Project 1 : Report

Ajay RR (IMT2017502) Ronak Doshi (IMT2017523) Ram S (IMT2017521)

Visual Recognition Course
International Institute of Information Technology Bangalore

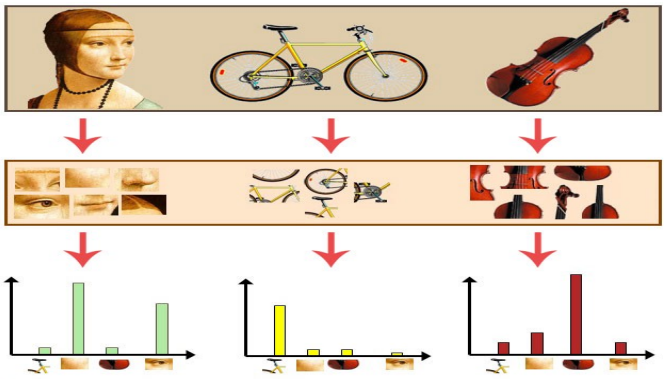
Contents

1 VLAD on CIFAR10

A comparative study between VLAD (Vector of Locally Aggregated Descriptors) and BoVW (Bag of Visual Words) was performed, using the CIFAR10 dataset.

1.1 BoVW (Nutshell)

The general idea of Bag of Visual Words (BOVW) is to represent an image as a set of features. Features consists of keypoints and descriptors. Keypoints are the "stand out" points in an image, so no matter the image is rotated, shrunk, or expanded, its keypoints will always be the same. The descriptor is the description of a keypoint. Keypoints and descriptors are used to construct vocabularies and represent each image as a frequency histogram of features that are in the image. Later from the frequency histogram, one can find similar images or predict the category an image could belong to.



1.2 VLAD

The Bag of Visual Words was used on the CIFAR10 dataset in assignment 2. In order to make this pre-processing method better, an algorithm called **Vector of Locally Aggregated Descriptors (VLAD)** was used.
VLAD is an extension of the BoVW concept. In VLAD we first assign each descriptor of the image to the corresponding cluster center (Same as BoVW) but instead of computing the histogram of cluster frequency, we sum the difference of the descriptors with the assigned cluster (of the descriptor) center.
I.e, After training K-Means on the dataset we get a model with k clusters:

$$C = c_1, c_2, , , , , c_k => Cluster\_centers$$

Then the VLAD feature vector for an image is given by:

$$v_i = \sum_{x|x=NN(c_i)} (x - c_i)$$

where:

x = d-dimensional descriptor
NN(c_i) = Gives all the descriptors assigned to cluster c_i

This makes $\boldsymbol{v}$ to be $\boldsymbol{k * d}$ sized vector. And hence the VLAD feature vector is nothing but a stretched version of this vector of size $\boldsymbol{1 * (k * d)}$

1.3 Advantages of VLAD over BoVW

- The primary advantage of VLAD over BoW is that in VLAD we add the difference of each descriptor from the cluster mean instead of just keeping the count of number of descriptors belonging to the cluster.
- This adds a more discriminative property to our feature vector as we are treating descriptors of the same cluster differently by adding the weights of each of them.
- This first order statistic adds more information to our feature vector and hence gives us better discrimination for our classifier.
- VLAD has compact visual representations with high discriminative ability, something which BoVW lacked.

1.4 CIFAR10 Dataset

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. The same can also be imported using keras backend in python. The classes that the CIFAR10 dataset contains is shown in Figure 1.

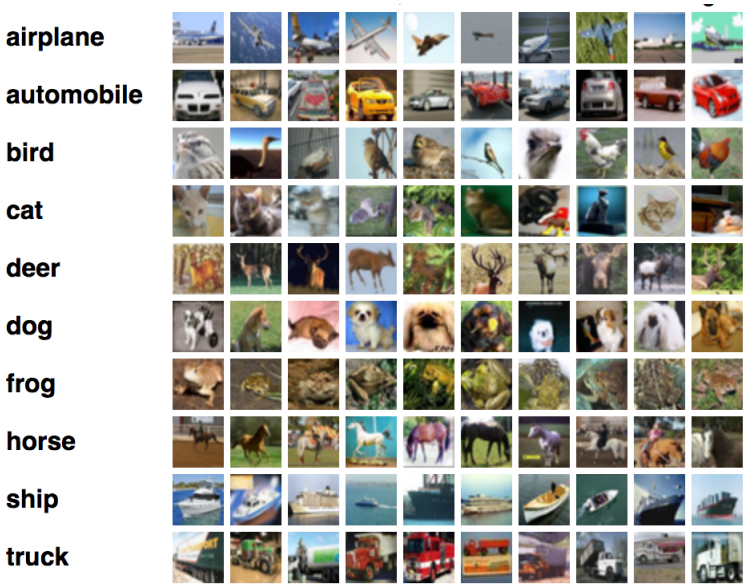


Figure 1: CIFAR10 Dataset

1.5 Pre-Processing

Each image was resized to 150 x 150 while preserving their aspect ratio. Inter Area interpolation and mask was used in-order to prevent loss of essential key-points while resizing.

```
def resize2SquareKeepingAspectRation(self,img, size, interpolation = cv2.INTER_AREA):
    h, w = img.shape[:2]
    c = None if len(img.shape) < 3 else img.shape[2]
    if h == w: return cv2.resize(img, (size, size), interpolation)
    if h > w: dif = h
    else: dif = w
    x_pos = int((dif - w)/2.)
    y_pos = int((dif - h)/2.)
    if c is None:
        mask = np.zeros((dif, dif), dtype=img.dtype)
        mask[y_pos:y_pos+h, x_pos:x_pos+w] = img[:h, :w]
    else:
        mask = np.zeros((dif, dif, c), dtype=img.dtype)
        mask[y_pos:y_pos+h, x_pos:x_pos+w, :] = img[:h, :w, :]
    return cv2.resize(mask, (size, size), interpolation)
```

After this, each image was converted to gray-scale in order to run the SURF algorithm on the images.

1.6 Feature Extraction

The steps followed for feature extraction are :

1. Extracting keypoints and descriptors of image using SURF
2. Applying KMeans on the training data's descriptors to form $\boldsymbol{k}$ clusters
3. Generating VLAD feature vectors

To determine the required number of clusters for KMeans, Elow method was used as shown in Figure 2. The number of clusters was thus selected to be 60.

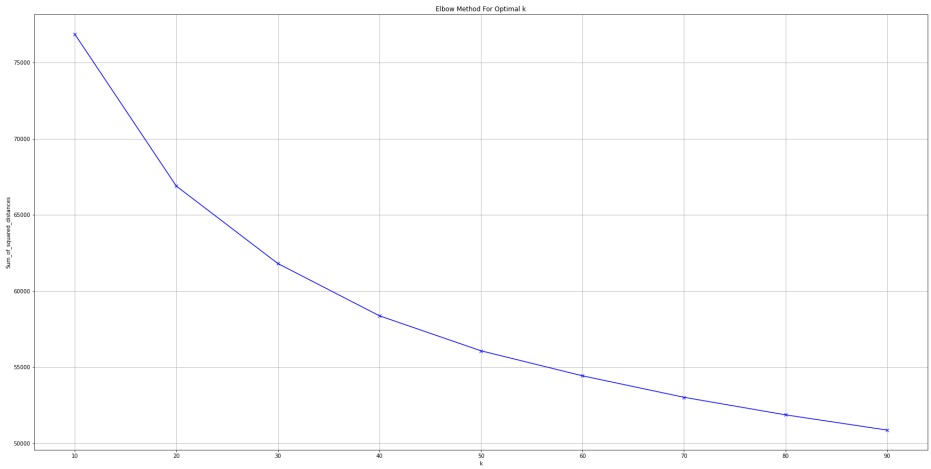


Figure 2: Elbow plot for deciding $\boldsymbol{k}$ for KMeans

After performing VLAD, the dataset was of size (50000 x 3840). This is because the number of clusters was 60 and SURF gives 64 descriptors, Hence the size of VLAD feature vector is $\mathbf{1x(k * d)}$ which is nothing but $\mathbf{1x(60 * 64) = 1x3840}$. And, as there were 50000 images, the dataset was of **(50000 x 3840)**

1.7 Data Visualization

In order to visualize high dimensional data (>3) we use something called T-Distributed Stochastic Neighbouring Entities (t-SNE). t-Distributed Stochastic Neighbor Embedding (t-SNE) is another technique for dimensionality reduction and is particularly well suited for the visualization of high-dimensional datasets. The 2-dimensional feature space of VLAD features of CIFAR10 dataset is shown in Figure 3.

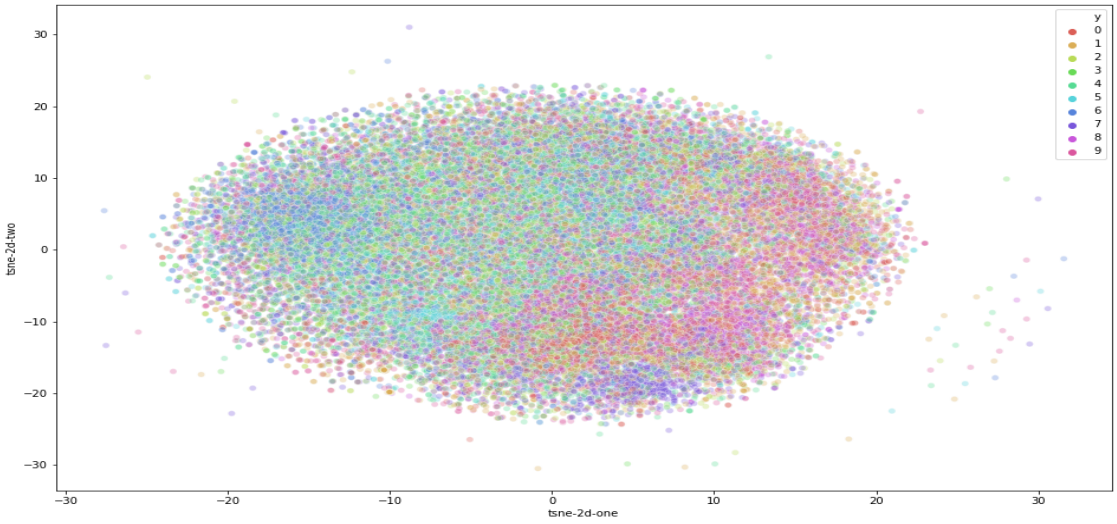


Figure 3: VLAD Data visualization of all classes using TSNE

1.8 Experiments

1.8.1 Principle Component Analysis (PCA)

In order to reduce complexity of a model and avoid over-fitting, dimensionality reduction was an option in order to enhance the model. Principal Component Analysis is a statistical procedure that uses an orthogonal transformation to convert a set of observations into a set of values of linearly uncorrelated variables called **principal components**.
This transformation is defined in such a way that the first principal component has the largest variance and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. The resulting vectors are an uncorrelated orthogonal basis set. The variance with respect to the number of components is shown in Figure 4.

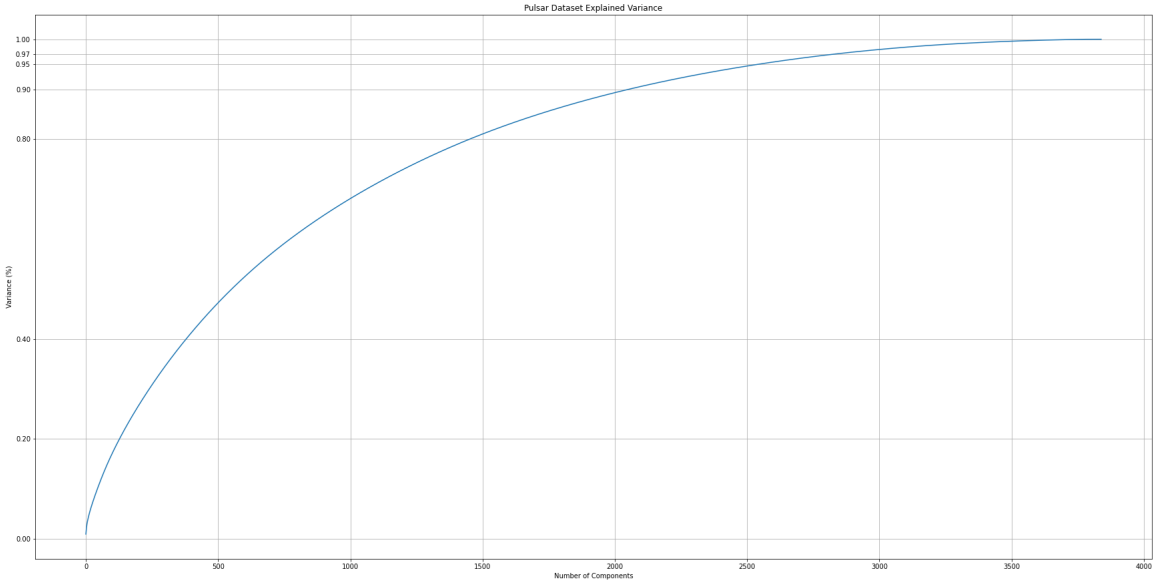


Figure 4: Variance w.r.t number of components

In-order to conserve 95% of variance, the dimension of the dataset was reduced to **2544** from **3840**

1.8.2 Linear Discriminant Analysis (LDA)

Apart from PCA, LDA is also very commonly used dimensionality reduction model. Major difference between PCA and LDA is that PCA is an un-supervised model whereas LDA is a supervised dimensionality reduction model. LDA is hence said to be supervised PCA.
Linear Discriminant Analysis (LDA) is a generalization of Fisher's linear discriminant. It is a method used in statistics, pattern recognition and machine learning to find a linear combination of features that characterized / separates two or more classes of objects or events. This resulting combination may be used as a linear classifier, or for dimensionality reduction before later classification. LDA reduces the dimensionality to the $\boldsymbol{number\_of\_class - 1}$ hence the dimensionality of the dataset became **(50000 x 9)** from **(50000 x 3840)**

1.8.3 SIFT and SURF

SIFT and SURF are two most common keypoint detection algorithms used. SURF is a faster version of SIFT.
For experimentation, both the above algorithms were used for detecting keypoints and generating descriptors. SIFT corresponding to a keypoint generated descriptor of size 128 whereas SURF created a descriptor of size 64. We used the SURF algorithm since it is faster and computationally less expensive.

1.9 Results

After performing data processing and feature extraction, a Support Vector Machine (SVM) and a Random Forest model were trained. 50000 images were used for training and 10000 images for testing. The results are shown in Table 1.

Table 1: Accuracy Table

Model and Experiment	**Accuracy**
SURF + BoVW + SVM	25.5%
SURF + VLAD + SVM	**44%**
SURF + VLAD + Random Forest	35.63%
SURF + VLAD + PCA + SVM	32.81%
SURF + VLAD + LDA + SVM	41.76%

1.10 Problems Faced

One of the major problem faced was the lack of computation power. Process of K-Means clustering, VLAD extraction, PCA/LDA, and model training were computationally intensive for a personal computer (considering the size of the dataset). Processing took lot of hours in one stretch. With RAM limitations and low processing power it was very difficult to gather results. A lot more experiments could have been performed if access to a more powerful GPU was available.

1.11 Conclusion

After seeing the accuracy score we can conclude that VLAD feature extraction algorithm performs better than Bag of visual words (BoVW). It can be seen that despite of reduced length VLAD outperforms BoVW which is considered as one of the golden technique for feature quantization.
VLAD performs approximately 20% better than BoVW. We can conclude that VLAD is an extremely compact, very discriminative and efficient in retrieval and classification tasks than BoVW.

2 Face Recognition

A pipeline of a face recognition system is shown in Figure 5. The implementation of each step is described in the subsequent sections.

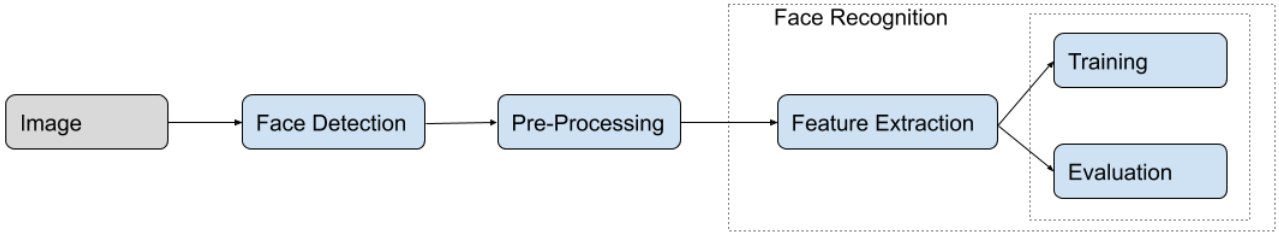


Figure 5: A Face Recognition System

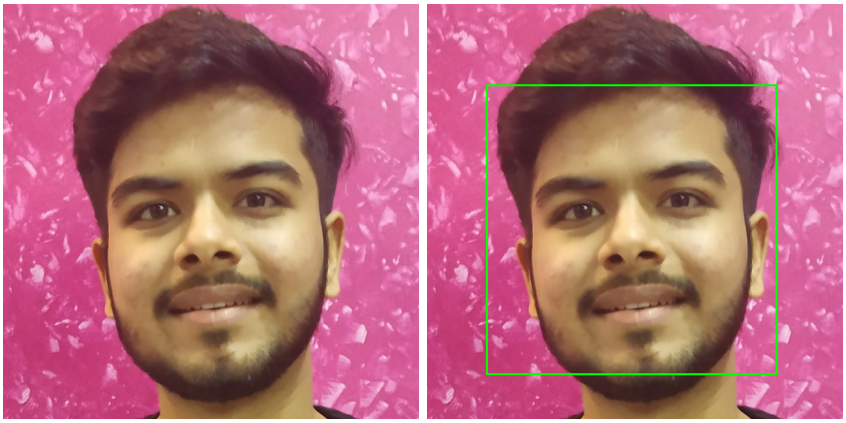
2.1 Dataset

The IIIT-B face dataset containing 20 images each of 43 different people was used for this project. Each image was of **1024*X*1024** pixel resolution.

2.2 Pre-Processing

Pre-processing is a very significant step in order to boost the performance of a face recognition model. The following pre-processing methods were applied to the face-dataset :

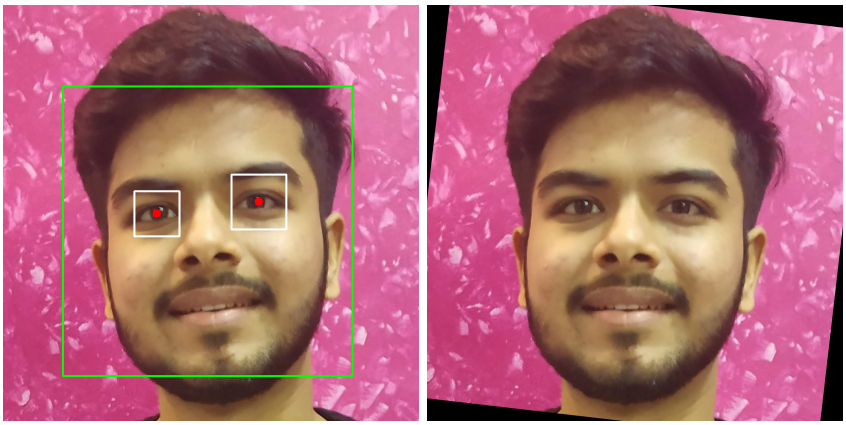
- **Conversion to Grayscale :** In order to avoid distractions, as well as reduce the computational requirement, the images were converted to grayscale. However, the results have been displayed on the original color image for better clarity.
- **Face Detection :** The two commonly used face detection methods are using **Haar Feature-based Cascade classifier** and **HOG + SVM classfier**. Both of these methods were applied, and the best one was chosen. Both the pre-trained classifiers are available for python. The evaluation results are shown in 2.3. The best out of the two was the **Haar Feature-based Cascade classifier**. The bounding box co-ordinates of the face obtained upon face detection was used to obtain the region of interest (the face). The result of face detection on a sample image is shown in Figure 6. Images in which a face was not detected were discarded from the dataset. Poor lighting, pose and angle could have been the possible causes of not being able to detect a face in an image. The face detection is evaluated in 2.3.



(a) Original Image (b) Detected Face

Figure 6: Face Detection

- **Eye detection :** A pre-trained Haar Feature-based Cascade Classifier was used to detect the eyes (this classifier is also capable of detecting eyes in faces with glasses). The eye detection was performed only in the region of interest which was obtained in the previous step. The centers of the eyes were then determined using the bounding box co-ordinates of the eyes. Poor lighting conditions, closed eyes, dark spectacles etc can hinder eye detection. As a result, both the eyes were not detected in some images. These images were still retained since the face was detected, and features can be extracted as explained in subsequent sections. The eye detection is evaluated in 2.3.
- **Face straightening :** After detecting both eyes, the image was straightened by calculating the angle between the eyes, and appropriately **rotating** the image. The results of eye detection and subsequent image straightening is shown in Figure 7.

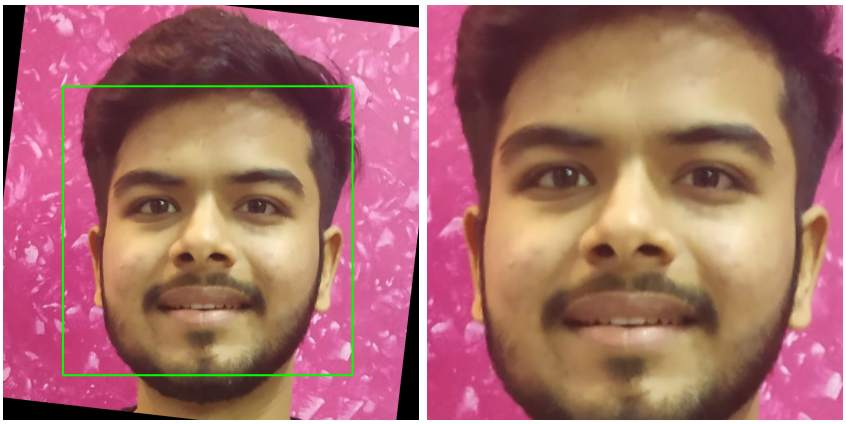


(a) Eye detection (b) Straightened face

Figure 7: Eye Detection and Face Straightening

- **Face cropping :** In order to improve the performance of face recognition, the external distracting information such as backgrounds, clothes, accessories etc. were removed by cropping the image to contain only the face.
- **Image resizing :** The images were resized to **256*X*256** pixels.

The results of face detection after straightening (to find the new region of interest), and the final cropped image that was obtained is shown in Figure 8. Figure 8b is the resulting **256*X*256** image.



(a) Face detection (b) Resized image

Figure 8: Face cropping and Resizing

2.3 Evaluation : Face and Eye Detection

The performance of face and eye detection is shown in Figure 9. The face detector successfully detected the face in **787** out of **859** images. This is an accuracy of **91.61%**. The accuracy of the HOG+SVM based classifier is **80%**. The eye detector successfully detected eyes in **645** out of **787** images, which is an accuracy of **81.95%**.

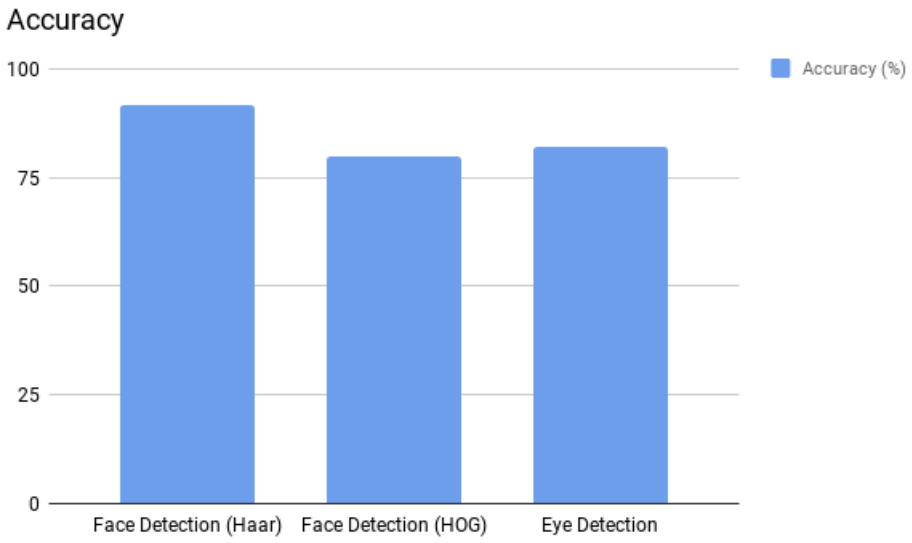


Figure 9: Performance of Face and Eye Detection

2.4 Feature Extraction

We applied three different feature extraction techniques to the pre-processed data set : For **PCA**, we sent the extracted features through a Fully Connected layer to allow us to classify the output whereas for **LDA** and **LBP**, we used the Fisher faces and **LBPH** Face Recognizer algorithms available in OpenCV.

2.4.1 PCA Implementation

Provided below is the general sequence of steps followed to implement the PCA algorithm:

1. Find the mean vector
2. Assemble all the data samples in a mean adjusted matrix.
3. Create the covariance matrix.
4. Compute the Eigen vectors and Eigen values.
5. Compute the basis vectors
6. Represent each sample as a linear combination of basis vectors.

2.4.2 Local Binary Patterns - LBP

Local Binary Patters (LBP) is a type of visual descriptor used for classification. It was first described in 1994. It has been found to be a powerful feature for texture classification. The LBP feature vector is created in the following manner :

1. Divide the examined window into cells.
2. For each pixel in a cell, compare the pixel to each of its 8 neighbors. Follow the pixels along a circle, i.e. clockwise or counter-clockwise.
3. Where the center pixel's value is greater than the neighbor's value, write "0". Otherwise, write "1". This gives an 8-digit binary number.
4. Compute the histogram, over the cell, of the frequency of each "number" occurring (i.e., each combination of which pixels are smaller and which are greater than the center). This histogram can be seen as a 256-dimensional feature vector.
5. Normalize the histogram.
6. Concatenate the normalized histograms of all cells. This gives a feature vector for the entire window.

2.5 Results

The features obtained when applying PCA, LDA, and LBP were then used for Facial Recognition.
A graph between the accuracy of prediction and the number of Eigen Faces while using PCA is shown in Figure 10. This graph shows the variation in Top-1 accuracy of PCA feature extraction as we vary the number of Eigen Faces retained. The results of Face Recognition are shown in Table 2.

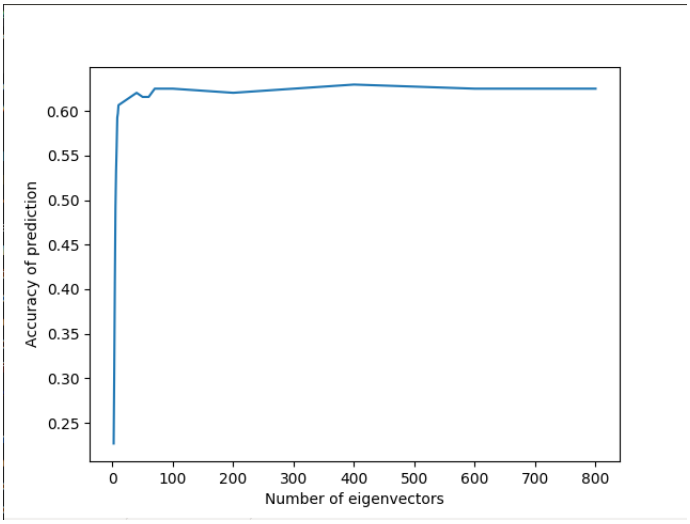


Figure 10: Number of Eigen faces vs Accuracy

	Top 1	Top 3	Top 10
PCA	0.6899	0.7868	0.8798
LDA	0.4263	0.5155	0.6007
LBP	0.8643	0.9341	0.9651

Table 2: Top-k results

2.6 Observations and Experiments

- LDA is performing worse than PCA. This is mainly because while PCA encodes information in an orthogonal linear space, LDA encodes discriminating information in a linear separable space using bases that are not necessarily orthogonal. While it is generally believed that LDA algorithms are superior to PCA algorithms, some recent work does show that when the training data set is small, PCA can outperform LDA and also PCA is less sensitive to different training data sets.
- From the Eigen Faces vs Accuracy graph, it is clear that retaining around 100 Eigen vectors is the ideal number for maximum accuracy.
- We can see that feature extraction is important and essential when it comes to Face Recognition as we effectively dealt with only 1/4 the amount of data for each image while maintaining a relatively high accuracy. This makes computation time tangible for real life applications.

References

[1] Vector of Locally Aggregated Descriptors - https://ameyajoshi005.wordpress.com/2014/03/29/vlad-an-extension-of-bag-of-words/

[2] PCA vs LDA Concept Comparision - https://sebastianraschka.com/Articles/2014_python_lda.html

[3] Image Preprocessing Methods in Face Recognition - https://ieeexplore.ieee.org/document/5780626

[4] Study of the Pre-processing Impact in a Facial Recognition System - https://www.semanticscholar.org/paper/Improving-Face-Recognition-Rate-with-Image-Dharavath-Talukdar/78e20189b96261188b73a5133a84d8e9f82b799a