Predicting Laptop Users Gaze

Contactless Selection of (Big) Buttons on the Screen

Rahul Gupta
Mentored By: Prof. Gaurav Sharma

Department of Computer Science
IIT Kanpur

UGP (CS395) Presentation

Abstract

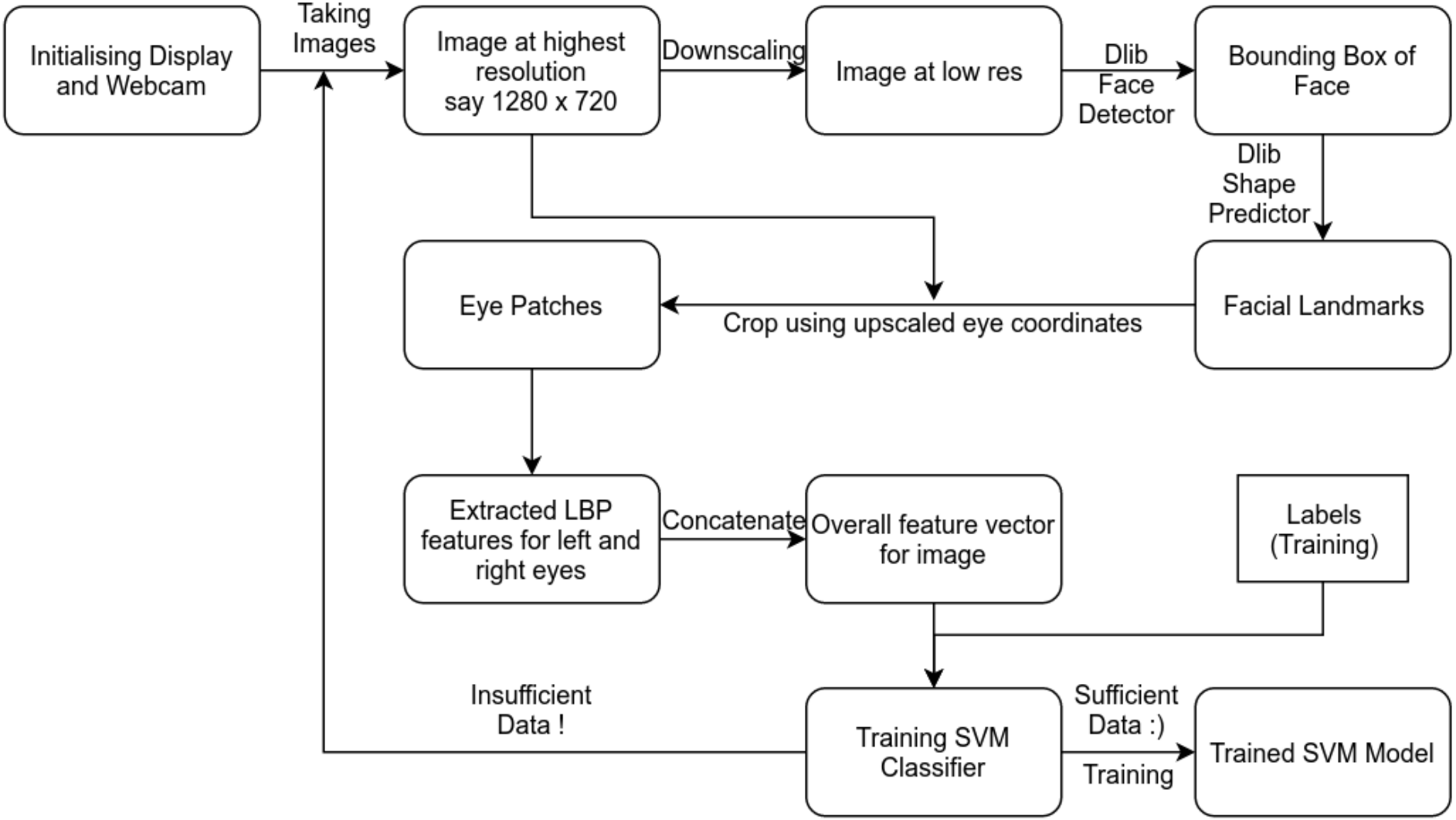
- The aim of the project was to build a software tool, based on basic computer vision technologies, which allows for an HCI(Human Computer Interface) to select buttons placed in a coarse grid of 2x2 of 2x3 on the screen, using just her gaze
- Future directions include making an end-to-end trainable deep network for the task.

Outline

Outline

Overview

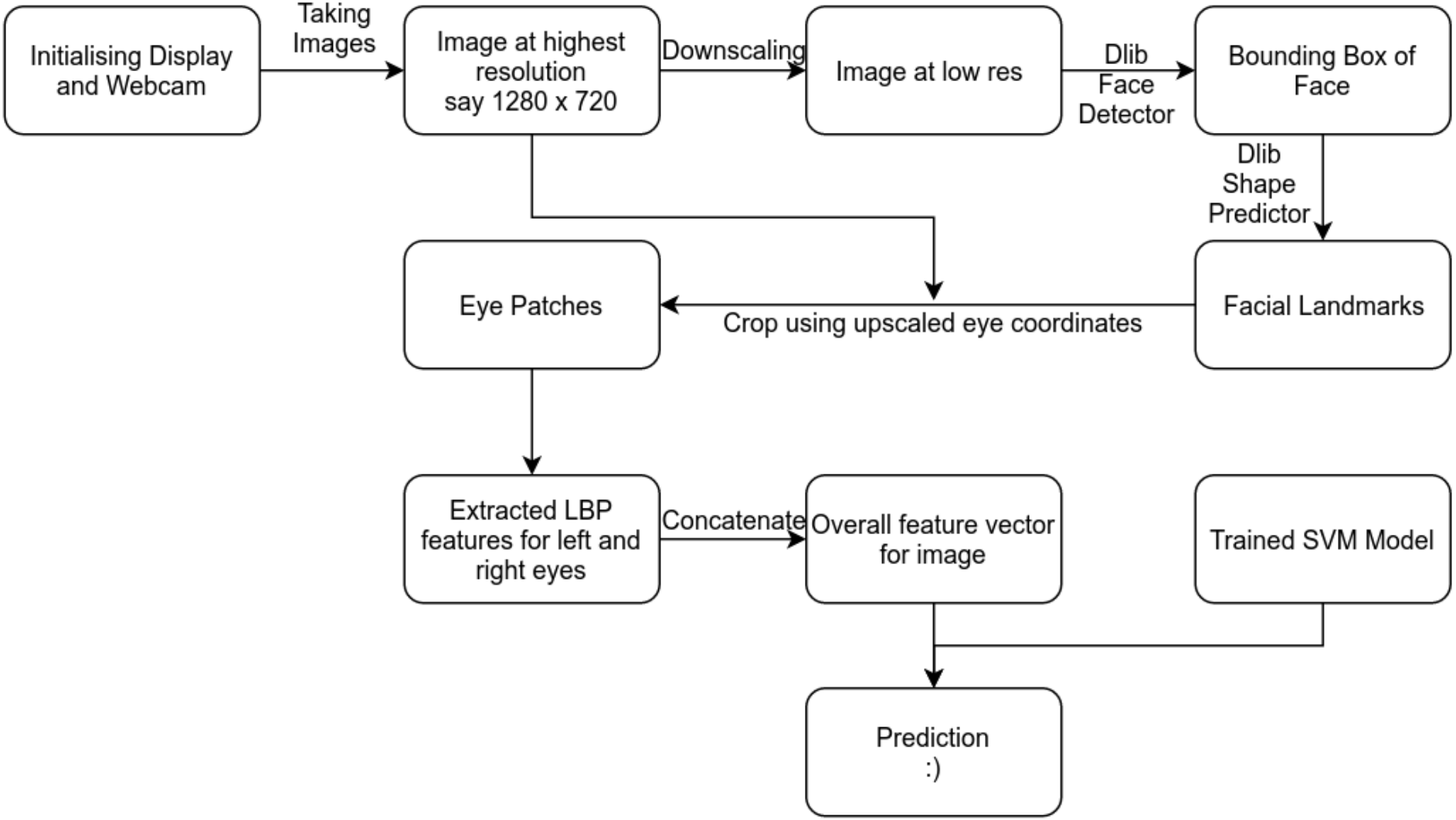
Training



Outline

Overview

Prediction



Outline

User Interface

Using Pygame

Pygame is a cross-platfrom library designed to make it easy to write multimedia software, such as games, in Python.
It consists of a camera module, which allows taking images and using them with several controls.

Outline

User Interface

Design

- Training Data Collection
  - Users are asked to focus on dots appearing at several places on screen.
  - These dots appear sequentially in four quadrants around corners.
  - From webcam, images are taken and region around eyes is cropped to feed the classifier.
  - This process takes about 20 seconds.
- Predicting Users Gaze
  - Users can focus on a quadrant of the screen.
  - A marker appears in the that quadrant . This can be used for selecting options in the future.

Outline

Face Detection

Dlib: A C++ Library

Dlib is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real world problems. It offers a frontal face detector that can be used for getting face landmarks as shown in figure

Figure: Dlib predicted landmarks

Outline

Getting Face Landmarks

Speeding up by downscaling

- Images are taken at 1280*720 or highest available resolution of webcam.

Getting Face Landmarks

Speeding up by downscaling

- Images are taken at 1280*720 or highest available resolution of webcam.
- Followed by cropping and downscaling it to 250*250 for face detection.

Getting Face Landmarks

Speeding up by downscaling

- Images are taken at 1280*720 or highest available resolution of webcam.
- Followed by cropping and downscaling it to 250*250 for face detection.
- Then we get the bounding box over faces in the picture.

Getting Face Landmarks

Speeding up by downscaling

- Images are taken at 1280*720 or highest available resolution of webcam.
- Followed by cropping and downscaling it to 250*250 for face detection.
- Then we get the bounding box over faces in the picture. For our purpose we assume there is only 1 face.

Getting Face Landmarks

Speeding up by downscaling

- Images are taken at 1280*720 or highest available resolution of webcam.
- Followed by cropping and downscaling it to 250*250 for face detection.
- Then we get the bounding box over faces in the picture. For our purpose we assume there is only 1 face.
- Face landmark detector is invoked within that bounding box.

Getting Face Landmarks

Speeding up by downscaling

- Images are taken at 1280*720 or highest available resolution of webcam.
- Followed by cropping and downscaling it to 250*250 for face detection.
- Then we get the bounding box over faces in the picture. For our purpose we assume there is only 1 face.
- Face landmark detector is invoked within that bounding box.
- Detected landmark points are upscaled.

Outline

The Classifier

Local Binary Patterns

- Local binary patterns (LBP) is a type of visual descriptor used largely for classification of textures in computer vision.

The Classifier

Local Binary Patterns

- Local binary patterns (LBP) is a type of visual descriptor used largely for classification of textures in computer vision.
- LBPs compute a local representation of texture. This local representation is constructed by comparing each pixel with its surrounding neighborhood of pixels.

The Classifier

Local Binary Patterns

- Local binary patterns (LBP) is a type of visual descriptor used largely for classification of textures in computer vision.
- LBPs compute a local representation of texture. This local representation is constructed by comparing each pixel with its surrounding neighborhood of pixels.
- The first step in constructing the LBP texture descriptor is to convert the image to grayscale.

The Classifier

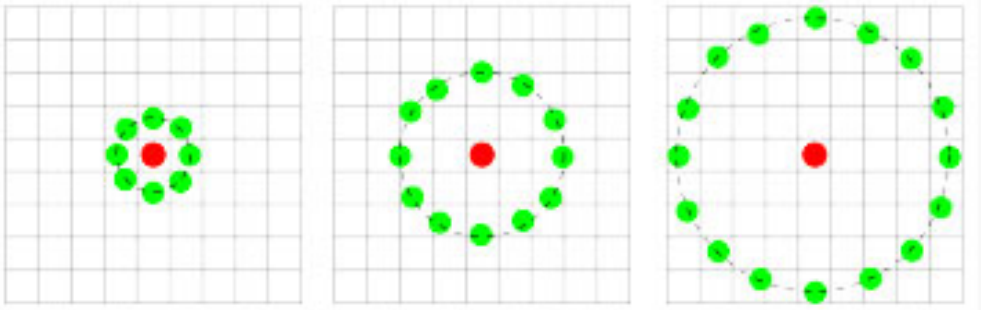
Local Binary Patterns

- Local binary patterns (LBP) is a type of visual descriptor used largely for classification of textures in computer vision.
- LBPs compute a local representation of texture. This local representation is constructed by comparing each pixel with its surrounding neighborhood of pixels.
- The first step in constructing the LBP texture descriptor is to convert the image to grayscale.
- LBP value is then calculated for each pixel and stored in the output 2D array with the same width and height as the input image.

The Classifier

Local Binary Patterns

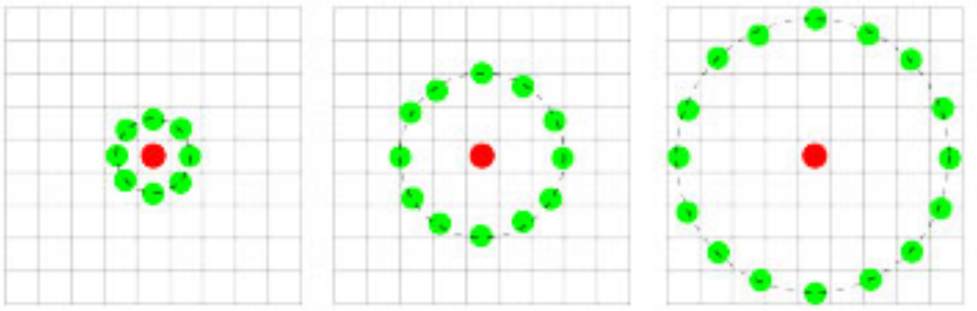
- For calculating a pixel's lbp, we threshold it against its neighborhood of n pixels(n can be set) within radius r.



The Classifier

Local Binary Patterns

- For calculating a pixel's lbp, we threshold it against its neighborhood of n pixels(n can be set) within radius r.



- If the intensity of the center pixel is greater-than-or-equal to its neighbor, then we set the value in copied neighbor to 1; otherwise, we set it to 0.

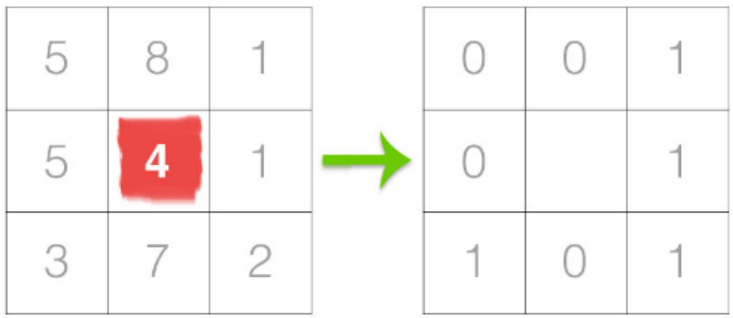


Figure: example for r=1 and n=8

The Classifier

Local Binary Patterns

- With n surrounding pixels, we have a total of 2^n possible combinations of LBP codes.

The Classifier

Local Binary Patterns

- With n surrounding pixels, we have a total of 2^n possible combinations of LBP codes.
- We start from a neighboring pixel and work our way clockwise or counter-clockwise consistently for all pixels in our image.

The Classifier

Local Binary Patterns

- With n surrounding pixels, we have a total of 2^n possible combinations of LBP codes.
- We start from a neighboring pixel and work our way clockwise or counter-clockwise consistently for all pixels in our image.
- The results stored in a binary array treated as a decimal value is the lbp value for that pixel.

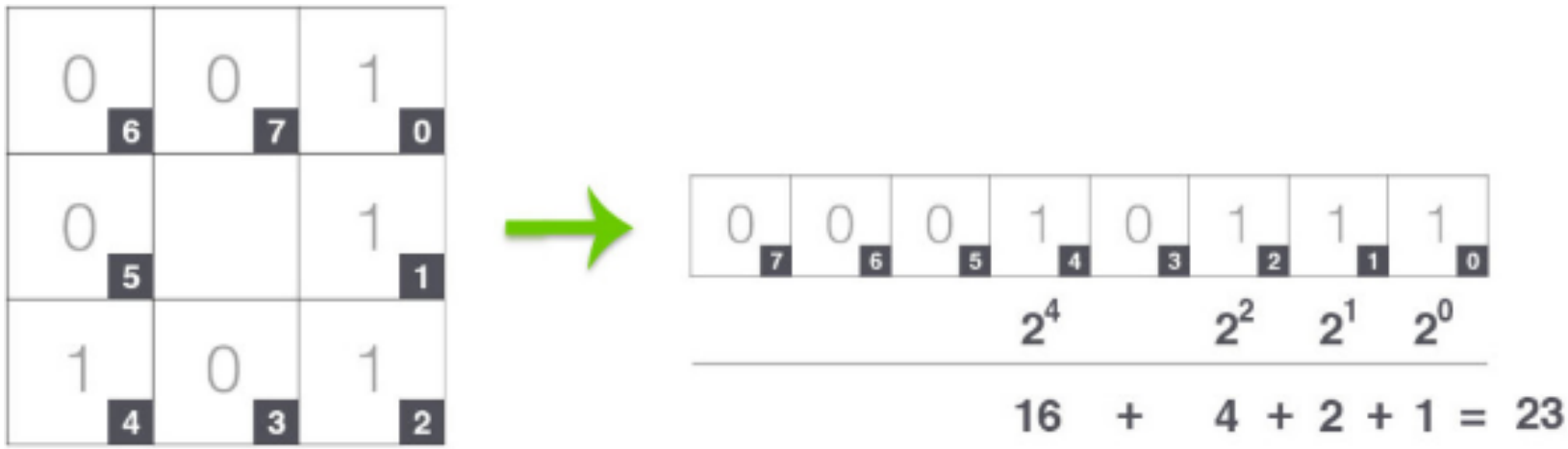


Figure: example for r=1 and n=8

The Classifier

Local Binary Patterns

- We have used Quantized LBP for getting the feature vector. I used the function provided by Prof. Gaurav Sharma for this task.

The Classifier

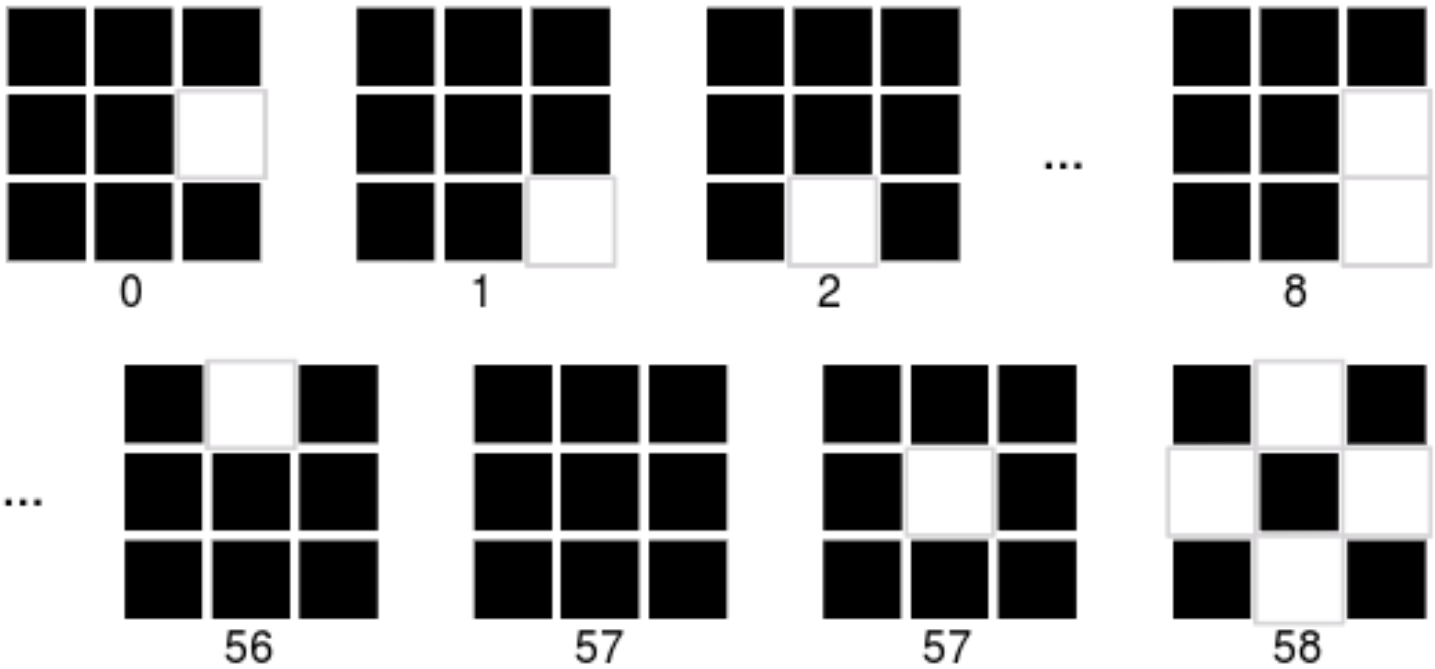
Local Binary Patterns

- We have used Quantized LBP for getting the feature vector. I used the function provided by Prof. Gaurav Sharma for this task.
- It is similar to LBP except that here instead of plotting histogram for all distinct 256 values of lbp, we group some of them.

The Classifier

Local Binary Patterns

- We have used Quantized LBP for getting the feature vector. I used the function provided by Prof. Gaurav Sharma for this task.
- It is similar to LBP except that here instead of plotting histogram for all distinct 256 values of lbp, we group some of them.



LBP quantized patterns.

Outline

The Classifier

Training

- For each of the images, we crop and concatenate the eye regions. We then calculate its LBP matrix.

The Classifier

Training

- For each of the images, we crop and concatenate the eye regions. We then calculate its LBP matrix.
- Our feature vector is the histogram computed over transformed image using local binary patterns.

The Classifier

Training

- For each of the images, we crop and concatenate the eye regions. We then calculate its LBP matrix.
- Our feature vector is the histogram computed over transformed image using local binary patterns.
- For all images taken using webcam, we compute the feature vector and store them against their quadrant label.

The Classifier

Training

- For each of the images, we crop and concatenate the eye regions. We then calculate its LBP matrix.
- Our feature vector is the histogram computed over transformed image using local binary patterns.
- For all images taken using webcam, we compute the feature vector and store them against their quadrant label.
- A Support Vector Machine is then trained over that data.

The Classifier

Training

- For each of the images, we crop and concatenate the eye regions. We then calculate its LBP matrix.
- Our feature vector is the histogram computed over transformed image using local binary patterns.
- For all images taken using webcam, we compute the feature vector and store them against their quadrant label.
- A Support Vector Machine is then trained over that data.
- The model is then saved and is used subsequently for making predictions.

Further Developments

- Making an end-to-end trainable deepnetwork for the task using torch.
- Work done in this direction:
  - Getting the average coordintates of eyes, nose etc from lfw database for face alignment.
  - Affine transformation for aligning the face.
  - Formatting MPII Gaze Dataset to be used for training the cnn.
- Things to be done:
  - Training an optimal neural network that classifies data with satisfactory accuracy.
  - Deploying it to complete the application.

References I


PyImageSearch
http://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/


PyGame
http://pygame.org


Scikit - SVM
http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html


PyImageSearch
http://scikit-image.org/docs/dev/api/skimage.html

References II

Coupled Projection multi-task Metric Learning for Large Scale Face Retrieval

Binod Bhattarai, Gaurav Sharma and Frederic Jurie

mtml cvpr 2016, CVPR, 2016