Person Identification Model

Duaa Naz *30668*
Hadia Sajid *31687*
Huzaifa Awais *31774*
Kashaf Ansari *30493*

***Abstract*—This report shows how the person identification system was developed, employing computer vision and machine learning. This system identifies individuals by first detecting faces with Haar cascades. After that, unique features are extracted from these faces using Histogram of Oriented Gradients (HOG). Finally, a Support Vector Machine (SVM) classifies these features for person identification.**

Understanding the decision-making process was a core focus. This was achieved through exploratory data analysis (EDA), revealing patterns within the dataset, aspect ratio analysis of faces, and feature visualization to illustrate the model's perception. This report details the implementation process, dataset preparation, algorithm selections, and the visualizations utilized to enhance comprehension.

Testing on real student facial data demonstrates strong performance, supporting the system's potential for practical use in educational or institutional environments.

***Index Terms*—Introduction, Libraries ,Data Processing and EDA ,Feature EXtraction, Model Training ,Confusion Matrix,Classification Metrics, Analysis , Future Work , Division Of Roles**

I. Introduction

This project aims to apply Computational Thinking (CT) to a real-world machine learning task—person identification through facial images. The goals are:

A. Computational Thinking Visualized

1) Decomposition: Break the task into data preprocessing, face detection, feature extraction, and classification.
2) Pattern Recognition: Identify similar and repeating features across face images.
3) Abstraction: Focus on the important/unique facial features for recognition.
4) Algorithmic Thinking: Design clear steps for training, testing, and interpreting the model.

B. Interpretability in Machine Learning

In order to understand **why** a prediction is made. We used visualizations (boxplots,graphs) and feature analysis(brightness,blurr,aspect ratio of each image) to interpret the behavior of the model.

For example, if a model predicts that a photo belongs to Student 07, interpretability asks: What features (like face shape, brightness, contrast, or image clarity) led the model to this conclusion?

This makes it easy for us to check if the model is making informed decisions,easier to debug it aswell.

.

II. Libraries

1) **os**
Helps work with folders and file paths.Used to get file names and directories of images.
2) **cv2**
Powerful image processing library. Used for reading images, converting to grayscale, and detecting faces.
3) **numpy**
Deals with numerical data and arrays. Used to handle image data in numeric form.
4) **pandas**
Used for handling and analyzing data in table form (like Excel). Stores and processes image paths, student IDs, and calculations.
5) **glob**
Finds all file names matching a pattern (e.g., all '.jpg' images). Used to collect paths of student images.
6) **matplotlib.pyplot**
Used to draw graphs and charts. Helps visualize things like image size and aspect ratio.
7) **skimage.feature**
Extracts important features from images for face recognition,using tools like HOG and LBP
8) **skimage.exposure** Adjusts image brightness and contrast (for better HOG visualization).
9) **sklearn.preprocessing.LabelEncoder**
Converts student IDs (like "01", "02") into numbers so the model can understand them.
10) **sklearn.model selection train test split**
Splits data into training and testing parts to evaluate model performance.
11) **sklearn.ensemble RandomForestClassifier** Type of decison tree classifier model used for classifying images based on features.
12) **sklearn.metrics** Measures how well the model works.
13) **accuracy score** Specifies how many predictions were correct.
14) **classification report**
Gives precision, recall, and F1-score.
15) **confusion matrix** Shows how often the model confused two students.
16) **ConfusionMatrixDisplay**' Displays the confusion matrix as a plot.
17) **plt.style.use('ggplot')** Sets the plot style to a cleaner and more readable format.
18) **tkinter** A built-in Python library used to create graphical user interfaces (GUIs). It helps build interactive win-

dows, buttons, and dialogs

19) **PIL (Python Imaging Library) / Image, ImageTk** Used for image manipulation and converting images into formats that work with Tkinter GUIs.

July 6th, 2025

III. EDA

Before training any Machine Learning model, it is crucial to understand the data we're working with. This process is called Exploratory Data Analysis (EDA).

1) How many images are there per student?
2) Are all images valid and readable?
3) How many images contain detectable faces?
4) What are the average image sizes and aspect ratios?

These are just some of the questions implementing EDA effectively gives us the answers to

A. Initial Step

The first part of the EDA grouped the image data by "student id" to see how many images we had for each student. This was important because it allowed us to check for balance in the dataset.

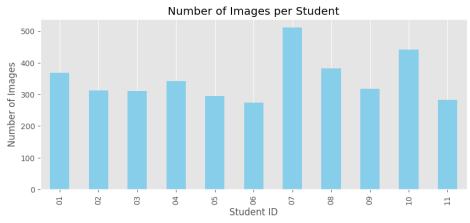


Fig. 1. Image count per student

This helped ensure that most students had a sufficient number of images for training and testing.A model trained on imbalanced data (e.g., more images for one student and fewer for another) could become biased. The given dataset comprises grayscale images of student faces, stored in separate folders (one per individual). Each image varies slightly in pose and lighting, simulating real-world variance.

B. Face Detection

Another crucial part of EDA was checking how many images actually contained detectable faces, images without a face would not be useful.

1) To read each image, convert it to grayscale, and use a Haar Cascade (These are simple rectangular patterns that capture differences in pixel intensities within an image and identifies regions that strongly match the learned Haar-like features of a face) face detector to check if a face was present.

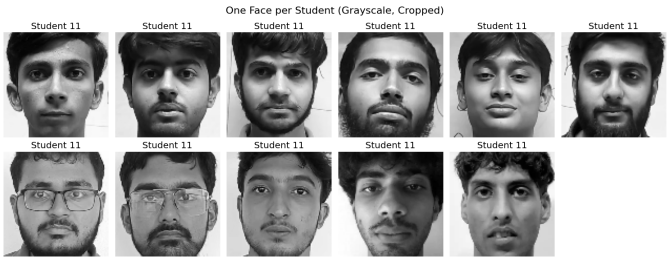


Fig. 2. Detected Faces

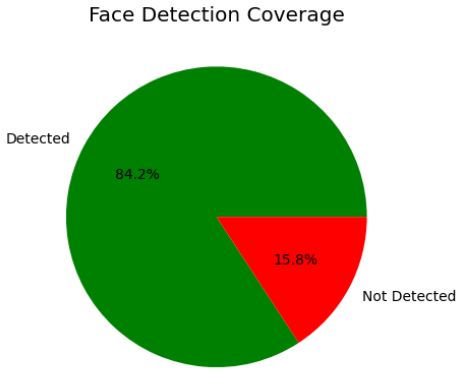


Fig. 3. Visualization showing loss in 15.8-percent of data due to faces not being detected

2) Blurring , reduces noise and fine details in images, making facial features more distinguishable for detection. HSV conversion, done with cv2 , transforms images from RGB to HSV color space, which separates color and brightness information, helpful in skin detection, lighting correction and HSV is closer to how humans perceive colors, so it's useful in visual tasks.
This helps in handling variations in lighting and improves image analysis. These preprocessing steps ensured cleaner input data, leading to more accurate model training and predictions.

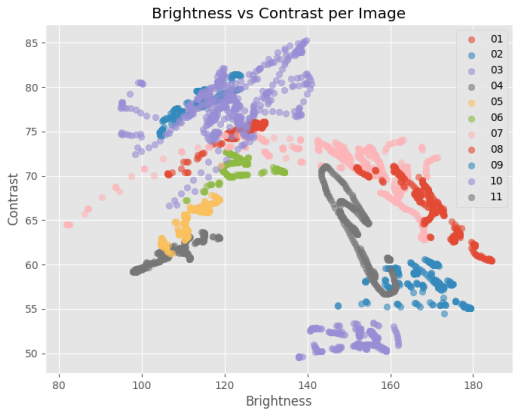


Fig. 4. To spot the outliers and biasedness in the data, we plotted the graph of brightness and contrast of images which each point representing one image. It aids us to decide the preprocessing procedures and normalize the data accordingly

3) The **aspect ratio** is calculated as **width / height**. It helps us understand the shape of the images. A consistent aspect ratio ensures better feature extraction and classification.

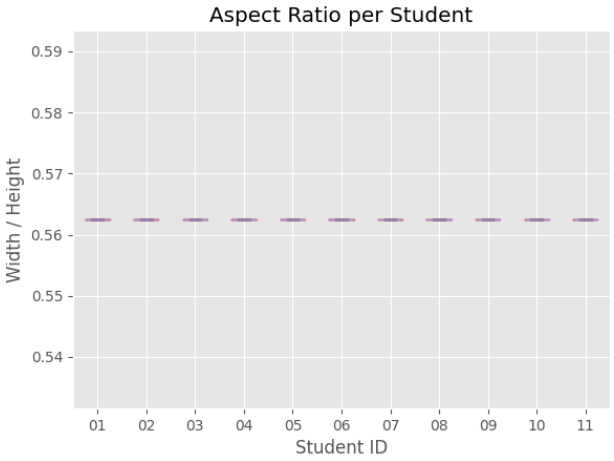


Fig. 5. this image shows that all the images have same shapes and hence no corrupt images are included in the dataset

4) The blur score plot shows how sharp each student's images are. Student 1 has the sharpest images with high blur scores, while students like 10 and 11 have mostly blurry ones. Some students (like 08 and 09) have a mix of sharp and blurry images. This matters because blurry images can make face detection and feature extraction harder, which may reduce the accuracy of the final model. Checking these scores helps us spot which students may need cleaner or retaken images.

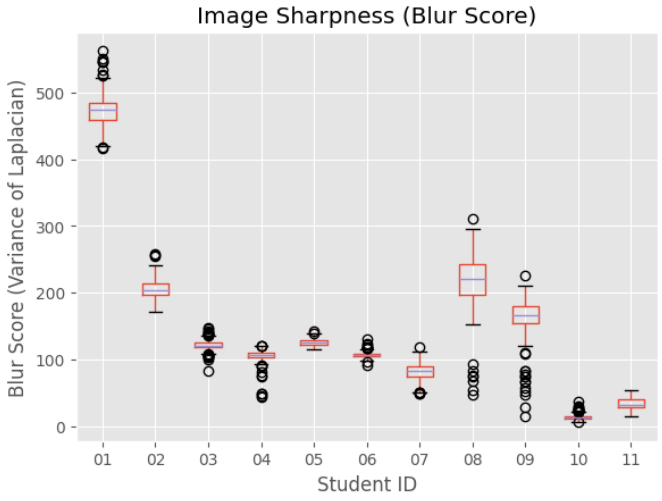


Fig. 6. Box Plot

C. Outliers

To get an accurate result, we must remove the outliers in the data so that the data is an optimal fit. To achieve that, we remove all such images in our data which aren't optimal for the training model. All blur and partial images will be removed and the optimal images will be taken and resized according to the requirements. In order to do that, we are using low variance threshold comparison which means that if the variance threshold of an image is low, it's blur and hence not included in the training dataset.

As the dataset is quite extensive, we can easily remove the outliers without impacting the training much. Only the standardized images, which are clear and totally visible are added in the dataset.

5) Understanding image dimensions is also important because inconsistent sizes can cause errors during preprocessing and feature extraction.

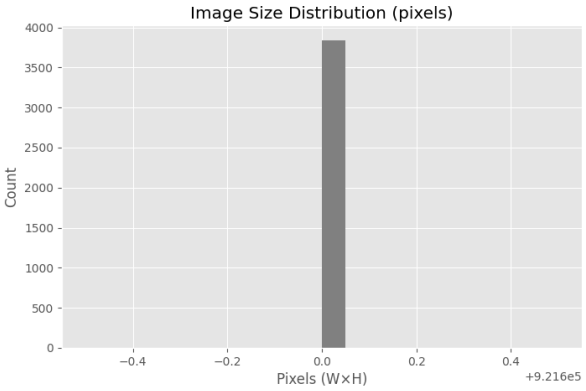


Fig. 7. This graph indicates that all images are of equal size which is essential for models like HOG and SVM

D. Analysis from the Results of EDA

The data analysis showed that we have around 3,800 images from 11 students, which is a good amount for training and testing our model. The face detection worked on about 84 percent of the images (3,230 out of 3,836), so the Haar Cascade method is working fairly well. All the images are about the same size, so we won't have issues when processing them. Also, the aspect ratio is almost the same across all students, which means the pictures are consistent. This makes our data well-prepared for the next steps like feature extraction and classification.

IV. Feature Extraction

We used a two-step method to extract features from face images. First, OpenCV's Haar Cascade to detect faces in grayscale images. It finds the largest face in each image, crops it, and resizes it to 100×100 pixels.

Once a face is detected, nested detectors identify eyes and smiles within the face region. Detected features are highlighted using colored rectangles. This process helps verify the quality and consistency of facial data before training the recognition model.

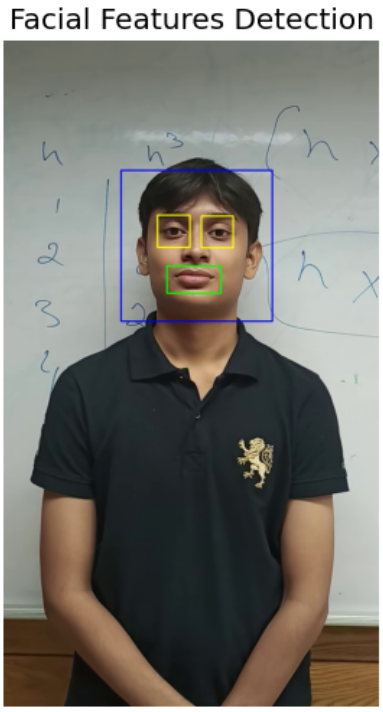


Fig. 8. Enter Caption

Next,using the HOG (Histogram of Oriented Gradients) method to describe the shape and edges of the face. HOG looks at how the brightness changes in different directions to capture facial patterns.

A. How it works:

We calculate how the brightness changes between each pixel and its neighbors to detect edges and their directions. Next, we divide the image into small squares (usually 8×8 pixels), and in each square, we create a histogram that counts how many edges point in different directions. Then, by combining all the information into a single long list of numbers (a feature vector), which gives a detailed description of the face's shape. [10]

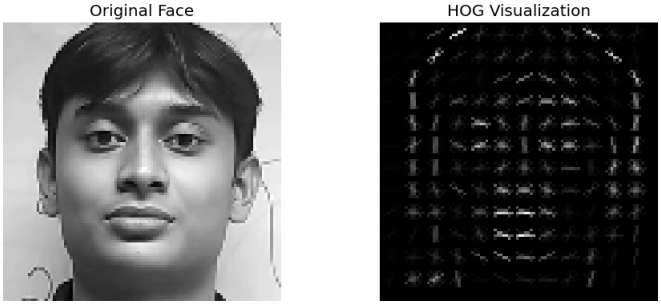


Fig. 9. HOG visualization

B. HOG+LBP

LBP (Local Binary Pattern) is useful for capturing small texture details on the face, like skin patterns, lines, and small changes in brightness. It works by comparing each pixel with its surrounding neighbors.
We set the radius to 1, meaning each pixel was compared with its immediate 8 neighbors (8*radius).
HOG, on the other hand, looks at the shape and edges of the face. It focuses on the direction of lines and outlines, which helps the system understand the overall structure of the face. HOG works well even if the lighting changes or if the face is slightly turned.
By using both LBP and HOG, we were able to create a stronger and more complete description of each face[15]. This helped our model recognize students more accurately, even when the image conditions were not perfect,such as changes in light, facial expression, or angle.

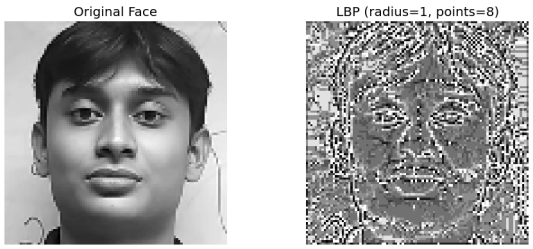


Fig. 10. LBP

The final features and their student labels were then divided into training and testing sets to train the model. [3]

V. Feature Visualization

The feature reduction is performed using Principal Component Analysis (PCA) for easier visualization and analysis of data. [8]
PCA is a dimensionality reduction technique. Face features (like those extracted by HOG) are typically very high-dimensional (many numbers representing characteristics of the face). PCA takes these many dimensions and transforms them into a new set of dimensions called "principal components" (PC1, PC2, etc.), which capture the most variance (information) in the data.

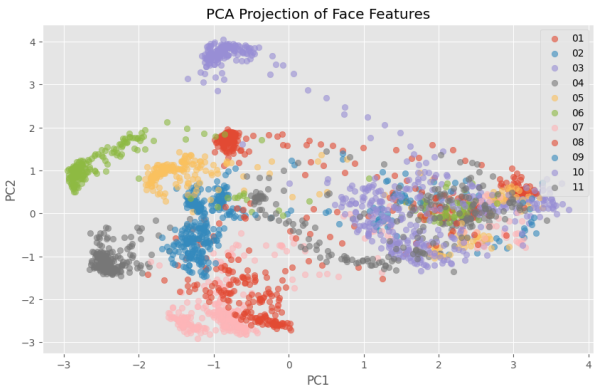


Fig. 11. PCA

Each point on the graph represents a single face feature vector from the dataset, projected onto the 2D space defined by PC1 and PC2. All points colored ”01” belong to the same person, all points colored ”02” belong to another person, and so on, up to ”11” distinct individuals.

A. Analysis from the graph

For some individuals like ”01” in red, ”06” in green, ”07” in light pink, the points tend to form relatively tight and somewhat separate clusters. This suggests that the face features for these individuals are quite distinct in this 2D PCA space, making it easier for a classifier (like an SVM) to tell them apart.
The scattered nature and significant overlap of for example, the ”03” cluster indicate that its features are not very distinct, making it a potentially challenging individual for the person identification system to classify accurately.

VI. Model Training

A. Splitting Data:

Split the dataset into two parts

- Training Samples
- Testing Samples

The Training Samples are about 80 percent, the rest of the 20 percent has been saved for testing. This way the model's accuracy can be easily tested.

B. Decision Tree

A Decision Tree is a machine learning model that makes decisions by asking a series of yes/no questions (or checking conditions) about the input data.
A decision tree is kind of like a flowchart that helps the computer make decisions based on the features of the data, like how blurry a face is, how bright the image is, or what angle the photo was taken from. At each step, the tree checks which feature helps it best split the data into groups, like separating one student from another. It keeps asking simple yes/no questions like “Is the blur score less than 100?” and branches off based on the answers. This keeps going until it reaches an end point (called a leaf), where it finally makes a decision, like guessing which student is in the image. It's basically a series of smart choices based on the data to reach a final answer.

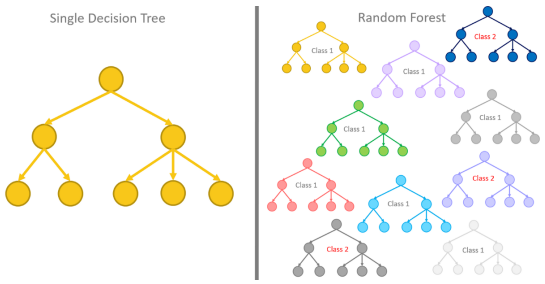


Fig. 12. Example of a Decision Tree

C. Random Forest Classifier

A Random Forest is an ensemble learning method that creates multiple decision trees and combines their outputs to make a final prediction[13]. This approach improves accuracy and reduces overfitting compared to a single decision tree. In our implementation, we used 100 trees (n estimators=100) with a maximum depth of 5 (max depth=5). Limiting the depth helps prevent the model from memorizing the training data too much, ensuring it doesn't overfit.
1) Implementation: During the implementation of this, our group did alot of experiments playing with the ”Depth” of the decision tree. ”Depth” refers to how many levels of decisions or splits the tree does from the first question to the last.
2) Problems Faced: The deeper the tree, the more complex decisions it can make, which often helps it learn better from training data. But too much depth can also cause overfitting, where the tree memorizes the training data too well and performs badly on new data.
However, since Random Forest combines the predictions of many decision trees trained on different subsets of the data, it naturally reduces overfitting compared to a single deep decision tree[14]. To further prevent overfitting, we set the maximum depth of each tree to 5.
3) Advantages: Highly interpretable (easy to visualize logic of classification).
Fast to train, even on moderate-sized datasets.
4) Drawbacks: Prone to overfitting without proper depth control or pruning.

VII. Evaluating Performance: The Confusion Matrix

To evaluate the accuracy of the face detection model, a *confusion matrix* is used as a standard performance metric in classification tasks [4]. The confusion matrix summarizes the outcomes of the classification by comparing the predicted labels with the actual labels. It consists of four components:

a) **True Positives (TP)**: Correctly detected faces.
b) **True Negatives (TN)**: Correctly identified non-faces.
c) **False Positives (FP)**: Non-faces mistakenly classified as faces.
d) **False Negatives (FN)**: Faces that were missed by the model.

Using these values, various performance metrics can be computed. For example, **accuracy** is given by:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

A. Confusion Matrix for Model Evaluation

To evaluate the performance, we used a **confusion matrix**, a common tool in classification problems that provides a detailed breakdown of prediction results. The confusion matrix compares the *true labels* with the

predicted labels for the test dataset and helps visualize the model's strengths and weaknesses in classification. In our system, the confusion matrix was generated using the utility *sklearn.metrics.ConfusionMatrixDisplay* after predicting the class labels for the test set. The diagonal elements of the matrix represent correctly classified samples (true positives), while the off-diagonal elements correspond to misclassifications.

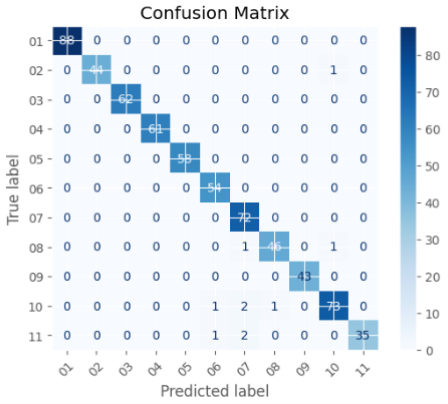
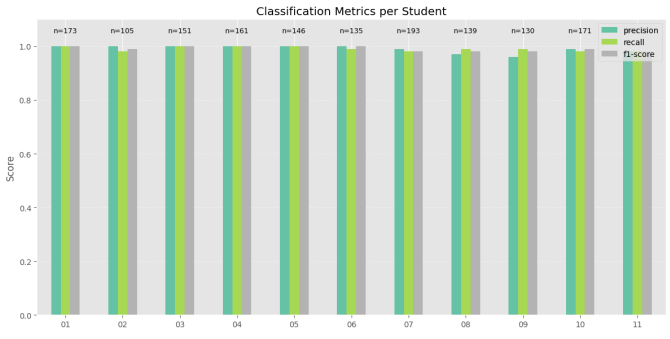


Fig. 13. Confusion Matrix showing prediction result

This visualization allows us to identify which student identities are frequently confused by the model, and serves as a useful diagnostic tool to understand potential issues in face detection, feature extraction, or data imbalance[12] . Additionally, it complements the overall test accuracy (e.g., 98.45%) by providing class-wise performance insights.

VIII. Classification Metircs



To evaluate how well the face recognition model performs for each individual student, we used

a) Precision:

$$Precision = \frac{TP}{TP + FP}$$

b) Recall:

$$Recall = \frac{TP}{TP + FN}$$

c) F1-score:

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

as performance metrics. These metrics were visualized in a grouped bar chart for every student ID. **Precision** shows how often the model was correct when it predicted a student's face, while **Recall** tells us how many of the actual images it identified correctly. The **F1-score** gives a balanced measure of both.
The chart includes the total number of test images used per student (e.g., n=173). Most students scored close to 1.0 (100), indicating high accuracy. Slightly lower scores for a few students may result from fewer images, blurry or poor quality data, or inconsistent facial features. This analysis is important for ensuring fairness, detecting potential bias, and improving model performance.

IX. Graphical User Interface

Our system has a user-friendly interface using the Tkinter library. The window allows users to:

a) Upload a photo for face recognition.
b) Activate their webcam for real-time recognition.
c) See the prediction result and the face bounding box displayed on-screen.

Once an image is selected via the "Upload Image" button, it is read using OpenCV, and the face is detected using a pre-trained Haar Cascade classifier. The detected face is first checked for sharpness using variance to avoid misclassification of blurry images. If a clear face is detected, the image is converted to grayscale and resized, and features are extracted using HOG+ LBP.

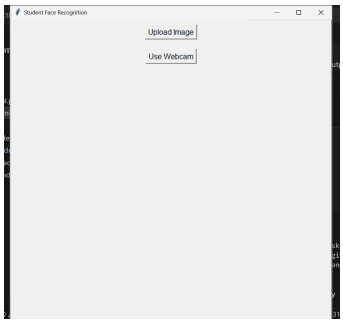


Fig. 14. Options for Image Data

These features are then passed to a previously trained RandomForestClassifier model, which outputs a class prediction along with a confidence score. If the score falls below a certain threshold (in this case, 0.5), the face is labeled as "Unknown." Otherwise, the label is decoded using a label encoder and displayed on the image along with the confidence percentage.

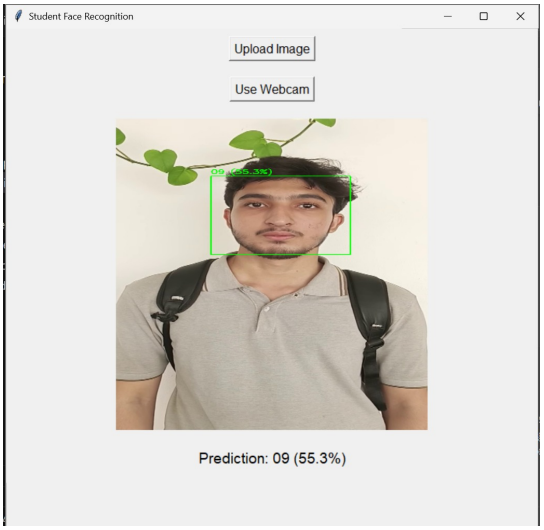


Fig. 15. Display

The GUI displays this annotated image along with the predicted name or result using PIL and ImageTk.

X. Webcam Integration

The application also provides a real-time face recognition feature through webcam access. When the user clicks the "Use Webcam" button, the program accesses the system's default webcam using OpenCV's VideoCapture functionality.

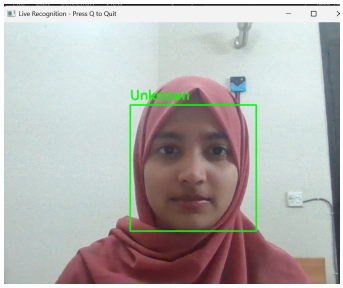


Fig. 16. Webcam Integration

A continuous loop is initiated to read each frame from the webcam feed. For every frame, the same face detection and recognition pipeline is applied as in the image upload function. The recognized name (or "Unknown") is overlayed on the video feed in real-time. The window remains active until the user presses the 'q' key, at which point the webcam is released and all OpenCV windows are closed.

XI. Analysis

Our face recognition system demonstrates the core principles of computational thinking, which guided the development of each part of the project.

a) **Decomposition** We broke down the complex problem of identifying students into smaller parts. This included:
Loading and preprocessing student images,
Detecting faces using Haar Cascade Classifier.
Extracting features using Local Binary Pattern (LBP) and Histogram of Oriented Gradients (HOG) from skimage.feature,
Training the prediction model using a Random Forest Classifier from sklearn.ensemble,
Creating a graphical interface using tkinter to make the system user-friendly. This separation allowed each component to be developed, tested, and improved independently.

b) **Pattern Recognition** Pattern recognition was fundamental to our system.
Haar Cascade used pretrained facial patterns to detect features like the eyes and nose in real time.
HOG identified the shape and structure of the face by focusing on gradients and edge directions.
LBP extracted texture patterns, such as skin lines and fine details, which are useful for distinguishing between similar-looking faces. By combining both, we captured both structural and textural information, resulting in better recognition performance.

c) **Abstraction** Abstraction was used to simplify data while retaining critical information.
Detected faces were cropped, resized, and converted to grayscale, reducing noise and computational load.
Instead of using the full image, we extracted only the useful features (LBP + HOG), which significantly reduced the input size for the classifier.

d) **Algorithm Design** The system follows a well-defined algorithmic process:
Capture or load the image ,
Detect the face using Haar Cascade,
Convert the face to grayscale and resize it,
Extract features using HOG and LBP,
Use the trained Random Forest model to predict the identity,
Display the result on the GUI .

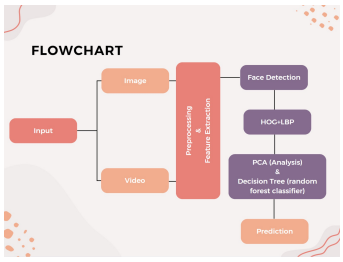


Fig. 17. Flow of the process

e) **Model Interpretability** Understanding why the model makes a certain prediction is just as important as whether it's correct. In our face recognition system, interpretability helped us build trust, catch errors, and improve fairness. We looked at things like brightness, blur, and image clarity to see how they affected the model's decisions. For exam-

ple, blurry images often led to wrong predictions. PCA visualizations also showed which students had clear, unique features and which ones didn't. Furthermore, to prevent the model from making wild guesses, we added a confidence threshold. If it's not at least 50 percent sure, it simply states "Unknown." This made the system much more reliable. We also used the confusion matrix to see who the model was mixing up. We checked the accuracy of the model through that Lastly, we know that face recognition isn't just a tech project, it affects real people. A wrong prediction can lead to missed attendance or denied access. That's why interpretability explainability matters a lot. It helps people trust the system, and helps us make it better.

XII. Division Of Roles

i) Duaa Naz has worked on code and research.
ii) Hadia Sajid has written the report and worked on research.
iii) Huzaifa Awais has written the report and worked on research.
iv) Kashaf Ansari has worked on code and research.

XIII. Link to Project

https://github.com/Duanaz20/face-recognition

References

[1] IBM. *Support Vector Machine (SVM)*. https://www.ibm.com/think/topics/support-vector-machine. Accessed: July 6, 2025.

[2] Correia, J. P., da Silva, L. R., & Silva, R. (2025). *Multifractal analysis and support vector machine for the classification of coronaviruses and SARS-CoV-2 variants*. Scientific Reports, 15, 15041. https://doi.org/10.1038/s41598-025-98366-5. Accessed: July 6, 2025.

[3] S. Dadi and S. R. Pillutla, "Improved Face Recognition Rate Using HOG Features and SVM Classifier," *IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)*, vol. 15, no. 2, pp. 24–29, 2020. [Online]. Available: https://doi.org/10.9790/2834-1502022429

[4] Kienzle, W., Bakır, G., Franz, M., and Schölkopf, B. (2005). *Face Detection — Efficient and Rank Deficient*. Max Planck Institute for Biological Cybernetics, Tübingen, Germany.

[5] Diyasa, I. G. S. M., Putra, A. H., Ariefwan, M. R. M., Atnanda, P. A., Trianggraeni, F., & Purbasari, I. Y. (2022). *Feature Extraction for Face Recognition Using Haar Cascade Classifier*. International Seminar of Research Month 2021, Volume 2022, NST Proceedings, pp. 197–206. http://dx.doi.org/10.11594/nstp.2022.2432

[6] Fung, G. M., Mangasarian, O. L., & Shavlik, J. W. (2003). *Knowledge-Based Support Vector Machine Classifiers*. Computer Sciences Department, University of Wisconsin-Madison.

[7] Jia, H., & Martinez, A. M. (2009). *Support Vector Machines in Face Recognition with Occlusions*. 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 136–141. https://doi.org/10.1109/CVPR.2009.5206862

[8] Santhosh, S., & Rajashekararadhya, S. V. (2023). *A Design of Face Recognition Model with Spatial Feature Extraction using Optimized Support Vector Machine*. 2023 2nd International Conference for Innovation in Technology (INOCON), Bangalore, India, pp. 1–8. https://doi.org/10.1109/INOCON57975.2023.10101149

[9] Mehta, S., Manoharan, M., & Reddy, P. (2017). *Real-Time Face Recognition System Using KPCA–LBP*. International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), 6(4), 509–512. https://d1wqtxts1xzle7.cloudfront.net/52029162/36$_{R}eal-Time_{F}ace_{R}ecognition_{S}ystem_{U}sing_{K}PCA_{L}BP-libre.pdf$

[10] Islam, K. T., Raj, R. G., & Al-Murad, A. (2017). *Performance of SVM, CNN, and ANN with BoW, HOG, and Image Pixels in Face Recognition*. 2017 2nd International Conference on Electrical & Electronic Engineering (ICEEE), Rajshahi, Bangladesh, pp. 1–4. https://doi.org/10.1109/CEEE.2017.8412925

[11] Li, P. (2023). *BioNet: A Biologically-Inspired Network for Face Recognition*. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 10344–10354.

[12] Arias-Duart, A., Mariotti, E., Garcia-Gasulla, D., & Alonso-Moral, J. M. (2023). *A Confusion Matrix for Evaluating Feature Attribution Methods*. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, pp. 3709–3714.

[13] Salhi, A. I., Kardouchi, M., & Belacel, N. (2012). *Fast and efficient face recognition system using random forest and histograms of oriented gradients*. In Proceedings of the 2012 BIOSIG - International Conference of the Biometrics Special Interest Group (BIOSIG), Darmstadt, Germany, pp. 1–11.

[14] Nugrahaeni, R. A., & Mutijarsa, K. (2016). *Comparative analysis of machine learning KNN, SVM, and random forests algorithm for facial expression classification*. In Proceedings of the 2016 International Seminar on Application for Technology of Information and Communication (ISemantic), Semarang, Indonesia, pp. 163–168. doi:10.1109/ISEMANTIC.2016.7873831

[15] Mady, H., & Hilles, S. M. S. (2018). *Face recognition and detection using Random forest and combination of LBP and HOG features*. In Proceedings of the 2018 International Conference on Smart Computing and Electronic Enterprise (ICSCEE), Shah Alam, Malaysia, pp. 1–7. doi:10.1109/ICSCEE.2018.8538377