Report

Team members:
Yang Zhang(zhang505), Ayoub Lassoued(alassoue), Shashant Devadiga(ssdevadi)

1. Baseline:
    - all the processed data will be store in the baseline/ folder
    - we grey scale the image and divide the value by 255 (we get better result and train faster when do this)
    - result: accuracy: 30 of 250 = 12%
    - It did not improve much when use color.

2. Eigenfood:
    - Resized the image and extracted features using SVD decomposition
    - Used top 10 Eigen vectors
    - Accuracy: 13 of 250 = 5.2%
    - Did not implement what the top k Eigen vectors look like

3. Haar-like features:
    - 1500 features consisting in sums and differences of squares have been generated : 500 two-rectangle features, 500 three-rectangle features and 500 four-rectangle features.
    - A vector of 1500 elements, featurevector, is generated for each image.
    - The training time is less than 30 minutes for the given training set.

4. Bags-of-words:
    - Resized the image to 100x100 pixels to reduce running time of training from 60 minutes to 20 minutes
    - Implemented kmeans clusting using OpenCV. The OpenCV modules required are added to git under folder opencv2
    - Saved the generated bad of words as an image so that it can used for classification
    - accuracy: 47 0f 250 = 19%
    - The first eigen value is quite high as compared to the second
5. Deep features:
    - the training process and testing process is quite slow. We have already put pre-process test data in deep/test-feature, the model is deep/c46.model
    - we assume the overfeat is location in overfeat/
    - we find some images are too small, so we need resize it to 231x231 to fit the soft requirement.
    - we mainly use the python script to deal with the data, we mainly use imagemagick to deal with the image.
    - `make_overfeat.py` is to conver raw image to feature files, which store in deep/
    - `overfeat_helper.py` is to combine these feature into a file to fit svm_multoclass format.
    - we assume the test image is under test/, we have already add python command into c++ (through system call).
    - result: accuracy: 191 of 250 = 76%

6. Future imporvement:
    - for speed, use cuda to increase the speed. Integral python & c++ smoothly.
    - we did not do this, but maybe we can use test/ for training.
    - overfeat is 14.2% when doing clasification (source: http://www.image-net.org/challenges/LSVRC/2013/slides/overfeat_ilsvrc2013.pdf), but we get 24%, so we could do much better, maybe using large pixels? or maybe combine several nets?

References:

http://docs.opencv.org/3.1.0/d7/d8b/tutorial_py_face_detection.html#gsc.tab=0
http://www.developerstation.org/2012/01/kmeans-clustering-in-opencv-with-c.html
http://dsp.stackexchange.com/questions/5979/image-classification-using-sift-features-and-svm