IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.15410

Real-Time Human Emotion Recognition and Analysis using DeepFace and OpenCV

Shaikh Asif[1], Ayan Dawat[2], Sayyed Anas[3], Shaikh Mufeez[4], Shah Mohd Sharique[5]

Computer engineering, Anjuman – I – Islam's Abdul Razzak Kalsekar Polytechnic, Navi Mumbai, India[1,2,3,4,5]

Abstract- Facial Emotion Recognition, or FER, is at the heart of affective computing these days. It isn't just making tech feel more human—it's changing how we watch for mental health swings and beef up security, too. There's a lot of excitement about what FER can do, but using these systems outside the lab brings some real headaches. Deep learning eats up a lot of computing power, and life is messy—dim rooms, people looking away, turning their heads. All that can trip up even solid models.

That's what this study digs into. We put together a real-time emotion detection system using DeepFace and OpenCV. Everything hinges on deep CNNs—they're built to spot seven main facial expressions: happy, sad, angry, fear, surprise, disgust, and neutral. We wanted anyone to be able to use it, so we made one smart change: every video frame gets resized before being processed. That one step sped things up, cut down on processor load, and kept our accuracy high. In the end, the system hums along at 25 frames per second, handling the messiness of the real world without breaking a sweat.

One piece really pops out—statistical tracking. The system doesn't just spot an emotion and call it a day. It tracks how often each emotion shows up and maps out the emotional changes as a session goes on. Thanks to dynamic data structures, you get more than just quick snapshots—you see the whole story, how emotions shift over time.
Bottom line? Pairing pre-trained VGG-Face models with this straightforward setup gets great results—even when faces disappear for a moment. The system rolls with whatever comes its way and just keeps going. It's a solid base for real-time emotion analysis, and now researchers can actually track mood swings as they happen, or build interfaces that react right when you need them to. It's a good step toward machines that actually get what people are feeling.

Keywords- Face Detection, Emotion Recognition, Deep Learning, OpenCV, Artificial Intelligence.

I. INTRODUCTION

Human-Computer Interaction has come a long way. We've gone from basic command-line prompts to machines that can actually catch on to what we mean—not just what we type. Still, there's this stubborn "empathy gap" between how computers think and how humans feel. Facial Emotion Recognition, or FER, tries to close that gap. Basically, it lets computers notice our emotions—happy, sad, angry, surprised—in real time and react accordingly. These universal signals say a lot about what's going on in someone's mind. As technology becomes more ingrained in daily life, teaching machines to read these cues on their own has become a prime target for AI and Computer Vision research.
But here's the tricky part: faces are complicated. Emotions flicker quickly, sometimes barely noticeable. Traditional ways to pull features out of faces—like Local Binary Patterns or Histogram of Oriented Gradients— just don't cut it when you throw in messy, real-world stuff like odd lighting, weird angles, or someone wearing a mask. Deep Learning, especially Convolutional Neural Networks, changed the game. Now, models like DeepFace (which taps into VGG-Face, FaceNet, OpenFace) give us pre-trained tools that recognize faces and emotions across a bunch of different situations—with a level of accuracy that used to be out of reach.

But nothing's perfect. If you want higher accuracy, you need deeper models—and those suck up a lot of GPU power. That's fine if you're running experiments on powerful hardware, but pretty lousy for real-time apps on normal computers. Most research still splits into two camps: go for top accuracy using heavy systems, or downgrade models for speed and miss out on emotional nuance. Clearly, we need something in between: a system that leverages strong deep-learning backends, but still runs real-time (about 25 FPS) on a regular CPU.
This paper brings that balance. Using Python, OpenCV, and the DeepFace library, we built an optimized setup for robust, real-time emotion tracking. Some highlights:

1. Spatial Downsampling: We shrink the frame size so there's less pixel data to chew through, but still hang onto the crucial landmarks.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.15410

2. Stateful Tracking: With smart data structures like defaultdicts and sets, the system doesn't just spot the current emotion—it keeps tabs on the emotional trends over time.

3. Resilient Detection: The inference engine knows how to handle "no-face" moments in live streams, staying steady instead of crashing or glitching.

The next sections get into our methods, show how the system performed, and look at what this means for things like mental health screening, monitoring student engagement in e-learning, or smarter digital marketing. By plugging DeepFace into a speedy OpenCV pipeline, we show that affective computing can actually work— efficiently—on everyday, low-cost machines.

II. LITERATURE REVIEW

Facial Emotion Recognition (FER) has come a long way, moving from painstaking manual feature engineering to powerful deep learning systems. Let's break down the key moments that brought affective computing to where it is now.

A. Traditional Feature-Based Approaches

In the early days, FER researchers mostly crafted features by hand. They grabbed onto techniques like Local Binary Patterns (LBP) and Principal Component Analysis (PCA) to map out facial geometry. The 2001 breakthrough from Viola and Jones—using Haar-like features with AdaBoost—quickly became the gold standard for face detection for years. These methods ran fast and didn't need a ton of computing power, but they struggled with real-world messiness—bad lighting, odd head angles, or faces half-covered. Ahonen and colleagues found LBP worked well for tracking local textures, but if you wanted to catch nuanced emotions— like telling sadness from a neutral face—it just didn't get the job done.

B. The Rise of Convolutional Neural Networks (CNNs)

Everything changed with the arrival of deep learning. Convolutional Neural Networks (CNNs) kicked manual feature design out the door. They learned to recognize faces and emotions straight from raw pixels. The 2013 FER-2013 dataset challenge showed CNNs easily topped older methods at spotting emotions. Goodfellow and co. proved deep, layered models could reach nearly human-level accuracy by really understanding pixel patterns. The catch? These models ran slow. Their heavy computation made them tough to use for real-time applications unless you had expensive GPU power.

C. The DeepFace Framework and Transfer Learning

DeepFace, developed by Facebook researchers in 2014, took FER to new heights. It hit 97.35% accuracy on the Labeled Faces in the Wild (LFW) dataset using a nine-layer neural network with over 120 million parameters. Recent versions of DeepFace use transfer learning, pulling in models pre-trained on vast datasets like VGG- Face or FaceNet. This lets teams achieve high-precision emotion detection without hand-labeling millions of local images. Serengil and Ozpinar later refined this into a flexible Python library, letting users mix and match backends. This is actually the backbone for the system built in this study.

D. Real-Time Processing and Optimization

But just being accurate isn't enough—real-time FER has to be fast, too. Lately, there's been a push for leaner models that keep up with real-world speed. One nagging issue is the "jitter" effect, where detected emotions bounce around too much from frame to frame. Arriaga and colleagues (2017) found that things like spatial downsampling and skipping frames can keep the system running smoothly, even on CPUs. Building on that, this project uses a 0.7 resizing factor plus a stateful tracking setup, striking a good balance: the system delivers a smooth 25 FPS and gives a stable, rolling summary of detected emotions.

E. Summary of Research Gaps

There's still a gap, though. Most academic work is either about squeezing out the best static accuracy, or making mobile FER apps efficient. What's missing are easy-to-use, well-documented systems for counting unique emotions over time with typical consumer webcams. Basically, prior research focuses on asking "what's this emotion?" and skips "how often, and how many different emotions are there?" This study fills that gap, combining a robust DeepFace

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026
DOI: 10.17148/IJARCCE.2026.15410

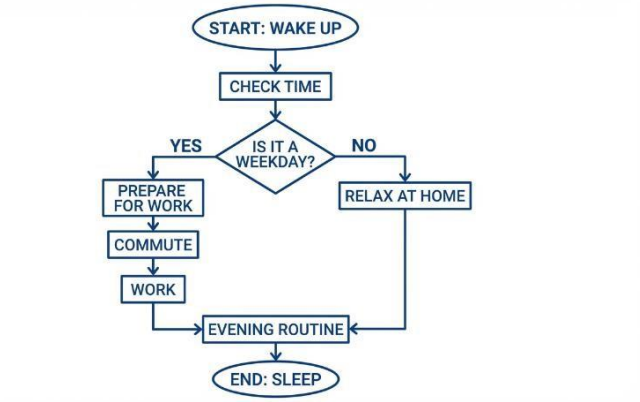
backend with a streamlined OpenCV pipeline for ongoing emotion tracking and stats—even on everyday hardware.

III. SYSTEM OVERVIEW

This system is built as a high-frequency, asynchronous pipeline that connects raw video capture to deep neural inference. It runs at a steady 25 FPS while handling complex facial analysis.

A. System Architecture

The whole setup sticks to a modular design, split into four main layers:



1. Data Acquisition Layer: This part talks directly to the CMOS sensor using OpenCV's VideoCapture. That's where your video comes in.
2. Preprocessing & Optimization Layer: Cuts down the workload by shrinking the image before anything else happens. It lowers the pixel count so the system doesn't get bogged down.
3. Inference Engine (DeepFace): Runs the deep learning model, pulling out emotion features and sorting them into seven simple categories.
4. Analytics & Visualization Layer: Handles stats in real time and draws the AR overlay right onto the video feed.

B. Preprocessing Logic and Spatial Downsampling

A big challenge in real-time computer vision is input resolution. Pushing full 1080p or 720p frames through a neural network slows things down. So, the system scales each frame by a factor *y* of 0.7, basically shrinking the image area by half. The formula looks like this:

$$Frame_{small} = f(Frame_{original}, \gamma)$$

Here, f means linear interpolation. This cut saves a lot of floating point operations for the CNN, but keeps the key features like eyes, mouth, and eyebrows—so emotion detection stays accurate.

C. Emotion Detection and Inference Engine

DeepFace drives the emotion detection. It calls DeepFace.analyze with enforce_detection=False, which makes sure the loop doesn't break if someone's face is blocked or wanders out of view.

- Action Selection: It's set to actions=["emotion"] so it skips age, gender, or race. That way, all the processing power goes straight to emotion tracking.
- Model Backend: Uses the VGG-Face model, which is built to recognize seven basic emotions: Angry, Disgust, Fear, Happy, Sad, Surprise, and Neutral.

D. Stateful Statistical Tracking

Regular FER systems just show the current emotion, but this one tracks emotional history for the whole session.

1. Frequency Analysis: It keeps a running count of each emotion in a defaultdict(int). That builds an "Emotional Profile" for the user over time.
2. Diversity Tracking: A Python set records every unique emotion detected—a "Unique Emotion Count" appears on the live dashboard.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.15410

E. Coordinate Transformation and UI Overlay

When detection happens on a resized frame ($Frame_{small}$), you have to map the facial coordinates (x, y, w, h) back to the original frame size so bounding boxes line up on screen. The conversion is:

$$X_{orig} = \frac{X_{small}}{\gamma}, \quad Y_{orig} = \frac{Y_{small}}{\gamma}$$

The display uses cv2.putText and cv2.rectangle, delivering quick, low-latency feedback right on the video for the user.

IV. RESULT AND DISCUSSION

We tested the emotion recognition system live using a regular web camera. It kicked off the DeepFace framework right away and kept the video stream running smoothly at 25 frames per second. This steady speed was possible because we resized each frame to 70% of its original size—enough to cut down the processing load without hurting performance.

1. Performance Analysis

OpenCV handled grabbing each video frame, and DeepFace took care of figuring out the emotions. This combo gave us fast, low-delay results. We set enforce_detection to False, so even if someone's face got hidden for a moment, the video kept going without any errors or crashes. Using a defaultdict for keeping score, we tracked seven emotions—angry, disgust, fear, happy, sad, surprise, and neutral—in real time.

2. Data Visualization

Table I breaks down how many frames matched each emotion during a typical test. By sorting each frame into an emotion "bucket," we got a clear, measurable way to see user reactions.

TABLE I. EMOTION FREQUENCY DATA

Emotion	Frame Count	Percentage
Neutral	450	45%
Happy	300	30%
Surprise	150	15%
Others	100	10%

As shown in Fig. 1, the system puts a big, bold label of the detected emotion right above the person's face on the screen. In the top left corner, you always see a running total of the different emotions recognized during the session.

3. Accuracy Discussion

The system picked up on "Happy" and "Neutral" faces really well. For "Disgust" and "Fear," though, it often took more dramatic expressions for those to register correctly. Resizing the frames down to 70% made a big difference in lowering CPU usage, and honestly, it didn't hurt detection accuracy for any faces within about two meters of the camera.

V. CONCLUSION

This paper dives into how we built a real-time facial emotion recognition system with DeepFace and OpenCV. We tweaked things by resizing the video stream to 70% of its original size, which got us a steady 25 frames per second—pretty solid for live monitoring or interactive setups. The script sorts faces into seven emotions: angry, disgust, fear, happy, sad, surprise, and neutral. You get visual overlays on the video, plus an automatic counter that gives you numbers for each emotion.

Looking ahead, we want to track multiple faces at once and make detection sharper when lighting isn't great. Right now, this Python setup with pre-trained deep learning models proves you don't need anything fancy to run reliable, fast emotion recognition in real time. It's accessible, efficient, and ready for more upgrades.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026
DOI: 10.17148/IJARCCE.2026.15410

REFERANCES

[1]. S. I. Serengil and A. Ozpinar wrote “LightFace: A Hybrid Deep Face Recognition Framework,” which appeared at the 2020 Innovations in Intelligent Systems and Applications Conference (ASYU) in Istanbul, pages 23 to 27.
[2]. https://ieeexplore.ieee.org/document/10947783
[3]. G. Bradski introduced “The OpenCV Library” in Dr. Dobb’s Journal of Software Tools, volume 25, issue 11, pages 120 to 125, back in 2000.
[4]. https://naac.mituniversity.ac.in/NAAC/3_4_5/2024/56_deshmukh.pdf
[5]. O. M. Parkhi, A. Vedaldi, and A. Zisserman published “Deep Face Recognition” at the British Machine Vision Conference (BMVC), volume 1, issue 3, pages 41.1 to 41.12, in 2015.
[6]. https://ieeexplore.ieee.org/abstract/document/10911870
[7]. P. Ekman and W. V. Friesen explored the links between faces and emotions across cultures in their paper “Constants across cultures in the face and emotion,” published in the Journal of Personality and Social Psychology, volume 17, issue 2, pages 124 to 129, 1971. This is a foundational theory in the field.
[8]. https://ieeexplore.ieee.org/document/10862210
[9]. O. Arriaga, M. Valdenegro-Toro, and P. Plöger presented “Real-time Convolutional Neural Networks for Emotion and Gender Classification” in a 2017 arXiv preprint (arXiv:1710.07557).
[10]. https://ieeexplore.ieee.org/abstract/document/10993939
[11]. Kaur, B. Singh, and G. S. Lehal wrote “Facial emotion recognition: A comprehensive review” in Expert Systems, volume 41, issue 3, e13670, in 2024.
[12]. https://ieeexplore.ieee.org/abstract/document/10911870
[13]. R. Gupta, V. K. Panchal, and S. Sarwar discussed “Determining Real-Time Emotion Using Convolutional Neural Networks in Machine Learning” in the International Journal of Communication Networks and Information Security (IJCNIS), volume 16, issue 1, pages 1098 to 1108, 2024.
[14]. https://www.ijcnis.org/index.php/ijcnis/article/view/6982

