**COMP3204 Computer Vision**

**Coursework 3: Scene Recognition**

**Team 33:**

cc14g17 Callum Connolly

khl1g17 Kwok Hung Liu

pd2g17 Peter Deng

Run #1: K-Nearest Neighbour and Tiny Image

Firstly, we identified the classes which were needed for the code, the Main class (App.java), the K-Nearest Neighbour Classifier class (KNNClassifier.java) and a Tiny Image class (TinyImage.java).

App.java handles the data. While the training dataset is downloaded from the specification straight away using VFSGroupDataset, the test data are downloaded straight into the /data directory because the output of each image must include its respective file name which cannot be done with VFSGroupDataset (or at least we did not managed to do so). After downloading the data, the set “training” is removed as it is the parent folder which contains all the folders for the different categories. The training dataset was then fed into an instance of the KNNClassifier class along the size of the tiny images (size) using the function addTrainingData (VFSGroupDataset, int).

In the KNNClassifier class, objects of TinyImage are created from the images from the dataset and a HashMap of vector values (FloatFV) and the category names (String) are stored.

We sorted the files in the training folder so the output will be in numeric order. We classify each image using the instance of the classifier class, turning them into a square tiny image before calculating a vector value. The vector value will then be used to be compared to the values stored in the HashSet (categories) where the difference will be stored in a priority queue. The first k-nearest values will then be used to retrieve the original FloatFV values which are the keys of the HashSet. Using the keys retrieved, a majority voting process will take place to determine the likely category of each image. If there are more than one category have the same number of votes, the output will join them with a keyword “and”.

Finally, the output will be processed and stored in a text file named “run1.txt”.

The time taken for run #1 to execute is about 30 seconds to a minute.

Run #2: Bag of Visual Words Classifier

When we first approached run #2, we tried to do it in a similar fashion to run #1 which did not work out in our favour as we realised specific methods and classes must be used. As a result, we made it in a similar style to the Tutorial of chapter 12.

Run3 – Jackey’s magical bag of stupid shit

**they were trained like this…. and tuned like this….and the specific parameters used for configuring the feature extractors and classifiers were…**

Contributions

Jackey- coding and testing of runs 2 and 3, editing and proofreading of report

Peter-Testing of run 2 and 3, contribution to report

Callum- coding and testing of run 1, organization of source control, editing and proofreading of report