theory

October 7, 2021

[ ]:

1 16720 (B) Bag of Visual Words - Assignment 2

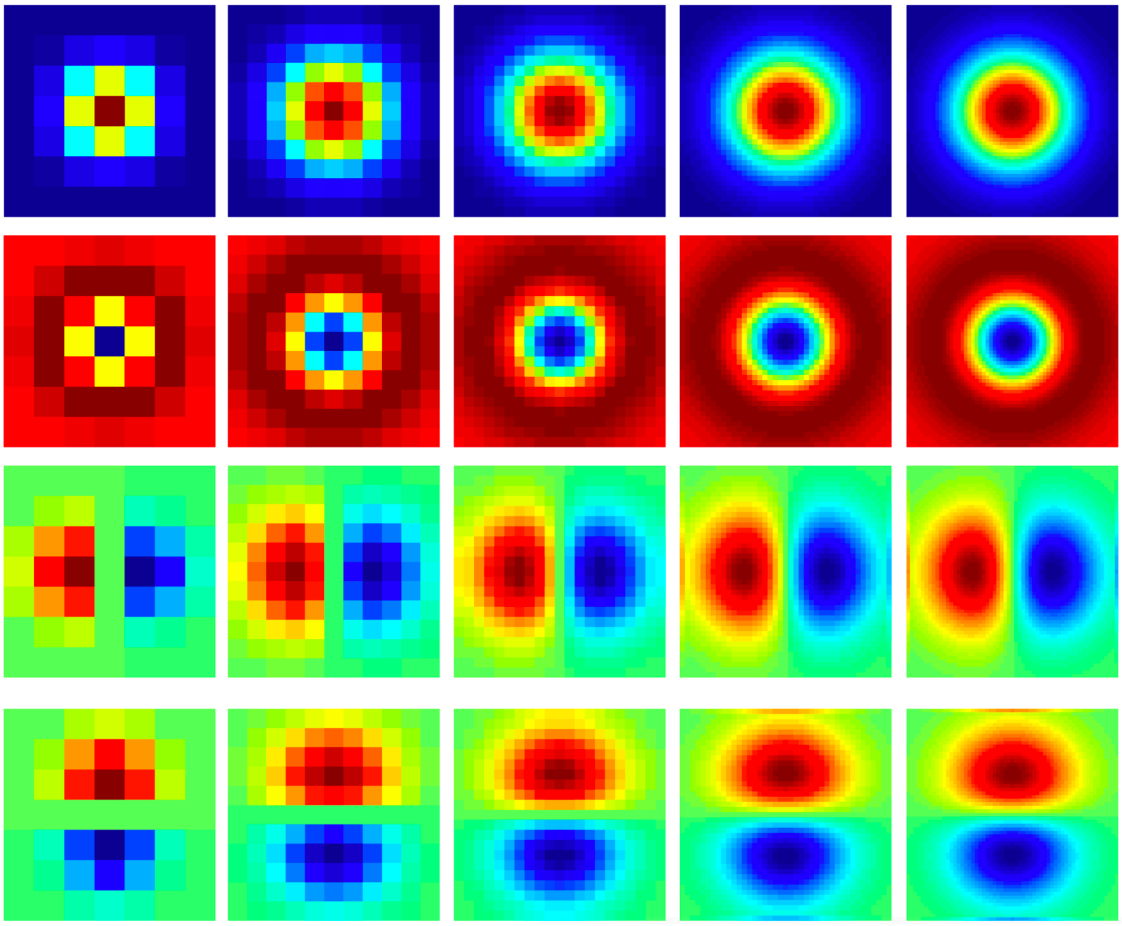
Instructor: Kris Kitani TAs:Paritosh (Lead), Rawal, Yan, Zen, Wen-Hsuan

1.1 Theory Questions

This section should include the visualizations and answers to specifically highlighted questions from P1 to P4. This section will be manually Graded

Q1.1.1 (5 Points WriteUp) What visual properties do each of the filter functions (See Figure below) pick up? You should group the filters into categories by its purpose/functionality. Also, why do we need multiple scales of filter responses? **Answer in the writeup. Answer in your write-up.**

Figure1. The provided multi-scale filter bank



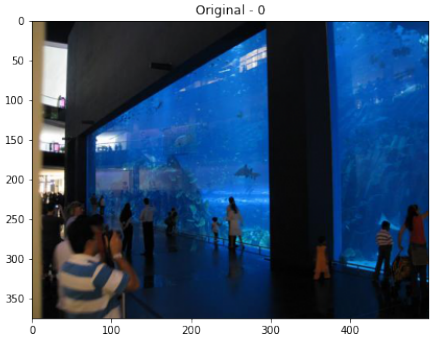
There are four types of filters used. 1 - Gaussian filters - These are low-pass filters that help get rid of the high-frequency components and thus, end up having a smoothening/blurring effect on an image. 2 - Laplacian filters - These help detect edges. These produce a peak at start of the change in intensity and then at the end of the change. 3 - Derivative of Gaussian in x - These help detect intensity changes in the x direction. Thus, help picking up vertical edges. 4 - Derivative of Gaussian in y - These help detect intensity changes in the y direction. Thus, help picking up horizontal edges. Based on the above, the laplacian and derivative filters kind of play a similar role in terms of picking up sharp changes while gaussian is slightly different as it subdues high frequency and noise. As the scale increase, it essentially, in this context, means that the sigma/variance or spread of the filter increases. It is equivalent in practice, to the same filter being applied on a sub-sampled image. With this, the filters would be able to capture visual properties at varied scales.

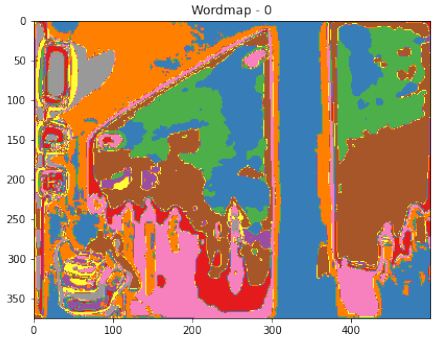
1.1.1 Q1.3.1 (5 Points WriteUp)

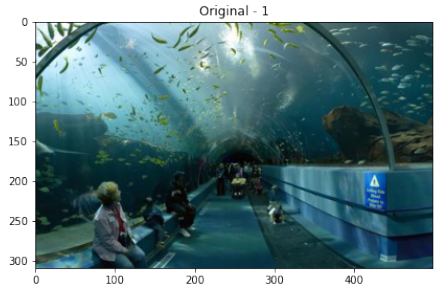
Visualize three wordmaps of images from any one of the category. **Include these in your write-up, along with the original RGB images. Include some comments on these visualizations: do the "word" boundaries make sense to you?**. We have provided helper function to save and visualize the resulting wordmap in the util.py file. They should look similar to the ones in Figure 2.

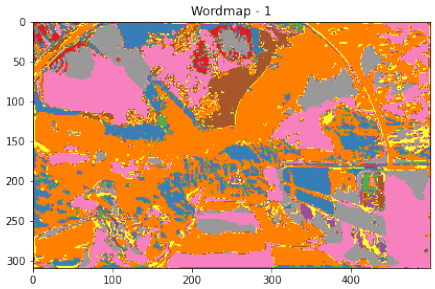
Figure 2. Visual words over images. You will use the spatially un-ordered distribution of visual

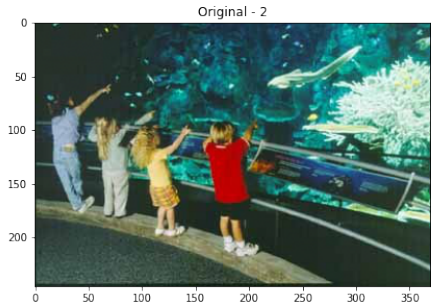
words in a region (a bag of visual words) as a feature for scene classification, with some coarse information provided by spatial pyramid matching [2]











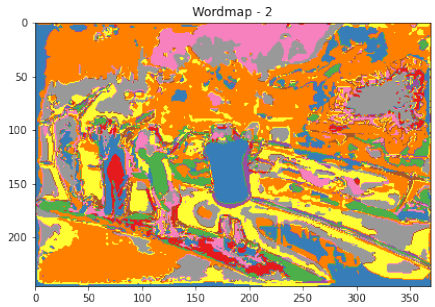
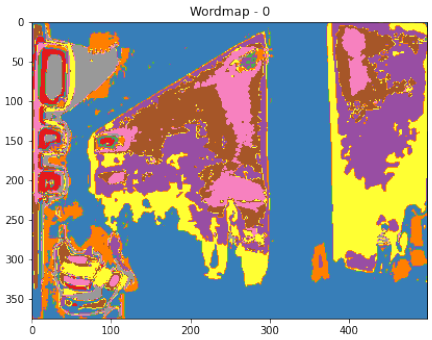
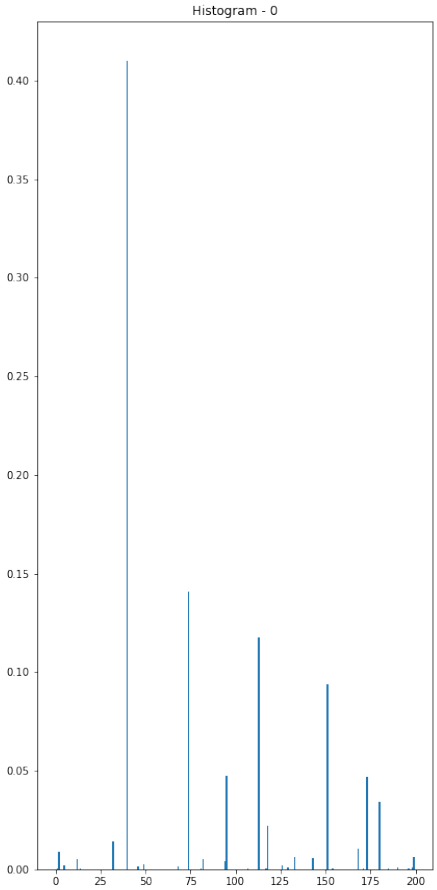


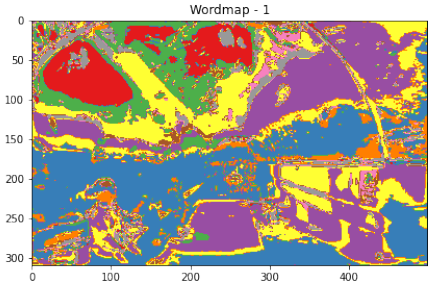
Image pixels with similar intensity and gradients have similar features and thus, all are associated with the same dictionary "word". As the pixel intensities vary, the features around those pixels differ as well, leading to different dictionary words in those regions. This can be seen in the above images as well.

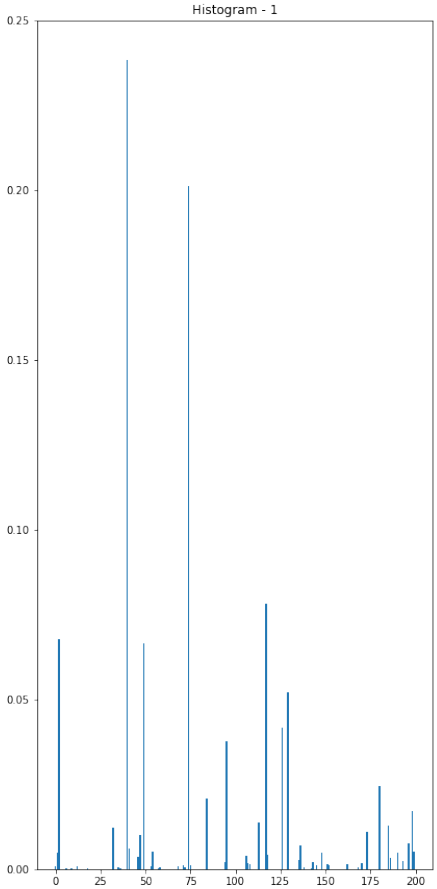
1.1.2 Q2.1

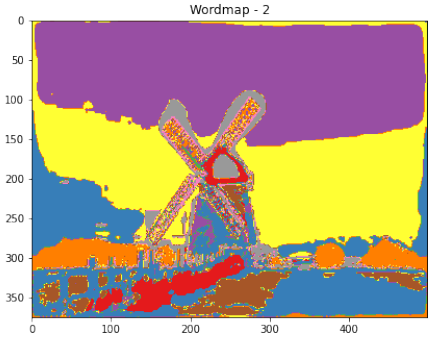
For 5 Images, include their visual word maps and histograms

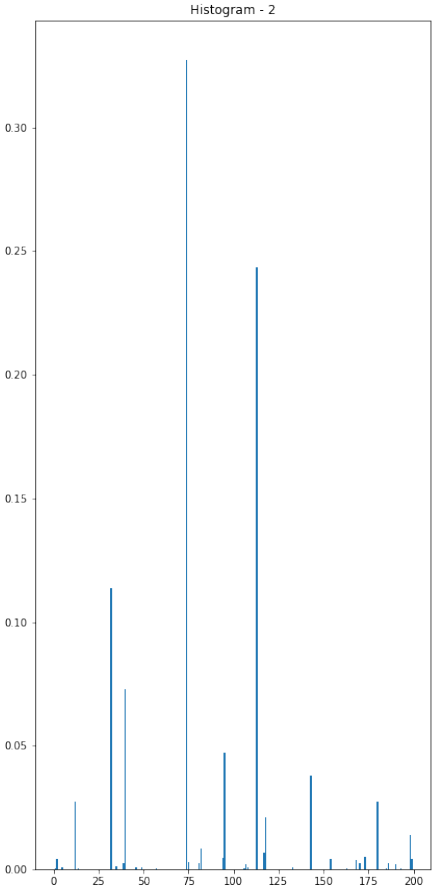


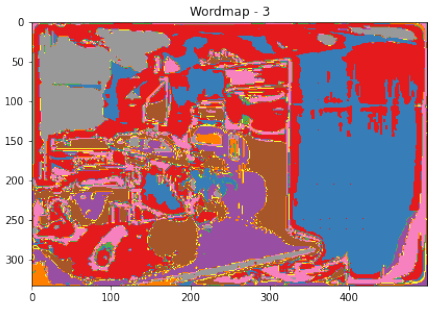


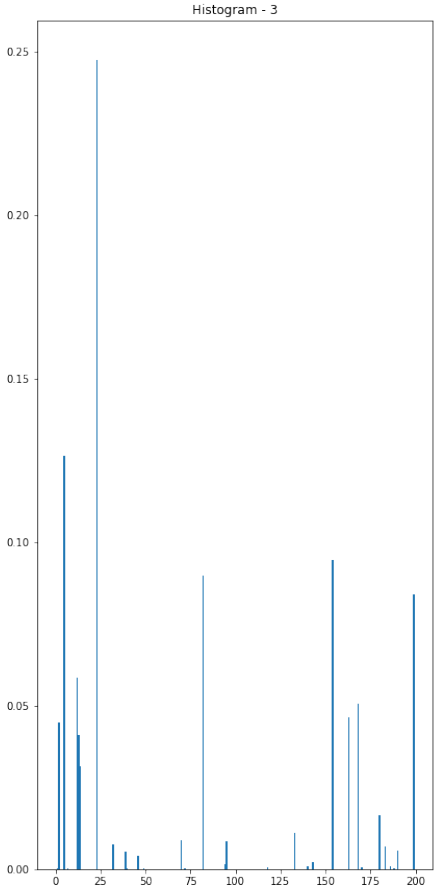

Wordmap - 1
0
50
100
150
200
250
300
0
100
200
300
400

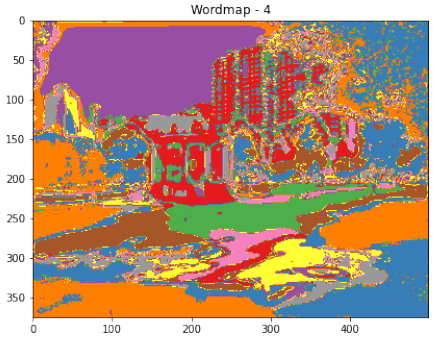

Histogram - 1
0.25
0.20
0.15
0.10
0.05
0.00
0
25
50
75
100
125
150
175
200

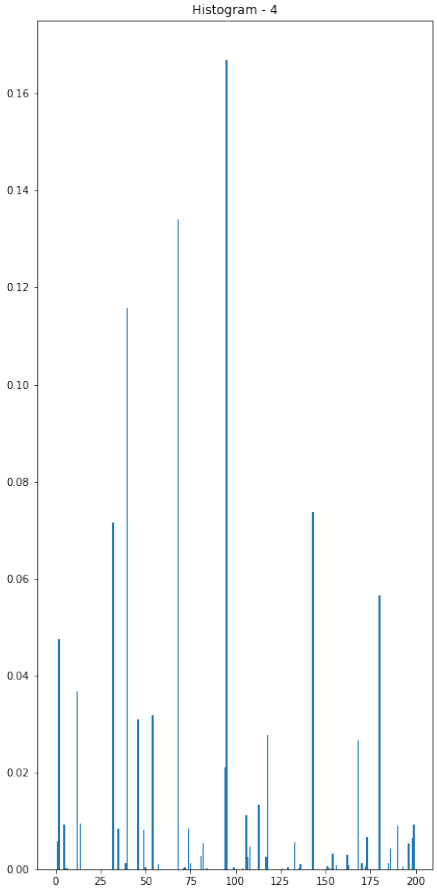

Wordmap - 2
0
50
100
150
200
250
300
350
0
100
200
300
400


Histogram - 2
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
25
50
75
100
125
150
175
200


Wordmap - 3
0
50
100
150
200
250
300
0
100
200
300
400


Histogram - 3
0.25
0.20
0.15
0.10
0.05
0.00
0
25
50
75
100
125
150
175
200



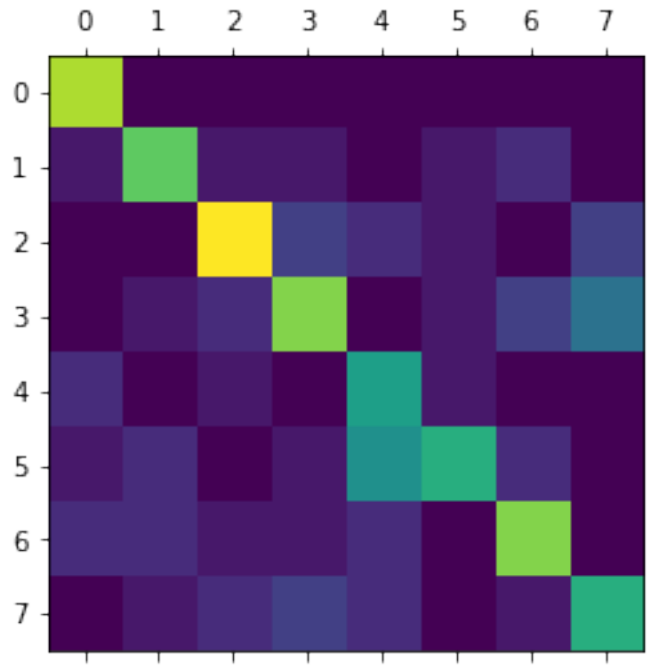


1.1.3 Q3.1.1

Submit the visualization of Confusion Matrix and the Accuracy value

Confusion Matrix -

```
[[14.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
[ 1., 12.,  1.,  1.,  0.,  1.,  2.,  0.],
[ 0.,  0., 16.,  3.,  2.,  1.,  0.,  3.],
[ 0.,  1.,  2., 13.,  0.,  1.,  3.,  6.],
[ 2.,  0.,  1.,  0.,  9.,  1.,  0.,  0.],
[ 1.,  2.,  0.,  1.,  8., 10.,  2.,  0.],
[ 2.,  2.,  1.,  1.,  2.,  0., 13.,  0.],
[ 0.,  1.,  2.,  3.,  2.,  0.,  1., 10.]]
```



Accuracy - 60.625%

Q3.1.2 (5 points WriteUp): As there are some classes/samples that are more difficult to classify than the rest using the bags-of-words approach, they are more easily classified incorrectly into other categories. **List some of these classes/samples and discuss why they are more difficult.**

As can be seen from the above confusion matrix, there are instances where images with true label 3 (highway) are being classified as class 7 (windmill). Similarly, images with true label 5 (laundromat) are being classified as class 4 (kitchen). These classes have quite a bit of similarity - 3,7 are both outdoors with blue skies, 2 lines (highway road, windmill); 4, 5 are both indoors with furniture, counter tops etc. Due to such similarities, it would be hard to distinguish between them, in some cases, based on BoW. Because of such similarities, a test image would end up having high word count for certain features which would be true for training examples of the other class thus, leading to misclassification.

For example, the above laundromat image was misclassified as kitchen. A lot of the kitchen images have similar wood/brown color components and so, the corresponding wordmaps/histogram of this image would have had high similarity with those images leading to the misclassification.

1.1.4 Q3.1.3 Extra Credit Manually Graded:

Now that you have seen how well your recognition system can perform on a set of real images, you can experiment with different ways of improving this baseline system.

Include the changes, modification you made and the impact it had on accuracy.

Tune the system you build to reach around 65% accuracy on the provided test set (`data/test_data.npz`). **In your writeup, document what you did to achieve such performance: (1) what you did, (2) what you expected would happen, and (3) what actually happened.** Also, include a file called `custom.py/ipynb` for running your code.

YOUR ANSWER HERE

1.1.5 Q3.1.4 Extra Credit:

Inverse Document Frequency: With the bag-of-word model, image recognition is similar to classifying a document with words. In document classification, inverse document frequency (IDF) factor is incorporated which diminishes the weight of terms that occur very frequently in the document set. For example, because the term "the" is so common, this will tend to incorrectly emphasize documents which happen to use the word "the" more frequently, without giving enough weight to the more meaningful terms.

In the homework, the histogram we computed only considers the term frequency (TF), i.e. the number of times that word occurs in the word map. Now we want to weight the word by its inverse

document frequency. The IDF of a word is defined as:

$$IDF_w = log \frac{T}{|\{d : w \in d\}|}$$

Here, T is number of all training images, and $|\{d : w \in d\}|$ is the number of images d such that w occurs in that image.

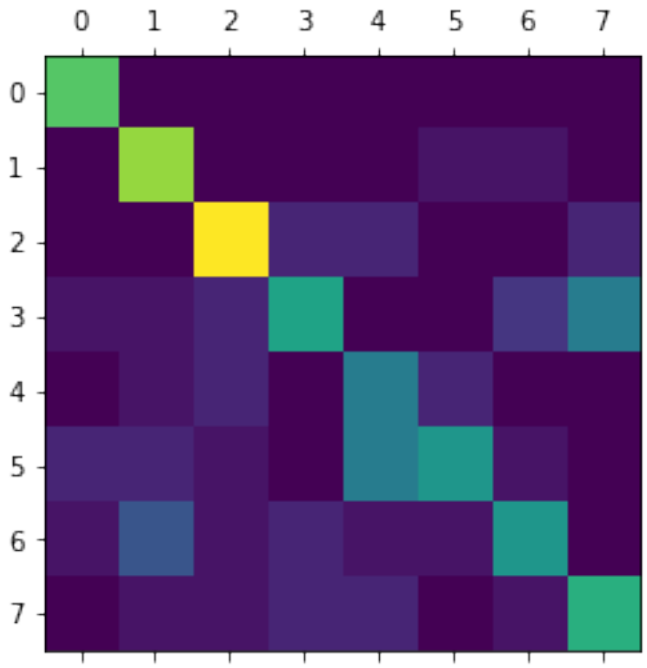
In your writeup: How does Inverse Document Frequency affect the performance? Better or worse? Explain your reasoning?

[ ]:
```
def evaluate_single_idf(args):
    file_path, dictionary, layer_num, K, trained_features, train_labels, idf =␣
 ↪args
    _, feature = get_image_feature(file_path, dictionary, layer_num, K)
    distances = distance_to_set(feature * idf, trained_features * idf)
    pred_label = train_labels[np.argmax(distances)]

    return pred_label

def compute_IDF():
    trained_system = np.load("trained_system.npz")
    trained_features = trained_system['features']
    dictionary = trained_system['dictionary']

    K = dictionary.shape[0]
    T = trained_features.shape[0]

    word_count = np.sum(trained_features != 0, axis = 0)
    idf = np.log(T / word_count)

    np.save('idf.npy', idf)

    return idf

def evaluate_recognition_System_IDF():
    test_data = np.load("./data/test_data.npz")
    trained_system = np.load("trained_system.npz")

    image_names = test_data['files']
    test_labels = test_data['labels']

    trained_features = trained_system['features']
    train_labels = trained_system['labels']

    dictionary = trained_system['dictionary']
```

```
    SPM_layer_num = trained_system['SPM_layer_num']
    SPM_layer_num = int(SPM_layer_num)
    K = dictionary.shape[0]

    print("Trained features shape: ", trained_features.shape)
    print("Test data shape: ", image_names.shape)

    idf = compute_IDF()

    num_images = len(image_names)
    ordered_labels = []

    for i in range(num_images):
        full_image_name = './data/' + image_names[i]
        ordered_labels.append(evaluate_single_idf([full_image_name,
                                                   dictionary,
                                                   3,
                                                   K,
                                                   trained_features,
                                                   train_labels,
                                                   idf.reshape(1, -1)]))


    ordered_labels = np.array(ordered_labels, dtype=int)
    assert ordered_labels.shape[0] == test_labels.shape[0]

    conf_matrix = np.zeros((8, 8))

    for i in range(ordered_labels.shape[0]):
        conf_matrix[test_labels[i], ordered_labels[i]] += 1

    accuracy = np.trace(conf_matrix) / np.sum(conf_matrix)
    np.save("./conf_matrix.npy", conf_matrix)

    return conf_matrix, accuracy
```

Confusion Matrix with IDF -

```
[[14.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
[ 0., 16.,  0.,  0.,  0.,  1.,  1.,  0.],
[ 0.,  0., 19.,  2.,  2.,  0.,  0.,  2.],
[ 1.,  1.,  2., 11.,  0.,  0.,  3.,  8.],
[ 0.,  1.,  2.,  0.,  8.,  2.,  0.,  0.],
[ 2.,  2.,  1.,  0.,  8., 10.,  1.,  0.],
[ 1.,  5.,  1.,  2.,  1.,  1., 10.,  0.],
[ 0.,  1.,  1.,  2.,  2.,  0.,  1., 12.]]
```



Accuracy - 62.5%

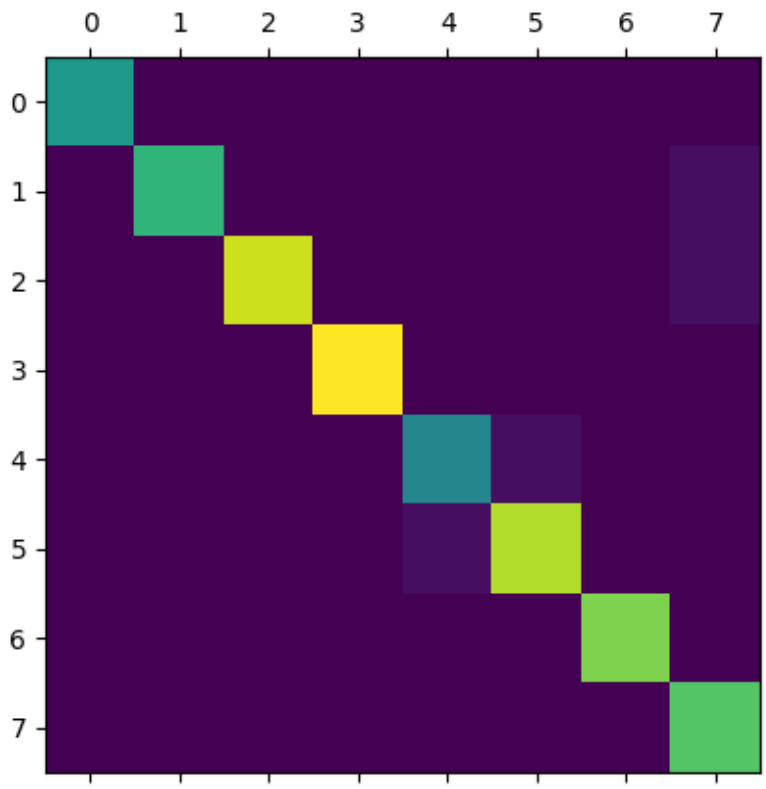
IDF has led to an improvement in performance. IDF helps deprioritize features that occur across a large number of images as they wouldn't be a good factor at actually differentiating images. Thus, by emphasizing features that are rarer or only occur in certain classes, we would be able to classify an image more accurately.

1.1.6 Q4.2.1

Report the confusion matrix and accuracy for your results in your write-up. Can you comment in your writeup on whether the results are better or worse than classical BoW - why do you think that is?

Confusion Matrix -

```
[[14.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
[ 0., 17.,  0.,  0.,  0.,  0.,  0.,  1.],
[ 0.,  0., 24.,  0.,  0.,  0.,  0.,  1.],
[ 0.,  0.,  0., 26.,  0.,  0.,  0.,  0.],
[ 0.,  0.,  0.,  0., 12.,  1.,  0.,  0.],
[ 0.,  0.,  0.,  0.,  1., 23.,  0.,  0.],
[ 0.,  0.,  0.,  0.,  0.,  0., 21.,  0.],
[ 0.,  0.,  0.,  0.,  0.,  0.,  0., 19.]]
```

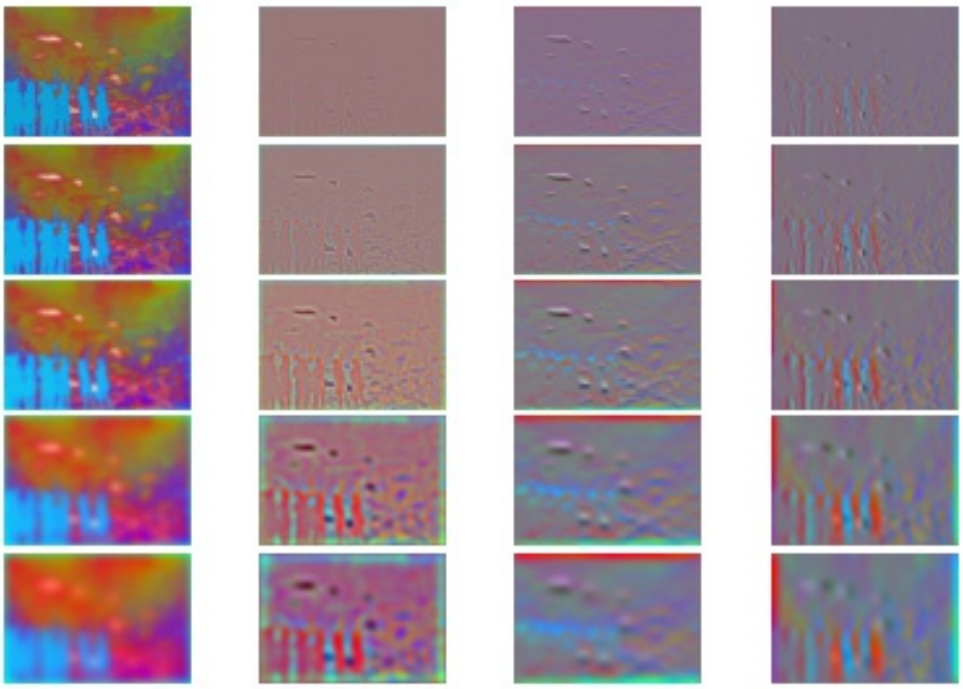


Accuracy - 97.5 %

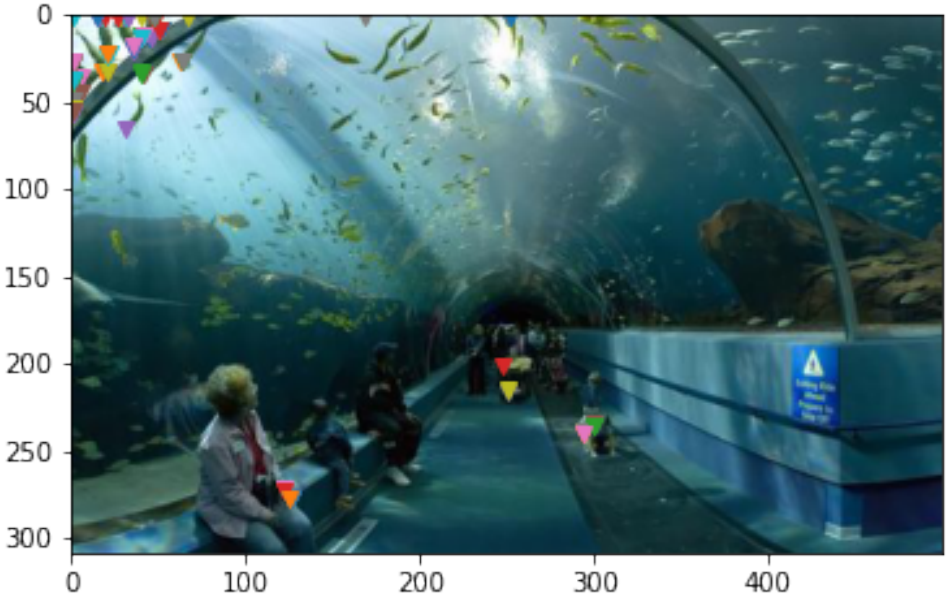
The VGG16 performs much better with an accuracy of 97.5% as compared to the 60.625% of the BoW. I believe this is because the network has learnt effective features for distinguishing on its own rather than the 4 types of features we used with BoW. In case of BoW, we rely on the relation of the frequency of low level features (responses to gaussian, laplacian, DoG in x and y) around keypoints occuring in the image. These can occur in all classes and so, might not be the most reliable when it comes to classifying. However, the network learns these low level features but with every deeper layer, it combines these to get a sense of higher level features as well. Thus, it is able to pick up complex understanding of the scene image and can classify it better.

1.1.7 Q1.1.2 Manually grade the image

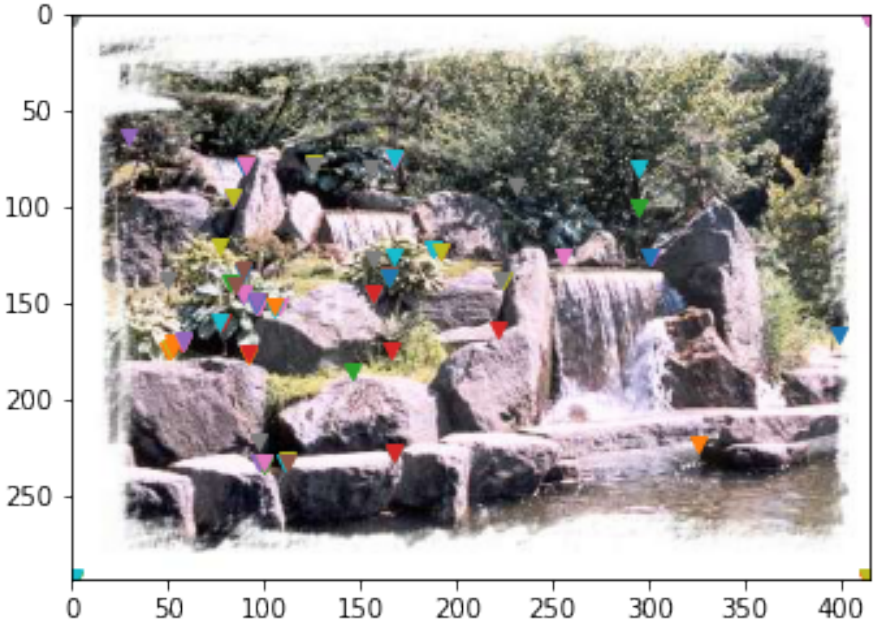
Can use sun_aztvjgubyrgvirup.jpg for visualizing the 20 image collage

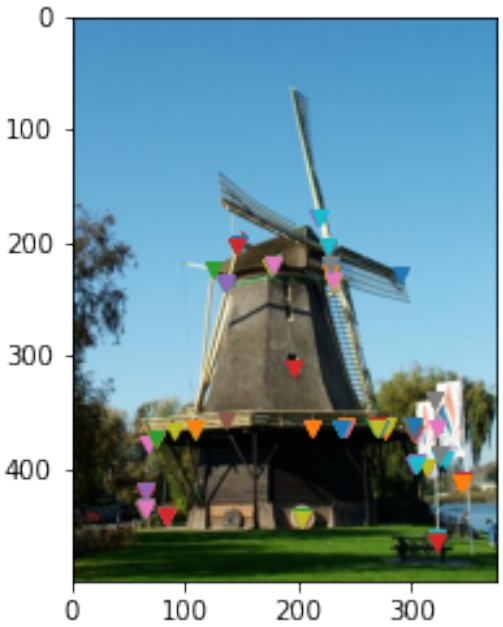


1.1.8 Q1.2.1 Manually grade 5 images for Harris Corner




0
50
100
150
200
250
300
0
100
200
300
400


0
50
100
150
200
250
0
50
100
150
200
250
300
350
400



1.1.9 References

[1] James Hays and Alexei A Efros. Scene completion using millions of photographs.ACM Transactions onGraphics (SIGGRAPH 2007), 26(3), 2007.

[2] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recogniz-ing natural scene categories. InComputer Vision and Pattern Recognition (CVPR), 2006 IEEE Conferenceon, volume 2, pages 2169–2178, 2006.

[3] Jian xiong Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recogni-tion from abbey to zoo.2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition,pages 3485–3492, 2010.14

[ ]: