International Journal of Scientific Research and Engineering Development-– Volume 9 Issue 2, Mar-Apr 2026
Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

AI Missing Person Identification System

Shubham.R.Jadhav, Sudarshan.M.Jadhav, Affan.A.Makandar, Swarup.S.Bodake, Miss.Soniya.R.Ghatage
Dr. Bapuji Salunkhe Institute of Engineering and Technology, Kolhapur, Department of Artificial Intelligence and Machine Learning
meshubu07@gmail.com, sudarshanjadhav7272@gmail.com, affanayubmakandar@gmail.com, swarupbodake07@gmail.com, soniyarg30@gmail.com

Abstract:
Missing persons' investigations are a serious problem for both law enforcement and society as a whole. Current methods for identifying missing persons rely almost entirely upon manual investigations, paper notices and by human observation of surveillance video. These traditional approaches are labor-intensive and do not allow for efficient processing. In addition to surveillance camera's becoming more common and digital evidence growing in volume, there is a need for an automated and smart approach to improve the rate at which missing persons can be identified and reduce errors.
The objective of this study was to develop an Artificial Intelligence (AI) Based Missing Person Identification System that uses Deep Learning and Facial Recognition Technology to automatically locate and identify people from images and/or live video feeds. The system will utilize a combination of computer vision algorithms for detecting faces and feature extraction using deep learning models. Once facial features have been extracted and converted into facial embeddings, they will then be compared against a centralized missing persons database using similar matching techniques to find possible matches.
Experiments were conducted to demonstrate the capability of the proposed system to provide accurate and timely results in real-time. The experiments also demonstrated that the proposed system provided improved efficiency for identifying missing persons when compared to manual investigative efforts. In addition to improving investigative efficiency, the system will assist in providing law enforcement agencies with automated alert capabilities, thereby allowing them to expedite search efforts. The proposed system has great potential to enhance missing person recovery rates, public safety and ultimately save lives.

Keywords: Artificial Intelligence, Deep Learning, Facial Recognition, DeepFace, Convolutional Neural Networks (CNN), Face Embedding, Cosine Similarity, Real-Time Surveillance, Computer Vision, Video Processin

1.Introduction:

The growing problem of missing persons and identity verification has become a real concern across the world. Every year, thousands of people go missing for all sorts of reasons—abduction, human trafficking, mental health issues, accidents, or even natural disasters. Quick identification is crucial. It keeps people safe and boosts the chance of finding those who are lost. Traditional identification methods just aren't keeping up. They're mostly manual, take too long, and rely almost entirely on people.

Usually, the process means printing flyers, scanning hours of CCTV footage by hand, and eyeballing photo comparisons. None of that works well when you're facing huge surveillance networks and mountains of video data. On top of that, humans get tired. Mistakes happen. That slows everything down—sometimes when time matters most.

But new advances in Artificial Intelligence (AI), Computer Vision, and Deep Learning are changing the game. Now, automated facial recognition systems can spot and identify people with remarkable accuracy. Deep learning models—especially Convolutional Neural Networks (CNNs)—can pull out unique features from faces and turn them into detailed digital representations for precise matching. Building on this technology, this research introduces a DeepFace-based Face Detection and Recognition System. It can automatically pick out and identify a particular

person from images and live video streams. The idea is to streamline the process, lighten the manual workload, and provide real-time identification that helps security and surveillance do their job—better and faster.

2.Literature Survey:

1. Facial recognition's been around for years as a hot topic in Computer Vision and Artificial Intelligence. At first, researchers used traditional machine learning and crafted features by hand—think Eigenfaces and Local Binary Patterns (LBP). Those early methods had decent accuracy, but struggled when faces had different lighting, expressions, or poses.
2. Deep Learning changed the game. Once Convolutional Neural Networks (CNNs) came onto the scene, facial recognition systems got a real boost. CNNs learn layered facial features straight from big datasets, giving results that are both more accurate and less prone to errors. Models like VGG-Face and FaceNet brought embedding-based recognition—essentially, each face gets translated into a number vector. With those embeddings, comparing faces becomes straightforward: just measure the distance between vectors using something like Euclidean distance or Cosine similarity.
3. DeepFace took things a step further, making state-of-the-art facial recognition easier to implement. It wraps several deep learning models into one interface and handles detection, alignment, feature extraction, and verification for real-time use. Studies show DeepFace-powered systems work well for surveillance, access control, and security monitoring. Still, most current setups don't come with handy web interfaces, job processing in the background, or automated alerts. The system proposed here improves on earlier work by blending deep learning recognition with real-time video processing, deploying the app through Flask, and adding email notifications—bringing these tools closer to practical use for finding missing persons.

3. Problem Identification:

Traditional methods for finding missing persons come with plenty of practical hurdles. Law enforcement and security teams mostly depend on manually comparing photos, distributing printed notices, and constantly watching CCTV footage. All that eats up time, demands a lot of effort, and hinges heavily on human attention. That opens the door to delays and mistakes.

Now, surveillance systems have exploded across airports, train stations, city streets—you name it. So the amount of video data has shot up. Trying to analyze all this footage by hand isn't just inefficient; it's nearly impossible. On top of that, the old systems don't have automation, real-time alerts, or database tools that can scale up. This makes them almost useless when speed really matters.

So there's an obvious need for a smarter, automated solution that can quickly spot and identify people from images and live video. That system has to be accurate, scalable, and able to work in real-time, all while slashing the manual workload. Tackling these challenges is where the proposed AI-based Face Detection and Recognition System comes in.

4. Methodology:

The Face Detection and Recognition System uses a clear, step-by-step approach to identify a specific person from images and video streams through deep learning. The main steps are: data preparation, face detection, feature extraction, similarity comparison, and generating results.

4.1 Data Collection

The system needs a reference image of the person you want to identify, plus one or more target images or video files for actual verification. It stores the reference image, then processes it to extract unique facial features. All uploaded files go into a secure folder in the system for processing.

4.2 Image Preprocessing

Before recognition starts, the system preprocesses each image and video frame:

- Resizes images to standard dimensions
- Converts images to RGB format
- Normalizes pixel values
- Removes noise if needed

Preprocessing helps boost detection accuracy and keeps the input format consistent for the deep learning model.

4.3 Face Detection

OpenCV's face detection backend handles face detection in both images and video frames. For every target image or frame:

- It detects all visible faces

- Extracts just the facial regions
- Generates bounding box coordinates

At this step, only the facial regions go forward to recognition.

4.4 Feature Extraction (Face Embedding Generation)

The system uses the DeepFace library for extracting facial embeddings from both the reference image and detected faces in the target. The deep learning model—by default, VGG-Face—turns each face into a high-dimensional vector that's unique for each person.

These embeddings make it possible to compare faces using math, not just pixels.

4.5 Face Verification and Similarity Matching

The system compares the extracted embeddings using similarity metrics like Cosine Similarity. Here's the formula:

Similarity = (A · B) / (||A|| × ||B||), where A is the reference embedding, and B is the target embedding.

If the score passes a set threshold, the system confirms there's a match.

4.6 Video Frame Processing

For videos:

- The system processes the video frame by frame.
- To keep things fast, it checks every 5th frame rather than all frames.
- It records which frame gets the highest confidence score.
- The best match frame gets saved and displayed.

This keeps the system both quick and accurate.

4.7 Result Generation

When there's a match:

- A green box appears around the matched face
- The system shows the confidence score
- It saves the result image
- Optionally, it sends the user an email notification

If there's no match, the system says detection failed.

This methodology brings accurate recognition, strong performance, real-time capabilities, and real-world usefulness, especially for surveillance tasks.

5. Implementation:

We built the Face Detection and Recognition System using Python and deep learning libraries to deliver accurate, real-time identification from images and video streams. The system includes three main components: the face recognition engine, the web application backend, and the user interface.

5.1 Face Recognition Engine

We handle core tasks in the face_detection.py module. This module:

- Loads the reference image
- Generates facial embeddings
- Detects faces in target images or video frames
- Compares embeddings to check for similarity
- Draws bounding boxes and confidence scores

We use DeepFace for high-dimensional facial embeddings with a pre-trained deep learning model (VGG-Face by default). We rely on OpenCV to read images and videos, detect faces, and display results.

5.2 Reference Image Encoding

At initialization, the system loads the reference image and turns it into a numerical embedding vector using DeepFace's represent() function. This embedding captures the individual's facial features and gets stored for later comparison.

5.3 Target Image Processing

For image verification:

1. The target image loads with OpenCV.
2. The system detects faces in the image.
3. Each face gets converted into an embedding.
4. We compare each embedding with the reference embedding.
5. If the similarity passes the threshold, the system confirms a match.

Matched faces get green bounding boxes. Unmatched faces get red ones.

5.4 Video Processing

For video verification:

- The system processes each video file frame by frame.
- To boost efficiency, we only analyze every fifth frame.

• Similarity scores are calculated for each detected face.
• The frame with the highest confidence match is saved as the best result.

This streamlines processing and keeps accuracy at a good level.

5.5 Web Application Integration

A Flask-based web app (app.py) lets users interact through their browsers. The backend manages:
• File uploads
• Background processing with Python threading
• Job status tracking
• Result generation

Each request gets a unique job_id. The frontend polls the API endpoints to check job status and update the progress bar.

5.6 Email Notification System

We also added an optional email alert using Python's SMTP library. If a match turns up and an email's provided:
• The best result image attaches
• The system sends a notification email to the user

This makes the system useful for surveillance and security.

5.7 System Environment

• Programming Language: Python
• Framework: Flask
• Libraries: DeepFace, OpenCV, NumPy, Matplotlib
• Concurrency: Python Threading
• Operating System: Windows/Linux
• Hardware: CPU-based processing (GPU recommended for faster video analysis)

5.8 Implementation Outcome

The system:
• Detects faces in images and videos
• Generates facial embeddings
• Matches faces by similarity
• Shows visual results in real time
• Supports web-based interaction and notification

It runs efficiently in controlled environments and can be expanded for large-scale surveillance **use.**

6. CONCLUSION:

This research introduces an AI-powered Face Detection and Recognition System that identifies specific individuals in photos and video feeds using deep learning methods. Using the DeepFace framework along with computer vision algorithms, the system automates facial feature extraction and matching, which cuts down on the need for manual monitoring. With real-time processing of both images and video, the solution becomes practical for real-world surveillance uses.

Testing shows that facial recognition based on embeddings achieves reliable accuracy when lighting and image quality are controlled. The system quickly processes still images and video frames, finds possible matches, and displays visual results alongside confidence scores. A web-based interface and email notification feature make the system easier to use and more accessible.

Of course, things like lighting, resolution, or a partially hidden face can affect performance. Still, this system lays a strong foundation for intelligent identification tools. With more optimization and wider deployment, it has real potential to improve security monitoring, help find missing people, and advance automated surveillance.

7. Future Scope:

The Face Detection and Recognition System we've designed forms a solid base for automated identification, but there's room to take it much further in terms of performance, scalability, and practical use. Down the line, you can link the system to a centralized cloud database so it runs across multiple sites at once. If you connect it with national or state missing person databases, you get instant cross-checking and much faster identification. Running the system on cloud platforms with GPU acceleration also boosts speed—this really matters when you're processing high-res surveillance footage.

You can push accuracy even further by adding advanced deep learning models, especially new face embedding architectures. These hold up better when conditions get tough—think low light, blocked faces (like with masks or caps), or unpredictable camera angles. Syncing up multiple cameras and plugging directly into live CCTV streams would seriously step up real-time monitoring. On top of that, there's potential for building a mobile app for law enforcement, letting

officers verify identities on the spot with just a smartphone camera.

Looking ahead, it's worth focusing on privacy-preserving tech and ethical AI frameworks to keep data secure and ensure everything runs responsibly. With all these upgrades, the system can grow into a smart, all-in-one surveillance and identification tool—ready for smart cities and environments where security on a large scale really matters.

References:

[1] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep Face Recognition," Proceedings of the British Machine Vision Conference (BMVC), 2015.

[2] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A Unified Embedding for Face Recognition and Clustering," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[3] S. Iandola et al., "DeepFace: Closing the Gap to Human-Level Performance in Face Verification," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

[4] G. Bradski, "The OpenCV Library," Dr. Dobb's Journal of Software Tools, 2000.

[5] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a Convolutional Neural Network," International Conference on Engineering and Technology (ICET), 2017.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," Advances in Neural Information Processing Systems (NeurIPS), 2012.

[7] S. Ranjan, C. D. Castillo, and R. Chellappa, "L2-Constrained Softmax Loss for Discriminative Face Verification," arXiv preprint arXiv:1703.09507, 2017.

