CS 548: Assignment 04

Appearance-Based Features and Machine Learning

NO Theory Questions

No theory questions for this assignment.

Programming Assignments (100%)

LBP PROGRAMMING LANGUAGE: C++
MACHINE LEARNING PROGRAMMING LANGUAGE: Python 3.x

For this assignment, you will implement LBP feature extraction software in C++. You will also write and run Python code to perform machine learning on these features.

DO NOT USE OPENCV'S FUNCTIONS FOR HISTOGRAMS OR FEATURE EXTRACTION. You must write these algorithms yourself.

Your C++ code will rely on the OpenCV library ONLY.

Your code should NOT use OS-specific libraries/functionality!

You can use the checkboxes to track whether you've met each requirement.

C++ LBP Feature Extraction (60%)

#	***Requirements***	
1	**Download the provided code files (available on the course webpage and Blackboard):**	
	CMakeLists.txt - same as BasicVision's version of the file, except this one uses Assign04 as the project name.	
	LBP.hpp - header file for functions you will write.	

	TEST_LBP.hpp - header file for testing functions.	
	TEST_LBP.cpp - source file for testing functions.	
	Assign04.cpp - the main program	
	The main program takes 3 arguments: 1. Path to input image 2. Path to output CSV FILE 3. Whether we are in testing mode or not (0 = no, 1 = yes) "Testing mode" will run a series of tests on your code. The normal mode will load the image, extract the LBP histogram, and APPEND the histogram to the CSV file.	
2	**Create a file "LBP.cpp" that defines all functions within LBP.hpp:**	
	int getPixel(Mat inputImage, int row, int col)	
	You may assume inputImage is a single-channel UCHAR image.	
	If row and col are within the bounds of the image, return the value of the pixel at (row, col).	
	Otherwise, return 0.	
	void getLBPNeighbors(Mat inputImage, int centerRow, int centerCol, int *neighbors)	
	You may assume inputImage is a single-channel UCHAR image.	
	You may also assume the neighbors ALREADY has NEIGHBOR_CNT doubles allocated.	

	You will call getPixel() on the following coordinates to fill in the values of neighbors[]: 0. (centerRow - 1, centerCol) 1. (centerRow - 1, centerCol + 1) 2. (centerRow, centerCol + 1) 3. (centerRow + 1, centerCol + 1) 4. (centerRow + 1, centerCol) 5. (centerRow + 1, centerCol - 1) 6. (centerRow, centerCol - 1) 7. (centerRow - 1, centerCol - 1)	
	void thresholdArray(int threshold, int *data, int cnt)	
	You may also assume the $data$ ALREADY has cnt ints allocated.	
	Do NOT assume there are 8 elements! Use cnt!	
	Loop through $data$. If $data[i]$ is GREATER THAN $threshold$, set $data[i]$ equal to 1.	
	Otherwise, set $data[i]$ equal to 0.	
	int getUniformLabel(int *data, int cnt)	
	You may also assume the $data$ ALREADY has cnt ints allocated.	
	Do NOT assume there are 8 elements! Use cnt!	
	You will compute the **rotation-invariant uniform LBP** labels.	
	If the number of 1-0 or 0-1 transitions is less than or equal to 2, then use the number of "one" neighbors to determine the label.	
	If the number of transitions is greater than 2, then use the label (cnt + 1).	

	When checking for transitions, **make sure you also compare the last and first elements in the array!**	
	You should only have a total of (cnt+2) possible labels.	
	void getLBPImage(Mat inputImage, Mat &outputImageLBP)	
	You may assume inputImage is a single-channel UCHAR image.	
	(Re)create outputImageLBP to be the same size and type as inputImage.	
	In this function, you may assume that the total number of samples is given by NEIGHBOR_CNT.	
	For each position (i,j) in inputImage: • Get the center pixel value using getPixel(). • Get the neighbors around this pixel using getLBPNeighbors(). • Threshold the array of neighbors using the center value using thresholdArray(). • Get the uniform label using getUniformLabel(). • Store the label at (i,j) in outputImageLBP.	
	void calculateHistogram(Mat image, double hist[], int length)	
	Copy this function from Assignment 02.	
	void computeLBPHistogram(Mat imageLBP, double *totalHistogram)	
	You may assume imageLBP is a single-channel UCHAR image.	
	Break the image up into 4 non-overlapping subregions of (imageLBP.rows/2) x (imageLBP.cols/2) size. If a dimension is odd, you can ignore the last row or column.	
	To get each subregion: Mat subRegion = imageLBP(Rect(subRegionCol*subWidth, subRegionRow*subHeight, subWidth, subHeight));	

	You may assume that totalHistogram has (4 * MAX_LABEL_CNT) elements in it.	
	For each subregion, use calculateHistogram() to get the histogram of MAX_LABEL_CNT elements.	
	Copy the subregion's histogram into totalHistogram at the appropriate spot. (Starting index = subRegionRow * MAX_LABEL_CNT * 2 + subRegionCol * MAX_LABEL_CNT)	

Python Machine Learning (40%)

#	***Requirements***	
1	**Create an account on XSEDE (if you don't already have one) with your school email address:** `https://portal.xsede.org`	
2	**Email me your username so I can add you to the allocation.**	
3	**Once you are added, you will have to create an account on PSC Bridges:** `https://apr.psc.edu/autopwdreset/autopwdreset.html`	
	This is NOT the same as your XSEDE username/password!	
4	**Download the provided code files (available on the course webpage and Blackboard):**	
	Train_Features.csv - LBP features for training set of CIFAR10 database.	
	Test_Features.csv - LBP features for training set of CIFAR10 database.	
	Assign04.job - job script file for Bridges.	
5	**Create a Python 3.x file, TrainAssign04.py**	
	If the number of arguments passed in is less than 3, print an error and exit.	
	The arguments passed in should be: Training Data CSV File = sys.argv[1] Testing Data CSV File = sys.argv[2]	
	Read the CSV files as Panda DataFrames.	

	Note that the files DO have headers, which is the default behavior.	
	Get the ground truth labels.	
	Grab the "ground" column and get the values (Numpy array).	
	Drop the "ground" column from the training and testing data.	
	Drop the "Filename" column from the training and testing data.	
	Get the values (Numpy arrays) of the remaining training and testing data.	
	Create and fit ANY two classifiers from the scikit-learn library to the training data, with the following options:	
	(Option 1) The SAME classifier but with DIFFERENT parameters (e.g., Adaboost with a substantially different number of weak classifiers)	
	(Option 2) Two DIFFERENT classifiers	
	Using your two classifiers, predict the results for the training data.	
	Using your two classifiers, predict the results for the testing data.	
	For each classifier, print out the following **(WITH TEXT THAT EXPLAINS WHICH NUMBER BELONGS TO WHICH STATISTIC, DATASET, AND CLASSIFIER):**	
	Training accuracy with classifier 1 (USE A DESCRIPTIVE NAME for the classifier, like "Adaboost with 100").	
	Training F1 score (average = "macro") with classifier 1.	
	Testing accuracy with classifier 1.	
	Testing F1 score (average = "macro") with classifier 1.	
	Training accuracy with classifier 2.	
	Training F1 score (average = "macro") with classifier 2.	

	Testing accuracy with classifier 2.	
	Testing F1 score (average = "macro") with classifier 2.	
6	**Copy all these files to a directory in your HOME directory on Bridges.**	
	You can ftp to Bridges using your BRIDGES username/password and this address: `data.bridges.psc.edu`	
7	**SSH into Bridges via the XSEDE Single Sign On hub**	
	You will have to first install and setup the DUO phone app: `https://portal.xsede.org/mfa`	
	Ssh into XSEDE: `login.xsede.org`	
	Once there, connect to Bridges: gsissh bridges	
8	**Go to where your job script is located and run the job.**	
	To run the job: **sbatch Assign04.job**	
	Do NOT directly run your Python scripts in the login node!!!	
	To list all jobs you have: **squeue -u <your Bridges username>**	
	To cancel your job: **scancel <job id>**	
	See the Bridges user guide for more information: `https://www.psc.edu/bridges/user-guide`	
9	**Once your job completes, check the output log to make sure everything went smoothly.**	
	The log will have the name "slurm-" + <job id> + ".out"	
	There should be NO errors that prohibit the script from completing!	
10	**You will submit both your code file (TrainAssign04.py) AND your log file.**	

Local Machine

For testing the Python code on your local machine, follow the instructions given in the slide deck "Introduction to Scikit-Learn and Keras" to install Anaconda (for Python 3), setup a virtual environment, and install all necessary libraries.

DO NOT follow these instructions for PSC Bridges!

C++ Code Testing Mode

The provided main program will perform a series of tests to check the implementation of your functions. **Your code should pass these tests.**

For testing mode:

```
Assign04 ../input/Image0.png ../output 1
```

For regular mode:

```
Assign04 ../input/Image0.png ../output 0
```

NOTE: This program will assume that you have the following folder structure:

- /build
- /input
  - Image0.png
  - Image1.png
  - Image2.png
  - Image3.png
  - Image4.png
  - Image5.png
  - Image6.png
- /ground
  - HISTOGRAMS.csv
  - LBP_Image0.png
  - LBP_Image1.png
  - LBP_Image2.png
  - LBP_Image3.png
  - LBP_Image4.png
  - LBP_Image5.png
  - LBP_Image6.png

Submission

You must do the following:

- Submit the following items as a *.tar, *.tar.gz, or *.zip file to "Assignment 04: Programming":
  - **LBP.cpp**
  - **TrainAssign04.py**
  - **Slurm log file**

Do NOT submit:

- **A screenshot**
- Your executable file(s)
- Your project files
- Input/output images
- Extracted data and/or .csv files

Grading

Your OVERALL assignment grade is weighted as follows:

- 60% - C++ Programming
- 40% - Python Programming

Future assignments may use a different grading distribution.

For the **"C++ PROGRAMMING"** portion of the assignment:

Issue	***Penalty***
Code that does not compile	**60% off**
getPixel() incorrect / not working	**up to 5% off**
getLBPNeighbors() incorrect / not working	**up to 5% off**
thresholdArray() incorrect / not working	**up to 5% off**
getUniformLabel() incorrect / not working	**up to 15% off**
getLBPImage() incorrect / not working	**up to 15% off**
calculateHistogram() incorrect / not working	**up to 5% off**
computeLBPHistogram2x2() incorrect / not working	**up to 10% off**
getPixel() not implemented	**up to 10% off**
getLBPNeighbors() not implemented	**up to 10% off**
thresholdArray() not implemented	**up to 10% off**
getUniformLabel() not implemented	**up to 30% off**
getLBPImage() not implemented	**up to 30% off**
calculateHistogram() not implemented	**up to 10% off**
computeLBPHistogram2x2() not implemented	**up to 20% off**
Using OpenCV's functions for feature detection or histogram calculation to compute your results	**If used in function, equivalent to not implementing that function**
NOTHING implemented	**100% off**
Code NOT submitted in a .tar, *.tar.gz, or *.zip file	**5% off**
Submission that does NOT meet the requirements listed in the "Requirements" table	**MINIMUM 5% off for each violation**
Sending an executable WITHOUT CODE	**100% off**
Nothing submitted	**100% off**

For the **"PYTHON PROGRAMMING"** portion of the assignment:

Issue	***Penalty***
Code with syntax errors	**60% off**
Log contains errors that prevent the script from completing.	**60% off**
Not checking for correct parameter count.	**5% off**
Only one classifier used.	**up to 30% off**
Classifiers not sufficiently different.	**up to 20% off**
Results not clearly printed out.	**up to 30% off**
Code that does NOT match log.	**100% off**
Missing code.	**100% off**
Missing log.	**100% off**
Code NOT submitted in a .tar, *.tar.gz, or *.zip file	**5% off**
Submission that does NOT meet the requirements listed in the "Requirements" table	**MINIMUM 5% off for each violation**
Nothing submitted	**100% off**

In general, these are things that will be penalized:

- **Code that does not compile / has syntax errors**
- Sloppy or poor coding style
- Bad coding design principles
- Code that crashes, does not run, or takes a VERY long time to complete
- Using code from ANY source other than the course materials
- Collaboration on code of ANY kind; this is an INDIVIDUAL PROJECT
- Sharing code with other people in this class or using code from this or any other related class
- Output that is incorrect
- Algorithms/implementations that are incorrect
- Submitting improper files
- Failing to submit ALL required files