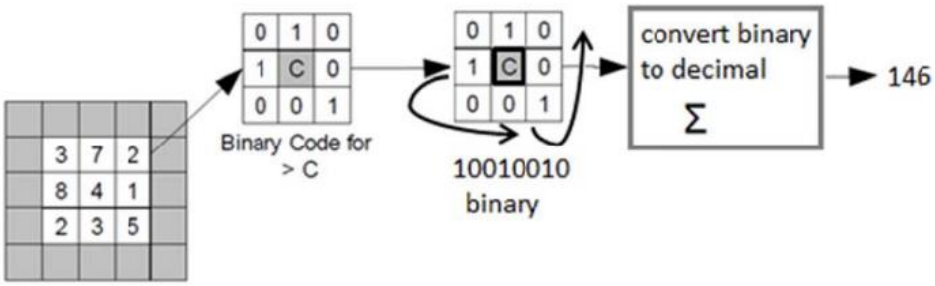
Computer Vision: *Implementing a Local Binary Pattern (LBP) Classifier from scratch*

Requirements:

Texture: Local binary pattern (LBP) is a feature used for texture representation and classification. The LBP operator describes the surroundings of a pixel by generating a bit-code from the binary derivatives of a pixel. The operator is usually applied to greyscale images and the derivative of the intensities. An example of how the LBP in its simplest form works is illustrated in the figure below. Note how a pixel with a value > C is assigned '1' and a pixel with a value < C is assigned '0'.



a) Write a function that divides an input (greyscale) image into equally sized non-overlapping windows, and returns the feature descriptor for each window as distribution of LBP codes. For each pixel in the window, compare the pixel to each of its 8 neighbours (on its left-top, left-middle, left-bottom, right-top, etc.). Follow the pixels in a counter-clockwise manner. Compute the histogram over the window - as the frequency of each "number" occurring. Normalize the histogram. The histogram is now a feature descriptor representing the window at hand.

b) Come up with a *descriptor* to represent the whole image as consisting of multiple windows. *Hint:* Think of combining several local descriptions into a global description.

c) Using the global descriptor you have created, implement a classification process to classify the images in the provided dataset into two categories: face images and car images. Comment the results, include images if necessary. *Hint:* You can use simple histogram similarities. Is the global descriptor able/unable to represent whole images of different types, e.g. faces vs. cars? Identify problems (if any). Suggest solutions if possible, include images if necessary.

d) Decrease the window size and perform classification again. Comment the results, include images if necessary.

e) Increase the window size and perform classification again. Comment the results, include images if necessary.

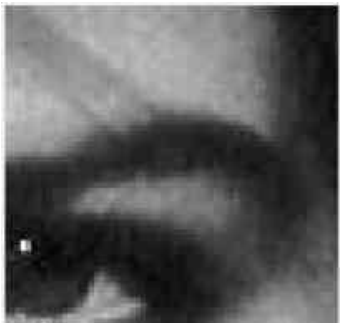
Report:

Question (a)

Three non-consecutive windows

W1

W2



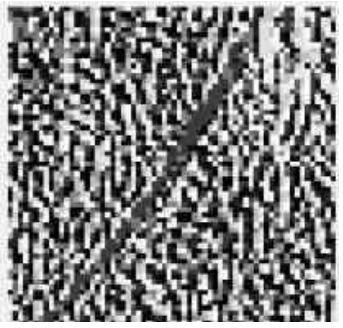
W3

LBP of windows

LBP1

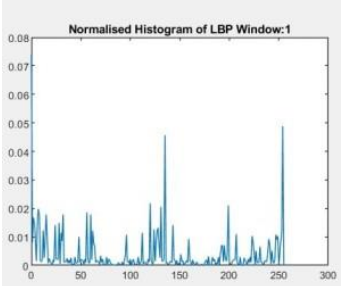
LBP2

LBP3

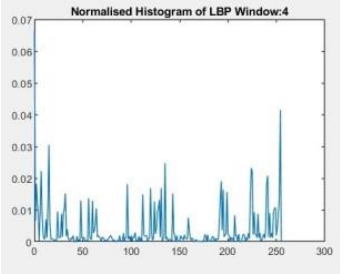


Histograms of LBPs

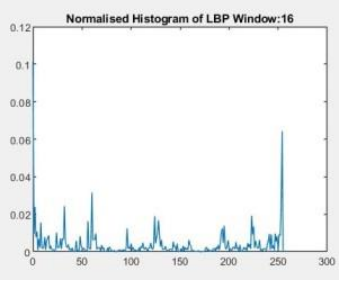
H1



H2



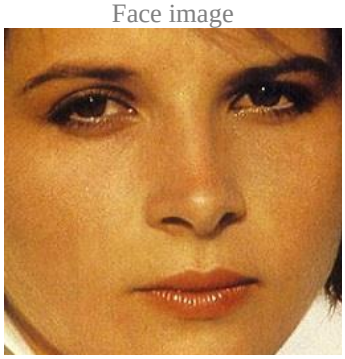
H3



Question (b)

Two example images:

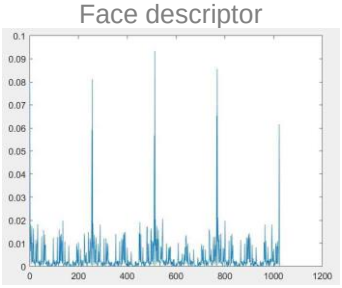
Face image



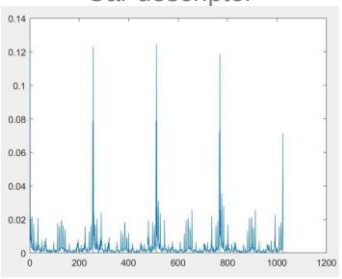
Car image

Descriptors:

Face descriptor



Car descriptor



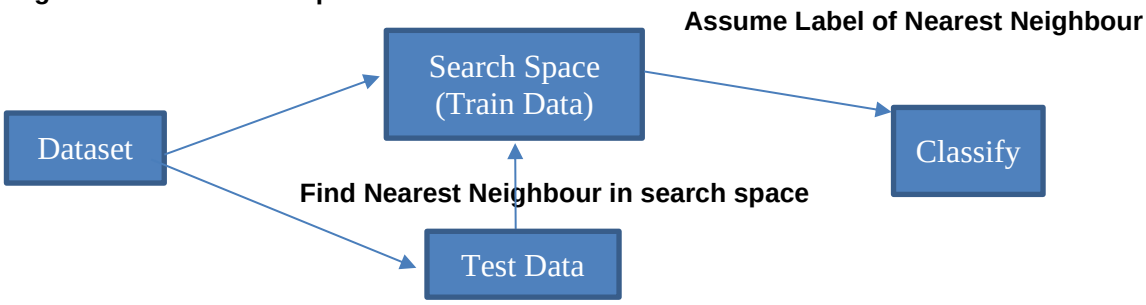
Comments:

a) I took the image, performed LBP on that image, split that image into windows and returned the windows, LBP windows and their normalised histograms. Each normalised histogram for each LBP window becomes the local descriptors of the image.

b) To create the descriptor I took the normalised histograms (local descriptors) representing each LBP window of the image (which I calculated in 6a) and concatenated them side by side into one global histogram (global descriptor) to represent the whole image consisting of multiple windows. This would store the features of each image in the order they appear in on the object allowing for ease of comparison when comparing similar objects for classification. The new formed global descriptor is a collection of local descriptors for the chosen image.

Question (c)

Block diagram of classification process:



Comments:

To classify the images in Dataset A I made a classifier which follows a Nearest Neighbour Classification approach. The classifier has 2 settings. 'Setting 1' uses 2 images as the train data and 'Setting 2' uses 4 images as the train data. What I refer to as train data in this case are the global descriptors of the images in the search space. First I split the dataset into train (labelled) and test (unlabelled) data. The global histograms (descriptors) of the train images are stored in the search space. Every time a test image is classified it compares its own global descriptor to the global descriptors in the search space. It then calculates the Euclidean distance (similarity) between itself and all existing global descriptors in the search space using the following formula:

$$D = \sqrt{\sum_{i=1}^{n}(b_i - a_i)^2}$$

Where 'D' is the Euclidean distance, 'b' is the global histogram of the test image and 'a' is the global histogram of a train image from the search space. The test image is classified using the label of whichever train image it is closest to using the smallest Euclidean distance. I found that 'Setting 2' of my classifier performed better than 'Setting 1' as using more train data resulted in a more accurate classification of the test data but at the cost of a longer execution time.

Setting 2 used 2 faces and 2 cars as train data whereas Setting 1 used 1 face and 1 car as train data.

In most cases I found just 2 images(1 face and 1 car) were enough to be used as train data for this dataset. The only time that Setting 1 misclassified was when face-2.jpg from Dataset A was used as the train image for faces. In this case the remaining 2 faces were misclassified as cars. The reason for this is because the global histograms of face-1.jpg and face-3.jpg were more similar to the global histograms of the cars used as the train image than they were to the global histogram of face-2.jpg. This however was overcome using Setting 2 of my classifier.(See below). For both settings the images with the black labels are the train data and the images with the red labels are the test data which were classified.

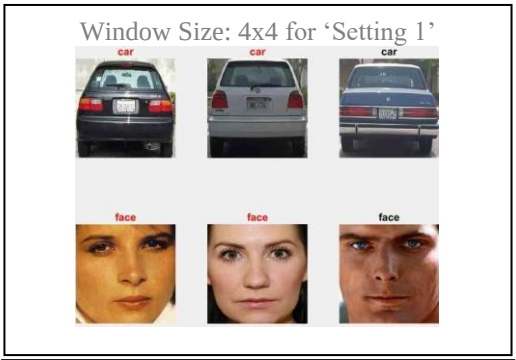
Setting 1: (4 Train Images) Results for Window Size(128x128 and 4x4)

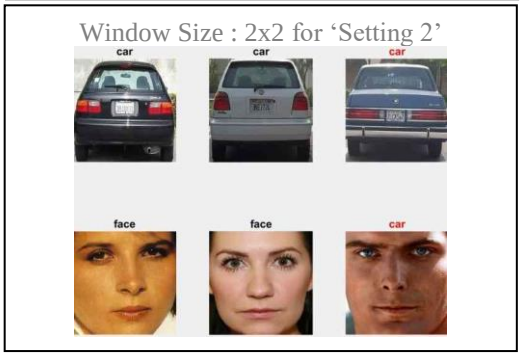
Test Images	Train Images	Classification Accuracy/Comments
Face-1, Face-2 Car-1, Car-2	Face-3 Car-3	100%
Face-1, Face-3 Car-1, Car-3	Face-2 Car-2	Both faces classified as cars, Both cars classified as cars. 50%
Face-3, Face-2 Car-3, Car-2	Face-1 Car-1	100%
Face-2, Face-3, Car-1,Car-3	Face-1 Car-2	100%
Face-2,Face-3, Car-1,Car-2	Face-1 Car-3	100%
Face-1,Face-3, Car-2,Car-3	Face-2 Car-1	Both faces classified as cars, Both cars classified as cars. 50%
Face-1,Face-3, Car-1,Car-2	Face-2, Car-3	Both faces classified as cars, Both cars classified as cars. 50%
Face-1,Face-2, Car-2,Car-3	Face-3, Car-1,	100%
Face-1,Face-2, Car-1,Car-3	Face-3, Car-2	100%

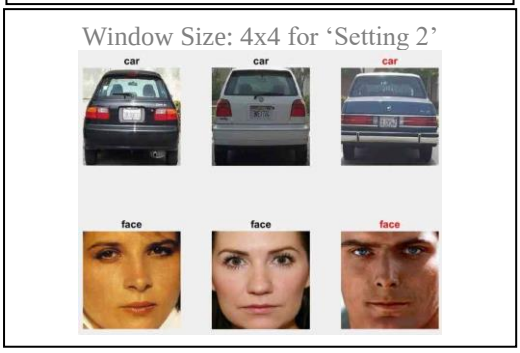
Figure 1: *(Shows all possible combinations using Dataset A and highlights in orange where classification was only 50% accurate and in green where classification was 100% accurate).Inaccuracies were overcome using Setting 2.*

Question (d)

Comments:





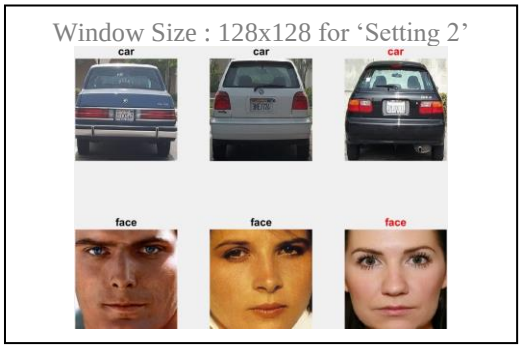


(Black labelled images are train data and red labelled images are the classified test images)

When using smaller window sizes I found my classifier performed significantly slower compared to larger window sizes. The results were significantly less accurate for window size 2x2. When using a window size this small many features from the image would not be extracted thereby leading to worse feature descriptors which would make the classifier perform poorly. As shown in the above example outputs of the classifier we can see even Setting 2 of my classifier performs poorly for a 2x2 window size.
Window size 4x4 performed much faster than window size 2x2 and provided similar results (more accurate results) to larger window sizes such as 128x128 (as show in Figure 1 p13).

Question (e)

Comments:



(Black labelled images are train data and red labelled images are the classified test images)

When testing for larger window sizes I found my classifier performed significantly faster and produced better results than when using a smaller window size like 2x2. Window sizes larger than or equal to 4x4 produced similar results to Figure 1 (p13) when using 'Setting 1' and the misclassifications were overcome using 'Setting 2'. The above example outputs show accurate classifications using 64x64 and 128x128 windows using 'Setting 2' of my classifier.

Question (f)

Comments

A Local Binary Pattern (LBP) operator is used for texture detection. It generates a bit code describing the surrounding pixels of a pixel and is typically applied in 2 planes (X and Y directions). Typically the surrounding 8 pixels are checked in an clockwise or anti-clockwise circular order whereby if the centre pixel value is higher than the current pixel it is assigned a 1 else it is assigned a 0.

These 8 values would make up the bit code which is then converted to a decimal value to aid with the extraction of feature descriptors through the generation of histograms. This process works well with static texture but must be modified to be applied to dynamic texture.

Dynamic texture is texture which involves motion. Some examples of dynamic texture include smoke or the waves in oceans. LBP can be modified to recognize dynamic textures by applying it in 3 planes (instead of just 2) where each plane is orthonormal to the others.

LBP is extracted from the proposed 3 directions to then form a global histogram which consists of all the local descriptors (histograms) concatenated together. Elliptical sampling can be used to fit the measure of space-time. This method extends the standard LBP approach by combining appearance and motion.