16-720B Computer Vision: Homework 1

Spatial Pyramid Matching for Scene Classification

Due: September 26 at 11:59pm

[redacted] (AndrewID: [redacted])

1 Representing the World with Visual Words

1.1 Extracting Filter Responses

Q1.1.1: What properties do each of the filter functions pick up? You should group the filters into broad categories (e.g. all the Gaussians). Also, why do we need multiple scales of filter responses?

Solution: Gaussian filters are used for reducing the noises and blurring the image but still preserve high frequency edges. It picks up the information in colors. Laplacian of Gaussian filters can detect all the edges in the image. Filters with derivative of Gaussian in the x direction help to detect the vertical edges and derivative of Gaussian in the y direction is for horizontal edges detection. The reason why we need multiple scales of filter responses is that we don't know the scale of targets in the images. With different scales of targets, we need to used different scales of filters to better pick up the features accordingly. With multiple scales of filters, it is more likely to have the better filter for objects with multiple scales, which can help to represent the feature better.

Q1.1.2: Function

```
visual_words.extract_filter_responses(image)
```

completed. It returns responses as `filter_responses`. The visualized responses are shown as Fig. 1.

1.2 Creating Visual Words

Q1.2: Functions

```
visual_words.compute_dictionary_one_image(args)
visual_words.compute_dictionary()
```

completed to generate the dictionary. File `dictionary.npy` that contains the dictionary of visual words is saved. The saved dictionary is generated with $K = 100$ and $\alpha = 50$.

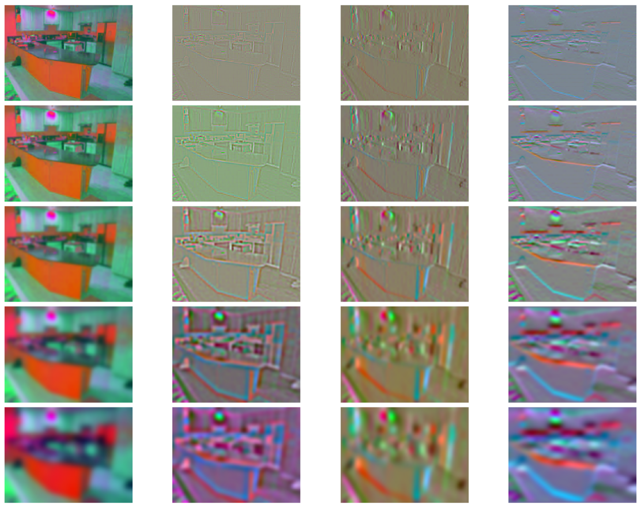

Figure 1: The filter responses of the input data

1.3 Computing Visual Words

Q1.3: Function

`visual_words.get_visual_words(image, dictionary)`

completed to generate the `worldmap`. Three worldmaps with their original RGB images are shown as Fig. 2:

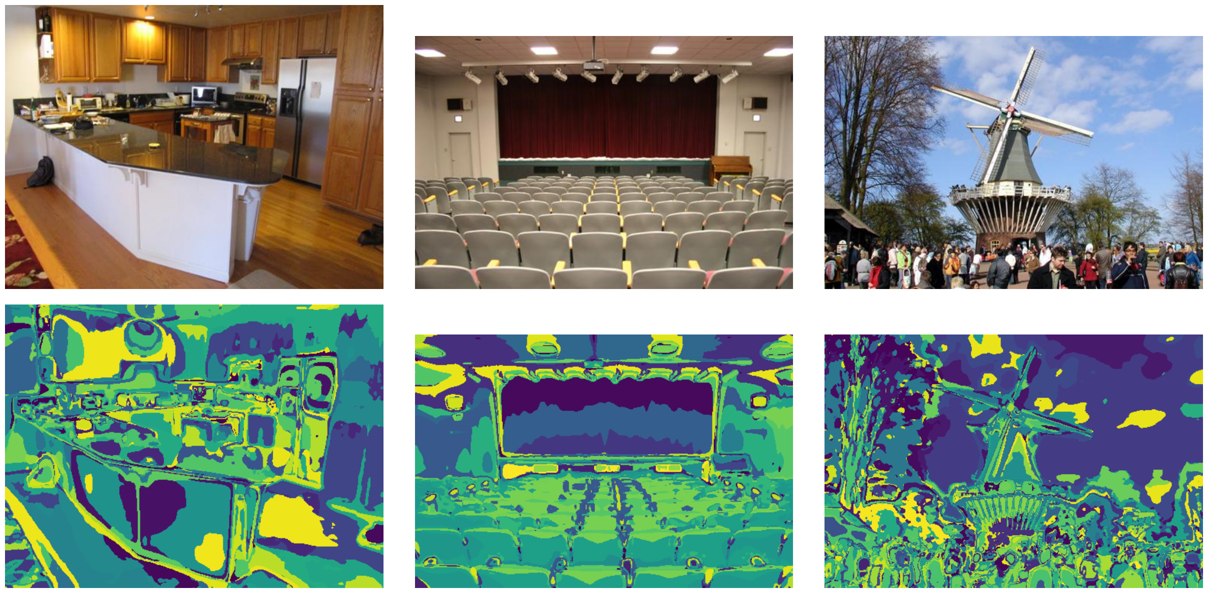

Figure 2: Three visual words with the original RGB images.

2 Building a Recognition System

2.1 Extracting Features

Q2.1: Function

`visual_recog.get_feature_from_wordmap(wordmap,dict_size)`

completed to get the histogram feature given a wordmap. The L_1 normalized histogram `hist` is returned.

2.2 Multi-resolution: Spatial Pyramid Matching

Q2.2: Function

`visual_recog.get_feature_from_wordmap_SPM(wordmap, layer_num, dict_size)`

completed to get multi-resolution representation for an image. The L_1 normalized `hist_all` is returned. In the function, 3-layer spatial pyramid ($L = 2$) is used.

2.3 Comparing images

Q2.3: Function

`visual_recog.distance_to_set(word_hist,histograms)`

completed calculate the similarity between the input feature `word_hist` and the reference training samples.

2.4 Building A Model of the Visual World

Q2.4: Function

`visual_recog.build_recognition_system()`

completed to generate file `trained_system.npz` that contains all information in the training. Helper functions

`visual_recog.get_image_feature(file_path,dictionary,layer num,K)`

is implemented to load the image from `file_path`, extract the word map and compute the SPM feature.

`visual_recog.get_feature_for_one_image(args)`

is implemented for parallel computation.

2.5 Quantitative Evaluation

Q2.5: Function

`visual_recog.evaluate_recognition_system()`

is implemented to test the recognition system. The returned confusion matrix is shown as below:

$$\begin{bmatrix} 9. & 0. & 2. & 0. & 3. & 3. & 3. & 0. \\ 1. & 10. & 2. & 1. & 0. & 0. & 0. & 6. \\ 2. & 0. & 11. & 1. & 1. & 1. & 0. & 4. \\ 0. & 1. & 0. & 13. & 0. & 0. & 0. & 6. \\ 5. & 0. & 1. & 0. & 10. & 2. & 2. & 0. \\ 3. & 0. & 0. & 0. & 5. & 11. & 1. & 0. \\ 1. & 0. & 0. & 2. & 0. & 4. & 13. & 0. \\ 0. & 1. & 0. & 6. & 0. & 0. & 2. & 11. \end{bmatrix}$$

The accuracy is **0.55**

The visualized confusion matrix is shown as Fig. 3:

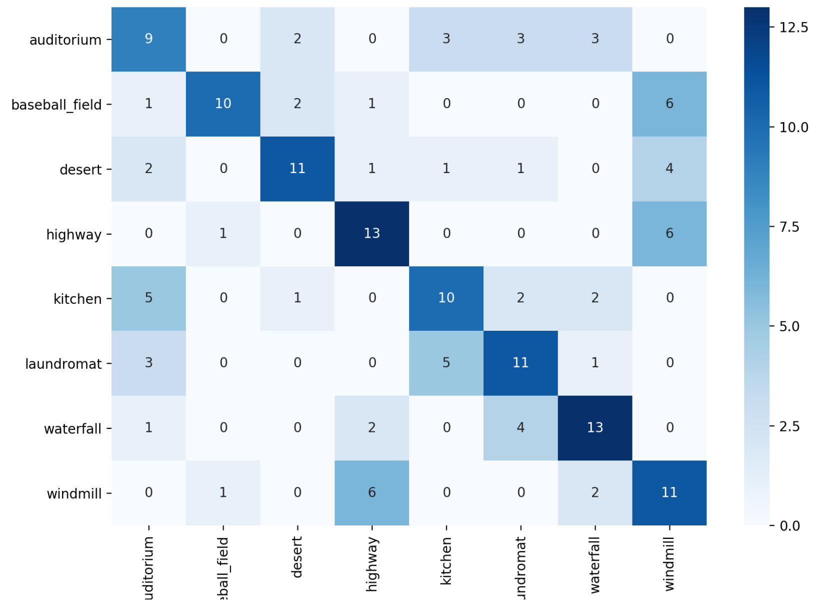

Figure 3: Confusion matrix of the recognition system.

2.6 Find the failed cases

Q2.6: We can see from the confusion matrix that many "baseball_field" are mis-classified as "windmill", many "highway" are mis-classified as "windmill" and many "windmill" are classified as "highway". Fig. 4 provides an example of the mis-classification for each situation mentioned.

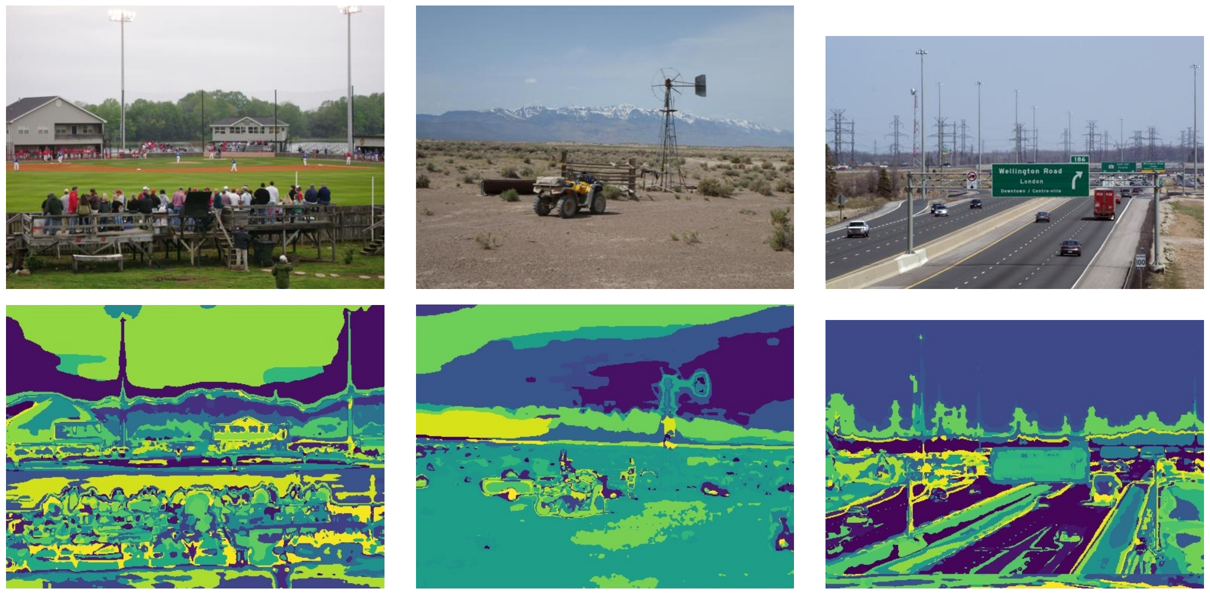

Figure 4: Three visual words with the original RGB images.

The first column is an example that "baseball_field" is classified as "windmill" and the third column is that "highway" is classified as "windmill". They both share similar features with "windmill" that they all have tall and windmill-like buildings, which might make it hard to classify them correctly. The second column is an example that "windmill" is classified as "highway". In this case, the windmill in the image is quite small while the vehicle is large and the whole background looks like a wide road. The similarity between these images might cause the mis-classification.

3 Deep Learning Features - An Alternative to "Bag of Words"

3.1 Extracting Deep Features

Q3.1: Function

`network_layers.extract_deep_feature(x, vgg16_weights)`

is completed to extract the deep features of the input image `x` given the VGG16 network weights `vgg16_weights`. To complete this, four sub-functions: `multichannel_conv2d(x, weight,bias)`, `relu(x)`, `max pool2d(x,size)` and `linear(x,W,b)` are completed for multiple channel convolution calculation, performing Rectified Linear Unit, performing max pooling and computing linear combination respectively. The obtained deep feature is a 4096-dimensional vector. The output is same as the output using pytorch.

3.2 Building a Visual Recognition System: Revisited

Q3.2: Function

```
deep_recog.build_recognition_system(vgg16)
```

is completed to build the recognition system with the deep features. This function generates file `trained_system_deep.npz`, which contains information about the training features and labels.

Function

```
deep_recog.eval_recognition_system(vgg16)
```

is completed to evaluate the recognition system. The returned matrix is shown as below:

$$\begin{bmatrix} 20. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 1. & 17. & 1. & 0. & 0. & 0. & 1. & 0. \\ 0. & 0. & 18. & 2. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 20. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 18. & 2. & 0. & 0. \\ 0. & 0. & 0. & 0. & 2. & 18. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 20. & 0. \\ 0. & 0. & 1. & 1. & 0. & 0. & 0. & 18. \end{bmatrix}$$

The accuracy is **0.93125**

The visualized confusion matrix is shown as Fig. 5:

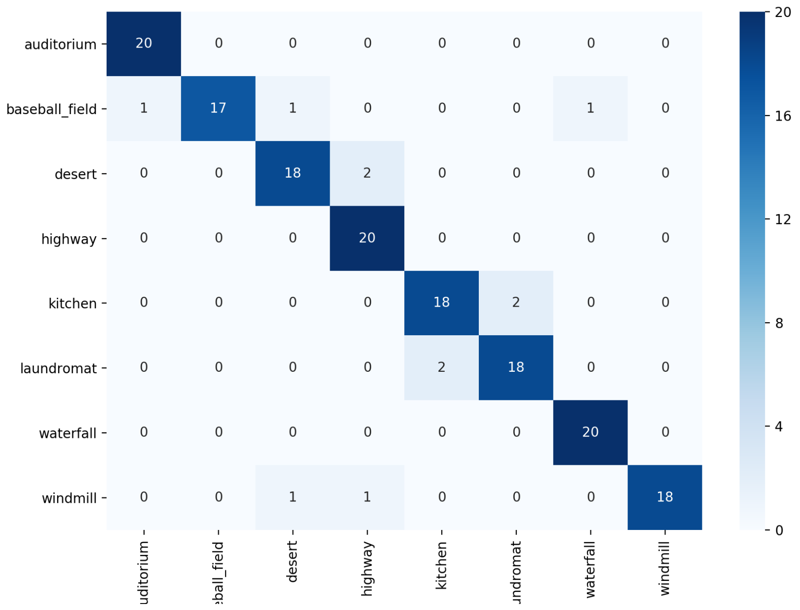

Figure 5: Confusion matrix of the visual recognition system with deep features.

The result is better than classical BoW. The reason should be that the weights of VGG network are trained based on many more training images, which contain the eight types of image labels we used here. After the pre-training, the network can help to represent these images very well with more information in texture, color etc., since it is trained with more data and more layers .