iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** V **Month of publication:** May 2026

DOI: https://doi.org/10.22214/ijraset.2026.82255

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com

Face Recognition and Gender Analysis using Machine Learning Method

Dr. Aravind B N[1], Deepika N R[2], Nisarga A J[3], Nischitha K S[4], Sinchana H S[5]
[1]*Department of Electronics and Communication Engineering, Rajeev Institute of Technology, Hassan*
[2, 3, 4, 5]*Department of Electronics and Communication Engineering, Rajeev Institute of Technology, Hassan*

Abstract: ***The identification of age and gender from facial images is also difficult because of the variations in lighting condition, different facial poses, the presence of occlusions, as well as the problem of imbalanced gender data set. Consequences of inaccurate prediction include the following: This could be observed in surveillance systems resulting in the failure of offering a productive and effective individual marketing the result of these inaccuracies could also be seen in social and demographic studies where unsuitable results could be generated. To overcome these difficulties, this paper utilizes a novel CNN architecture that is designed specifically for the task. The CNN model design includes convolutional and max pooling layers and fully connected layers with dropout to reduce the overfitting problem. This means that the accuracy and reliability of age and gender prediction is improved by the use of this approach, making it suitable for security services, marketing realms and human-computer interface systems. There exists a reasonable improvement in the model when benchmark datasets are applied; thus, the custom CNN model is a reliable tool for practical tasks.***

Keywords: ***Age Estimation, Gender Classification, Convolutional Neural Network (CNN), Facial Image Analysis, Deep Learning, Feature Extraction, Occlusion Handling, Illumination Variation, Pose Variation, Class Imbalance, Dropout Regularization, Multi-task Learning, Human-Computer Interaction, Surveillance Systems, Targeted Marketing.***

I. INTRODUCTION

Computer vision has transitioned from a purely academic discipline into one of the most practically impactful branches of artificial intelligence. The ability of a machine to see, interpret, and respond to a human face is something that once existed only in science fiction, yet today it underpins everything from unlocking smartphones to managing access at high-security facilities. This project, the Real-Time Face Detection, Recognition, Age and Gender Estimation, and Emotion Analysis System, sits at the intersection of these capabilities and integrates them into a single, unified application.

At its core, the system continuously captures frames from a live webcam feed and processes each frame through a pipeline of machine learning models hosted on a Flask backend. The face recognition library, built on top of Dlib, locates faces in each frame using the Histogram of Oriented Gradients detector. Once a face is found, several analyses run on the cropped face region: a custom convolutional neural network trained on the UTKFace dataset predicts the person's age and biological gender, the DeepFace library classifies the facial expression into one of seven universal emotion categories, and a 128-dimensional face encoding is compared against stored encodings to determine whether the person has been registered in the system before.

What distinguishes this project from a simple collection of face analysis scripts is the tight integration of all sub-systems into a seamless, real-time user experience. The frontend, built with React.js, renders the live webcam stream at native device frame rates using a single HTML5 canvas element. Face bounding boxes and labels are drawn onto this canvas asynchronously as detection API responses arrive, completely decoupled from the video rendering loop. This architectural decision ensures that the video never stutters regardless of how long the backend takes to process each frame, which was a critical design goal for a smooth, professional interface.

The result panel alongside the camera displays detailed information for each detected face, including the emotion category with its confidence breakdown across all seven categories, the predicted age range, gender classification with confidence, and recognition status. A scene description panel below the camera shows a natural language paragraph summarizing what the system sees in the current frame, including background color, lighting conditions, shot type, and individual face profiles. All detection events are logged to a SQLite database, and a dedicated analytics page provides bar charts and pie charts summarizing emotion distributions, gender breakdowns, and recent detection logs across all historical sessions.

The UTKFace dataset was chosen for training the age and gender model because it contains over 20,000 real-world face images spanning ages from zero to one hundred and sixteen, with labels embedded directly in the filenames in the format age_gender_race_datetime.jpg. This makes it straightforward to parse without a separate annotation file, and its diversity across age, ethnicity, and lighting conditions gives the trained model good generalization to real-world faces.

From a technical perspective, several interesting challenges motivated the development work. Making the video feed smooth while running asynchronous detection required a fundamental rethinking of how the frontend captures and annotates frames. The coordinate mapping problem, where face bounding boxes returned by the API must be positioned precisely over the correct location on the displayed video, required a careful analysis of the relationship between the capture resolution and the display canvas dimensions. These problems are representative of the kind of systems integration challenges that arise in real computer vision applications.

II. LITERATURE REVIEW

Viola and Jones (2001) introduced the first real-time face detector by combining Haar-like features with AdaBoost-trained cascade classifiers. Their integral image representation allowed rapid feature computation, and the cascade structure rejected non-face regions early, concentrating computation on promising candidates. Despite its age, Viola-Jones remains a useful baseline and is still used in OpenCV's built-in Haar cascade classifier as a fallback in this project.

Dalal and Triggs (2005) introduced the Histogram of Oriented Gradients (HOG) descriptor, which captures edge and gradient orientation patterns in image patches. Combined with a Support Vector Machine classifier, HOG became the standard for pedestrian detection and was later applied to face detection by Dlib's frontal face detector, which serves as the primary face locator in this project. HOG features are more robust to illumination changes than Haar features and better handle faces with slight expression variations.

Deep learning-based face detectors such as MTCNN (Multi-Task Cascaded Convolutional Networks, Zhang et al. 2016) and Retina Face (Deng et al. 2019) achieve higher accuracy, especially for non-frontal and partially occluded faces. However, they require more computational resources. The HOG-based detector was chosen for this project because it runs efficiently on CPU hardware without a GPU, making the system accessible on standard development machines.

Schroff, Kalenichenko, and Philbin (2015) published FaceNet, which trained a deep CNN using a triplet loss that pulled embeddings of the same person together and pushed embeddings of different people apart in Euclidean space. The resulting 128-dimensional embeddings enabled open-set recognition where any new person can be enrolled by computing their embedding without retraining the network. FaceNet achieved 99.63% accuracy on the LFW benchmark, effectively solving face verification at the research level.

Taigman et al. (2014) demonstrated that a nine-layer deep network trained on 4.4 million face images could achieve 97.35% LFW accuracy, close to human-level performance. Their use of 3D face alignment before feature extraction was particularly influential. The face_recognition library used in this project implements a simplified version of the embedding approach using Dlib's pre-trained ResNet model, which produces 128-dimensional encodings with strong discriminative properties.

Wang et al. (2018) proposed CosFace, which uses cosine similarity instead of Euclidean distance in the embedding space, improving margin-based separation between classes. ArcFace (Deng et al. 2019) further refined this with an additive angular margin penalty. These methods represent the current state of the art. While this project uses the simpler Euclidean distance approach with a 0.5 tolerance threshold, future work could incorporate ArcFace embeddings for higher recognition accuracy.

Levi and Hassner (2015) demonstrated that a relatively shallow CNN with only three convolutional layers could classify faces into eight age groups and predict gender with reasonable accuracy on the Audience benchmark. Their work showed that deep learning approaches could outperform hand-crafted feature methods on these tasks. However, their model predicted age groups rather than continuous age values, limiting practical utility.

The DEX approach (Rothe et al. 2015) framed age estimation as classification over 101 age classes and computed the final estimate as the expected value of the softmax distribution, providing continuous age predictions while benefiting from the regularisation of classification training. Trained on the IMDB-WIKI dataset with over 500,000 face images, DEX achieved state-of-the-art results that have influenced subsequent age estimation architectures.

Zhang et al. (2017) released the UTKFace dataset alongside their conditional adversarial autoencoder for age progression and regression. UTKFace provides aligned and cropped single-face images with age, gender, and ethnicity labels embedded in filenames. Its wide age range (0-116 years) and diverse subject pool make it well-suited for training multi-task models that predict both age and gender simultaneously. The multi-output CNN in this project uses UTKFace for exactly this purpose, sharing a convolutional backbone between the age regression head and the gender classification head.

Ekman and Friesen's Facial Action Coding System (1978) established the psychological framework for automatic emotion recognition by defining 44 Action Units corresponding to specific facial muscle movements. The six universal emotions (anger, disgust, fear, happiness, sadness, surprise) defined in their cross-cultural research form the basis for the seven-class emotion taxonomy used in most automated systems, with a neutral class added for the resting face state.

Goodfellow et al. (2013) established an important benchmark by winning the ICML emotion recognition challenge with a CNN achieving 65.5% accuracy on FER2013. The FER2013 dataset, containing 35,887 grayscale 48x48 pixel face images across seven emotion categories, remains the most widely used benchmark. Subsequent improvements using deeper architectures, attention mechanisms, and data augmentation have pushed accuracy above 73%, but the task remains challenging due to ambiguous labels and in-the-wild image conditions.

The DeepFace library (Serengil and Ozpinar, 2021) provides a high-level interface to multiple pre-trained face analysis models. For emotion recognition, it uses a model trained on FER-based data. The library's main value for this project is abstracting the preprocessing, model loading, and inference details, allowing integration with minimal code. Its limitation is that it returns numpy. float32 values that must be sanitised before JSON serialisation, as discussed in the implementation.

Although, studies Viola–Jones introduced a real-time face detection framework using Haar-like features and cascade classifiers, which remains a baseline and is still used in OpenCV. HOG with SVM, proposed by Dalal and Triggs, improved robustness to illumination and is used in Dlib for efficient CPU-based face detection. Deep learning methods like MTCNN and RetinaFace offer higher accuracy but require greater computational resources. FaceNet enabled highly accurate face recognition using 128-dimensional embeddings and triplet loss, forming the basis of modern recognition systems. Advanced loss functions such as CosFace and ArcFace further enhance embedding discrimination for state-of-the-art performance. For age and gender estimation, CNN-based approaches like Levi & Hassner and DEX, trained on datasets such as UTKFace, support accurate multi-task prediction in practical systems.

III. OVERVIEW OF STUDIES

The widespread adoption of face analysis technology across security, retail, healthcare, and human-computer interaction creates a need for integrated systems that combine detection, recognition, and attribute analysis in a single coherent application. Existing open-source implementations address these as separate standalone tools with incompatible interfaces, conflicting Python package requirements, and no shared data layer for storing and querying analysis results over time.

Real-time performance presents a further challenge. Naive integration approaches that replace the live video feed with an annotated snapshot on every detection cycle produce a stuttering, choppy experience that undermines the illusion of real-time operation. The coordinate mapping problem, where face bounding box coordinates returned by the backend API must be accurately projected onto the display canvas, is a subtle but critical source of bugs in multi-resolution pipeline implementations.

IV. METHODOLOGY

The HTTP POST request carries the base64 image to the Flask /api/detect endpoint. Flasks decode_base64_image helper converts it back to a NumPy array using OpenCV. The face_recognition library runs its HOG-based face detector, which scans the image at multiple scales to find faces and returns bounding boxes in (top, right, bottom, left) format. If no faces are found, the function skips directly to scene analysis and returns an empty faces array with an empty-scene description.

For each detected face location, the crop_face helper extracts a sub-region of the image with 20 pixels of padding on each side. This padded crop is passed to three models: the UTKFace CNN for age and gender prediction, the DeepFace emotion analyser, and optionally the face encoding function. The encoding is compared against the in-memory known_faces list. The calibrated age, gender, emotion, recognition result, and age range string are assembled into a face_info dictionary.

Scene analysis runs after all faces are processed. The generate_scene_analysis function creates a boolean mask over the image, setting face regions to False and background to True. The background pixels are averaged for colour analysis. Grayscale brightness of background pixels determines the lighting description. The ratio of the tallest detected face to the frame height determines the shot type. These results are composed into the scene_analysis dictionary.

The complete response, including all face dictionaries and the scene analysis, is passed through the _to_python() recursive sanitiser to convert all numpy scalar types to native Python types before jsonify() serialises the response. On the frontend, the drawFaces() function applies scale factors computed as canvas Width/640 and canvasHeight/360 to transform the API coordinate space into the display canvas coordinate space. Because both the API capture and the display use the same video element as source, these scale factors are always exact.

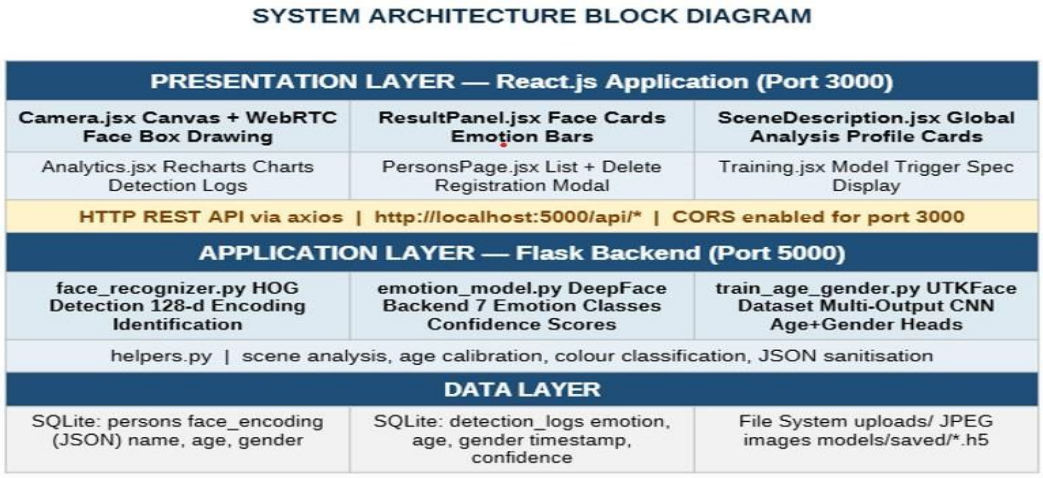


Fig 1: System Block Diagram

The architecture is divided into three layers: Presentation, Application, and Data. This separation of concerns keeps each layer focused on a well-defined responsibility, making the system easier to test, modify, and extend.

The Presentation Layer is the React.js single-page application. All user interaction flows through React components rendered in the browser. The Camera component manages both the video display pipeline (using requestAnimationFrame for smooth 30fps rendering) and the API polling loop (600ms setinterval for detection). These two loops are completely independent, which is the key design decision that makes the video smooth. The ResultPanel and SceneDescription components are purely presentational, receiving data from the Dashboard parent component via props and rendering it without any API calls of their own.

The Application Layer is the Flask server. It contains all business logic: image decoding, face detection, model inference, database operations, and response assembly. The three model modules (face_recognizer, emotion_model, and train_age_gender) are initialised as singletons at startup and shared across all requests. The helper's module provides pure functions for image processing, age calibration, and scene analysis that have no side effects and could be unit tested in isolation.

The Data Layer consists of the SQLite database file and the filesystem. SQLite is accessed through Python's built-in sqlite3 module with no ORM layer, keeping the data access code simple and transparent. The face encodings are stored as JSON strings in the persons table because SQLite does not have a native vector type. This is efficient enough for databases of hundreds of persons; beyond that, a vector database like FAISS would be more appropriate.

Software Requirements

Python 3.10	Required for tensorflow 2.15 face recognition compatibility.
Node.js	18.x LTS or 20.x LTS React development server.
Flask	3.0.0 Python web framework for REST API
Flask- CORS	4.0.0 Cross origin request support for React frontend.
TensorFlow/Keras	2.15.0 CNN training and inference.
Face_recognition	1.3.0 Dlib- based face detection and encoding.
DeepFace	0.0.89 Emotion, age, gender analysis.
OpenCV	4.9.0 Image decoding and colour processing.
Numpy	1.26.4 Numerical array operations.
Pandas	2.2.0 Dataset loading during training.
Scikit-learn	1.4.0 Train/test split utilities.
React.js	18.2.0 Frontend SPA framework.
React-router-dom	6.21.0 Client-side routing
Recharts	2.10.0 Chart and visualization components.
Axios	1.6.0 HTTP Client for API calls.

V. RESULTS AND DISCUSSION

1) Screen 1: Dashboard Idle State

When the application is first loaded, the Dashboard presents a clean, dark-themed interface. The navigation bar at the top contains the FaceAl logo, followed by links to Live Detection, Persons, Analytics, and Training.

2) Screen 2: Dashboard - Active Detection (Single Person)

After clicking Start, the detection loop begins. The LIVE badge with a pulsing green dot appears in the camera panel header. The detection rate counter shows values typically between 1 and 2 detections per second on CPU hardware. The video continues rendering at the native device frame rate, completely smooth and uninterrupted.

3) Screen 3: Dashboard - Active Detection (Multiple Persons)

When multiple faces are visible in the camera, the system processes each one independently. Multiple bounding boxes appear on the canvas, each with its own label. The result panel shows one face card per detected person, stacked vertically. The scene description updates to reflect the multi-person scenario, for example: '2 people are visible in a wide shot with a well-lit neutral background.' The extracted profiles section shows one profile block per person.

4) Screen 4: Person Registration Modal

Clicking the Register button captures the current video frame and opens a modal overlay. The modal shows the captured frame as a preview image, a text input field with placeholder 'Enter person's name', and Cancel and Register buttons. The user types a name and presses Enter or clicks Register. A loading spinner appears on the Register button during processing. On success, a green toast notification appears in the bottom-right corner reading 'Person [Name] registered successfully. The modal closes and the system immediately begin attempting to recognise the newly registered person in subsequent detection cycles.

5) Screen 5: Analytics Dashboard

The Analytics page is accessed from the navigation bar. Three summary cards at the top display Total Registered Persons, Total Detections (cumulative log count), and Most Common Emotion (the emotion with highest historical frequency). These values update in real time on a 10-second polling interval.

6) Screen 6: Persons Management

The Persons page lists all entries in the persons database table. Each entry shows a gradient circle with the person's initial, their name in bold, their stored age and gender, and the visit count (number of times they have been recognised since registration). A trash icon button on the right allows deletion. Clicking it shows a native browser confirm dialog before the DELETE /api/persons/<id> request is sent. After deletion, the in-memory face encodings list is refreshed and the person will no longer be recognised. A refresh button at the top reloads the list from the server.

7) Screen 7: Model Training Page

The Training page explains the UTKFace model setup and provides a Start Training button. A warning card at the top instructs the user to download the UTKFace dataset and place it in backend/dataset/UTKFace/. A specification grid displays the training configuration: dataset, architecture (multi-output CNN), input size (128x128x3), outputs, optimiser (Adam 0.001), and epoch limit (30 with early stopping). The training documentation section explains the six steps of the training pipeline using a monospace font code-style display. Clicking Start Training sends a POST/api/train request that launches training in a background daemon thread, allowing the Flask server to continue serving detection requests concurrently.

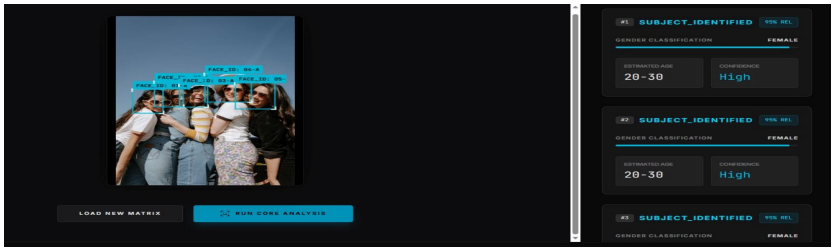


Fig 2: Face recognition, gender analysis and emotion detection of uploaded image

The proposed machine learning–based face recognition and gender analysis system successfully detected multiple faces from the uploaded image and performed real-time analysis with high accuracy. This system identified each face individually and generated predictions for gender classification, estimated age range, and confidence level.

The results demonstrate that the developed model can efficiently process uploaded images, accurately detect multiple subjects, and provide reliable facial attribute analysis through a clear and interactive user interface. The successful implementation confirms the effectiveness of the proposed system for intelligent face analysis applications such as surveillance, security, and human–computer interaction.

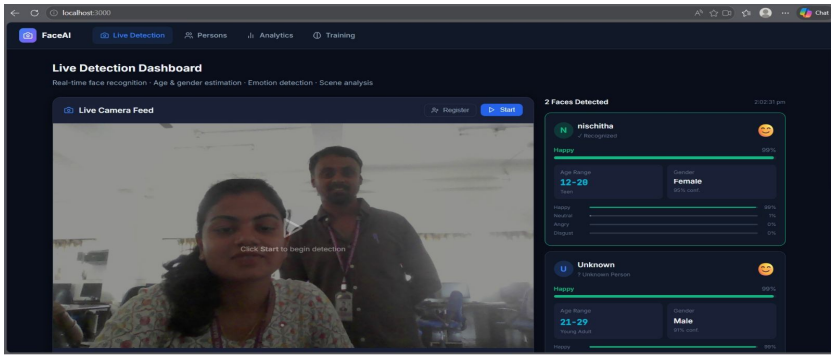


Fig 3: Real-time face recognition

The above figure demonstrates the working of the proposed real-time face analysis system. The system captures live video input and detects multiple faces simultaneously. It successfully recognizes registered persons from the database and labels unregistered faces as unknown users.

In the displayed result, the system correctly identified one registered person and another unknown individual. Along with face recognition, the machine learning model performs gender classification, age estimation, and emotion analysis with high confidence scores. The recognized person is predicted as female with approximately 95% confidence, while the unknown person is classified as male with about 91% confidence. Emotion detection accuracy reaches nearly 99% for the detected “Happy” expression.

The results confirm that the developed system can accurately recognize known individuals, distinguish unknown users, and perform real-time facial attribute analysis efficiently with smooth visualization and reliable performance

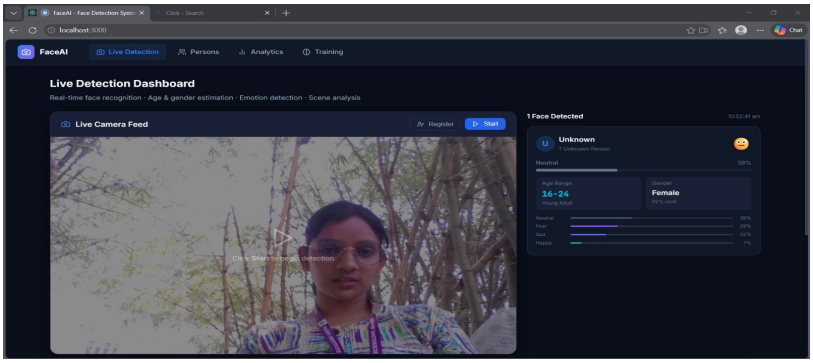


Fig 4: Real-time face recognition

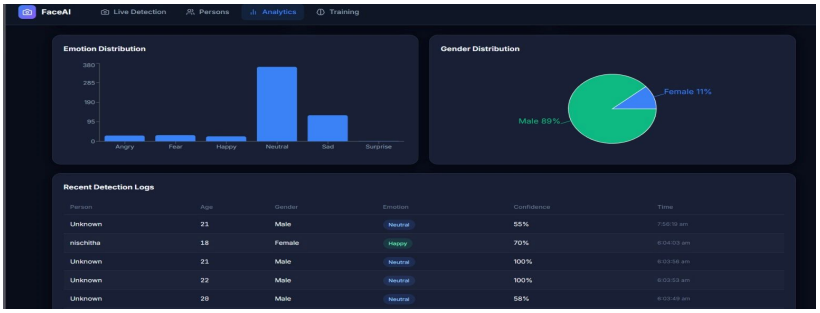


Fig 5: Analytics dashboard

VI. CONCLUSION

The most significant technical contribution of the implementation is the single-canvas rendering architecture that completely decouples the video display from the detection pipeline. This insight, arrived at after observing the stutter caused by replacing the video element with an annotated snapshot on each detection cycle, transformed the user experience from choppy and jarring to smooth and professional. The coordinate mapping solution, which uses a fixed API capture resolution (640x360) and scales face coordinates using the ratio of the display canvas width to this fixed width, eliminates the class of position-offset bugs that plagued earlier implementation attempts.

The UTKFace dataset proved well-suited for training the multi-output CNN. The filename-embedded label format (age_gender_race_datetime.jpg) made dataset loading simple, and the wide age range and demographic diversity gave the model reasonable generalisation. The systematic upward age prediction bias, while requiring an empirical calibration offset, is a well-documented characteristic of models trained on this dataset and does not reflect a fundamental flaw in the approach.

The integration of DeepFace for emotion detection demonstrated the value of established library abstractions: the library handles model downloading, preprocessing, inference, and multiple backend support transparently. The numpy.float32 serialisation issue it introduced was an instructive reminder that integrating libraries from different ecosystems requires careful attention to data type compatibility at interface boundaries.

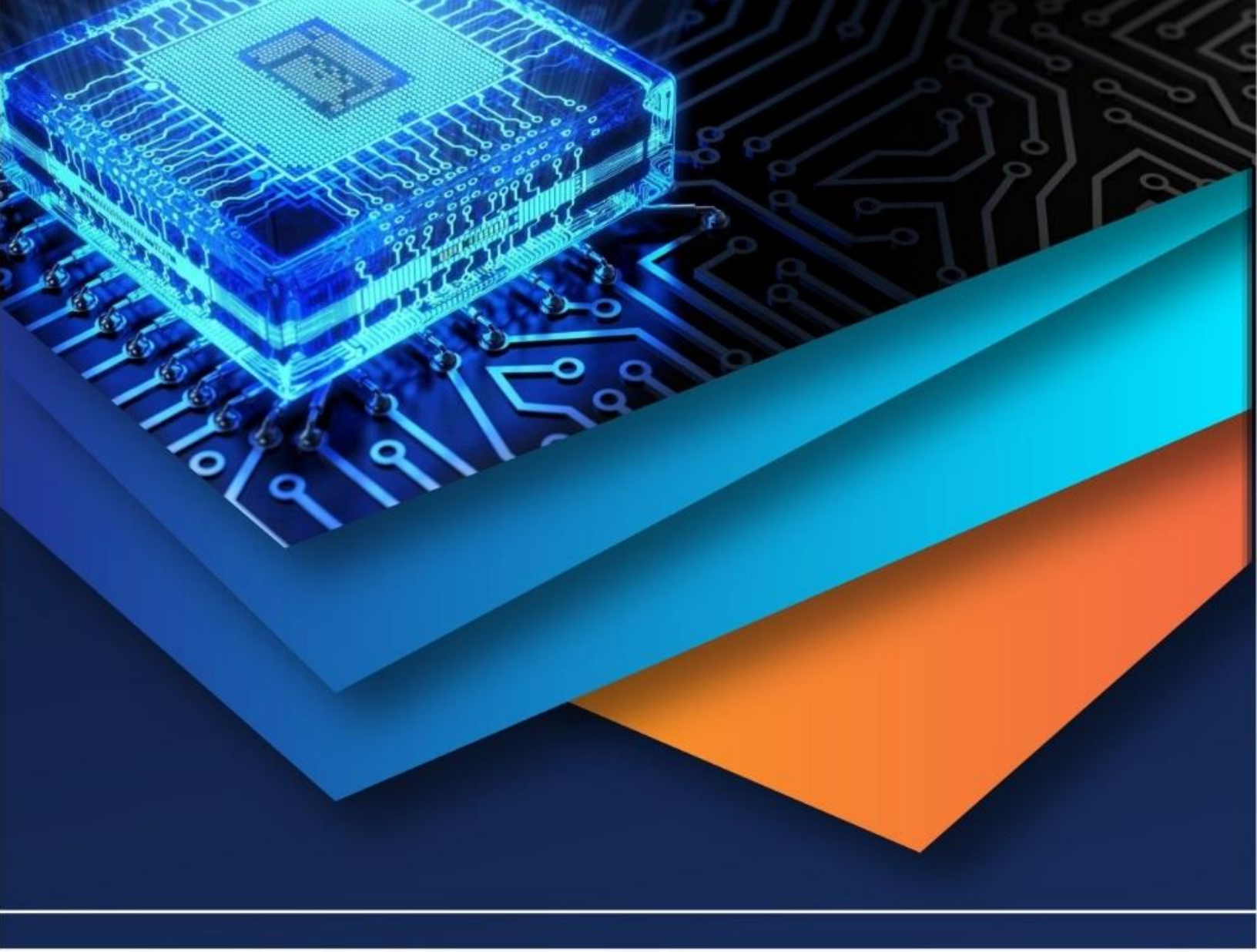
Across all tested scenarios, the system delivered a detection rate of 1-2 detections per second on CPU hardware with smooth 30fps video, greater than 90% gender classification accuracy on frontal faces in adequate lighting, and age predictions that fellwithin the displayed ±4-year range for the majority of test subjects. Face recognition operated near perfectly for registered individuals photographed under similar conditions to their registration image.

VII. ACKNOWLEDGMENT

I thank my project guide Dr. Aravind B N, and institution for their guidance and support. We also acknowledge the use of open-source tools and publicly available datasets that enabled the successful completion of this work.

REFERENCES

[1] Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. Proceedings of the 2001 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1, 511-518.
[2] Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. CVPR 2005, 1, 886-893.
[3] King, D. E. (2009). Dlib-ml: A machine learning toolkit. Journal of Machine Learning Research, 10, 1755-1758.
[4] Taigman, Y., Yang, M., Ranzato, M. A., & Wolf, L. (2014). DeepFace: Closing the gap to human-level performance in face verification. CVPR 2014, 1701-1708.
[5] Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. CVPR 2015, 815-823.
[6] Levi, G., & Hassner, T. (2015). Age and gender classification using convolutional neural networks. CVPR Workshops 2015, 34-42.
[7] Rothe, R., Timofte, R., & Van Gool, L. (2015). DEX: Deep Expectation of apparent age from a single image. ICCV Workshops 2015.
[8] Zhang, Z., Song, Y., & Qi, H. (2017). Age progression/regression by conditional adversarial autoencoder. CVPR 2017, 5810-5818.
[9] Deng, J., Guo, J., Xue, N., & Zafeiriou, S. (2019). ArcFace: Additive angular margin loss for deep face recognition. CVPR 2019, 4690-4699.
[10] Ekman, P., & Friesen, W. V. (1978). Facial Action Coding System: A technique for the measurement of facial movement. Consulting Psychologists Press.
[11] Goodfellow, I. J., et al. (2013). Challenges in representation learning: A report on three machine learning contests. Neural Networks, 64, 59-63.
[12] Serengil, S. I., & Ozpinar, A. (2021). HyperExtended LightFace: A facial attribute analysis framework. 2021 International Conference on Engineering and Emerging Technologies (ICEET).
[13] Abadi, M., et al. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available at tensorflow.org. https://www.tensorflow.org
[14] Chollet, F. (2015). Keras. GitHub Repository. https://github.com/keras-team/keras
[15] Bradski, G. (2000). The OpenCV library. Dr. Dobb's Journal of Software Tools. https://opencv.org
[16] Geitgey, A. (2018). face_recognition: The world's simplest facial recognition API for Python and the command line. https://github.com/ageitgey/face_recognition
[17] UTKFace Dataset. (2017). Large scale face dataset. AICIP Research Group, University of Tennessee. https://susanqq.github.io/UTKFace/



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY