Assignment 2

TDS3651 Visual Information Processing

Out: **January 13, 2017** (Week 9)
Due: **February 14, 2017** (Week 14)

1 Introduction

1.1 Objective

To design suitable features and methods to improve the performance of a food image retrieval system. A large database comprising of 1,000 food images belonging to 10 popular Malaysian food is provided for this task. The food categories are '**Ais Kacang**', '**Banana Leaf Rice**', '**Cendol**', '**Curry Laksa**', '**Durian**', '**Maggi Goreng**', '**Nasi Lemak**', '**Pisang Goreng**', '**Roti Canai**' and '**Satay**'. Figure 1 shows some sample images from the database.

Figure 1: Sample images from the food database

Generally, an image retrieval system is a computer system for browsing, searching and retrieving images from a large database of digital images. More specifically, the retrieval of images can be done without text/keywords by using an image as query instead. This is sometimes known as *content-based image retrieval*, where information derived from image content is used to retrieve visually similar images.

1.2 Guidelines

- This is a group assignment, with a maximum of 2 persons in each group.
- In regards to the objective of this assignment, you are **NOT** allowed to use third-party packages to assist you in this problem. Packages '`pip`'-ed from Python Package Index (PyPI) are acceptable.
  - **Important:** Do not upload your own packages or algorithms that you have created for this assignment to public repositories such as Github/BitBucket, which may allow others to "use" portions of these packages to solve the same assignment. (You can do so after the assignment is over). All assignments will be scrutinized in this aspect.

- You can use Spyder (which comes with iPython console), but you can use your own favourite IDE or opt for basic text editor & command line.
- This assignment is worth 20% of coursework marks.
- Late-Day policy applies.
- Submission deadline: January 14, 2017, 11.59PM (Tuesday of Week 14). There will be 3 extra days before the hard deadline.

2 Data

A large database of 1,000 Malaysian food images is provided to prototype a food image retrieval system. There are 10 food categories in total, with 100 images per category. All images have a fixed square dimension of 640 × 640 pixels. Figure 1 shows a sample image from each category of the database.

The images in this database were extracted from Instagram under full anonymity (no knowledge of who took these photos). Also, no associated meta-information (i.e. hash-tags, geo-tags, user IDs, filters used) are available in this database. Hence, you are to only rely on visual information alone.

Table 1 shows the abbreviations used in the scripts that are used to indicate the categories, and also the range of image numbers associated with each category.

Abbreviations	**Category**	**Image numbers in database**
AK	Ais Kacang	0 – 99
BL	Banana Leaf Rice	100 – 199
CD	Cendol	200 – 299
CL	Curry Laksa	300 – 399
DR	Durian	400 – 499
MG	Maggi Goreng	500 – 599
NL	Nasi Lemak	600 – 699
PG	Pisang Goreng	700 – 799
RC	Roti Canai	800 – 899
ST	Satay	900 – 999

Table 1: Food categories in the database, with their abbreviations and corresponding image numbers

Unlike the previous assignment, there are no pickle data files for the images, since all the images are provided in the `fooddb` folder. Each image is labeled with a unique number, *e.g.* `'5.jpg'` is an image of Ais Kacang, `'423.jpg'` is an image of Durian.

Besides, another 4 new **query images** (not from the database) are provided to further validate the performance of your system (see Figure 2).

Figure 2: Four new query images that are not from the food database

Do not tamper with the information in these data files or manually alter the images. All your submitted codes will be re-run with the original data!

3 Performance Measures

A retrieval or search system should be evaluated for its performance. For that, we need performance measures, or *metrics*. The following metrics come together with your evaluation scripts.

A standard metric used to measure the performance on a single query image q is the

$$\text{Average Precision (AP)}_q = \frac{1}{m}\sum_{k=1}^{K}\frac{r_k}{k} \tag{1}$$

where k is the number of retrieved images and r_k is the number of relevant images (correct category) retrieved from among the k images. When all Q images in the database are to be evaluated to obtain a single performance value, we use

$$\text{Mean Average Precision (MAP)} = \frac{1}{Q}\sum_{q=1}^{Q}(\text{AP})_q \tag{2}$$

This MAP metric can also be determined for each category as well, by simply constraining the calculation of MAP to only images of the same category. This gives us a better idea of the performance of each category.

Another metric, recall rate, measures the accuracy of retrieving images of the correct category, given the total number of images retrieved,

$$\text{Recall rate} = \frac{1}{Q}\sum_{q=1}^{Q}\frac{r_k}{R} \tag{3}$$

3.1 Benchmarking with Precision-Recall curve

By varying the number of images retrieved R, we can get pairs of precision and recall values, which can be plotted to give a precision-recall (PR) curve. A PR curve is a standard benchmarking tool for comparing the performance of different methods. You are to write a new script or function that can plot the PR curves.

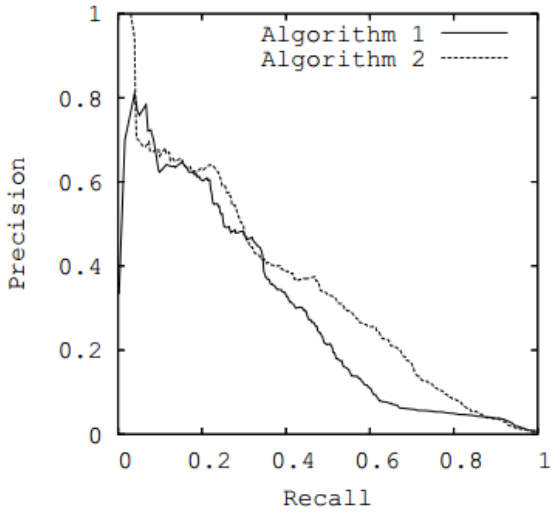

Figure 3: A sample Precision-Recall (PR) curve which can be used to compare the performance of different algorithms

4 Scripts and Functions

4.1 Codes to Write

There are two working functions – `computeFeatures` and `computeDistances`, that you need to write.

4.1.1 `computeFeatures()`

Input:

- `img`: 3-D array of an RGB color image. You may safely assume that the color image read by OpenCV in BGR ordering has been converted to RGB (see the calling scripts).

Output:

- `featvect`: A D-dimensional vector of the input image `img`.

Function `computeFeatures` is called by the `featureExtraction.py` and `queryEval.py` scripts.

4.1.2 `computeDistances()`

Input:

- `fv`: A $N \times D$ array containing D-dimensional feature vector of N number of data (images)

Output:

- `D`: $N \times N$ square matrix containing the pairwise distances between all samples, *i.e.* the first row shows the distance between the first sample and all other samples (columns).

Function `computeDistances` is called by the `fullEval.py` and `queryEval.py` scripts.

4.2 Scripts Provided

There are three functions in total – one feature extraction script and two evaluation scripts, provided for you to test the system:

- `featureExtraction.py`: Perform feature extraction on the entire database of images. The extracted features will be pickled to `'feat.pkl'` while the total time taken is also shown.
- `fullEval.py`: Run a full evaluation on the entire database. It returns the mean average precision (MAP) and recall rates, and displays the MAP per-category plot.
- `queryEval.py`: Run a query-based evaluation using a single image as query. It returns the average precision (AP) and recall rate, and displays the top 10 closest matches.

All functions are runnable on Anaconda Prompt or standard command-line prompt (if necessary path settings have been configured). You can use the `'-h'` switch to get further help on how to use these functions, and what other options are there.

4.2.1 Example of Usages

This command performs feature extraction on the entire database:

```
>> python featureExtraction.py
```

To perform a full evaluation on the database (this run uses the default baseline features). By default, the number of images retrieved is 50:

```
>> python fullEval.py

Features loaded
Mean Average Precision, MAP@50: 0.1035
Recall Rate@50: 0.0800
```

For better convenience, you may decide to run feature extraction first, followed by a full evaluation. For that you can do this:

```
>> python fullEval.py -t
```

To retrieve a different number of images,

```
>> python fullEval.py -r 150

Features loaded
Mean Average Precision, MAP@150: 0.0472
Recall Rate@150: 0.2600
```

To run a retrieval based on a single query image, that is chosen from among the database images, specify the image number. In this case, image `'100.jpg'` is selected as the query image (see Figure 4).

```
>> python queryEval.py -d 100

Features loaded
Query image: BL
Average Precision, AP@50: 0.1814
Recall Rate@50: 0.2800
```

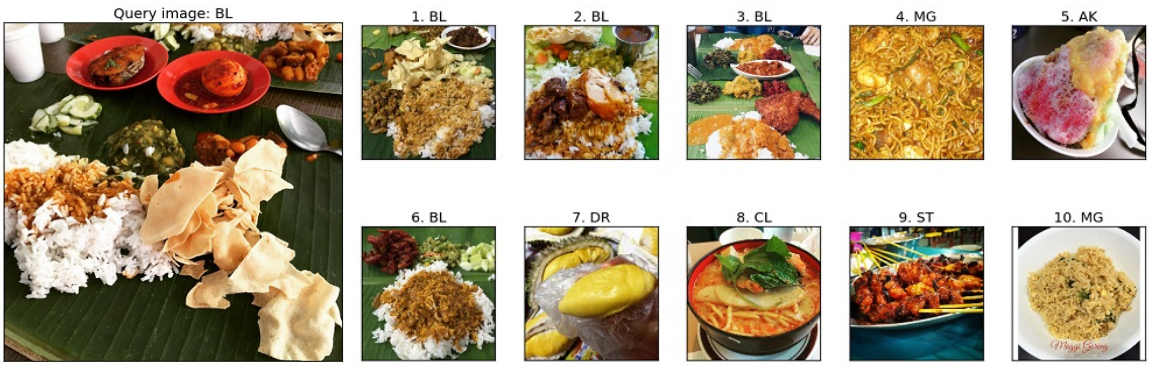

Figure 4: A sample query image (BL) and the top 10 retrieved results using the default method.

To run a retrieval using an external query image (not from the database), specify the file name:

```
>> python queryEval.py -q q_ST.jpg

Features loaded
Query image: ST
Average Precision, AP@50: 0.0997
Recall Rate@50: 0.1400
```

Alternatively, you can run these evaluation scripts from iPython console as well, using the `runfile` command. E.g.

```
In [1]: runfile('queryEval.py', args='-q q_ST.jpg');
```

5 Submission

Submit the following in a ZIP file via MMLS Assignment page:

- **Code**:
  - `computeFeatures.py`
  - `computeDistances.py`
  - Script\function for plotting PR curves, and all other additional supporting codes.
- **Report (in PDF)**: Proposed outline:
  - Abstract (short)
  - Introduction (short)
  - Description of Methods Used
  - Results & Analysis
  - Suggestions for Improvement

Please **do not** submit anything in hardcopy (report) or in stored media (CD/DVD) form.

5.1 Mark Distribution

The following table shows the mark distribution for this assignment:

Code (10%)	Methods Used	5
	Creativity/Originality in Solution	2
	Visualization: Precision-Recall Curves	2
	Clarity/Readability	1
Report (10%)	Abstract (short) & Introduction	2
	Description of Methods	3
	Results & Analysis	4
	Suggestions for Improvement	1
Bonus (max. 2%) – for exceptional achievement in retrieval performance		

End of Assignment 2 Guideline
John See, 2017