Mini Project 3 - Scene Recognition

is it in the pic?

Sohaila Zaki 201-800-998

Ibrahim Ibrahim 201-800-739

Objective:

Perform scene recognition with three different methods. We will classify scenes into one of 15 categories by training and testing on the 15-scene database

Implement three scene recognition schemes:

- Tiny images representation (get_tiny_images()) and nearest neighbor classifier (nearest_neighbor_classify()).
- Bag of words representation (build_vocabulary(), get_bags_of_words()) and nearest neighbor classifier.
- Bag of words representation and linear SVM classifier (svm_classify()).

Implementation:

1. get_tiny_images():

```
size = 16
tiny_images = np.ndarray((len(image_paths), size * size))
i = 0
for image in image_paths:
    img = resize(imread(image), (size, size), anti_aliasing=True)
    img = img.reshape(1, -1)
    tiny_images[i, :] = cv2.normalize(img, None, alpha=0, beta=255, norm_type=cv2.NORM_MINMAX)
    i += 1
print(f"Tiny images are generated successfully and its size is {tiny_images.shape}.")
return tiny_images
```

In this function, the images are rescaled into small sizes (16x16). The functions is used to represent the whole images with less amount of information by keeping the low frequencies only and getting rid of the high frequencies.

2. build_vocabulary():

```
Feature_Vectors = []
cells_per_block = (2, 2)
pixels_per_cell = (8, 8)
for image in image_paths:
    feature_vector = hog(imread(image), orientations=9, pixels_per_cell=pixels_per_cell,
                         cells_per_block=cells_per_block, block_norm='L2-Hys', visualize=False,
                         transform_sqrt=False, feature_vector=True, channel_axis=None).reshape(-1, 2 * 2 * 9)
    Feature_Vectors.append(feature_vector)
Feature_Vectors = np.vstack(Feature_Vectors)
clf = MiniBatchKMeans(n_clusters=vocab_size, max_iter=100).fit(Feature_Vectors)
print(f"Vocabularies are built successfully and centroids size is {clf.cluster_centers_.shape}.")
return clf.cluster_centers_
```

In this function, we build the vocabulary bag that will compare the feature vector wit to build the histogram.

3. get_bags_of_words():

```
vocab = np.load('vocab.npy')
print('Loaded vocab from file.')
cells_per_block = (2, 2)
pixels_per_cell = (8, 8)
hist = np.zeros((len(image_paths), vocab.shape[0]))
i = 0
for image in image_paths:
    # Read image grey scale
    img = imread(image)
    # Using HOG to detect key points
    feature_vector = hog(img, orientations=9, pixels_per_cell=pixels_per_cell,
                         cells_per_block=cells_per_block, block_norm='L2-Hys', visualize=False,
                         transform_sqrt=False, feature_vector=True, channel_axis=None).reshape(-1, 2 * 2 * 9)
    h = np.zeros(vocab.shape[0])
    # using eucliddean to compute distance
    distance = cdist(vocab, feature_vector, 'euclidean')
    # find the feature with minimum distance
    j = np.argmin(distance, axis=0)
    idx, count = np.unique(j, return_counts=True)
    h[idx] += count
    h = h / np.linalg.norm(h)
    # append the bin to histogram
    hist[i] = h
    # Increment
    i += 1

return hist
```

In this function, feature vectors are extracted from each image in our dataset and compare it with other generated vocabularies to calculate the histogram of the image features.

4. svm_classify():

```
SVC = LinearSVC(C=700.0, class_weight=None, dual=True, fit_intercept=True,
                intercept_scaling=1, loss='squared_hinge', max_iter=100,
                multi_class='ovr', penalty='l2', random_state=0, tol=1e-4,
                verbose=0)
SVC.fit(train_image_feats, train_labels)

predicted = SVC.predict(test_image_feats)
return predicted
```

In this function, linear support vector machine is used to model the data by training on the data and fitting the test data to classify the images according to their feature vectors.

5. **nearest_neighbor_classify():**

```
k = 1
categories = np.unique(train_labels)
predicts = []
distances = cdist(test_image_feats, train_image_feats, 'euclidean')
for d in distances:
    labels = []
    index = np.argsort(d)
    for i in range(k):
        labels.append(train_labels[index[i]])
    amount = 0
    for item in categories:
        if labels.count(item) > amount:
            label_final = item
    predicts.append(label_final)
return predicts
```

In this function, KNN is used as a classifier to model the data by training on the data and fitting the test data to classify the images according to their feature vectors.

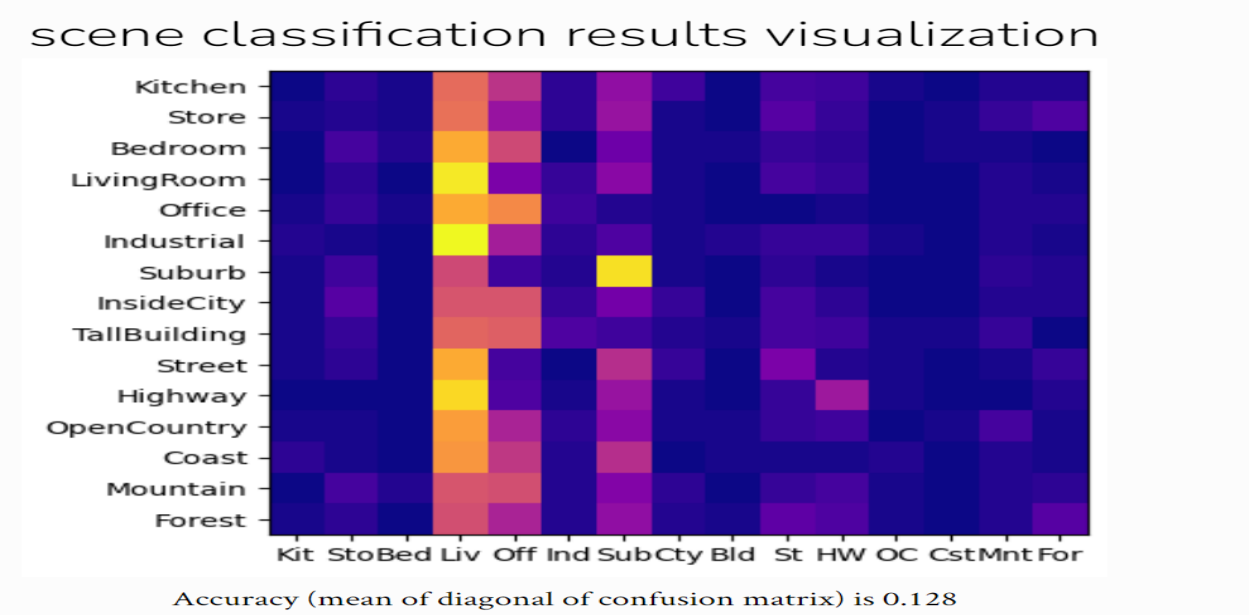
Parameters

Setting cells per blocks equals 2 and pixels per cell equals 8 the following results for both Supported Vector machine and Nearest Neighbor.

Results:

Tiny Image and SVM (12%)

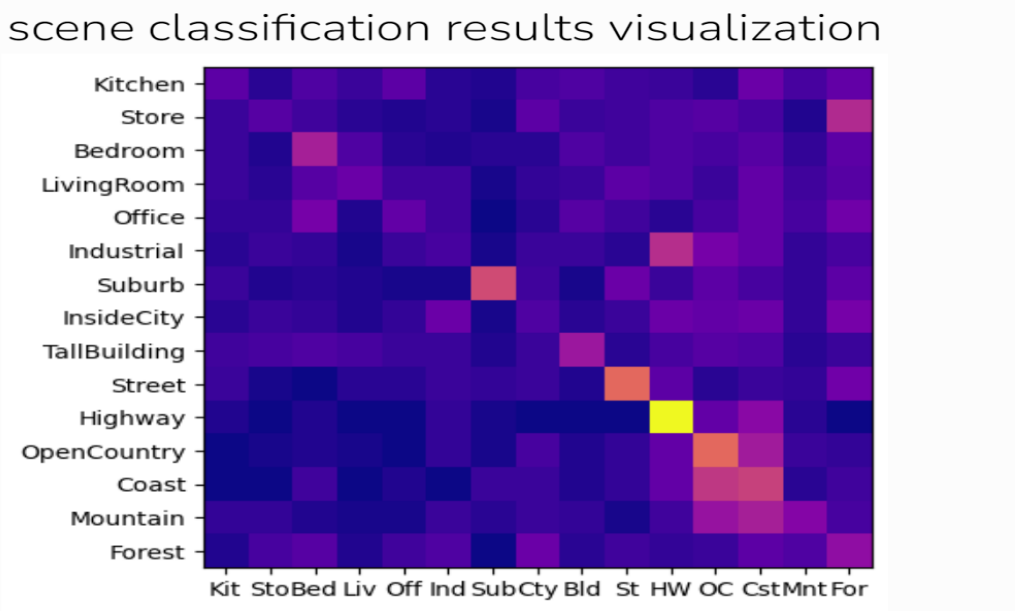
```
PS C:\Users\Ibrahim\Downloads\Compressed\drive-download-20220422T142554Z-001\code> python main.py -f  tiny_image -c support_vector_machine
Getting paths and labels for all train and test data.
Using tiny_image representation for images.
Loading tiny images...
Tiny images are generated successfully and its size is (1500, 256).
Tiny images are generated successfully and its size is (1500, 256).
Tiny images loaded.
Using support_vector_machine classifier to predict test set categories.
Creating results_webpage/index.html, thumbnails, and confusion matrix.
Accuracy (mean of diagonal of confusion matrix) is 12.800%
Wrote results page to results_webpage/index.html.
```



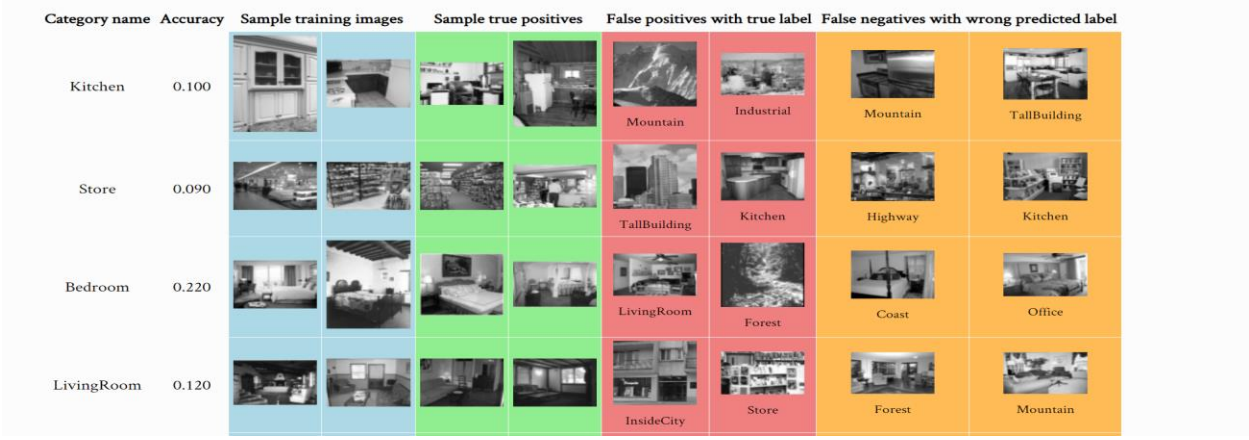
Category name	Accuracy	Sample training images	Sample true positives	False positives with true label	False negatives with wrong predicted label
Kitchen	0.000			 TallBuilding  Coast	 Office  Suburb
Store	0.020			 Street  OpenCountry	 LivingRoom  Suburb
Bedroom	0.020			 Kitchen  Mountain	 LivingRoom  LivingRoom
LivingRoom	0.510			 Industrial  Kitchen	 Office  Mountain
Office	0.380			 Highway  Mountain	 LivingRoom  LivingRoom

Tiny Image and NN (21%)

```
PS C:\Users\Ibrahim\Downloads\Compressed\drive-download-20220422T142554Z-001\code> python main.py -f  tiny_image -c nearest_neighbor
Getting paths and labels for all train and test data.
Using tiny_image representation for images.
Loading tiny images...
Tiny images are generated successfully and its size is (1500, 256).
Tiny images are generated successfully and its size is (1500, 256).
Tiny images loaded.
Using nearest_neighbor classifier to predict test set categories.
Creating results_webpage/index.html, thumbnails, and confusion matrix.
Accuracy (mean of diagonal of confusion matrix) is 21.800%
Wrote results page to results_webpage/index.html.
```

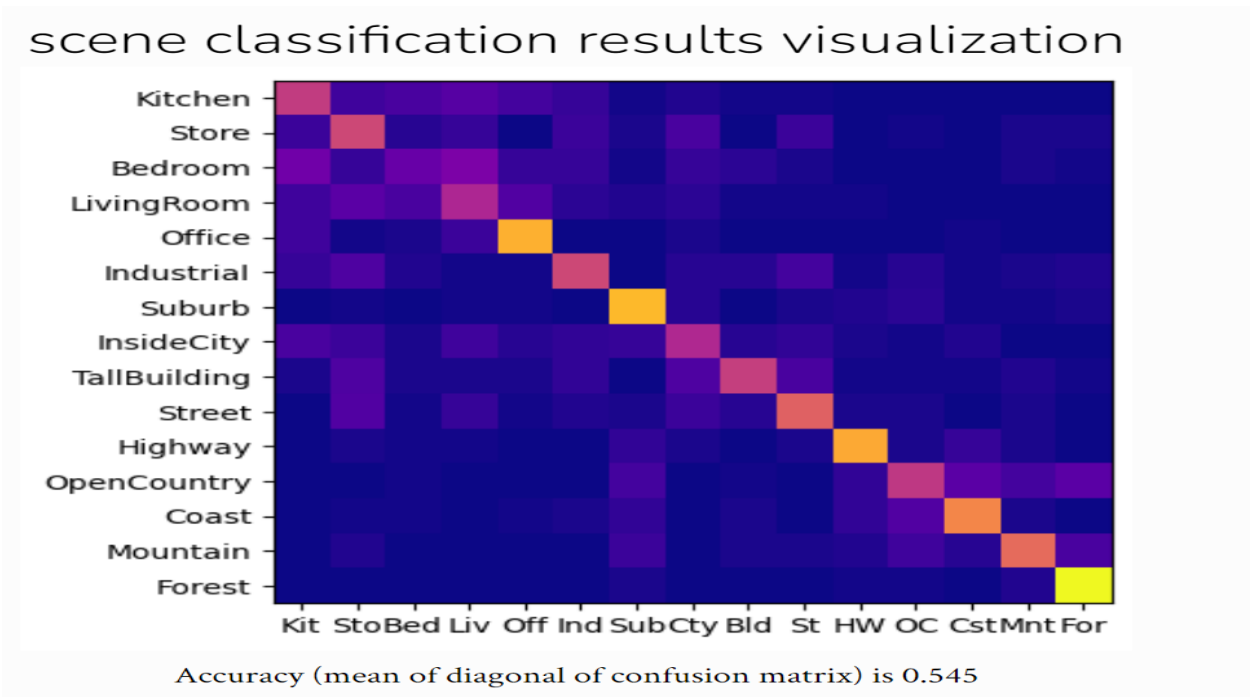


Accuracy (mean of diagonal of confusion matrix) is 0.218



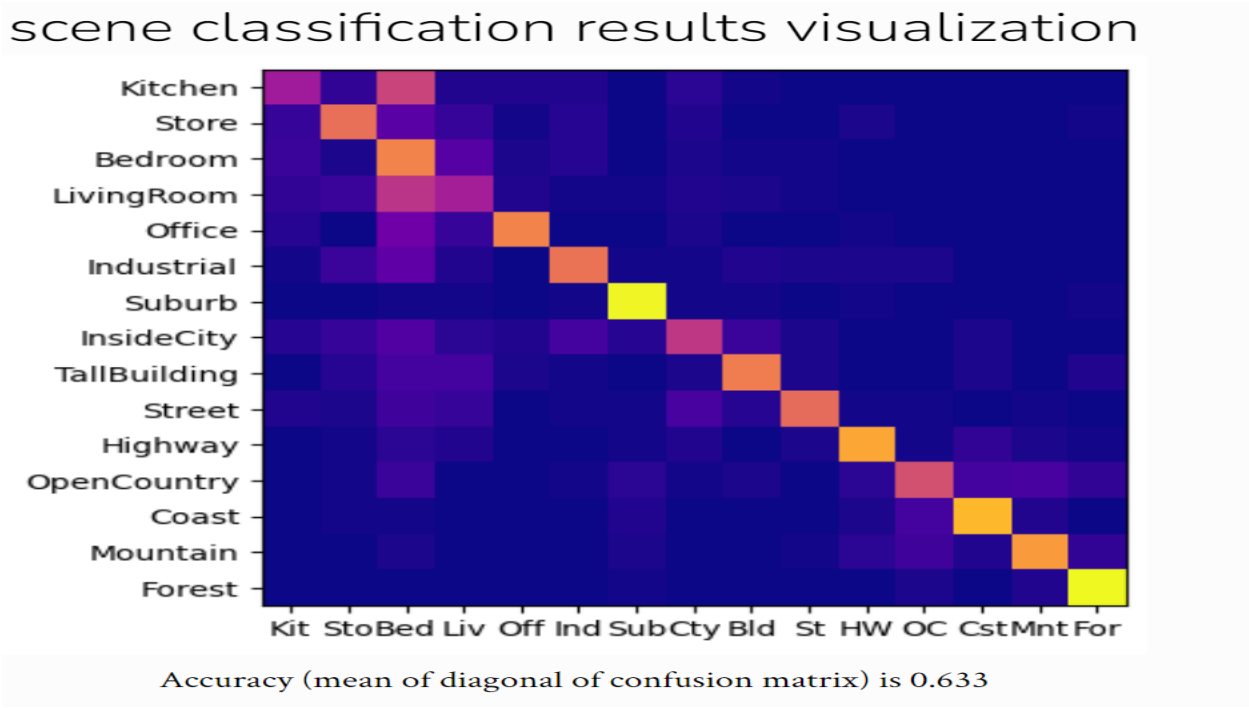
Bag of Words and NN (54%)

```
PS C:\Users\Ibrahim\Downloads\Compressed\drive-download-20220422T142554Z-001\code> python main.py -f  bag_of_words -c nearest_neighbor
Getting paths and labels for all train and test data.
Using bag_of_words representation for images.
Loaded vocab from file.
Loaded vocab from file.
Using nearest_neighbor classifier to predict test set categories.
Creating results_webpage/index.html, thumbnails, and confusion matrix.
Accuracy (mean of diagonal of confusion matrix) is 54.533%
Wrote results page to results_webpage/index.html.
```

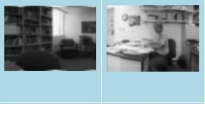
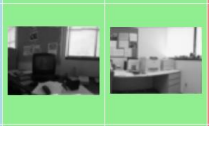


Bag of Words and SVM(63%)

```
PS C:\Users\Ibrahim\Downloads\Compressed\drive-download-20220422T142554Z-001\code> python main.py -f  bag_of_words -c support_vector_machine
Getting paths and labels for all train and test data.
Using bag_of_words representation for images.
Loaded vocab from file.
Loaded vocab from file.
Using support_vector_machine classifier to predict test set categories.
Creating results_webpage/index.html, thumbnails, and confusion matrix.
Accuracy (mean of diagonal of confusion matrix) is 63.267%
Wrote results page to results_webpage/index.html.
```



Accuracy (mean of diagonal of confusion matrix) is 0.633

Category name	Accuracy	Sample training images	Sample true positives	False positives with true label	False negatives with wrong predicted label
Kitchen	0.330			Store, LivingRoom	InsideCity, Bedroom
Store	0.600			InsideCity, Kitchen	LivingRoom, Bedroom
Bedroom	0.660			LivingRoom, LivingRoom	LivingRoom, Store
LivingRoom	0.340			Store, Bedroom	InsideCity, Bedroom
Office	0.660			Bedroom, LivingRoom	Bedroom, LivingRoom