Assignment 2
Due April 28, 2020 at 11.59 PM via ilearn Assignment-2 Submission folder

IMAGES. All images may be found in the TEST IMAGES folder on iLearn or in the folder uploaded specific to this assignment.

Independent Reading.
https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

Problem 1. [2 pts]

(a) On the `house.tif` and `lena_gray_256.tif` images, perform edge detection using the a) Laplacian of the Gaussian and b) Canny edge detector. Report the results, especially the choice of thresholds in computing the edges.

(b) Using the output of the canny edge detector, implement a Hough transform to detect the lines in the `house.tif` image. You should not to use the built-in Hough transform related functions.

Write scripts for your solutions.

Problem 2. [4 pts]

(a) Implement the Shi-Tomasi corner detector as a function named `getCorners.m`, which is available with this package, but needs to be completed. Details of i/p and o/p can be obtained from the script. Use the following:

- Sobel operator to obtain the gradients
- Window size of 5×5 for w (see course slides for details)
- Suppress points which are not local maxima within a 5×5 window.

Run the corner detector on the `house.tif` image and display the top 50 corners according to the minimum eigen value criterion in the algorithm.

(b) This part of the problem is on HoG and SIFT features. `featurematching.m` is the main code for this problem. This code can be broken down into the following steps:

- The code loads the `blocks.png` image and then applies one of the two affine transforms `tform1` or `tform2` to obtain a transformed image. Then, it calls the `getCorners.m` function to obtain the corner points.
- The images along with their corresponding corners are used as inputs to the `getFeatures.m` function, which is provided with this package, but needs to be completed. Details about i/p and o/p to this function may be obtained inside it. This function should extract the Histogram of Gradient (HoG) features for 8×8 patches around each of the corner points. Use 16 bins to obtain the HoG features. You should not use the built-in function of HoG features in Matlab.
- The matches between the feature points of the original and transformed images are obtained using the `getMatches.m` function, which needs to be completed and its i/p and o/p details may be found inside the function. For each feature point, choose the nearest (according to HoG feature) as its match if the distance measure is greater than a certain threshold (hand picked). Use the normalized cross-correlation as a distance measure. The matches are then displayed.
- Finally, the `featurematching.m` code extracts the SIFT features using the vlfeat package (you do not need to implement this) and displays the matches.

Run the `featurematching.m` code on both the transformed images and analyze the displayed results. The HoG features perform better on which transformed image and why ? Which feature (HoG or SIFT) perform well on the transformed images, and why ?

DO NOT change the i/p and o/p arguments and their semantic meaning of the provided functions.

Problem 3. [4 pts] In this problem, you are required to extract the Deep Convolutional Neural Network (CNN) features for CIFAR10 dataset. The dataset is available in PyTorch (using Torchvision), TensorFlow and Keras. This dataset is for object recognition with 10 categories. The PyTorch starter code for this problem is extract_features.py. Some portions of the code is already filled in for convenience. Fill the missing portion of the code for the following:

(a) **Feature Extraction.** Use the pre-trained AlexNet and VGG-16 architecture available in models folder to extract features for the `train_data` and

`test_data` provided in the started code. These models will output the features of penultimate layer of the model.

This code will save features and their labels in 'vgg16_train.mat' and 'vgg16_test.mat' for VGG-16 features and 'alex_train.mat' and 'alex_test.mat' for AlexNet features. This train file for VGG16 should contain 'feature' of dimension 50000×4096, where 50000 is the number of images and 4096 is the feature dimension obtained using VGG-16. Similarly the 'feature' in train file for AlexNet is of size 50000×256.

(b) **K-Nearest Neighbours**. For each feature in 'vgg16_test.mat' and 'alex_test.mat' find K-Nearest Neighbours for label assignment in the 'vgg16_train.mat' 'vgg16_train.mat', respectively. Report accuracy for K = 1, 3 and 5.

Submission Protocol. You should submit codes as well as explanations to each problem (if required). You should add comments to your codes to make them reader friendly. All coding related problems should be in separate scripts named after the problem number. For e.g. Problem 1 has two parts (a) and (b). The codes corresponding to them should be in Problem1a.x and Problem1b.x (x = m if you are using MATLAB and x = py for Python). Keep all the images necessary to run a code in the same folder as the code, while you are submitting. You MUST also include a report written electronically (using the likes of LaTeXor MS Word). It should contain explanations, images, etc (as required). The total length of the report cannot be more than 5 pages with 11 point font and single spacing.

You can also submit a Jupyter Notebook which should contain the write-up with all the details and images along with the code. If you code works and write-up is missing you HW will be considered incomplete.

Each student must do the assignment independently, although you may discuss prior to that. While discussion is allowed, we will be particularly careful about any plagiarism, whether from each other or from other sources.