Assignment 3

CSCI B657 – Computer Vision

ssurendr-cshelke-dipiband

Shwethambari Surendran
Cyril Shelke
Dipika Bandekar

1. FILES SUBMITTED:

SR. NO	FILE NAME	INCLUDED/MODIFIED
1	a3.cpp	MODIFIED
2	Classifier.h	MODIFIED
2	Makefile	MODIFIED
3	svm.h	INCLUDED
4	deep.h	INCLUDED
5	haar.h	INCLUDED
6	pca.h	INCLUDED
7	bow.h	INCLUDED

- Classifier.h – Contains the code and core logic for Part 1, Part 2 Section 2, Part 3 Questions of the assignment
- Pca.h – Contains the code and core logic for Part 2 Section 2
- Bow.h- Contains the code and core logic for Part 2 Section 3

2. HOW TO RUN OUR CODE?

1. PART 1:

- **SECTION 3**

1. **Training SVM :** ./a3 train svm
2. **Testing SVM:** ./a3 test svm

2. PART 2:

- **SECTION 1**

  1. **Training SVM :** ./a3 train eigen
  2. **Testing SVM:** ./a3 test eigen

- **SECTION 2**

  1. **Training SVM :** ./a3 train haar
  2. **Testing SVM:** ./a3 test haar

- **SECTION 3**

  1. **Training SVM :** ./a3 train bow
  2. **Testing SVM:** ./a3 test bow

3. PART 3:

1. **Training SVM :** ./a3 train deep
2. **Testing SVM:** ./a3 test deep

Note: In general respective train, model, test and prediction files are generated along with the accuracies.

3. ANALYSIS:

1. SVM Learn and Classify

Function Used: (Section 1, 2 and 3)

1. void svm(const Dataset & filenames, string value)

Following is the brief outline of the algorithm

a. SVM train:

- Step 0: Each Input image in the train folder is taken as an input image
- Step 1: Each input image is resized (values used for testing 40, 50, 55, 60, 70, 30)
- Step 2: resized image is unrolled along x axis and vector of similar class is prepared
- Step 3: SVM multiclass learn package is used to learn images using system()

References: https://www.cs.cornell.edu/people/tj/svm_light/svm_multiclass.html.

- Format of the input train file used : target(1-25) feature : value (pairs) #info

b. SVM test:

- Step 0: Each Input image in the test folder is taken as an testing image
- Step 1: Each input image is resized (values used for testing 40, 50, 55, 60, 70, 30)
- Step 2: resized image is unrolled along x axis and vector of similar class is prepared
- Step 3: SVM multiclass classify package is used to classify using system()

References: https://www.cs.cornell.edu/people/tj/svm_light/svm_multiclass.html.

- Format of the input train file used : target(1-25) feature : value (pairs) #info
- Final resize value chosen = 50 * 50 (Accuracy 20% Obtained)

Answers to specific questions Part 1:

1. When images are resized using different dimensions various accuracies are generated
2. Images resized with color perform better as compared to the ones without color
3. As the resize dimensions increase above 60 the accuracy begins to decrease
4. For grayscale images accuracy is constant around 10.4% as shown in the table below
5. Model_file_svm has been added for reference

Find below the table with detailed analysis of different factors:

Sr. No.	Resize Dimensions	Accuracy %	Incorrectness %	Color	Time in sec	CORRECT
1	40x40	17.6	82.40%	YES	303	44
2	40x40	10.4	89.60%	NO	225	26
3	50x50	10.4	89.60%	NO	497	26
4	50x50	20	80%	YES	486.2	50
5	60x60	11.6	88.40%	NO	646.71	29
6	60x60	20.8	79.20%	YES	684	52
7	70x70	18.4	81.60%	NO	1227.25	46
8	55x55	19.6	80.40%	YES	543.38	49

2. EIGEN/HAAR/BOW

Section 1:

Randomly chose k to be 400 so that we had reduced features by 25%.

We used the symmetric_eigen() function available in CImg library.

When the eigen value were printed, we were able to see the following pattern:

The eigen values decrease from 9.4363e+08 to -1.5435e-05.

Function Used:

Inside pca.h

1. void train(const Dataset & filenames)

Following is the brief outline of the algorithm

For each image while training, we first converted into greyscale and unrolled it into 1 X 1600 resolution. Subtracted the mean and got the covriance of this matrix. Used the symmetric_eigen() function to generate the eigen values and multiplied this by the unrolled image matrix. We got the final reduced matrix of 1250 X 400 which we gave to SVM as the training model.

Passed each image in the test set to get the accuracy and got an accuracy of around 6%.

Section 2:

Function Used:

1. void train_test_haar(const Dataset &filenames, string value)

Following is the brief outline of the algorithm

a. Haar train:

- Step 0: Each Input image in the train folder is taken as an input image
- Step 1: Each input image is resized to dimensions 50x50
- Step 2: Random x(0-34), y(0-34) , height(1-8), width(1-6) values are generated
- Step 3: Respective rectangles formed and total sum of all pixel values inside the rectangle is calculated
- Step 4: For each (x, y) coordinate the pixel values below it is also calculated and sum of all pixels is taken
- Step 5: The absolute difference between them is calculated and values are stored as final features
- Step 6: 1000 values per image is given to the svm for training in the required format

References: https://www.cs.cornell.edu/people/tj/svm_light/svm_multiclass.html.

https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf

- Format of the input train file used : target(1-25) feature : value (pairs) #info

b. Haar test:

- Step 0: Each Input image in the test folder is taken as an testing image
- Step 1: Each input image is resized (50 x 50)
- Step 2: Same steps as above
- Step 3: SVM multiclass classify package is used to classify using system()

References: https://www.cs.cornell.edu/people/tj/svm_light/svm_multiclass.html.

- Format of the input train file used : target(1-25) feature : value (pairs) #info

- Accuracy achieved is 4.8%

Section 3:

Function Used:

1. void svm(const Dataset & filenames, string value)

Following is the brief outline of the algorithm

Analysis:

1. The accuracies of haar and bow are below the baseline (ie. 20%)
2. The time taken by haar detector is comparatively less as compared to svm in step 1
3. Time taken by bag of words is longer as compared to baseline

3. DEEP FEATURES

Function Used:

1. void train_test_deep(const Dataset &filenames, string value)

Following is the brief outline of the algorithm

a. Deep train:

- Step 0: Each Input image in the train folder is taken as an input image
- Step 1: Each input image is resized to dimensions (231x231, 240x240, 250x250 etc)
- Step 2: The
- Step 3: Respective rectangles formed and total sum of all pixel values inside the rectangle is calculated
- Step 4: For each (x, y) coordinate the pixel values below it is also calculated and sum of all pixels is taken
- Step 5: The absolute difference between them is calculated and values are stored as final features
- Step 6: 1000 values per image is given to the svm for training in the required format

References: https://www.cs.cornell.edu/people/tj/svm_light/svm_multiclass.html.

https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf

- Format of the input train file used : target(1-25) feature : value (pairs) #info

b. Haar test:

- Step 0: Each Input image in the test folder is taken as an testing image
- Step 1: Each input image is resized (50 x 50)

- Step 2: Same steps as above
- Step 3: SVM multiclass classify package is used to classify using system()

  References: https://www.cs.cornell.edu/people/tj/svm_light/svm_multiclass.html.

- Format of the input train file used : target(1-25) feature : value (pairs) #info
- Accuracy achieved is 4.8%

Analysis:

1. The accuracies of haar and bow are below the baseline (ie. 20%)
2. The time taken by haar detector is comparatively less as compared to svm in step 1
3. Time taken by bag of words is longer as compared to baseline

TIME Analysis

Sr. No.	Model file Name	Time
1	model_file_svm	486s
2	model_file_haar	160s
3	model_file_deep	
4	model_file_pca	
5	model_file_bow	