CS 4641: Final Report

Group 44

Introduction and Background

The project aims to develop a machine learning (ML) model that is capable of automatically categorizing and sorting group photos based on individual faces. Our primary objective in this case is to have a model that is able to accurately identify faces and whose face it is amongst our 5 group members.

Literature Review

Face Mesh accurately identifies facial features by mapping key points, and enhancing recognition. Sobel's Gray-Scale method aids edge detection, focusing on significant facial areas while reducing noise. Background removal isolates the face, ensuring the model concentrates on relevant features. Breaking down face scans into images allows effective data categorization for training datasets. SVM classifies extracted features, while GNN improves accuracy by measuring distances between facial features. InsightFace can provide insights for model development. Using CNN to identify numbers from images, they got 97% accuracy.

Data Collection and Dataset Description

To avoid infringing privacy and to simplify data collection efforts, this project will only use images of the team members. The images will be collected by extracting individual frames from recorded videos of each team member's face using a Python script [4]. These images will be organized into test and training data with team members' names as labels.

As the project progresses, and given time, we plan to expand the dataset by including more group photos to enhance the models' recognition capabilities and accommodate multiple individuals in each image.

Dataset Link

Dataset Link

Problem Definition & Motivation

At large events like weddings and conferences, sorting through numerous photos can be time-consuming. To address this inefficiency, we aim to develop an ML solution that automates the categorization of images based on facial recognition of various faces in the photo. Users will submit a video of their face, and our model will then identify photos in which said user appears.

Methods

Data Pre-Processing Method(s):

The data preprocessing method used in creating our model includes extracting individual faces from 5 different videos uploaded by the user (videos with different facial expressions, different lighting, different contexts, etc.) and extracting frames from those videos to use for our dataset. This process involved first getting users to upload said videos, creating a python script to extract a high number of .jpg frames from the videos, and saving those image frames into separate files. Further preprocessing efforts include image resizing/ scaling, image alignment, image labeling, and simple enhancements to concentrate on the user's face.

Image 1: Preprocessed JPG frames extracted from training videos of team members' faces

ML Algorithms/Models Implemented:

For image processing, we used MTCNN for facial detection because of its deep learning-based architecture which allows it to detect faces quickly and accurately, even in images with varied orientations, scales, and complex scenes. Once a face is detected, we implement FaceNet to generate embeddings of compact, numerical representations of facial features and capture the essential characteristics of a face. These embeddings are more accurate than traditional methods (like Eigenfaces or Fisherfaces) for distinguishing between individuals, even under varied lighting, poses, and expressions.

After image processing, for the supervised model, an SVM classifier is employed to classify these embeddings, leveraging its ability to define clear margins and complex decision boundaries. SVMs are computationally efficient and often outperform simpler models for this purpose, providing high accuracy while avoiding overfitting. Additionally, with a smaller training data, SVM is still effective. Together, these models create a powerful pipeline that excels in detecting, encoding, and classifying faces reliably and efficiently.

After image processing, unsupervised models such as DBSCAN and K-Means were employed to cluster facial embeddings. DBSCAN was chosen for its ability to discover clusters of arbitrary shape and manage noise effectively, making it well-suited for handling variations in poses, lighting, or occlusions. Unlike many clustering methods, DBSCAN does not require prior knowledge of the number of clusters and can automatically exclude outliers, such as false detections or non-facial regions. In parallel, K-Means clustering was utilized for its simplicity and efficiency in partitioning data into a predefined number of clusters. By assigning facial embeddings to one of five clusters, corresponding to each team member, K-Means provided a structured approach to grouping similar faces. Together, these methods offered complementary strategies for organizing facial data.

Supervised Learning or Unsupervised Method Implemented

We implemented a supervised learning approach for Face Recognition using face embedding and Support Vector Machine (SVM) Classification.

- Generating face embeddings involves training a neural network on a labeled dataset, where each face image is tagged with an identity. This supervised training enables the model to learn distinct features for each person, mapping them into a vector space.
- The SVM classifier relies on the labeled embeddings generated from the face embedding model. It uses these labeled vectors to determine a boundary that best separates the identities based on the embedding feature, a process that requires labeled data for each individual in the training set.

We implemented two unsupervised learning approaches for Facial Clustering (intended to be used before Facial Recognition) using face embedding and Principal Component Analysis (PCA) prior to using DBSCAN and K-Means respectively.

- Generating face embeddings involves training a neural network on a labeled dataset, where each face image is tagged with an identity. This supervised training enables the model to learn distinct features for each person, mapping them into a vector space.
- PCA reduces the dimensionality of facial embeddings before applying DBSCAN and K-Means. By identifying key features that capture the most variance, PCA simplifies the data and improves clustering efficiency, focusing on the most informative facial characteristics
- DBSCAN does not use labeled embeddings, instead clustering facial embeddings based on the density of data points. It forms clusters by grouping closely packed points while treating sparse regions as noise, making it well-suited for unsupervised facial grouping tasks.
- K-Means does not use labeled embeddings, instead relying on a predefined number of centroids to partition the data. It iteratively assigns data points to clusters, minimizing the variance within each cluster based on the proximity to these centroids.

Results and Discussion

Visualizations

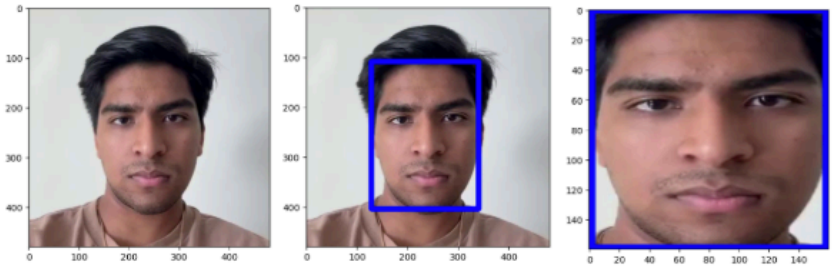


Image 2: (a) Testing Image, (b) Bounding boxes around face(s) detected in the image, (c) cropped, compressed, and resized face

Images 3 and 4: Model Predicts only when the Confidence Level is over a Specific Threshold (Here, 0.64). Else, Model Labels it as “unknown”

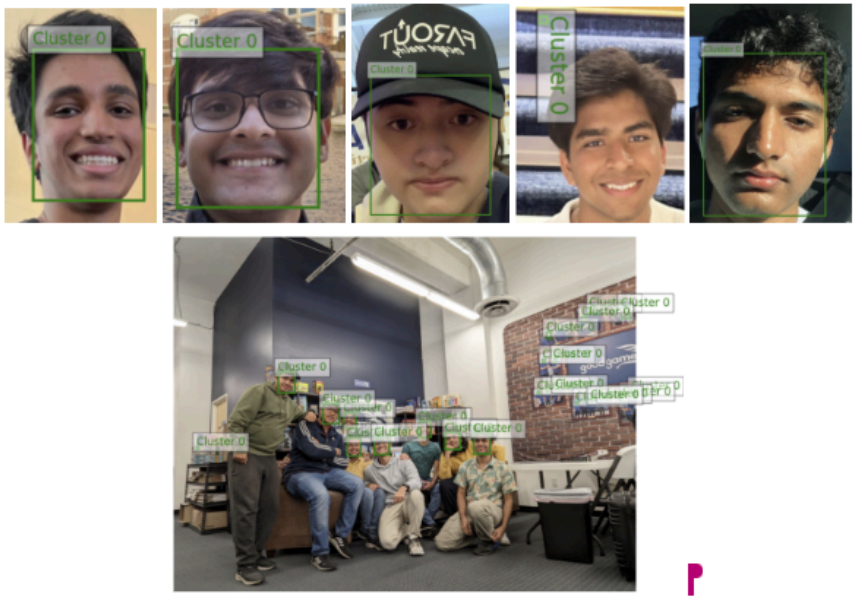


Image 5: Results of DBSCAN Clustering. Note, every face is under Cluster 0

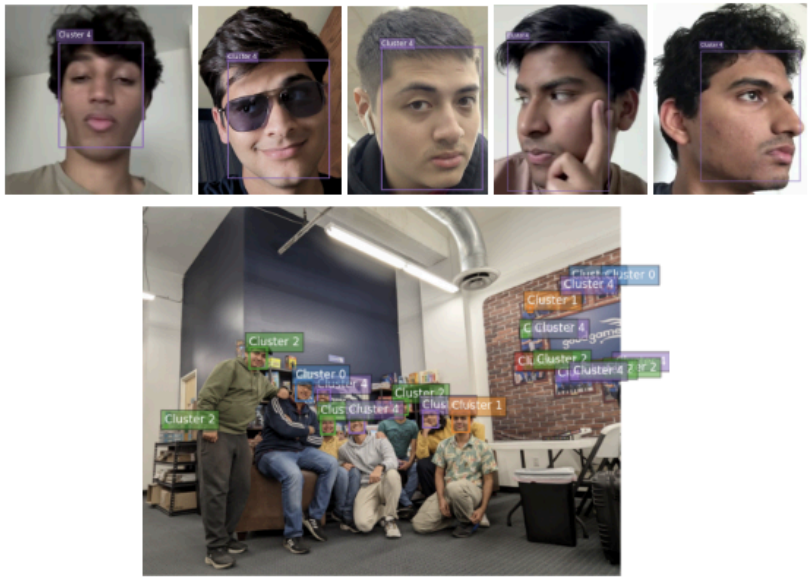
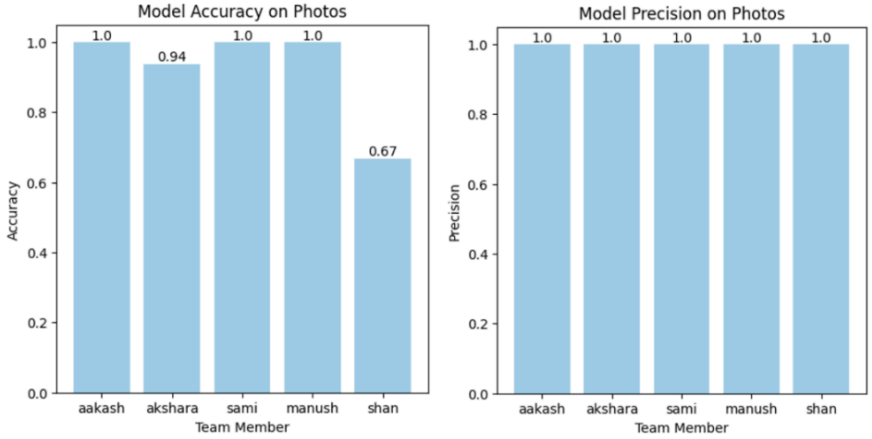


Image 6: Results of K-Means clustering. Note, all team members are under Cluster 4

Quantitative Metrics:

We tested our supervised model on multiple photos of each group member, including individual photos as well as group photos. This model correctly labeled 58 faces out of 59 with either a team member's name or as "unknown" depending on the model's confidence levels. This resulted in an overall accuracy of around 0.98.



Images 7 and 8: Model Accuracy and Precision on Photos per Team Member

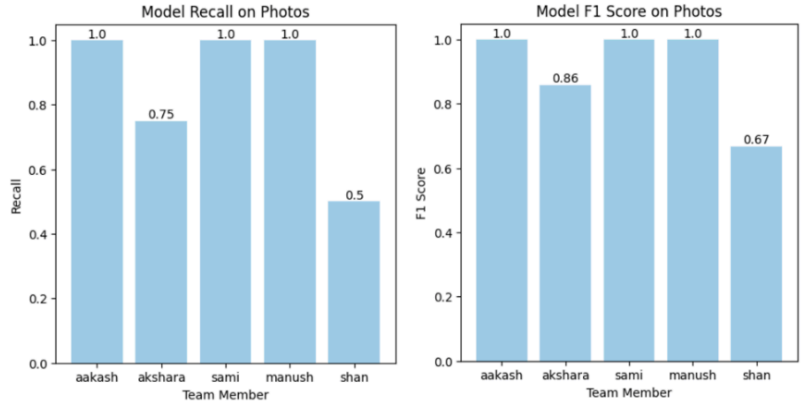


Image 9 and 10: Model Recall and F1 Score on Photos per Team Member

We observe that Shan's accuracy, recall, and F1 scores are lower than those of the other team members, while his precision score remains consistent. This suggests that the model is failing to recognize Shan in many of his images, indicating a potential imbalance in the dataset or inconsistent generalization of his specific features.

Upon further analysis of the photos that failed to be recognized, we observed that the subjects are often under different lighting conditions or have different facial hair compared to the images in the training dataset. While the training data accounted for various angles and facial expressions, it did not include sufficient variation in lighting or facial hair. Expanding the dataset to incorporate these conditions would enhance the model's ability to generalize and improve its recognition performance across a wider range of real-world scenarios.

There are no quantitative metrics for the unsupervised models, as they operate without labeled data, making it impossible to directly evaluate the accuracy of their clustering or "predictions." However, both DBSCAN and K-Means consistently clustered all team members into a single group due to the high similarity in our facial embeddings. Naturally, this

means that they had low accuracy and low precision.

Analysis of 3+ Algorithm/Model:

Supervised Models: MTCNN and SVM

The combined use of MTCNN, FaceNet, and SVM creates a streamlined and effective face recognition process. First, MTCNN (Multi-Task Cascaded Convolutional Network) is responsible for detecting faces within an image, outputting bounding box coordinates (x, y, w, h) that specify the facial regions. Once MTCNN identifies a face, FaceNet generates a facial embedding—a compact, high-dimensional vector of 128 dimensions that encodes the unique features of the face, making it easier to distinguish between different individuals. Finally, the SVM model uses these embeddings as input features to classify the faces, assigning them to known identities or verifying if they match a specific person. Together, these models enable accurate and efficient face detection, feature encoding, and classification.

Unsupervised Models: DBSCAN and K-Means

In order to keep the image processing consistent between the supervised and unsupervised models, we used MTCNN and FaceNet to detect and embed faces before facial clustering, respectively.

DBSCAN is used to cluster these embeddings, automatically grouping faces from the same individual together based on their density. K-Means is used to cluster these embeddings into predefined groups based on their proximity to centroids. K-Means assigns each facial embedding to one of the clusters, with the goal of minimizing variance within each group. Coupled with the facial detection and embedding by MTCNN and FaceNet, each of these models enables effective face detection, feature encoding, and unsupervised clustering of facial data.

Comparison of 3+ Algorithms/Models

In our testing of DBSCAN and K-Means for unsupervised facial clustering, both models faced significant challenges in differentiating between the faces of group members.

Initially, DBSCAN clustered all faces under Cluster -1 (noise). To enhance clustering effectiveness, we adjusted the epsilon parameter to achieve optimal feature separation for accurate cluster formation. Additionally, we employed Principal Component Analysis (PCA) to reduce dimensionality. Yet, the model continued to cluster all faces under Cluster 0. In an attempt to enforce multiple clusters, we implemented K-Means with five predefined clusters - one for each team member. But, this model too clustered all team members into a single group. K-Means showed some success in clustering non-group members in a photo with 15 people but struggled with our own team members' faces, which seems to exhibit minimal variability when embedded.

Since this behavior did not change even on trying multiple techniques mentioned above to improve performance, we have come to a soft conclusion that there is very little variability within the group members' photos for a simple unsupervised model to effectively differentiate between them. This behavior highlights the limitations of these models in scenarios requiring precise differentiation among visually similar individuals.

The reason SVM performed better than clustering may be because it is a supervised learning algorithm that leverages labeled data to explicitly learn the boundaries between classes, whereas clustering relies solely on the inherent structure of the data. Additionally, SVMs are well-suited for small datasets because they focus on the most critical data points, called support vectors, to find the optimal decision boundary, reducing reliance on large amounts of data. They use regularization to balance training accuracy and generalization, minimizing the risk of overfitting.

- SVM uses labeled examples to directly learn decision boundaries between group members, making it robust even for subtle differences.

- Clustering (e.g., DBSCAN, K-Means) has no labels and relies entirely on unsupervised patterns in the data, which may not be distinct for group members with highly similar embeddings.
- SVM can better handle noisy or overlapping data regions by using kernel methods to project data into higher dimensions if needed.
- Clustering is more sensitive to noise and often misclassified points when clear separations are not present, which was present in our implementation where either the lighting or the noise confused the model and misclassified each face.
- Lastly, SVM is designed to identify patterns in highly similar data by emphasizing discriminative feature

Next Steps

- Enable Real-Time Detection for Efficient Group Organization to support live photo organization, we will incorporate real-time detection capabilities. This addition allows the model to quickly identify and sort photos based on each individual's presence, creating a seamless experience for users as they manage group photos.
- Enhance Facial Recognition Capabilities to handle Facial Hair and Coverings Currently, our model has shown to struggle to recognize individuals if they are wearing opaque glasses, have grown excessive facial hair, are present in abnormal lighting, etc. Enhancing our model's capabilities to recognize individuals despite these changes would improve the model's robustness and practical applicability in real-world scenarios where such variations are common.

References

1. MediaPipe, "MediaPipe Python Tasks - Face Landmarker," Available: https://colab.research.google.com/github/googlesamples/mediapipe/blob/main/examples/face_landmarker/python/%5BMediaPipe_Python_Tasks%5D_Face_Landmarker.ipynb. [Accessed: 04-Oct-2024].
2. L. Haidar, "Edge Detection Techniques: Sobel vs. Canny," GitHub, 2024. Available: https://github.com/lina-haidar/Edge-Detection-Techniques-Sobel-vs.-Canny. [Accessed: 04-Oct-2024].
3. G. Shperber, "Background Removal with Deep Learning," Medium, 2024. Available: https://towardsdatascience.com/background-removal-with-deep-learning-c4f2104b3157. [Accessed: 04-Oct-2024].
4. D. Sadek, "Advanced Face Recognition System," Medium, 2024. Available: https://medium.com/the-modern-scientist/advanced-face-recognition-system-a392787cfe6c. [Accessed: 04-Oct-2024].
5. DeepInsight, "InsightFace: CNN-based Face Recognition," GitHub, 2024. Available: https://github.com/deepinsight/insightface. [Accessed: 04-Oct-2024].
6. DataCamp, "A Comprehensive Introduction to Graph Neural Networks (GNNs)," DataCamp, 2024. Available: https://www.datacamp.com/tutorial/comprehensive-introduction-graph-neural-networks-gnns-tutorial. [Accessed: 04-Oct-2024].
7. Scikit-learn, "Model Evaluation: Metrics," Scikit-learn, 2024. Available: https://scikit-learn.org/stable/modules/model_evaluation.html. [Accessed: 04-Oct-2024].
8. Abdirayimov, Sardor "Face Recognition in Python | Part 1 | FaceNet, MTCNN, SVM", Available: https://www.youtube.com/watch?v=bG2tNYs7gw8. [Accessed: 04-Nov-2024].

Contribution Table

Name	Proposal Contributions
Aakash Asthana	Introduction/Background, Problem Definition, Data Preprocessing Methods, References
Akshara Joshipura	Gantt Chart, Methods, Analysis of Models, Comparison of Models, Quantitative Metrics, Visualizations, Contribution Table

Manush Patel	Machine Learning Models & Preprocessing Approaches, Supervised Learning Model, Literature Review, Presentation & Video
Sami Moussa	Data Preprocessing Methods, Literature Review, GitHub, Supervised Learning/Unsupervised Method Implemented, Presentation & Video
Shan Patel	Introduction/Background, Problem Definition, Reference Formatting, Presentation, Video, GitHub, Quantitative Metrics