Deepfake detection from your pulse: how it works and how to build it
Hari (@harixganesh)

The quick version

Each heartbeat pushes blood under your skin, which absorbs light differently, so your skin reflects slightly less green light. A camera records it. Measuring a pulse with light is **photoplethysmography (PPG)**. Doing it from ordinary video of a face, without touching anyone, is **remote PPG (rPPG)**.

In 2020, Ciftci, Demir and Yin (Binghamton University and Intel Labs) built a detector on this called FakeCatcher. Intel made it a product in 2022.

How it works

Take a 10-second clip at 30 frames a second.

1. Find the skin. A face-landmark model marks points on the face, and you cut out skin patches (forehead, cheeks). Each patch is a **region of interest (ROI)**.

2. One number per frame. Every pixel is three numbers: how red, green and blue it is. Average the green of a patch in each frame (the simple version uses green; CHROM and POS, which combine all three colours, are better). That is 10 x 30 = 300 numbers, and it is all the computer is given (made-up values):

Frame	Average green
1	142.3
2	142.5
3	142.9
...	...
300	142.4

3. Separate the pulse from the junk. The pulse is the **signal**. Lighting, movement and compression are the **noise**, and it's usually bigger. A **band-pass filter** keeps only the frequencies where a heartbeat can live. Beats a minute = hertz x 60, so a band of 0.7 to 4 Hz is 42 to 240 beats a minute.

4. Read the pulse. A **Fourier transform** splits a signal into the frequencies inside it; take the strongest one. A pulse of 72 beats a minute is 1.2 Hz: 12 beats in your 10 seconds, one every 25 frames. With 300 samples the frequency steps are 0.1 Hz apart, so 1.2 Hz lands on step 12, and 1.2 x 60 = 72. That is a heart-rate reader. (FakeCatcher keeps 0.7 to 14 Hz, because the leftover "noise" helps it spot fakes.)

5. Compare the patches. On a real face they rise and fall together; on a fake they are less consistent. FakeCatcher feeds 32 face regions to a trained **classifier**, a model that says real or fake. The researchers got 96% on one older test set and about 91% on their own harder one. And good face-swap deepfakes can now copy the pulse from the real video they are made from (2025), so a pulse alone is not proof.

One outside test: in 2023 the BBC tried it on about a dozen clips. It got every fake right bar one, but several times called a real video fake. Pixelated video made the pulse harder to pick up.

How you would make one

- Read the video and find the face.
- Average skin patches, frame by frame.
- Filter the signal and read the beats a minute.
- Check that number against a real pulse.
- Compare the patches and train a classifier.
- Test it on people it has never seen.

Free, a laptop is enough, and you need some Python. About a weekend for the heart rate (my guess), a few more evenings for the detector.

That's the gist. If you actually want to build it, keep going.

If you actually want to build it

1. Video and face. OpenCV (cv2.VideoCapture, cap.read()) pulls out frames. Read the real frame rate with cap.get(cv2.CAP_PROP_FPS), because every beats-a-minute number depends on it. MediaPipe Face Landmarker gives 478 landmarks per face. It works when the points you draw on a frame sit on the face. About an hour.

2. Patches and averages. Forehead and both cheeks. Plot the landmark numbers on one frame to see which sit where, then take each patch's mean red, green and blue per frame. You should get one number per frame per patch. (Start with the forehead, it was strongest in the 2008 paper below.) About an hour.

3. Clean the signal and read the beats. Run scipy.signal.detrend to remove the slow drift. Then scipy.signal.butter(..., btype='bandpass', fs=fs, output='sos') and scipy.signal.sosfiltfilt. Start with 0.7 to 3.0 Hz. When it works, the line should look like a fairly regular wave. Then numpy.fft.rfft and rfftfreq, strongest bin in the band, times 60. The tallest spike should sit near your real pulse, counted at your wrist. About an hour. (Skip CHROM and POS until green works.)

4. Check against a real pulse. Compare with a finger pulse oximeter, or the UBFC-rPPG dataset, which has one recorded in. Score it by mean absolute error: beats a minute off, averaged over many clips. Start with still, well-lit video.

5. Compare the patches. For each window of video, measure how closely the filtered signals of 3 to 6 patches agree (scipy.signal.coherence). A few evenings.

6. Turn that into real or fake. Each window of video becomes a short list of numbers, labelled real or fake from videos you already know. A classifier (logistic regression or an SVM is enough) learns what the numbers look like for each, then every window of a new video votes and the majority wins. About an evening.

7. Split by person. Keep every clip of one person, and every fake made from one real video, on the same side of the train/test line (scikit-learn GroupKFold). (FaceForensics++ has real and fake pairs, but the download needs a form and is about 10 GB.)

How you know it works

Heart rate: mean absolute error against the oximeter. Detector: accuracy on people the model never saw, ideally from a dataset it never saw.

A homemade one will land well below 96%. Above chance on new people is a good result for a learning project.

Where people mess up

- Compressed or moving video. Even light compression degrades the pulse signal a lot, and the FakeCatcher authors' own false alarms came from movement and uneven light.
- Clips that are too short. A 10-second clip only tells beats 6 a minute apart; a 30-second one, 2.
- Testing on the people you trained on. Clips of one person look alike, so a random split flatters your score.
- Skin tone. Plain heart-rate readers get worse on the darkest skin type. Intel says its tests showed no gap but gave no data.
- Quoting your result as 96%. That's one older test set; the same paper got about 91% on harder videos.

Start here

In this order. All free. Times are my guesses.

- yarPPG Deep dive (45 to 60 min). Working Python from video to heart rate. A hobby project, so use it to see it work. No sign-up.
- Verkruysse 2008 (30 min). Pulse from ordinary light and a cheap camera. Free full text.
- The FakeCatcher paper (1 to 1.5 h). Intro, Section 4.4, Section 7. Free PDF.
- "High-quality deepfakes have a heart!" (30 to 45 min). Why a pulse alone no longer proves a video real. Open access.
- UBFC-rPPG (download time). Videos with a real pulse recorded alongside. No form; files are behind a link on the page.
- rPPG-Toolbox (half a day). Run POS or CHROM on UBFC-rPPG; it prints the error. Free.

If you build it, send me what you get!