than $N = 140$ and $k = 1$. The reason behind my logic is that even though we increase dimension by 211.1% percent we only get 1.327% increase in the test accuracy which is not significant and might be the indication of overfitting to the data. I obtained **Accuracy on the test set: 0.95625**.

```python
# best parameters
k = 1
n = 45

# Project training images onto the PCA subspace
pca_test = pca_fit.transform(np.subtract(x_test, mean_face.reshape(-1)))

x_train_normalized = pca_fit.transform(np.subtract(x_train, mean_face.reshape(-1)))
y_train = np.array(y_train)

# train and evaluate the k-Neighbours model using the chosen parameters
kNN_opt = KNeighborsClassifier(n_neighbors = k)
kNN_opt.fit(x_train_normalized[:,:n], y_train)
pred = kNN_opt.predict(pca_test[:,:n])
print("Accuracy on the test set:", accuracy_score(y_pred = pred, y_true = y_test))
```

Question 3:

Part 3.1:

I couldn't describe an image by just seeing random patches from an image. Only cases I got correct are the ones where I can identify definite shapes such as yellow edges of the bananas. Also knowing that I only have 4 classes make my guess more easier since I have 25% correct guess possibility with a random guess. On top of that, I did a quick look over the dataset which make my guesses easier but since I am overfitting/memorizing the dataset it is not an indication of being able to identify images by just looking at image patches.

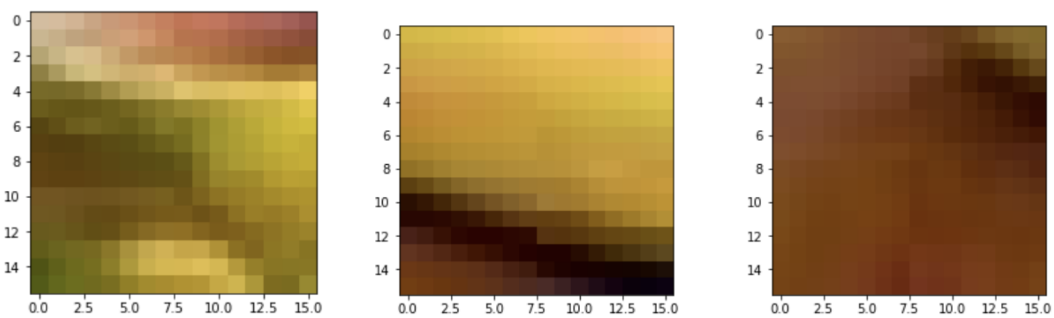


Figure 4: 3 random banana patches

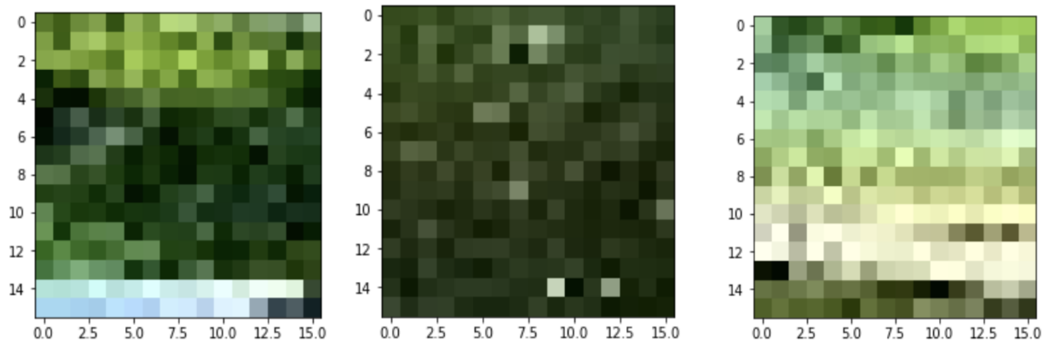


Figure 5: 3 random fountain patches

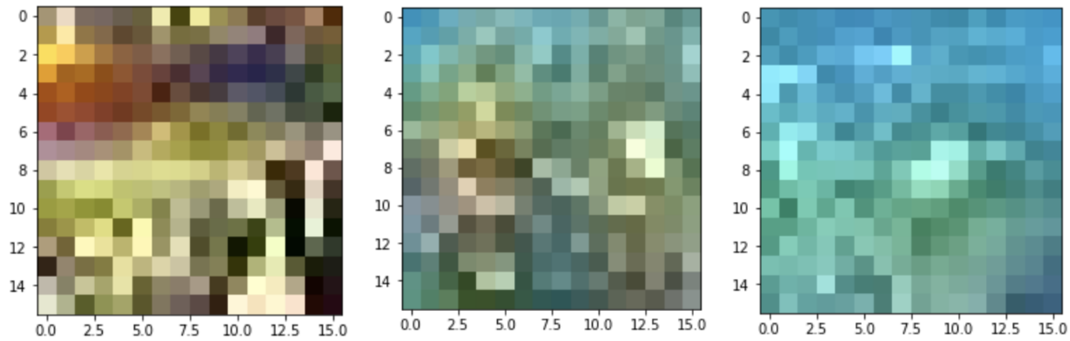


Figure 6: 3 random reef patches

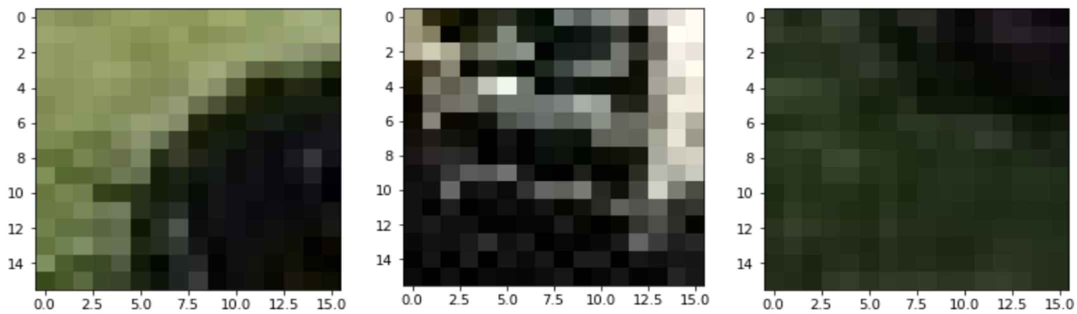


Figure 7: 3 random tractor patches

Part 3.2:

```
# Converting image patches to 1-d and storing
X_train_patches = []
X_test_patches = []

# Note: X_train_patch contains non flattened image patches
for img in X_train_patch:
    for patch in img:
```

```python
        patch = patch.reshape(-1)
        X_train_patches.append(patch)

for img in X_test_patch:
    for patch in img:
        patch = patch.reshape(-1)
        X_test_patches.append(patch)

X_train_patches = np.array(X_train_patches)
X_test_patches = np.array(X_test_patches)

print("X_train_patches shape: ", X_train_patches.shape)
# X_train_patches shape:  (24000, 768)
print("X_test_patches shape: ", X_test_patches.shape)
# X_test_patches shape:  (8000, 768)

# Configure k-means with C = 15 and maximum iteration = 5000
kmeans = KMeans(n_clusters=15, random_state=0, max_iter=5000).fit(X_train_patches)

centroids = kmeans.cluster_centers_
kmeans_clusters = kmeans.labels_

# Construction of 3 dimensional PCA subspace
pca = PCA(n_components=3)
pca_X_train_patches = pca.fit_transform(X_train_patches)

# get 6 random clusters
sample_C = 6
cluster_samples = np.random.choice(np.arange(15), size=sample_C, replace=False)

# get the centroids of the sampled clusters
centroid_samples = centroids[cluster_samples]
pca_centroids = pca.transform(centroid_samples)
```

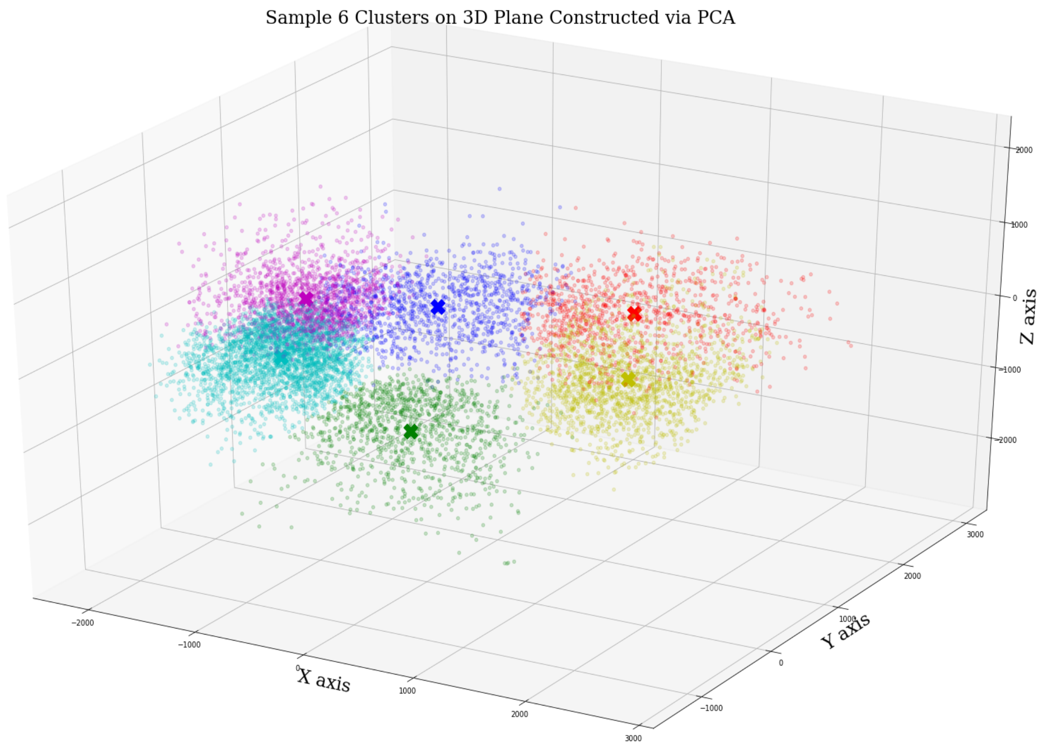


Figure 8: Visual words together with their associated features on the PCA subspace

Part 3.3:

Visualizing histogram of Softmax images for each category of images. The histograms created by constructing a Softmax matrix by calculating the euclidean distance between each cluster centroids and the training images. After reciprocal of the eucledian distance normalized, BoW calculated via taking maximum of the each encoded features in that dimension.

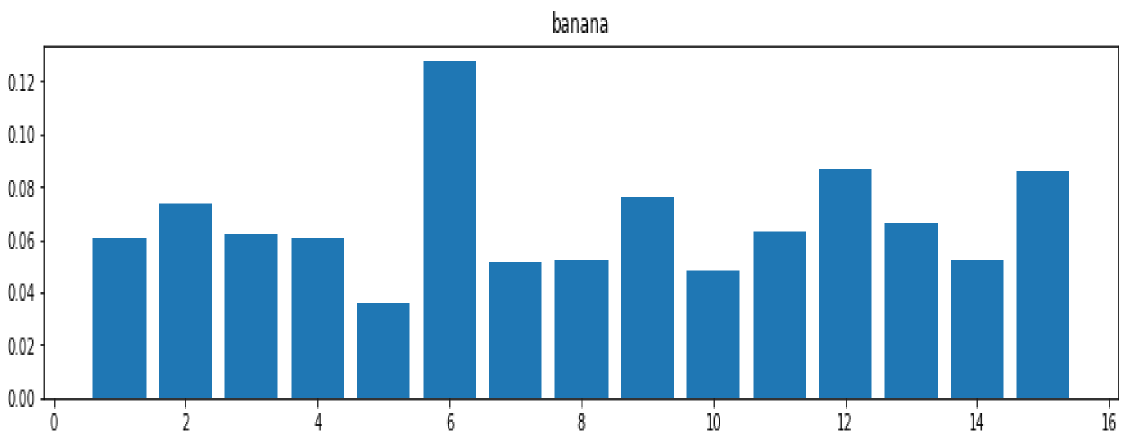


Figure 9: Histogram of the BoW: Banana

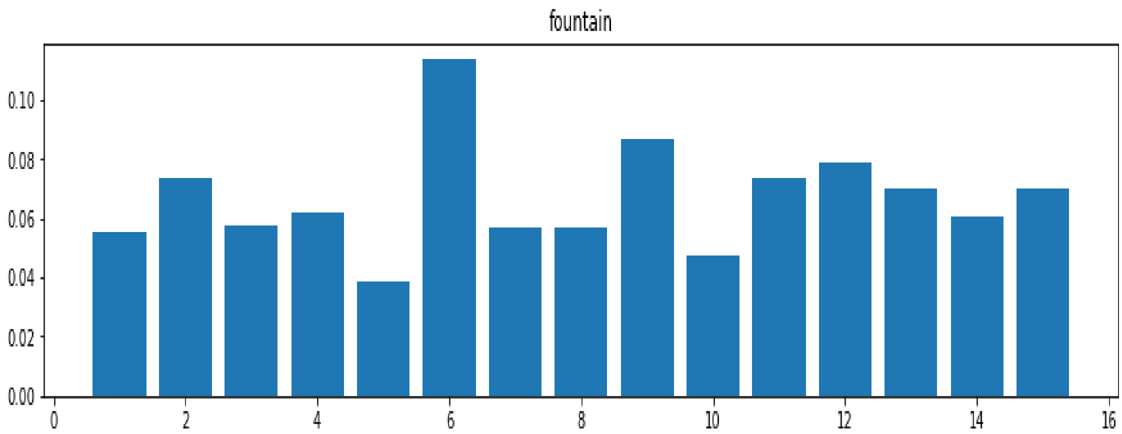


Figure 10: Histogram of the BoW: Fountain

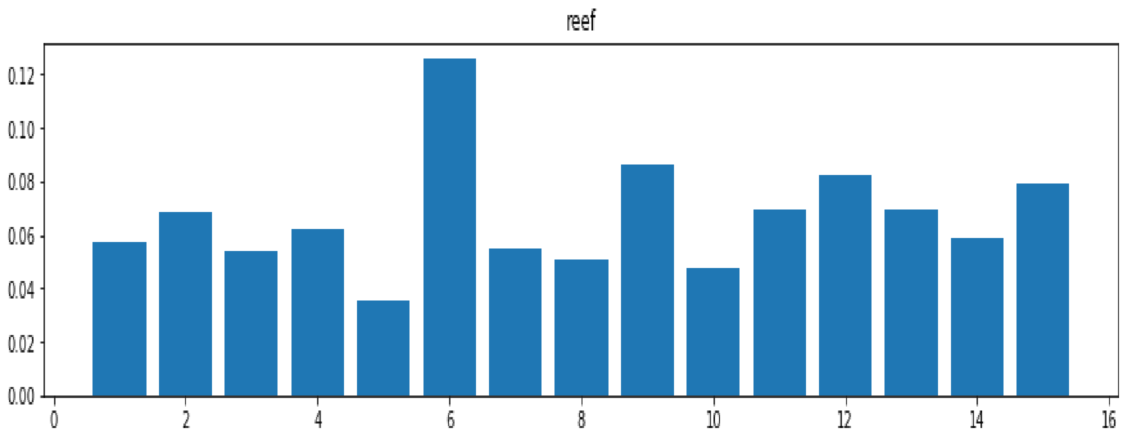


Figure 11: Histogram of the BoW: Reef

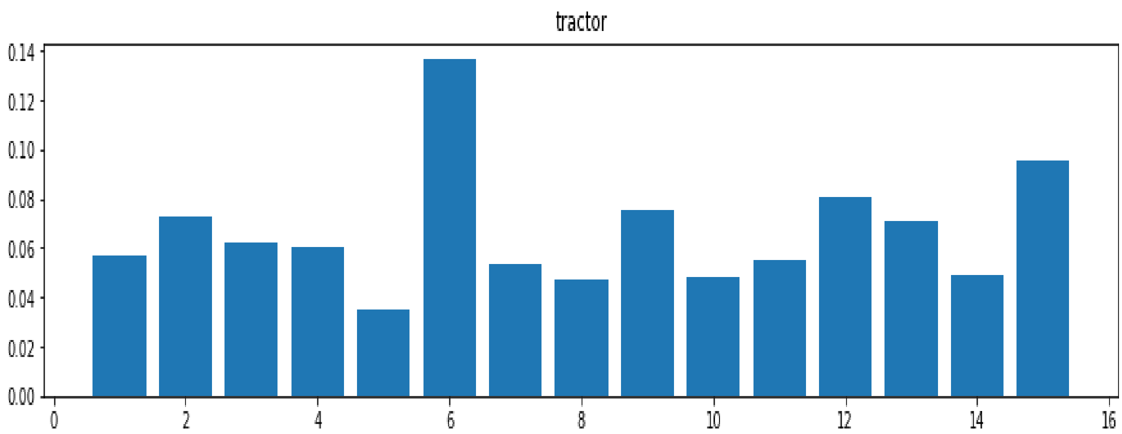


Figure 12: Histogram of the BoW: Tractor

Part 3.4:
Accuracy of Softmax on the test set: 0.552

```python
# Configure kNN with Number of neighbours is 5
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train_softmax, Y_train)

print(X_test.shape)
# (500, 12288)

# Compute softmax of the test set
X_test_softmax = Softmax(X_test, centroids)

print("Accuracy of Softmax on test set: ",
    accuracy_score(y_pred = knn.predict(X_test_softmax), y_true=Y_test))
```

Collaborators

I used Andrew Ng's Coursera Machine Learning Course as a reference and Scikit-learn documentation. Other than that no collaborators.

1. https://scikit-learn.org/stable/documentation.html