Face Detection Using Machine Learning

Submitted By: Vedansh Singh Tomar

Course: CSE1021

Date: 23/11/2025

Introduction:

Face detection is a computer vision technique used to identify and locate human faces in images or videos. It serves as a foundation for applications like facial recognition, surveillance, emotion detection, and human–computer interaction. Machine learning algorithms enable efficient and accurate detection by learning patterns from large datasets.

Problem Statement:

Traditional manual monitoring systems struggle to detect and analyze faces accurately. The goal is to develop an automated face detection system using machine learning that can reliably identify human faces in real-time or static images.

Functional Requirements:

- The system must detect human faces in input images.

- The system should draw bounding boxes around detected faces.
- It should process real-time video streams (optional).
- It must support multiple face detection in one frame.

Non-functional Requirements:

- Accuracy: High detection accuracy under different lighting conditions.
- Performance: Real-time or near real-time processing.
- Scalability: Should handle multiple images or video frames.
- Usability: Easy to integrate into applications.

System Architecture:

- Input image/video
- Preprocessing module
- Machine learning-based face detection model (e.g., Haar Cascade, CNN-based detectors)
- Output module to visualize detection results

Code:

Untitled1.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

Share

Connect

```
import imutils
import numpy as np
import cv2
from google.colab.patches import cv2_imshow
from IPython.display import display, Javascript
from google.colab.output import eval_js
from base64 import b64decode
```

```
def take_photo(filename='photo.jpg', quality=0.8):
  js = Javascript('''
    async function takePhoto(quality) {
      const div = document.createElement('div');
      const capture = document.createElement('button');
      capture.textContent = 'Capture';
      div.appendChild(capture);

      const video = document.createElement('video');
      video.style.display = 'block';
      const stream = await navigator.mediaDevices.getUserMedia({video: true});

      document.body.appendChild(div);
      div.appendChild(video);
      video.srcObject = stream;
      await video.play();
```

Untitled1.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

Share

Connect

```
      google.colab.output.setIframeHeight(document.documentElement.scrollHeight, true);

      await new Promise((resolve) => capture.onclick = resolve);

      const canvas = document.createElement('canvas');
      canvas.width = video.videoWidth;
      canvas.height = video.videoHeight;
      canvas.getContext('2d').drawImage(video, 0, 0);
      stream.getVideoTracks()[0].stop();
      div.remove();
      return canvas.toDataURL('image/jpeg', quality);
    }
    ''')
  display(js)
  data = eval_js('takePhoto({})'.format(quality))
  binary = b64decode(data.split(',')[1])
  with open(filename, 'wb') as f:
    f.write(binary)
  return filename
```

```
image_file = take_photo()
```

```
image = cv2.imread(image_file)
image = imutils.resize(image, width=400)
(h, w) = image.shape[:2]
```

Untitled1.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

Share

Connect

```
cv2_imshow(image)
```

```
!wget -N https://raw.githubusercontent.com/opencv/opencv/master/samples/dnn/face_detector/deploy.prototxt
!wget -N https://raw.githubusercontent.com/opencv/opencv_3rdparty/dnn_samples_face_detector_20170830/res10_300x300_ssd_iter_140000.caffem
```

```
print("[INFO] loading model...")
prototxt = 'deploy.prototxt'
model = 'res10_300x300_ssd_iter_140000.caffemodel'
net = cv2.dnn.readNetFromCaffe(prototxt, model)
```

```
image = imutils.resize(image, width=400)
blob = cv2.dnn.blobFromImage(cv2.resize(image, (300, 300)), 1.0, (300, 300), (104.0, 177.0, 123.0))
```

```
print("[INFO] computing object detections...")
net.setInput(blob)
detections = net.forward()
```

```
for i in range(0, detections.shape[2]):
  confidence = detections[0, 0, i, 2]
  if confidence > 0.5:
    box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
    (startX, startY, endX, endY) = box.astype("int")
    text = "{:.2f}%".format(confidence * 100)
    y = startY - 10 if startY - 10 > 10 else startY + 10
```

```
    (startX, startY, endX, endY) = box.astype("int")
    text = "{:.2f}%".format(confidence * 100)
    y = startY - 10 if startY - 10 > 10 else startY + 10
    cv2.rectangle(image, (startX, startY), (endX, endY), (0, 0, 255), 2)
    cv2.putText(image, text, (startX, y),
      cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 0, 255), 2)
```

```
cv2_imshow(image)
```

Workflow:

1. Load image
2. Preprocess
3. Apply ML detection model
4. Output results

Sequence Diagram:

User → System → ML Model → System Output

Component Diagram:

- Image Processor
- Face Detector
- Display Module

Design Decisions & Rationale:

- Python and OpenCV used due to wide library

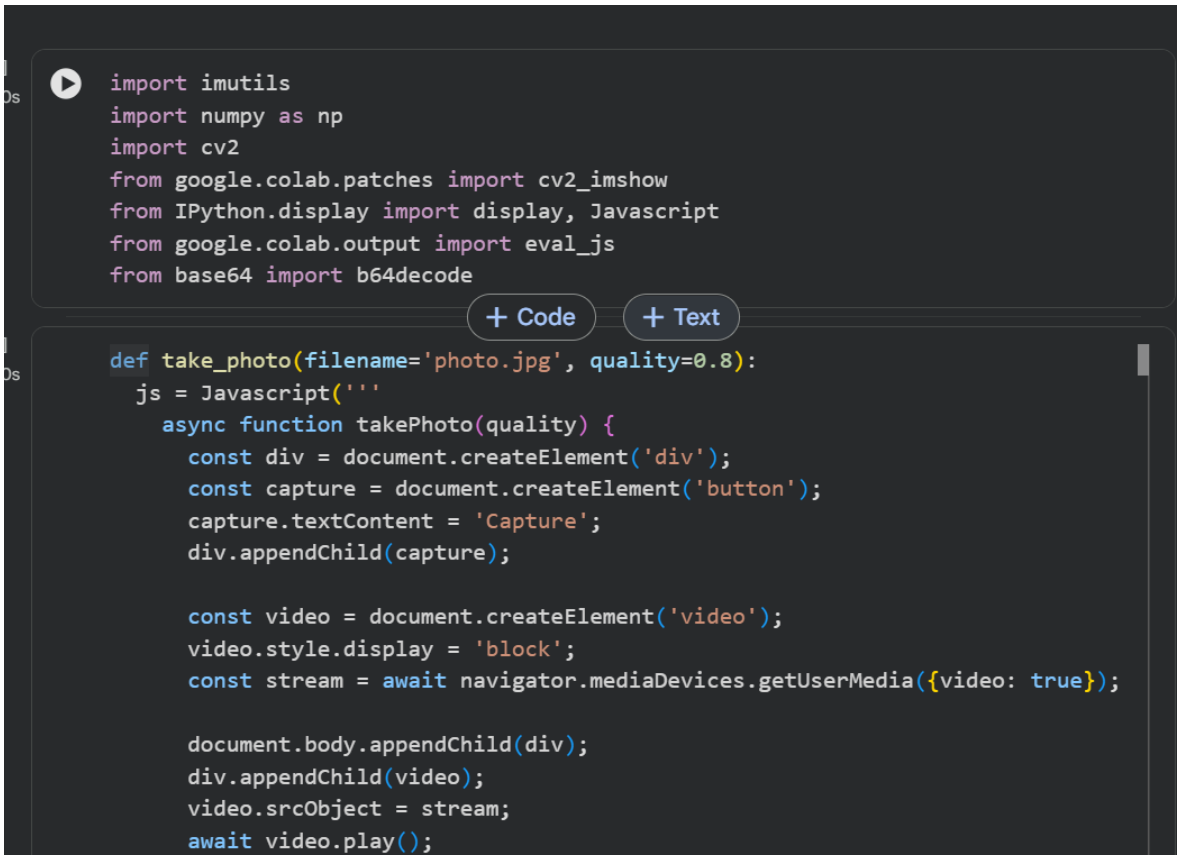
support.

- Modular architecture ensures reusability.

Implementation Details:

- Programming Language: Python
- Libraries: OpenCV, NumPy

Results:



```
import imutils
import numpy as np
import cv2
from google.colab.patches import cv2_imshow
from IPython.display import display, Javascript
from google.colab.output import eval_js
from base64 import b64decode
```

+ Code + Text

```
def take_photo(filename='photo.jpg', quality=0.8):
  js = Javascript('''
    async function takePhoto(quality) {
      const div = document.createElement('div');
      const capture = document.createElement('button');
      capture.textContent = 'Capture';
      div.appendChild(capture);

      const video = document.createElement('video');
      video.style.display = 'block';
      const stream = await navigator.mediaDevices.getUserMedia({video: true});

      document.body.appendChild(div);
      div.appendChild(video);
      video.srcObject = stream;
      await video.play();
```

[4] ✓ 0s

```
        google.colab.output.setIframeHeight(document.documentElement.scrollHeight,

        await new Promise((resolve) => capture.onclick = resolve);

        const canvas = document.createElement('canvas');
        canvas.width = video.videoWidth;
        canvas.height = video.videoHeight;
        canvas.getContext('2d').drawImage(video, 0, 0);
        stream.getVideoTracks()[0].stop();
        div.remove();
        return canvas.toDataURL('image/jpeg', quality);
      }
      ''')
    display(js)
    data = eval_js('takePhoto({})'.format(quality))
    binary = b64decode(data.split(',')[1])
    with open(filename, 'wb') as f:
      f.write(binary)
    return filename
```

[9] ✓ 41s

```
image_file = take_photo()
```

[10]

```
image = cv2.imread(image_file)
```

[10] ✓ 0s

```
image = imutils.resize(image, width=400)
(h, w) = image.shape[:2]
print(w,h)
cv2_imshow(image)
```

```
400 300
```

[11] ✓ 0s

```
!wget -N https://raw.githubusercontent.com/opencv/opencv/master/samples/dnn/face_
!wget -N https://raw.githubusercontent.com/opencv/opencv_3rdparty/dnn_samples_fac
```

```
--2025-11-23 16:15:35--  https://raw.githubusercontent.com/opencv/opencv/master/sa
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.110.133
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.110.13
HTTP request sent, awaiting response... 200 OK
Length: 28104 (27K) [text/plain]
Saving to: ‘deploy.prototxt’

deploy.prototxt     100%[===================>]  27.45K  --.-KB/s    in 0.001s

Last-modified header missing -- time-stamps turned off.
2025-11-23 16:15:35 (18.5 MB/s) - ‘deploy.prototxt’ saved [28104/28104]

--2025-11-23 16:15:35--  https://raw.githubusercontent.com/opencv/opencv_3rdparty/
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.111.133
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.111.13
HTTP request sent, awaiting response... 200 OK
Length: 10666211 (10M) [application/octet-stream]
Saving to: ‘res10_300x300_ssd_iter_140000.caffemodel’

res10_300x300_ssd_i 100%[===================>]  10.17M  --.-KB/s    in 0.07s

Last-modified header missing -- time-stamps turned off.
2025-11-23 16:15:36 (143 MB/s) - ‘res10_300x300_ssd_iter_140000.caffemodel’ saved
```

[12] ✓ 0s

```
print("[INFO] loading model...")
prototxt = 'deploy.prototxt'
```

[12] ✓ 0s

```
model = 'res10_300x300_ssd_iter_140000.caffemodel'
net = cv2.dnn.readNetFromCaffe(prototxt, model)
```

```
[INFO] loading model...
```

[13] ✓ 0s

```
image = imutils.resize(image, width=400)
blob = cv2.dnn.blobFromImage(cv2.resize(image, (300, 300)), 1.0, (300, 300), (104
```

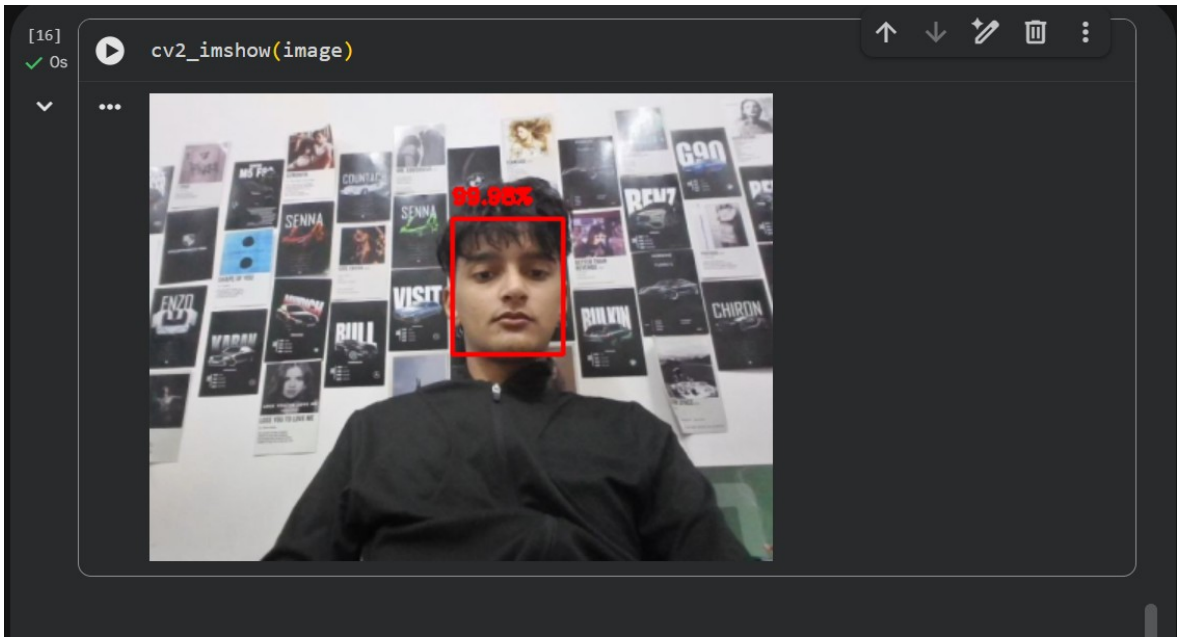
[14] ✓ 0s

```
print("[INFO] computing object detections...")
net.setInput(blob)
detections = net.forward()
```

```
[INFO] computing object detections...
```

[15] ✓ 0s

```
for i in range(0, detections.shape[2]):
  confidence = detections[0, 0, i, 2]
  if confidence > 0.5:
    box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
    (startX, startY, endX, endY) = box.astype("int")
    text = "{:.2f}%".format(confidence * 100)
    y = startY - 10 if startY - 10 > 10 else startY + 10
    cv2.rectangle(image, (startX, startY), (endX, endY), (0, 0, 255), 2)
    cv2.putText(image, text, (startX, y),
      cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 0, 255), 2)
```



Testing Approach:

- Test with multiple images under varying lighting conditions.
- Evaluate detection accuracy and false positives.
- Performance testing for real-time video.

Challenges Faced:

- Low light affecting detection accuracy.
- Handling occluded faces.
- Real-time processing speed.

Learnings & Key Takeaways:

- Understanding computer vision concepts.
- Hands-on experience with OpenCV.
- Improved understanding of ML model performance.

Future Enhancements:

- Use deep learning models (e.g., MTCNN, SSD).
- Add facial recognition.
- Deploy as a web or mobile app.

References::

- OpenCV Documentation
- Machine Learning textbooks
- Research papers on face detection
- https://www.geeksforgeeks.org/machine-learning/face-recognition-using-artificial-intelligence/
- https://amanxai.com/2020/11/17/face-mask-detection-with-machine-learning/
- https://pyimagesearch.com/2020/05/04/covid-19-face-mask-detector-with-opencv-keras-tensorflow-and-deep-learning/
- https://en.wikipedia.org/wiki/Face_detection