Real-Time Facial Recognition System with Web Client and Explainable Backend

Hiba Fatima[#1], Reham Hafeez[#2], Kabeer Jaffri[#3]

Department of Computer Science, Institute of Business Administration Karachi

[1]h.fatima.29005@khi.iba.edu.pk
[2]r.hafeez.30574@khi.iba.edu.pk
[3]k.jaffri.29027@khi.iba.edu.pk

***Abstract*—** This project proposes the implementation of a real-time facial recognition system, which uses a web-based client, an optimized lightweight two stage model architecture. The system is to be trained on a custom dataset of approximately 350-400 image frames for each of 11 individuals provided by the instructor. The client-side component, implemented as a web-application, will perform lightweight face detection to draw bounding boxes around detected faces (up to N individuals) directly on a live video feed, cropping and then delivering through RPC calls, thereby optimizing network bandwidth. The backend will host a Local Binary Patterns Histograms (LBPH) model, to perform individual identification. This report outlines the initial problem analysis, architecture, methodology, and discusses the feasibility of traditional AI algorithms and edge model face detection for a seamless UX.

Keywords— Web-Assembly, Face Detection, Multi-staged modeling, Real-time

I. Introduction

Facial recognition technology has become important in many modern applications, ranging from security to Snapchat® filters. This project aims to develop a real-time facial recognition system focusing on a specific dataset of 11 individuals.The core objective is to train a reliable recognition model using C/C++ without higher level libraries, one of our goals is to maximize the use of matrix multiplications in hopes to achieve more efficient deployment on specialized hardware in the future.

II. Problem Understanding

The central problem is to develop a facial recognition system capable of identifying individuals from a predefined dataset in real-time. Goal is not to create a accurate model, but a proof-of-concept, Key challenges associated with this include:

A. Variability in Lighting, Pose, and Orientation:

Real-world scenarios introduce diverse environmental conditions that can significantly impact detection and recognition accuracy.

B. Maintaining Image Quality Consistency:

Real-world scenarios introduce diverse environmental conditions that can significantly impact detection and recognition accuracy.

C. Real-Time Model Response

Achieving low latency between capturing a frame and presenting the identification result, which is critical for a "real-time" system. This project specifically addresses this by proposing a split architecture:

I. *Client-Side (Web Application):* Responsible for capturing the webcam feed, maintaining web-socket connection, performing initial face detection, and drawing bounding boxes directly in the browser and cropping images to send to the backend host. This offloads computational burden and reduces data transfer to the backend.

II. *Backend (Server):* Responsible for face identification using the trained model, receiving only cropped face images from the client.

III. Literature Review

All paragraphs must be indented. All paragraphs must be justified, i.e. both left-justified and right-justified.

A. LBPH (Local Binary Patterns Histograms) for Face Recognition

The Local Binary Patterns Histograms (LBPH) algorithm is a widely recognized and effective face recognition technique. It operates by extracting local texture features from grayscale images. The image is first divided into small regions, and for each region, Local Binary Pattern operators are applied to generate a histogram. These histograms are then concatenated to form a feature vector, which is used for classification. LBPH is known for its computational efficiency, and good performance on small datasets like ours.

B. Viola-Jones Algorithm for Face Detection

The Viola-Jones algorithm is an influential work in real-time object detection, again particularly for faces. It uses HAAR[1]-like features, and a cascade structure to quickly discard non-face regions. This cascaded approach makes it exceptionally fast and suitable for real-time applications, including its use as the initial face detection step in many recognition systems.

C. Deep Learning Approaches (DeepFace, FaceNet, VGGFace)

Deep learning approaches have revolutionized facial recognition. Models like DeepFace, FaceNet, and VGGFace utilize Convolutional Neural Networks (CNNs) while some use Vision Transformers to learn robust, high-level feature representations (embeddings). These embeddings are then used for classification or other tasks.

I. *Pros*: Significantly higher accuracy, robust to variations in pose, lighting, and ability to generalize to unseen data using drop-out layers and other methods.

II. *Cons*: Generally requires massive datasets for training, computationally intensive (demanding powerful hardware like GPUs), often act as "black boxes" and good visualization tools are only feasible on smaller models, xAI (Explainable AI, not the company) does make it easier to create models which are explainable and visualizable, in general while highly accurate, DL's computational demands and lack of inherent explainability might make them less suitable for this project's constraints on hardware and model interpretability, especially given the provided dataset size.

D. Justification for Selecting LBPH

LBPH is selected as the primary face recognition algorithm for this project for several compelling reasons:

I. *Dataset Suitability:* The project utilizes a relatively small custom dataset of 11 individuals (350-400 images each). LBPH performs efficiently and achieves acceptable accuracy on such constrained datasets.

II. *Real-Time Performance:* LBPH, being less computationally intensive than deep learning models, can offer real-time performance on standard CPU hardware, making it suitable for backend processing where dedicated GPUs might not be available for inference.

III. *Explainability/Interpretability:* While not a decision tree in the traditional sense, the LBP features themselves and the histogram comparison process offer a more transparent and understandable mechanism than the complex, multi-layered operations of a CNN. This aligns with the project's interest in a "visualizable" or "explainable" model concept.

IV. *Project Scope:* It aligns well with foundational computer vision concepts and avoids the extensive infrastructure requirements associated with deep learning libraries, allowing focus on the system integration and web client.

IV. Dataset Description and EDA

Exploratory Data Analysis will be conducted to gain insights into the dataset's properties and quality.

I. *Image Distribution*: Charts to be generated to visually represent the exact count of images available for each of the 11 individuals, ensuring no large imbalances exist that could bias the training.

II. *Sample Images*: A head / grid of sample images will be displayed for each person to visually inspect the variability in pose, expression, color, lighting and background within their respective image sets.

III. *Consistency Checks*: Image sizes, resolutions, and color profiles will be checked for any inconsistencies that might require pre-processing steps. Observations will be noted regarding image quality, (e.g., blurry images). Some steps may also be needed to remove RGB channels, and preserve the alpha channel.

Task Distribution

I. ***Kabeer Jaffri****: Data Loading and Preparation*
Job: Writing core function logic to work with the file system, discover the classes (people) and how many pics of each. This is the base level which provides the data structures for everyone else.
Function: load_and_prepare_data()

II. ***Reham Hafeez****: Data Visualization*
Responsibility: Duties and Responsibilities: Generate all EDA's visual deliverables. This includes showing a bar chart that shows the distribution of the classes, as well as a grid of images to get a qualitative feel for the dataset. This position is all about turning raw data into compelling graphics.
Functions: visualize_class_distribution()

III. ***Hiba Fatima****: Data Visualization*
Responsibility: To perform a deeper technical analysis of the data and to generate the final summary report. This includes checking image properties like dimensions and color modes for consistency and presenting the findings in a clean, tabular format.
Functions: analyze_image_properties(), generate_summary_report()

V. Methodology

Our methodology is divided into client and server implementations, orchestrating real-time face detection and recognition for a seamless user experience.

I. *Face Detection (Client-Side)*
The web client will access the user's webcam feed.

A. *Real-Time Frame Processing:* JavaScript will continuously capture frames from the video stream.

B. *Client-Side Face Detection Model:* An instance of a lightweight face detection model or a custom implementation (e.g., TF.js's versions of Blaze Face, or Face Landmark Detection Models) will run directly in the browser on Web Assembly

C. *Feasibility:* Running face detection in the browser is highly feasible and widely adopted. OpenCV.js compiled OpenCV's C++ library to WebAssembly, enabling near-native performance for image processing tasks directly in the browser. This negates the need to send entire video frames to the backend, drastically reducing bandwidth requirements.

D. *Bounding Box Drawing*: Upon detecting faces (designed to handle up to 2 individuals simultaneously for a cleaner demonstration), the client will draw bounding boxes and confidence scores directly onto the video canvas in real-time, and show a loading indicator.

E. *Data Preparation for Backend*: For each detected face, the client-side script will crop the face region from the original frame, resize it to a target dimension, convert it to grayscale, and serialize and send a RPC request.

II. *Real-Time Testing and Seamless Integration*

I. *Frame Rate Control:* To prevent overwhelming the backend and network, the client will implement a frame rate control mechanism. Instead of sending every live frame, it sends frames at a fixed reduced rate (e.g., 5-10 frames per second, clock cycle implementation), or only when a new face enters/leaves the frame, or when the confidence of an existing recognition drops below a threshold.

II. *Visual Feedback:* During the round-trip communication with the backend, the client will display visual cues (e.g., "Identifying...", a loading spinner within the bounding box, or temporary labels like "Unknown") to inform the user that processing is happening in the background. Once the backend responds, the detected bounding box will be updated with the predicted name and confidence score.

VI. Conclusion and Future Work

This project aims to deliver a functional real-time facial recognition system accessible via a web client, employing a pragmatic two-stage processing pipeline. The selection of LBPH for backend identification ensures a balance between performance, dataset suitability, and relative interpretability. The critical decision to implement face detection client-side using WASM, is pivotal for achieving a smooth and responsive user experience, significantly reducing the data processing load on the backend and network.

Planned Improvements and Future Work:

I. *Alternative Feature Extractors/Models:* Experiment with other classical face recognition algorithms (e.g., Eigenfaces, Fisherfaces) for comparison against LBPH's performance and characteristics.

II. *Deep Learning for Enhanced Accuracy:* If computational resources permit, explore integrating lightweight CNN models (e.g., MobileNet, SqueezeNet) as feature extractors, with a simpler classifier (like SVM or even a Random Forest on the extracted embeddings) for identification, to potentially boost accuracy while retaining some explainability (via feature visualization).

III. *Dataset Augmentation:* Implement data augmentation techniques (e.g., rotations, flips, brightness adjustments) to increase the effective size and variability of the training dataset, which can improve model generalization.

IV. *Deployment Optimizations:* Optimize the web client for various browsers and devices. Further investigate WASM for backend inference if the performance of the Python server becomes a bottleneck for larger datasets or more users.

V. *GUI/User Interface Enhancements:* Develop a more sophisticated user interface for model management, adding new users, or viewing recognition logs.

Appendix [A]

[1] Exploratory Data Analysis Outputs. “Contains face counts, channel structures and other distributions regarding data”

Acknowledgment

We acknowledge the creators of Pandas, Google colab (Google INC) as well as creators of other libraries that made this project possible.

References

[1] T. Ahonen, A. Hadid, and M. Pietikainen, "Face Description with Local Binary Patterns: Application to Face Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2006.

[2] P. Viola and M. J. Jones, "Robust Real-Time Face Detection," *International Journal of Computer Vision*, 2004.

[3] Y. Taigman et al., "DeepFace: Closing the Gap to Human-Level Performance in Face Verification," *CVPR*, 2014.

[4] F. Schroff et al., "FaceNet: A Unified Embedding for Face Recognition and Clustering," *CVPR*, 2015.