Person Identification System Using Facial Recognition with Interpretability Emphasis

Jawad Hussain, Abdullah Siraj Khan, Abdul Hadi
Department of Computer Science, IBA University Karachi

***Abstract*—This report discusses our initial understanding and the setup phase of our project focused on developing a person identification system using facial recognition. The focus is not only on achieving high accuracy but also on ensuring model interpretability through computational thinking principles. A dataset shared by the TA was explored, and image preprocessing has been initiated using OpenCV and Matplotlib to prepare for model training.**

I. Introduction

Face recognition has become an essential element in modern identification systems, utilized across security, authentication, and surveillance domains. However, the lack of interpretability, often regarded as the "black box" nature of machine learning models, raises concerns regarding trust, bias, and accountability [1]. This project aims to develop a lightweight yet interpretable machine learning-based face identification system by applying computational thinking principles such as decomposition, abstraction, and pattern recognition.

II. Initial Problem Understanding

The dataset comprises facial image frames of 11 individuals, with approximately 400 frames per person. Upon inspection, the images show variations in eye contact, face orientation, presence of glasses, and lighting conditions. These variations simulate real-world challenges in feature extraction and consistent classification. The system aims to identify individuals by analyzing key facial features such as eyes, eyebrows, lips, birthmarks, jawline, and nose.

III. Literature Review

Numerous facial recognition methods have been proposed. Traditional approaches, such as Eigenfaces and Fisherfaces, utilized Principal Component Analysis (PCA)-based dimensionality reduction [2]. More recently, Convolutional Neural Networks (CNNs) have achieved state-of-the-art performance in facial recognition [9]. However, deep learning models often lack transparency.

Interpretability techniques like saliency maps [4], Local Interpretable Model-Agnostic Explanations (LIME) [8], and Grad-CAM [6] provide visualization of model decisions. These methods align with computational thinking principles and will be considered for integration into this project.

IV. Dataset and Preprocessing Insights

The dataset consists of image frames grouped into folders, each representing one class. Images are resized to 256x256 pixels and converted to grayscale to reduce dimensionality and computational complexity. This step reflects decomposition by breaking down the problem into manageable components such as feature detection, input standardization, and classification.

Core libraries such as NumPy, Pandas, Matplotlib, and OpenCV have been installed and verified. Preliminary visualization indicates that pose normalization or pose-invariant features may enhance accuracy. Project repositories have been created on GitHub for collaborative development.

Fig. 1. Sample image frames from one subject in the dataset.

V. Work Distribution

- Jawad Hussain: Responsible for preprocessing, including dataset setup, resizing images, and grayscale conversion using OpenCV and Matplotlib.
- Abdul Hadi Javed: Leads feature extraction using HOG and CNN filters and will develop the GUI.
- Abdullah Siraj Khan: Manages classification, model training, accuracy testing, and the addition of interpretability tools.

VI. Conclusion

The team has initiated the development of an interpretable facial recognition system by exploring the dataset, establishing preprocessing workflows, and reviewing relevant literature. Next steps involve implementing feature extraction methods, experimenting with classifiers, and integrating interpretability tools. Computational thinking principles will guide future development and evaluation.

VII. Exploratory Data Analysis

A. Introduction

This Exploratory Data Analysis (EDA) explores and highlights our Face Recognition Dataset, comprising images of 12 individuals. The objective is to understand the dataset's weaknesses, distribution, and useful insights.

B. Dataset Summary

- Total Persons: 12
- Total Images: 4025
- Image Format: RGB

C. Class Distribution

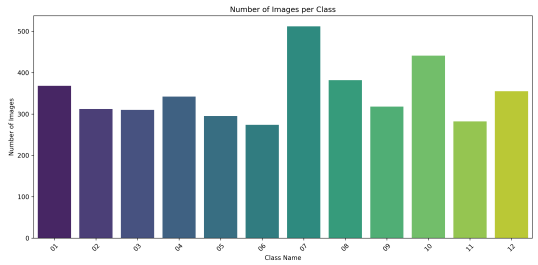

Fig. 2. Number of images per class.

Observation: The dataset shows a slight class imbalance. Class 7 has the highest number of images, while Classes 06 and 11 have fewer images. This imbalance can potentially affect our model's performance.

D. Sample Images

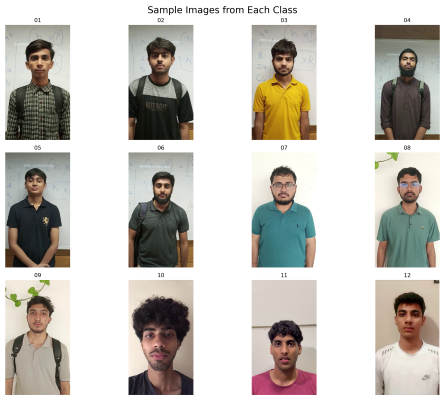

Fig. 3. Sample images from each class.

Observation: Visual inspection reveals variations in lighting, facial orientation, and background consistency across the dataset.

E. Average Pixel Intensity per Class

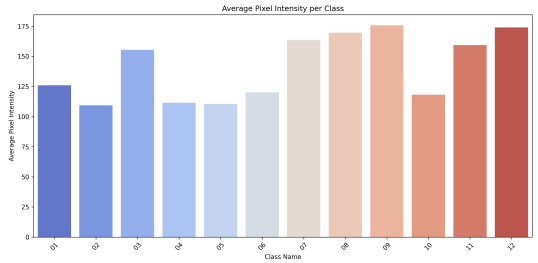

Fig. 4. Average pixel intensity per class.

Observation: Some classes exhibit slightly brighter or darker images on average, which could affect feature extraction consistency.

F. Pixel Intensity Distribution

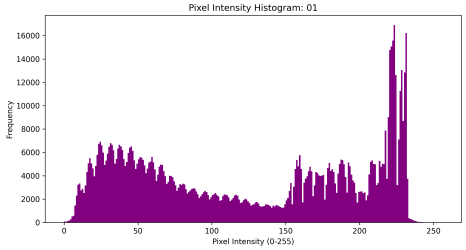

Fig. 5. Pixel intensity histogram for Class 01.

Observation: Peaks are visible in both the dark (0–100) and bright (180–250) pixel ranges, indicating high contrast images.

G. Image Size Distribution

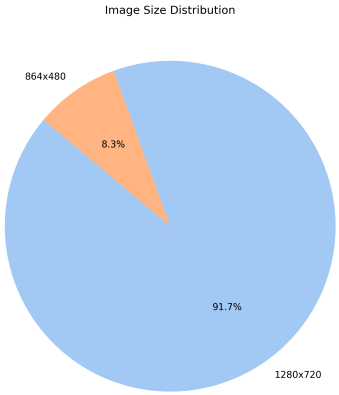

Fig. 6. Image size distribution.

Observation: Most images share the same size; only one class has different image dimensions.

H. Conclusion and Next Steps

- The dataset exhibits mild class imbalance.
- Face variability supports robustness for real-world applications.
- Brightness normalization could enhance model performance.

Recommendations include applying face alignment, using image augmentation, and considering class balancing techniques.

VIII. Model Training

In the initial phase of model training, we used the Decision Tree Classifier algorithm to classify the extracted facial features. The Decision Tree was chosen for its simplicity and interpretability, making it ideal for small datasets like ours.

The model was implemented using the Scikit-learn (sklearn) library in Python. The dataset was split in an 80:20 ratio for training and testing. Features were extracted using the Histogram of Oriented Gradients (HOG) method, which then served as input to the Decision Tree model.

The model achieved an initial accuracy of 95%.

Challenges Encountered:

- Limited dataset size affecting generalization
- Variations in shirt patterns and backgrounds
- Class imbalance favoring individuals with more images
- Lighting, facial angle, and expression inconsistencies

Future work will involve advanced models like SVMs, face-only cropping, and more extensive training.

IX. Feature Extraction

We used the Histogram of Oriented Gradients (HOG) technique for feature extraction due to its effectiveness in edge and shape-based recognition tasks.

Steps:

- Grayscale conversion
- Gradient computation (angle and magnitude)
- Dividing the image into cells and computing histograms
- Merging histograms into final feature vectors

Why HOG:

- Captures facial structure through edges
- Handles lighting and background variation well
- Produces compact vectors suitable for interpretable models like Decision Trees

We also used HOG visualization to increase interpretability, with emphasis on features like eye shape, eyebrows, and hair.

X. Project Pipeline

This section outlines our complete project development process and pipeline. It includes all the major phases from idea formulation to deployment and future working.

A. A. 1. Idea / Problem Definition

The core problem we aimed to solve was the lack of interpredebility in facial recognition systems. Most of the used models provide high accuracy but failed to explain how prediction are made, making them classic black box models. Our goal was to build a lightweight face recognition system that not only identifies individuals correctly but also visualizes and explain the decision'-making process behind each prediction.

B. B. 2. Requirements Gathering

We began by listing the technical requairemnts of our system:

- **Programming Language:** Python
- **Libraries:**
  - OpenCV (for image processing)
  - scikit-learn (for training Decision Tree)
  - matplotlib (for graph visualizations)
  - skimage (for HOG feature extraction)
  - Tkinter (for GUI)
- **Dataset:** 4025 labeled facial images of 12 individuals
- **Target Outcome:** An interpretable face recognition system with a working GUI

C. C. 3. Planning and Design

We structured our system as a modular pipeline with the following major blocks:

- Data loading and preprocessing
- Feature extraction using HOG
- Training a Decision Tree model
- Evaluation and interpretability tools
- A GUI for end-user interaction

D. D. 4. Development / Implementation

We implemented the full pipeline using Python. Here's how the development was structured:

- **Dataset Preprocessing:** Converted images to grayscale, resized to 256x256, organized by labels like (person 1, 2, etc.)
- **Feature Extraction:** Used HOG (Histogram of Oriented Gradients) to extract shape and edge-based features.
- **Model Training:** A Decision Tree Classifier was trained using scikit-learn on extracted HOG features.
- **Decision Path Extraction:** Used scikit-learn's decision path methods to trace which rules led to the prediction.
- **GUI Development:** Built a Tkinter-based interface where users can upload an image, predict the identity, and visualize the decision path. Our GUI also supports live face detection and recognition.

Fig. 7. Graphical User Interface Preview

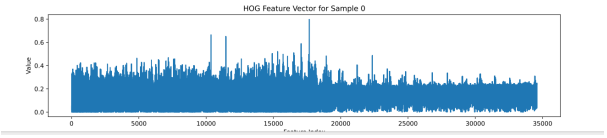

Fig. 8. HOG Feature Representation

E. E. 5. Testing

We tested the model's accuracy and system usability:

Model Testing:

- Dataset split: 80% training, 20% testing
- Accuracy achieved: initially 95 then increased to 99%
- Evaluation tools: Confusion matrix, classification report

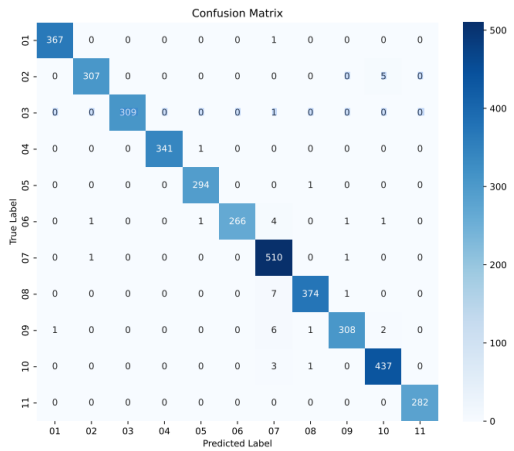

Fig. 9. Model Performance Visualization

GUI Testing:

- Verified predictions with multiple image inputs.
- Checked that HOG and decision path updates correctly.

F. F. 6. Deployment

The entire project runs locally. The user only needs to:

- Run `main.py` to train the model (optional)
- Run `gui.py` to use the system

All code and setup instructions are available on GitHub.

G. G. 7. Maintenance and Future Work

Although the system works as intended, there are possible future improvements that can be made if we consider polishing our system:

- Improve GUI aesthetics and layout
- Use face alignment and brightness normalization
- Replace Decision Tree with more scalable models like SVM or Random Forest while preserving interpretability
- Make an exe file for user

XI. Graphical User Interface (GUI)

We developed a graphical user interface (GUI) using Python's Tkinter to make our face recognition system more interactive and user-friendly.

The GUI provides two main functions:

- **Upload Image:** The user can select an image from the device. The system will extract HOG features, predict the person's identity using the trained Decision Tree model, and display the corresponding label of the person alongside some visual explanation.
- **Live Camera Support:** The user can also turn on the webcam for real-time face detection. The system automatically detects faces and predicts the identity on the live input.

Our GUI also shows the feature visualization and the decision path taken by the model to reach its prediction. This helps make our system transparent and interpretable.

Fig. 10. Screenshot of the GUI with prediction and decision path explanation.

XII. Source Code and Reproducibility

The complete source code, training pipeline, dataset builder, and GUI application are available on GitHub. Due to file size constraints, trained model files (`.pkl`) are not uploaded. However, they can be regenerated using the included scripts.

GitHub Repository: https://github.com/Jawad-Hussain-dev/Face$_{D}etection$

REFERENCES

[1] D. Castelvecchi, "Can we open the black box of AI?" *Nature*, vol. 538, no. 7623, pp. 20–23, 2016.

[2] M. Turk and A. Pentland, "Eigenfaces for Recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

[3] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A Unified Embedding for Face Recognition and Clustering," in *Proc. CVPR*, 2015.

[4] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps," *arXiv preprint arXiv:1312.6034*, 2013.

[5] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why Should I Trust You? Explaining the Predictions of Any Classifier," in *Proc. KDD*, 2016.

[6] R. R. Selvaraju et al., "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization," in *Proc. ICCV*, 2017.

[7] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *Proc. IEEE CVPR*, vol. 1, pp. 886–893, 2005.

[8] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why Should I Trust You?: Explaining the Predictions of Any Classifier," in *Proc. ACM SIGKDD*, pp. 1135–1144, 2016.

[9] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A Unified Embedding for Face Recognition and Clustering," in *Proc. IEEE CVPR*, pp. 815–823, 2015.