Face Detection, Recognition, and Clustering

Professor - Dr. Pratik Mazumder

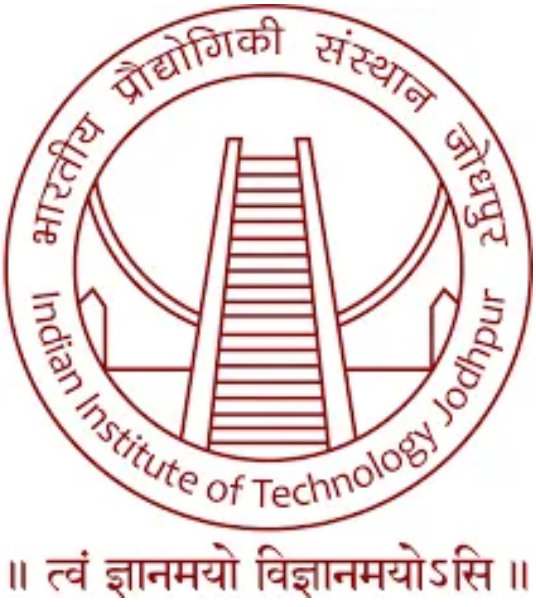
Team Members -

Kashvi Jain (B21CS037) ,

Arun Raghav S (B21CS015) ,

Vidit Garg (B21AI045) ,

Vidit Ajay Agrawal (B21AI058)



- ## MOTIVATION

Imagine the joy of flipping through an album filled with memories, each picture capturing a moment in time. Now, picture being able to effortlessly organize these cherished moments, grouping them by the faces of the people we love. This is the vision that drives our project: ***to create a face grouping system inspired by the seamless functionality of Google Photos.***

- ## PROBLEM STATEMENT

Facial recognition and detection technology has gained significant traction in recent years due to its wide range of applications in security, surveillance, authentication, and personalized user experiences. However, despite advancements, several challenges persist in the field of face detection, recognition, and clustering, necessitating innovative solutions to enhance accuracy, efficiency, and reliability.

Face Detection: The primary challenge lies in accurately detecting faces within images, especially under varying lighting conditions, poses, occlusions, textures, and backgrounds. Existing face detection algorithms often struggle with detecting faces in crowded scenes, low-resolution images, or when faces are partially obscured.

Face Recognition: Once faces are detected, the next challenge is to accurately recognize and identify individuals from a database or a set of known faces. This involves handling variations in facial expressions, aging, changes in appearance (e.g., hairstyle, accessories), and different viewpoints.

Face Clustering: In scenarios where multiple instances of the same individual appear across different images or video frames, the task of clustering these instances into coherent groups poses a significant challenge. This involves dealing with variations in lighting, poses, expressions, and occlusions while ensuring that faces belonging to the same individual are correctly grouped together.

The key objective is to develop robust face detection algorithms capable of accurately localizing faces under diverse environmental conditions and enhance face recognition models to achieve high accuracy in identifying individuals while mitigating the impact of variations in appearance. Then, efficient face clustering techniques are implemented to group instances of the same individual across multiple images or video frames.

- **METHODOLOGY**

Face Detection → Several face detection algorithms have emerged over the years, each with its strengths and applications:

Viola-Jones Algorithm: This classic algorithm uses ***Haar-like features*** and a cascade of classifiers to efficiently detect faces in images. It's fast and robust, making it suitable for real-time applications.

Histogram of Oriented Gradients (HOG): HOG extracts local intensity gradients and computes histograms of gradient orientations to detect faces. It's effective in various lighting conditions but may struggle with occluded faces.

Convolutional Neural Networks (CNNs): CNNs, especially architectures like ***ResNet, VGG, and MobileNet,*** have shown remarkable performance in face detection tasks. They learn hierarchical representations of features, making them highly accurate but computationally intensive.

Deep Learning-based Detectors: Recent advancements like Single Shot Multibox Detector (SSD), You Only Look Once (YOLO), and Faster R-CNN integrate deep learning with object detection, achieving impressive accuracy and speed for face detection tasks.

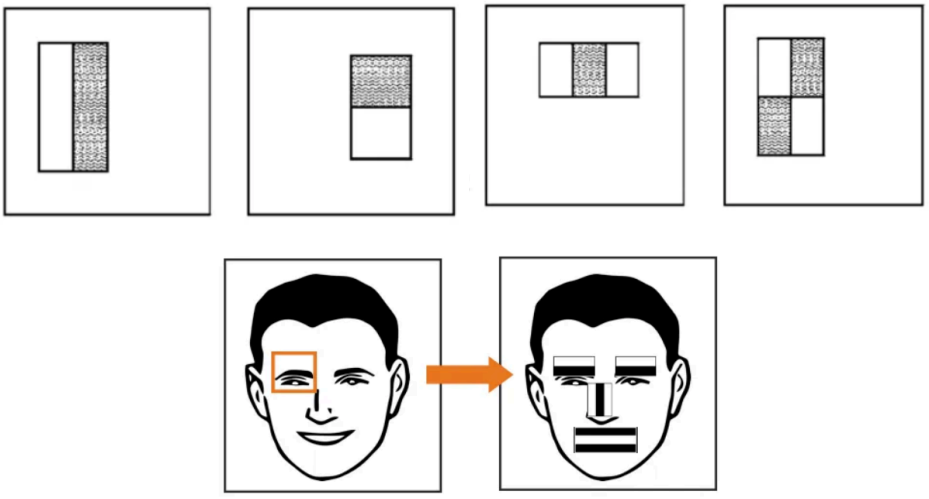
MTCNN (Multi-task Cascaded Convolutional Networks): MTCNN employs a ***cascaded architecture to detect faces, refine bounding boxes, and perform facial landmark detection simultaneously***. It's widely used for its accuracy and efficiency.

So for the following project, we have used some of these techniques to implement the feature of face detection from a particular image and use these faces for further functionalities of the face grouping system.

Describing some of these techniques in detail →

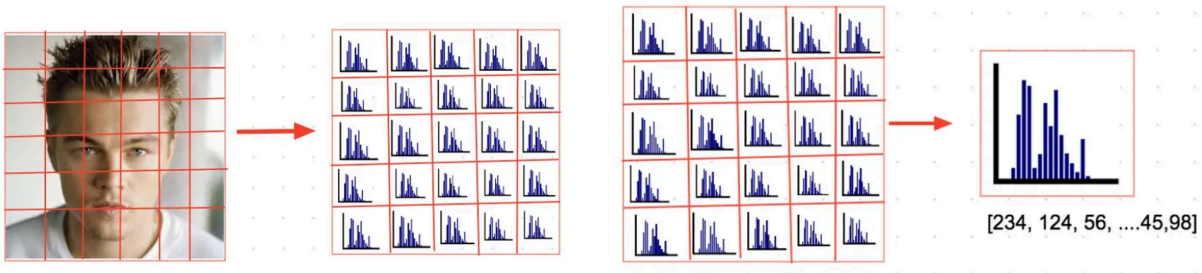
The Viola-Jones algorithm is a classic face detection method known for its speed and accuracy. It operates by dividing an image into small rectangular regions and evaluating simple features called Haar-like features within each region. These features are based on the contrast between adjacent rectangular regions of the image.

The algorithm uses a cascade of classifiers, where each classifier is trained to determine whether a particular region of the image contains a face or not , the training data majorly includes two major classes of data one which consists of the faces of different human beings and the other are nonfaces images such as plants, trees, cars and other objects similar to these. It employs AdaBoost, a machine learning technique, to select a subset of the most informative haar features/haar vectors and train the classifiers(generally classifiers such as SVM are used). During detection, the algorithm slides a window over the image, applying the cascade of classifiers to each region. If a region passes all classifiers, it's considered a face. This approach enables rapid face detection, making it suitable for real-time applications like video surveillance and facial recognition systems.

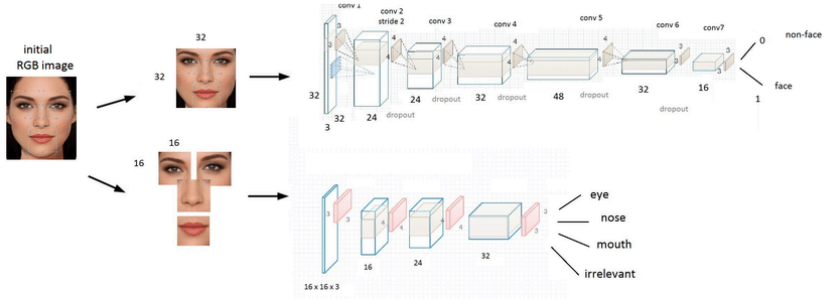


The Histogram of Oriented Gradients (HOG) face detection technique extracts local intensity gradients from an image and computes histograms of gradient orientations to detect faces.

- ***Gradient Computation:*** HOG begins by calculating the gradient magnitude and orientation for each pixel in the image. This step captures the local intensity variations and edges in the image.
- ***Cell Division:*** The image is divided into small cells, typically square regions. Each cell contains a fixed number of pixels.
- ***Histogram Calculation:*** Within each cell, a histogram of gradient orientations is computed. The histogram bins represent different orientations of gradients, quantizing the gradient orientations into discrete values.
- ***Block Normalization:*** To enhance robustness to changes in illumination and contrast, adjacent cells' histograms are grouped into larger blocks. Each block undergoes a normalization process, which typically involves dividing the histogram values by a normalization factor computed from the block's content.
- ***Descriptor Formation:*** The histogram values from all blocks are concatenated to form a descriptor, representing the local gradient distribution within the image.
- ***Detection:*** A sliding window is moved across the image, and the descriptor is computed for each window position. The descriptors are then fed into a classifier, such as a Support Vector Machine (SVM), to determine whether a face is present in the window.



CNN-based Approach for Face Detection → CNN-based face detection employs deep learning to automatically learn features from raw pixel data. Through convolutional layers and training on labeled datasets, it discerns whether image patches contain faces. Renowned for high accuracy and adaptability, it's pivotal in various applications, including surveillance and facial recognition systems.



<u>Face Recognition</u> → Face recognition is a technology that identifies or verifies individuals by analyzing and comparing facial features from digital images or video frames. It works by capturing facial features, such as the distance between the eyes or the shape of the nose, and matching them against a database of known faces.

Traditional Feature-Based Methods: These methods extract and analyze specific facial features, such as the eyes, nose, and mouth, to create a unique facial template for each individual. Matching involves comparing these templates for similarity.

Eigenfaces: Eigenfaces represent faces as vectors in a high-dimensional space and use Principal Component Analysis (PCA) to extract the most significant features. Faces are recognized by comparing their eigenfaces with those stored in a database.

Deep Learning-based Methods: Deep learning techniques, particularly Convolutional Neural Networks (CNNs), have revolutionized face recognition. CNNs automatically learn features from raw pixel data, leading to superior accuracy and robustness.

3D Face Recognition: This method utilizes depth information to capture the three-dimensional structure of the face, making it more resistant to variations in pose and lighting.

Face Detection + Recognition: In this approach, face detection algorithms first locate faces in an image or video frame, which are then passed to a recognition algorithm for identification or verification.

Some of the techniques that we have used for the project are →

Eigenfaces is a technique used for face recognition that represents facial images as vectors in a high-dimensional space. A dataset of facial images is collected, typically comprising grayscale images of faces with consistent alignment and lighting conditions. ***For the implementation of the project we have used the ATT Database of images.***

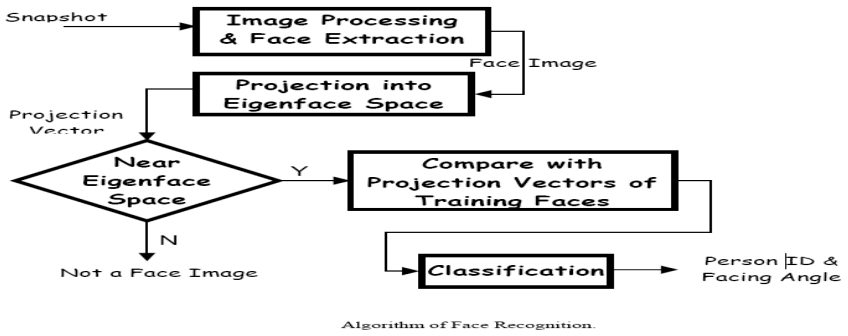
Preprocessing: Each facial image is preprocessed to standardize factors like size, alignment, and illumination. This ensures consistency across the dataset.

Feature Extraction: Principal Component Analysis (PCA) is applied to the preprocessed images to extract the most significant features. PCA reduces the dimensionality of the data by finding the eigenvectors (principal components) that capture the maximum variance in the dataset.

Eigenface Calculation: The eigenvectors obtained from PCA represent the ***eigenfaces.*** These eigenfaces are the basis vectors of the face space and encode variations in facial appearance across the dataset.

Face Representation: Each facial image in the dataset is then represented as a linear combination of the eigenfaces. This representation captures the unique characteristics of each face in terms of the eigenfaces' coefficients.

Recognition: To recognize a face, a new facial image is projected onto the eigenface space by computing its eigenface coefficients. These coefficients are compared to those of known faces in the dataset using metrics like Euclidean distance or cosine similarity. The face is recognized as the closest match based on these coefficients.



Algorithm of Face Recognition.

Deep Learning based approaches for face recognition → Deep learning-based methods for face recognition utilize Convolutional Neural Networks (CNNs) to automatically learn features from raw pixel data.

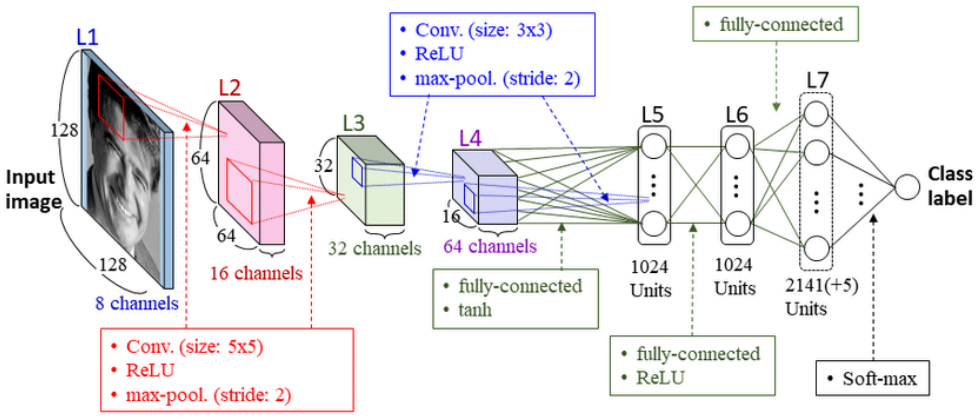
A CNN architecture is designed, comprising convolutional layers, pooling layers, and fully connected layers. These layers extract hierarchical features from the input images.

Training: The CNN is trained on the dataset using backpropagation and optimization techniques like gradient descent. During training, the network learns to recognize patterns and features relevant to face recognition by adjusting its parameters.

Feature Learning: The convolutional layers learn to extract features like edges, textures, and facial structures from the input images. As the network progresses through the layers, it learns to capture increasingly abstract and discriminative features.

Classification: The fully connected layers of the CNN are responsible for mapping the learned features to specific individuals or identities. The network outputs a probability distribution over the known identities, indicating the likelihood of each identity given the input image.

Recognition: To recognize a face, the input image is fed into the trained CNN, which outputs a probability distribution over the known identities. The face is recognized as the identity with the highest probability.



Face clustering → Face clustering is the process of grouping together similar faces in a collection of images or videos without prior knowledge of the identities of individuals. Here's a brief overview and some techniques:

Face clustering aims to organize a set of faces into groups based on similarity in facial appearance. It's useful for tasks like organizing photo collections, social media tagging, and content-based image retrieval.

Feature Extraction/Feature embeddings: Extract facial features such as embeddings using techniques like deep learning-based face embeddings (e.g., FaceNet, VGGFace). These embeddings represent faces in a high-dimensional feature space. We have also trained an autoencoder, the latent space generated can be a candidate for the image feature map. Using these feature mapsvid we can cluster images.

Distance Metrics: Define a distance metric (e.g., Euclidean distance, cosine similarity) to measure the similarity between pairs of face embeddings or we can also use methods like SIFT and SSIM score to calculate similarities between images. Set a threshold on the distance metric or the obtained scores to determine when to merge or split clusters. Faces with distances below the threshold are grouped together.

Clustering Algorithms: Apply clustering algorithms to group similar face embeddings. Commonly used algorithms include K-means clustering, hierarchical clustering, and spectral clustering.

Face clustering techniques play a crucial role in organizing large-scale face datasets and facilitating downstream tasks like face recognition and analysis.

Some of the techniques that we have used for clustering of faces are →

Feature Extraction: Deep neural networks are trained on large datasets of facial images to learn representations that capture important facial features. These networks, like FaceNet or VGGFace ***(in our case we have used the VGGFace (pre-trained model))***, comprise convolutional layers that extract hierarchical features from input images.

Embedding Formation: Once trained, the deep network maps each facial image to a fixed-length vector in a high-dimensional feature space. This vector, known as an embedding, encodes the unique characteristics of the face.

Semantic Similarity: The embeddings are designed to preserve semantic similarity, meaning that similar faces are mapped to nearby points in the embedding space. This enables measuring the similarity between faces by computing distances or similarities between their embeddings.

Distance Metric: Common distance metrics like Euclidean distance or cosine similarity are used to quantify the similarity between face embeddings. A smaller distance or a higher similarity score indicates greater resemblance between faces.We have used Cosine similairty for comparing the relation or similarity between the two faces.

We have used the below two methods where we take two images and performing comparison between the images using the SIFT score and the SSIM score →

SIFT Score: The SIFT (Scale-Invariant Feature Transform) score measures the similarity between two images based on distinctive features detected by the SIFT algorithm. It computes the number of matches between keypoints extracted from both images, providing a robust measure of similarity invariant to changes in scale, rotation, and illumination.

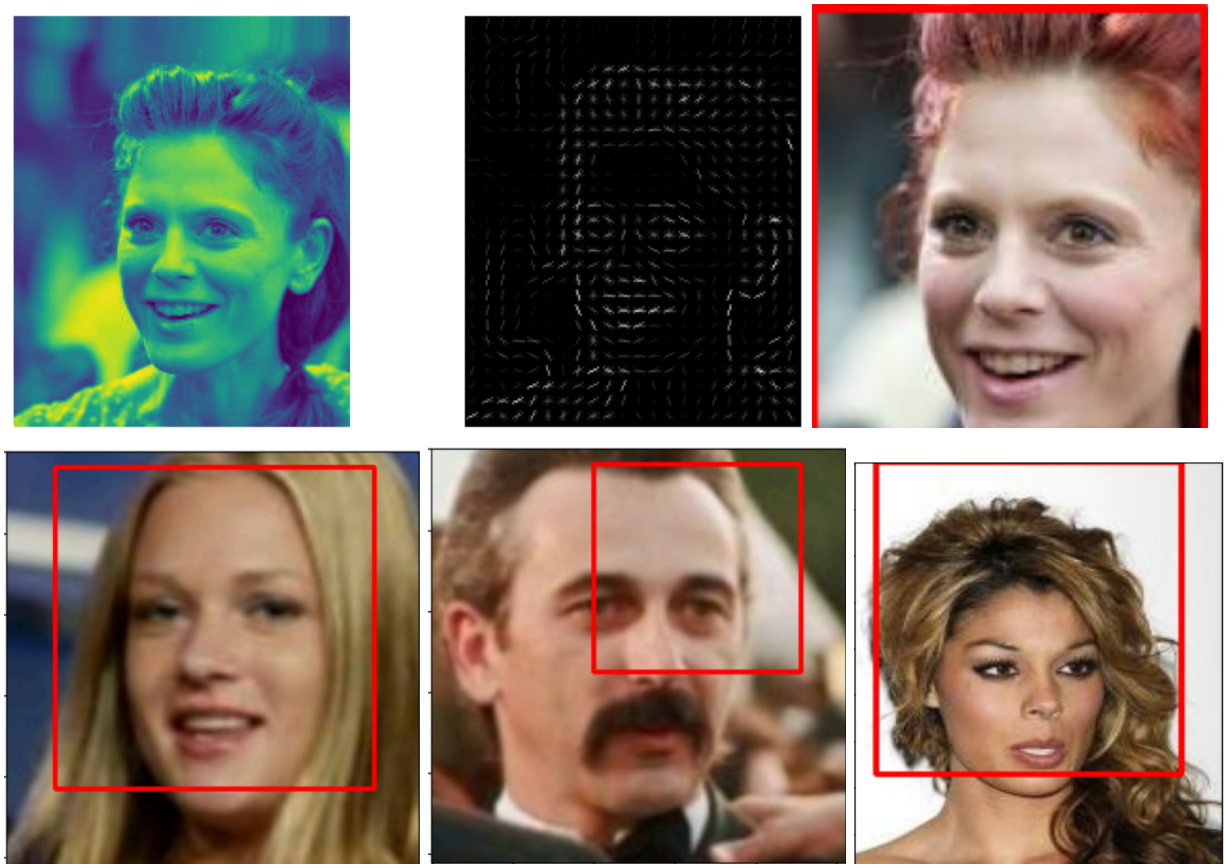
SSIM Score: The SSIM (Structural Similarity Index Measure) score assesses the similarity between two images by comparing their structural information. It considers luminance, contrast, and structure, producing a value between -1 and 1, where 1 indicates

perfect similarity. SSIM is commonly used in image quality assessment and image processing tasks.

RESULTS →

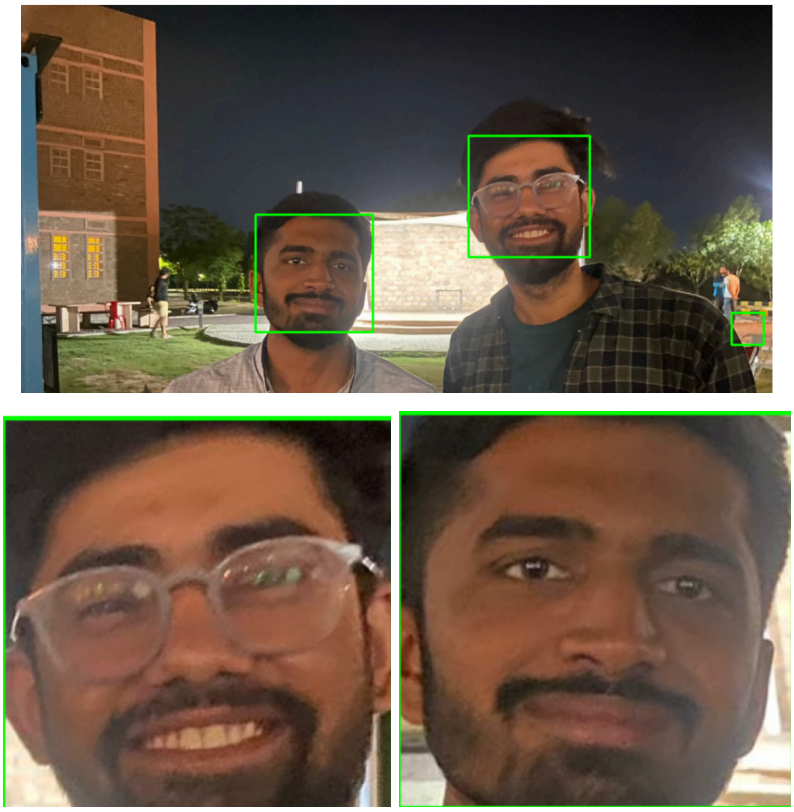
Face Detection →

So for the face detection implementation, we used three major approaches where the first approach was face detection using the Hog descriptor, the other was face detection using the Haar classifiers and the third one was face detection using the deep learning methods.



The above figure displays the original image on to the right and the middle in the centre shows the Hog features that capture information about the edges, textures, and structural details of the face, And the last image is the bounding box representation of a face where with the use of sliding window we traverse the input image and find which part of the

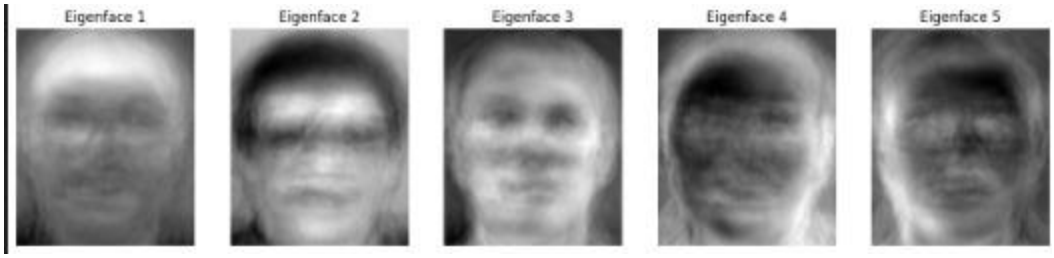
image are classified as face and all the windows classified as face are then aggregated and a bounding box is formed for the overlapping windows.



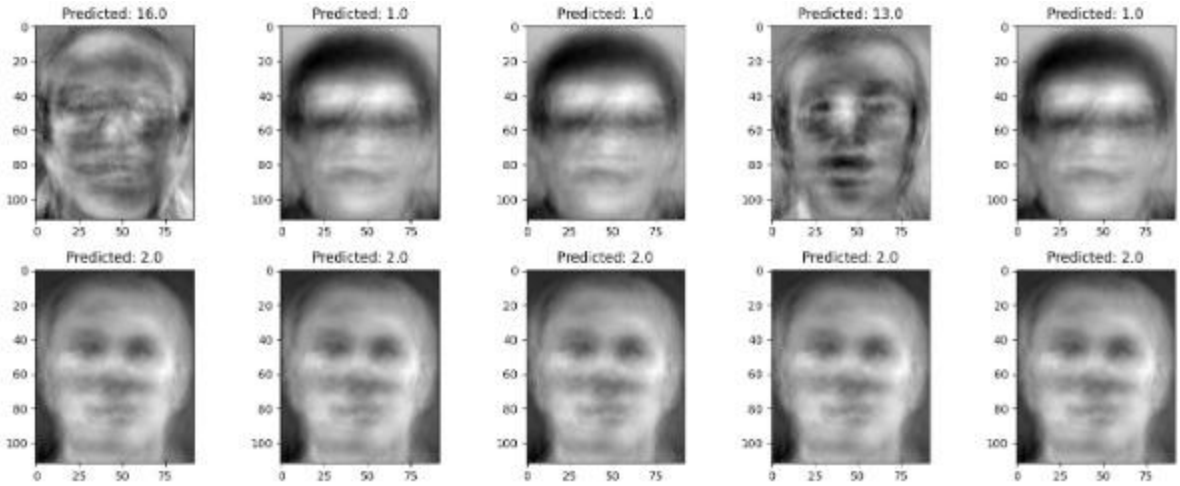
The above figure shows the results of the ***implementation of the Haar feature / Haar Cascade classification*** where the left image shows the bounding boxes or the face detected in the particular input figure and the left image is the face extracted using the region of the bounding box formed.

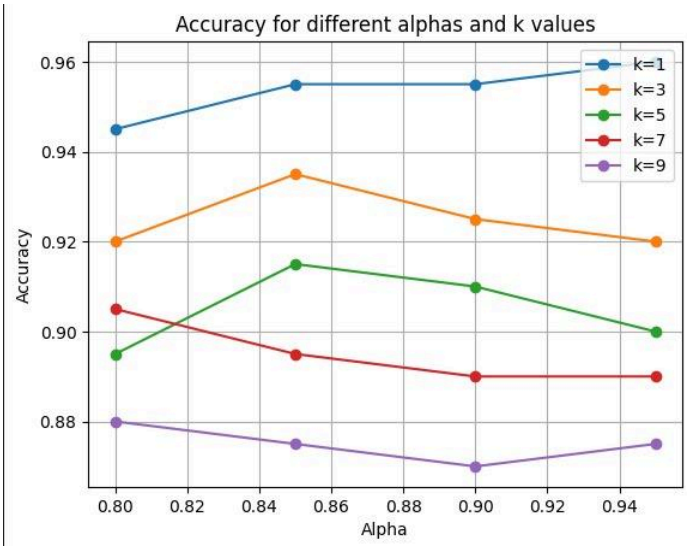
Face Recognition → For the face recognition, first, the images are centralized by subtracting the mean face from each individual face image.

The training and test data are projected onto the selected eigenvectors (eigenfaces) to transform the original high-dimensional data into a lower-dimensional subspace.

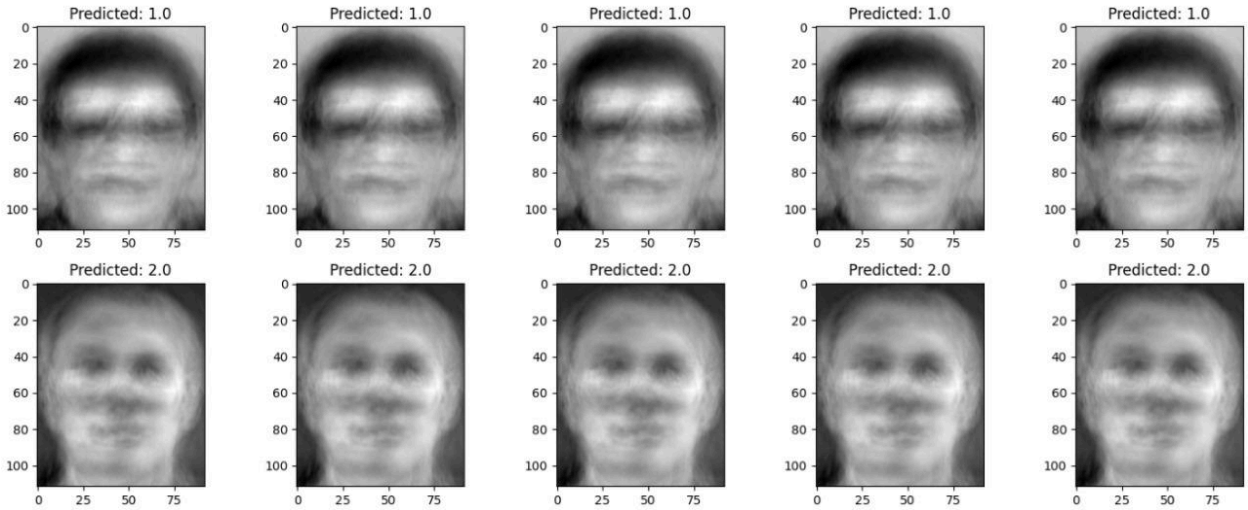


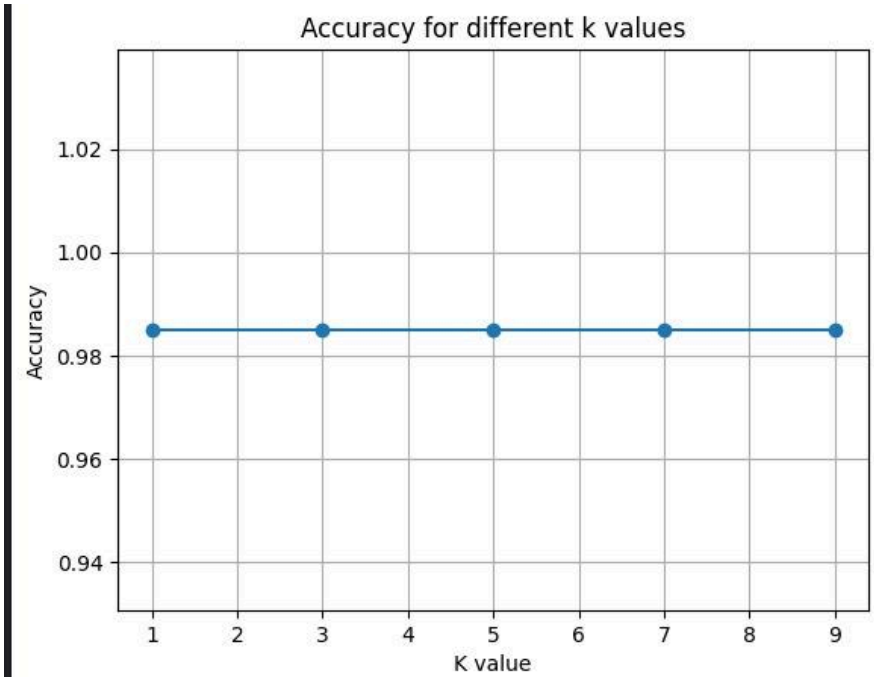
KNN results:



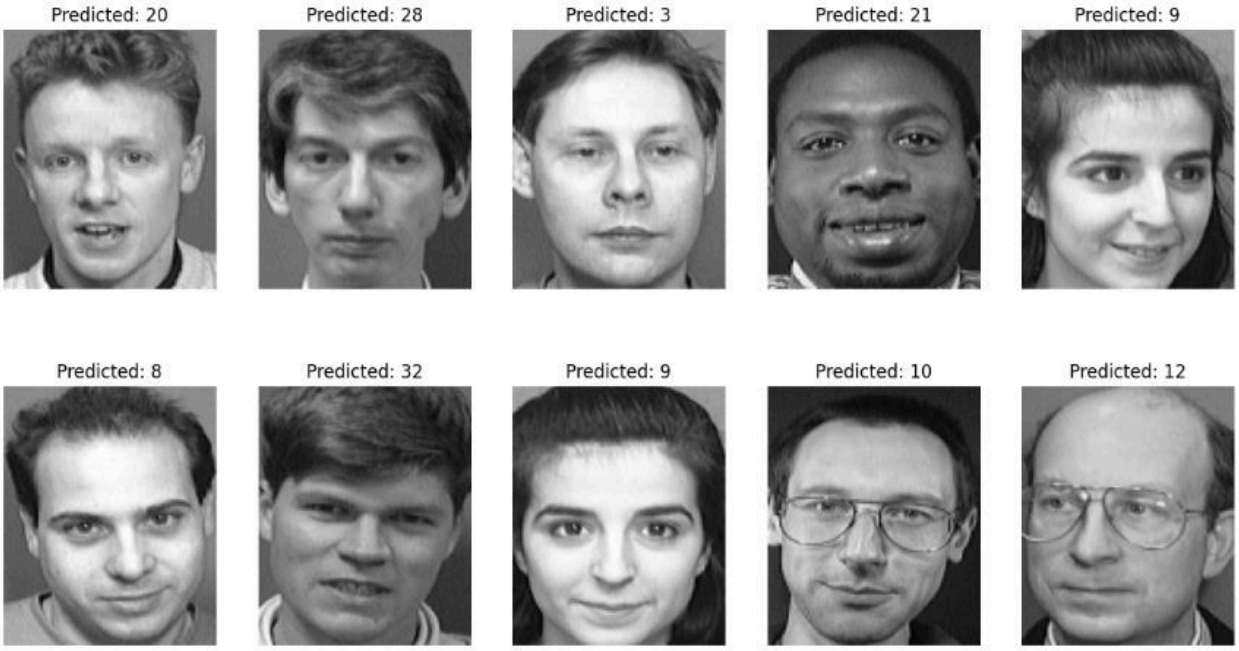


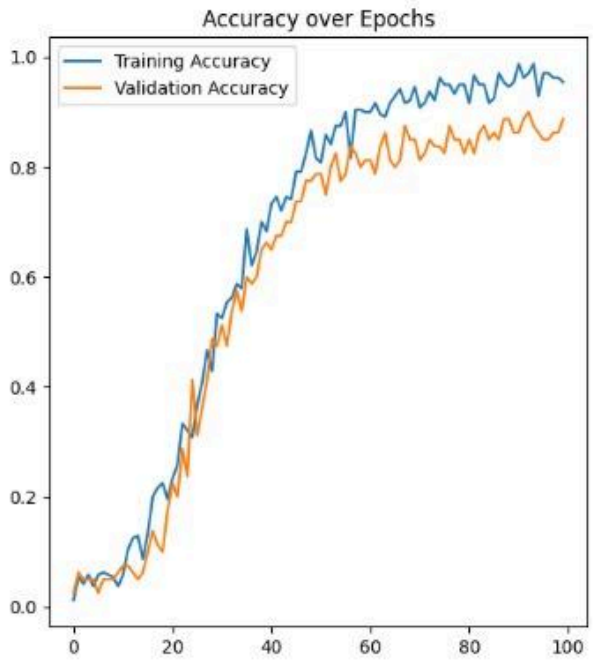
Then, within-class scatter matrix and the between-Class Scatter Matrix are computed. Eigenvalues and eigenvectors are extracted from the generalized eigenvalue problem $SW{-}1\ SB$. The training and test data are projected onto the selected eigenvectors using the projection matrix. Then, KNNis applied.

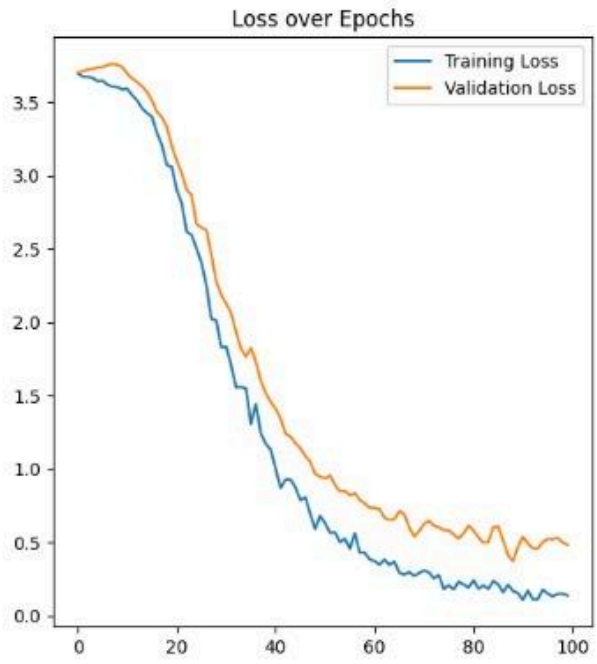




Results for DL model:



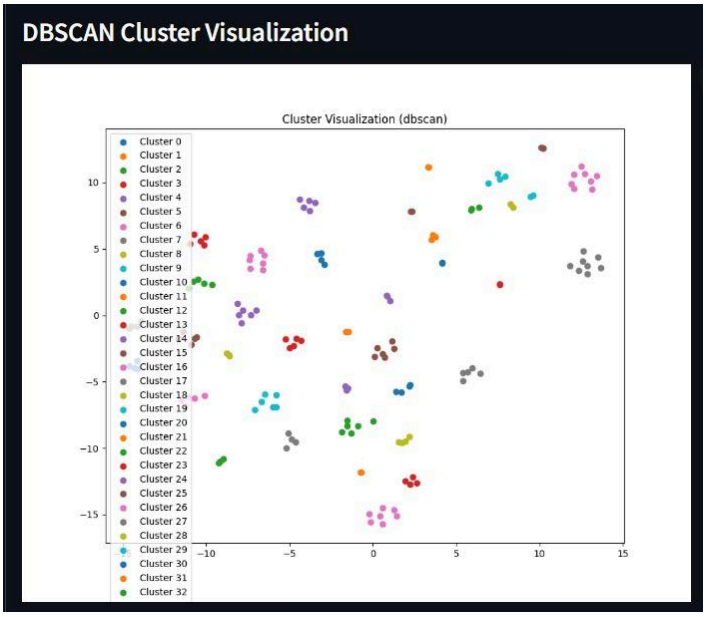


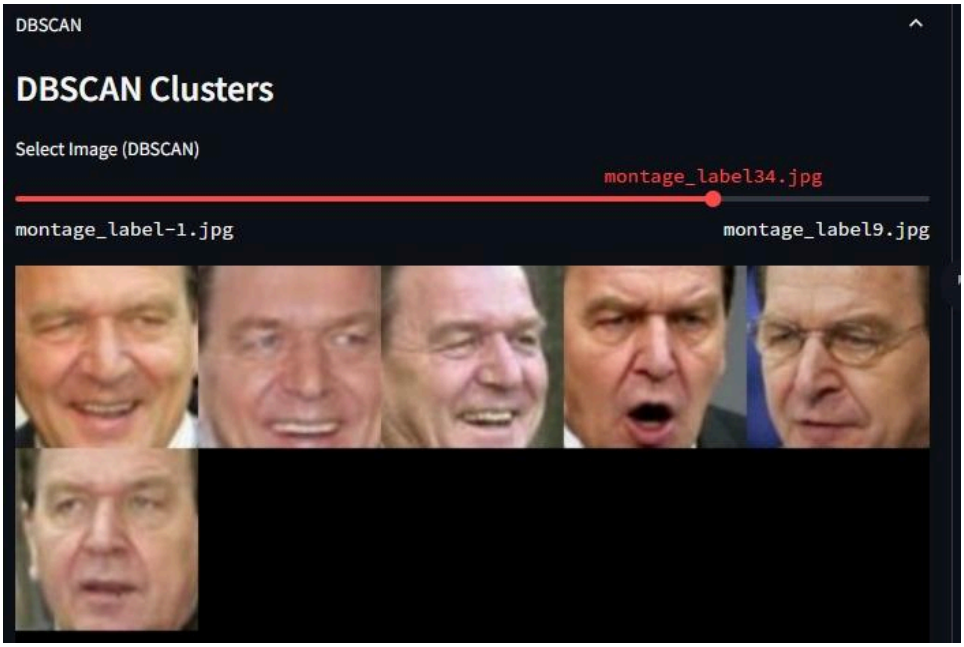


Face Clustering →

Silhouette Scores

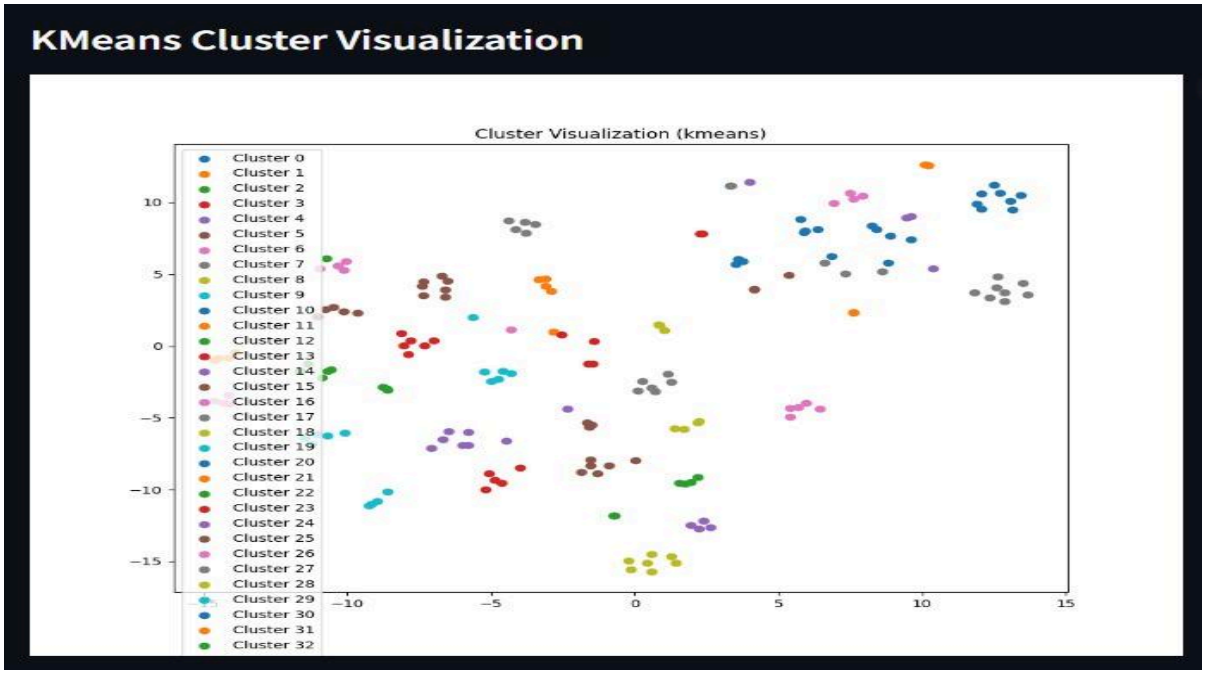
DBScan Clusters

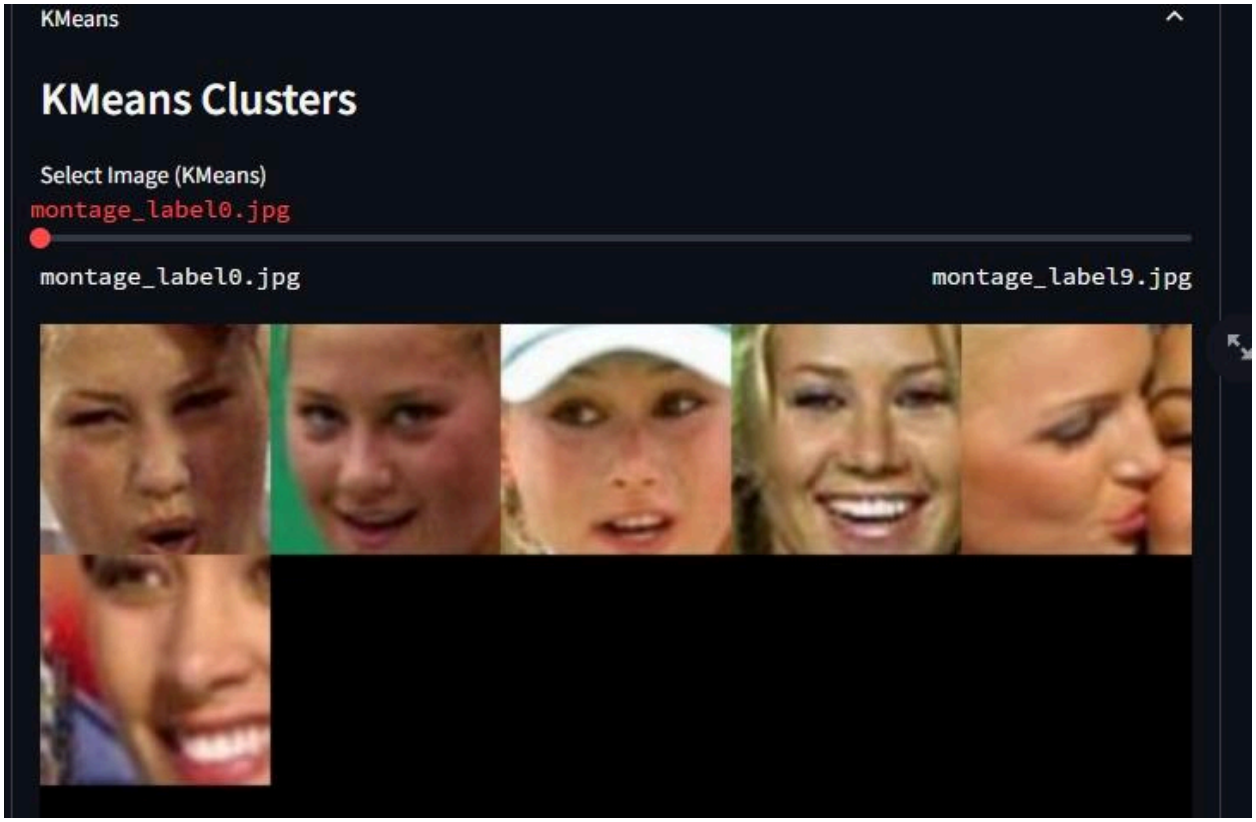

DBSCAN Cluster Visualization
Cluster Visualization (dbscan)
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Cluster 8
Cluster 9
Cluster 10
Cluster 11
Cluster 12
Cluster 13
Cluster 14
Cluster 15
Cluster 16
Cluster 17
Cluster 18
Cluster 19
Cluster 20
Cluster 21
Cluster 22
Cluster 23
Cluster 24
Cluster 25
Cluster 26
Cluster 27
Cluster 28
Cluster 29
Cluster 30
Cluster 31
Cluster 32
10
5
0
−5
−10
−15
−10
−5
0
5
10
15

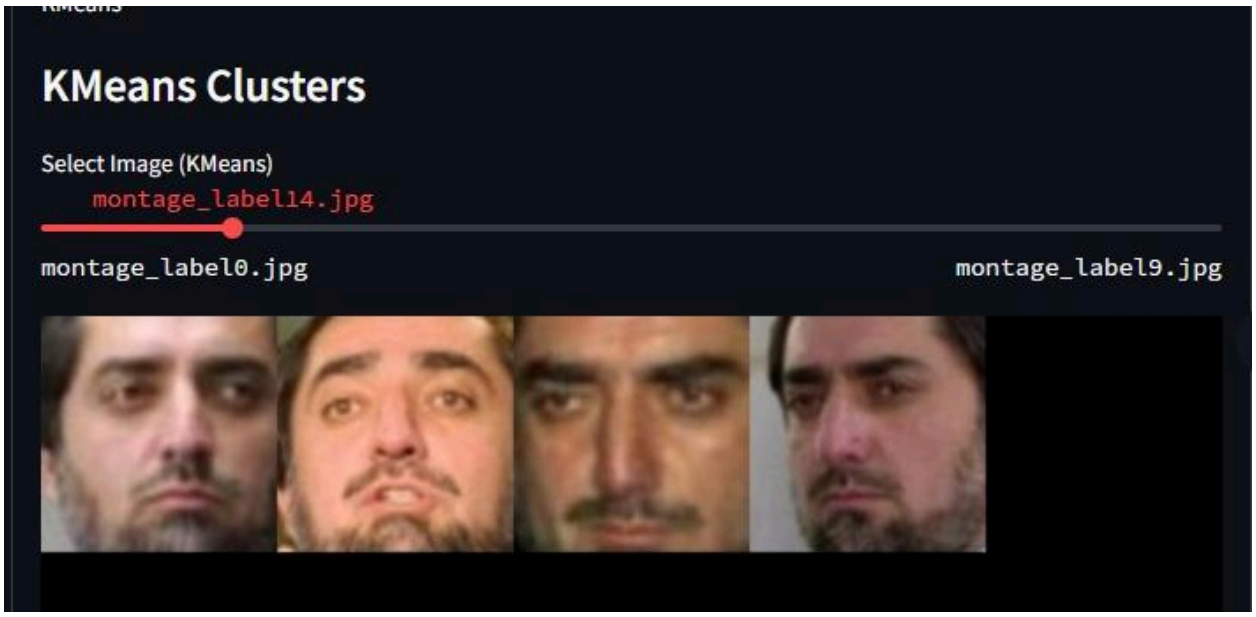

DBSCAN
DBSCAN Clusters
Select Image (DBSCAN)
montage_label34.jpg
montage_label-1.jpg
montage_label9.jpg

Kmeans Clusters

Cluster Visualization

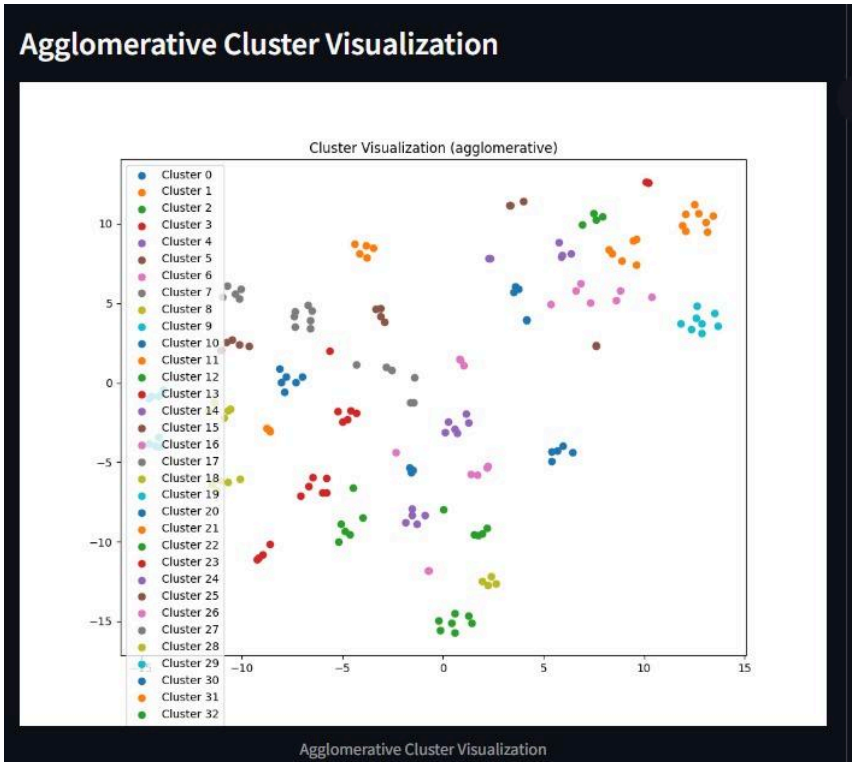


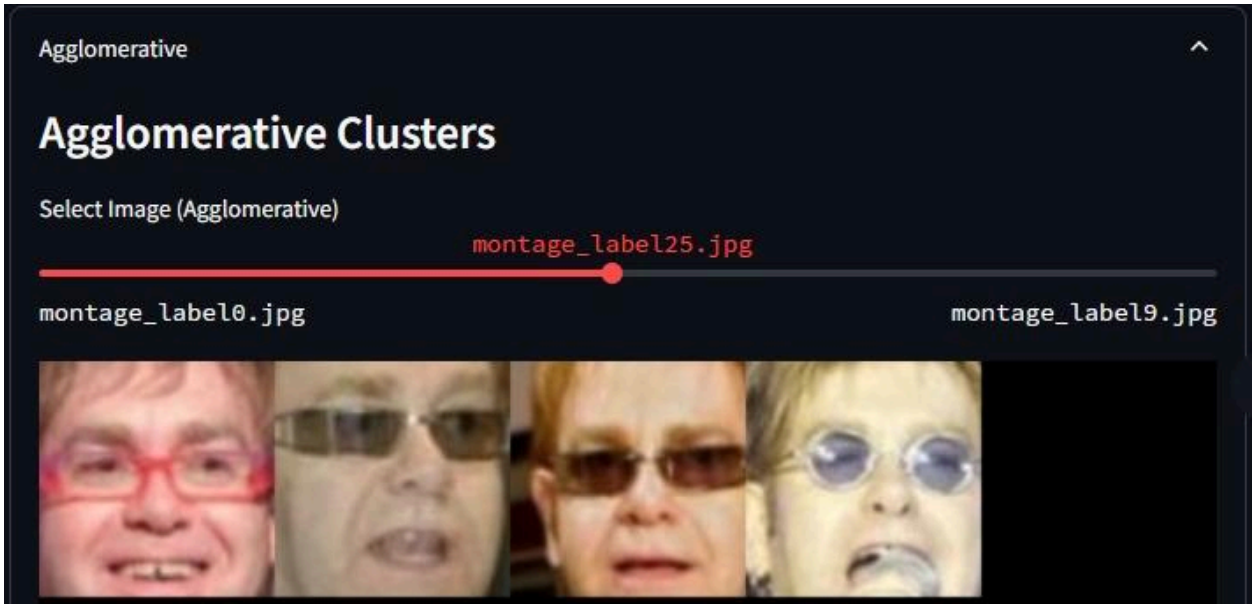

KMeans
KMeans Clusters
Select Image (KMeans)
montage_label0.jpg
montage_label0.jpg
montage_label9.jpg


KMeans Clusters
Select Image (KMeans)
montage_label14.jpg
montage_label0.jpg
montage_label9.jpg

Agglomerative Clusters

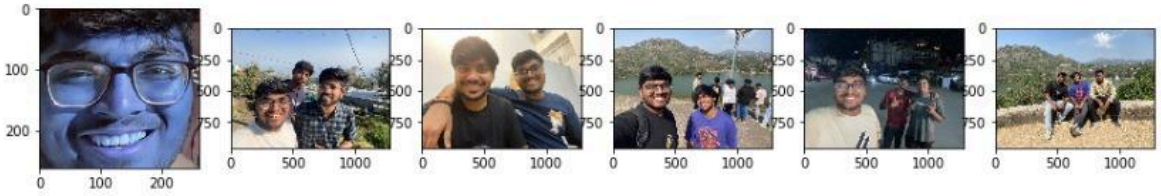
Cluster Visualization





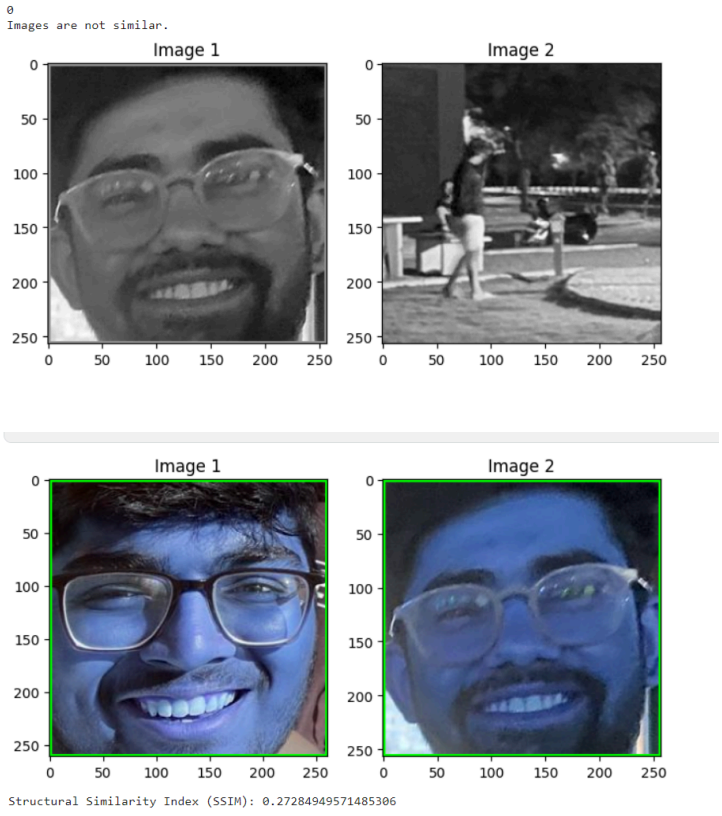
Clustered faces using the embeddings of VGG16. Similarity between two images was defined using cosine similarity between the embeddings of images.

This figure shows that the similar faces of the person are grouped together and represented as a cluster using the cosine similarity score on the face embeddings of the individuals in the different images.



Cluster of all the images that contain the representation face (ie the first image in the figure). The above plot basically is clustering of all the images that contain the face of the person shown on the leftmost image.

Clustering using SIFT AND SSIM SCORES



OBSERVATIONS AND CONCLUSION →

Face Detection →

For face detection the Hog feature descriptor was comparatively too slow and also could not perform efficiently in the case of multi face detection whereas the Cascade feature detection technique was performing comparatively better on the multiple faces was also taking much lesser time so for the rest of the project we move forward with the cascade classifier for the face detection technique.

Face Recognition→

For face recognition, the KNN accuracy after applying PCA came out to be 0.955 and for LDA, it was 0.985. When the classifiers were tuned, the PCA model had the highest accuracy for k=1 for all the alpha values. However, for LDA, all the accuracies were same. For the DL model, the final val accuracy after 100 epochs came out to be 0.8875.

So, it can be concluded that the best-performing model out of all of these was the LDA one. However, it was very time-consuming. So, PCA could be preferred with a slightly lower accuracy.

Face Clustering →

For face clustering the training of autoencoder took time for training on the input images, and also did not have satisfying results. Autoencoder-based clustering may lack interpretability, as the learned latent representations may not have clear semantic meanings. This can make it challenging to understand and interpret the resulting clusters. Then we used the VGG16 model to get the feature map of the images. This required no training on the input data. Clustering was done on the new feature maps of images. It was

observed that some of the faces that were present at a very less frequency were given a label -1, and the faces that were frequently present were clustered according to identity formed due to wrong detection faces by the haar cascades. The drawback of the method was that we used a pre-trained model. It may happen that the data we get may be different from the training data of the model, it may have a bias. Similarity between the face images was also calculated using the SSIM scores and the SIFT descriptors. The computation efficiency was also not high. SIFT descriptors are invariant to changes in scale and rotation, allowing them to effectively match features across different scales and orientations in images. So in conclusion we can say for image clustering purposes we can use the distance similarity measures using the SIFT descriptors.

Thankyou