Coursework 2 (Group) – Scene Recognition

Brief

This is a group coursework: please work in teams of four people.
Due date: Wednesday 7th January, 16:00.
Development data download: training.zip in the coursework (CW) folder
Testing data download: testing.zip in the CW folder
Handin: https://handin.ecs.soton.ac.uk/handin/2526/COMP3204/2/
Required files: report.pdf; code.zip; run1.txt; run2.txt; run3.txt
Credit: 25% of overall module mark

Overview

The goal of this project is to introduce you to image recognition. Specifically, we will examine the task of scene recognition starting with very simple methods -- tiny images and nearest neighbour classification -- and then move on to techniques that resemble the state-of-the-art.

This coursework will run following the methodology used in many current scientific benchmarking competitions/evaluations. You will be provided with a set of labelled development images from which you are allowed to develop and tune your classifiers. You will also be provided with a set of unlabelled images for which you will be asked to produce predictions of the correct class.

Details

You will need to write software that classifies scenes into one of 15 categories. We want you to implement three different classifiers as described below. You will then need to run each classifier against all the test images and provide a prediction of the class for each image.

Data

The training data consists of 100 images for each of the 15 scene classes. These are arranged in directories named according to the class name. The test data consists of 2985 images. All the images are provided in JPEG format. All the images are grey-scale, so you don't need to consider colour.

Objective measure

The key classification performance indicator for this task is *average precision*; this is literally the proportion of number of correct classifications to the total number of predictions (i.e. 2985).

Run conditions

As mentioned above, you need to develop and run three different classifiers. We'll refer to the application of a classifier to the test data as a "run".

Run #1: You should develop a simple *k*-nearest-neighbour classifier using the "tiny image" feature. The "tiny image" feature is one of the simplest possible image representations. One simply crops each image to a square about the centre, and then resizes it to a small, fixed resolution (we recommend 16x16). The pixel values can be packed into a vector by concatenating each image row. It tends to work slightly better if the tiny image is made to have zero mean and unit length. You can choose the optimal *k*-value for the classifier.

Run #2: You should develop a set of linear classifiers (an ensemble of 15 one-vs-all classifiers) using a bag-of-visual-words feature based on fixed size densely-sampled pixel patches. We recommend that you start with 8x8 patches, sampled every 4 pixels in the x and y directions. A sample of these should be clustered using *K*-Means to learn a vocabulary (try ~500 clusters to start). You might want to consider mean-centring and normalising each patch before clustering/quantisation. **Note:** we're not asking you to use SIFT features here - just take the pixels from the patches and flatten them into a vector & then use vector quantisation to map each patch to a visual word.

Run #3: You should try to develop the best classifiers you can! You can choose whatever feature, encoding and classifier you like. Potential features: the GIST feature; Dense SIFT; Dense SIFT in a Gaussian Pyramid; Dense SIFT with spatial pooling (commonly known as *PHOW* - Pyramid Histogram of Words), etc. Potential classifiers: Naive bayes; non-linear SVM (perhaps using a linear classifier with a Homogeneous Kernel Map), ...

Run prediction format

The predictions for each run must be written to a text file named `runX.txt` (where X is the run number) with the following format:

```
<image_name> <predicted_class>
<image_name> <predicted_class>
<image_name> <predicted_class>
...
```

For example:

```
0.jpg tallbuilding
1.jpg forest
2.jpg mountain
3.jpg store
4.jpg store
5.jpg bedroom
...
```

Restrictions

- You are not allowed to use the testing images for anything other than producing the final predictions **They must not be used for either training or learning feature encoding.**

The report

The report must be no longer than 4 sides of A4 with the given Latex format for CW2, and must be submitted electronically as a PDF. The report must include:

- The names and ECS user IDs of the team members
- A description of the implementation of the classifiers for the three runs, including information on how they were trained and tuned, and the specific parameters used for configuring the feature extractors and classifiers. We expect that your "run 3" section will be considerably longer than the descriptions of runs 1 & 2.
- A short statement detailing the individual contributions of the team members to the coursework.

What to hand in

You need to submit to ECS Handin the following items:

- The group report (as a PDF document in the CVPR format same as CW2; max 4 A4 sides, no appendix)
- Your code enclosed in a zip file (including everything required to build/run your software and to train and use your classifiers; please don't include binaries or any of the images!)
- The run prediction files for your three runs (named "run1.txt", "run2.txt" and "run3.txt").
- A plain text file listing the user ids (e.g. xx1g20) of the members of your team; one per line.

Marking and feedback

Marks will be awarded for:

- Successful completion of the task.
- Well structured and commented code.
- Evidence of professionalism in implementation and reporting.
- Quality and contents of the report.
- The quality/soundness/complexity of approach used for run 3.

Marks will not be based on the actual performance of your approach (although you can expect to lose marks if runs 1 and 2 are way off our expectations or you fail to follow the submission instructions). We will open the performance rankings for run 3. 😊

Standard ECS late submission penalties apply.

Individual feedback will be given to each team covering the above points. We will also give overall feedback on the approaches taken in class when we announce the winner!

Useful links

- **Matlab**
  - Image processing toolbox tutorials
  - **Recommended:** VLFeat
    - Example of using VLFeat to perform classification
  - Linear and non-linear SVMs
- **Python**
  - numpy, PIL, sklearn (Scikit-learn), OpenCV, etc.
- **C and C++**
  - OpenCV
  - **Recommended:** VLFeat
  - Example of using VLFeat to perform classification (Note this code is Matlab, but most of the functionality is available in the C/C++ API)
- **Java**
  - **Recommended:** OpenIMAJ
    - Chapter 12 of the tutorial deals with image classification
  - BoofCV

Questions

If you have any problems/questions, use the Q&A channel on Teams