Pixels
EVERY PIXEL IS KNOWLEDGE

Real-Time Object Detection with OpenCV

Represented by : Basma Ahmed .

Pixels
EVERY PIXEL IS KNOWLEDGE

The content for today:

1. What Are Pre-trained Models and Neural Networks?
2. Face and Object Detection .
3. Object Tracking and Color-Based Tracking .
4. Masking and Drawing on Video.
5. Using Pretrained AI Models in OpenCV.
6. Real-Time Detection with MobileNet-SSD.
7. Hands-On: Live Object Detection with Webcam.

1. Pre-trained Models and Neural Networks

Pre-trained Model : is a model that was already trained by experts on a huge dataset (like millions of images).

- Instead of building a new model from scratch, we **reuse** it to do tasks like recognizing or detecting objects in images or videos.

What is a Neural Network?

- A **neural network** is a system that learns how to recognize patterns for example, what a cat or a car looks like.

- A special type called a **Convolutional Neural Network (CNN)** is very good at understanding images.

1.2 Why Do We Use Pre-trained Models?

- **Saves time and effort** — no need to collect and train on huge datasets.
- **Gives high accuracy** — these models already know how to detect many objects.
- **Works well with OpenCV** — you can easily load the model and start detecting objects live from your camera.

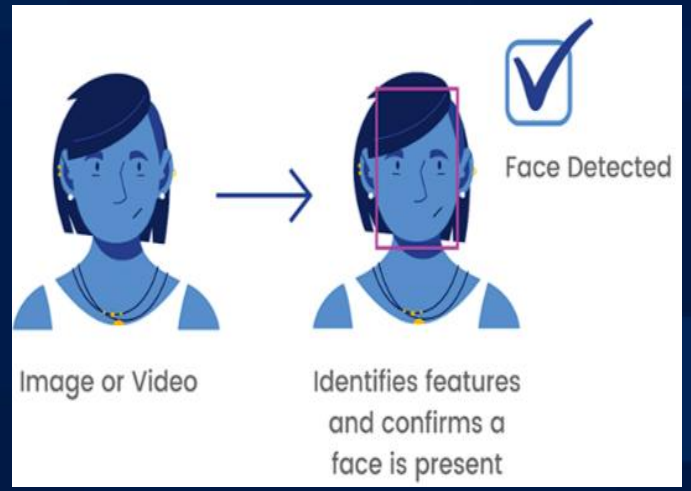
2. Face and Object Detection:

1. What is Face Detection?

- The process of identifying and locating human faces in digital images or video streams.
- **Not** the same as face recognition (which identifies *who* the person is).

2. Why Use OpenCV?

- OpenCV is an open-source computer vision library with built-in tools for **face detection.**
- Supports both traditional **(Haar)** and deep learning-based methods.



2.2 methods used for face detection :

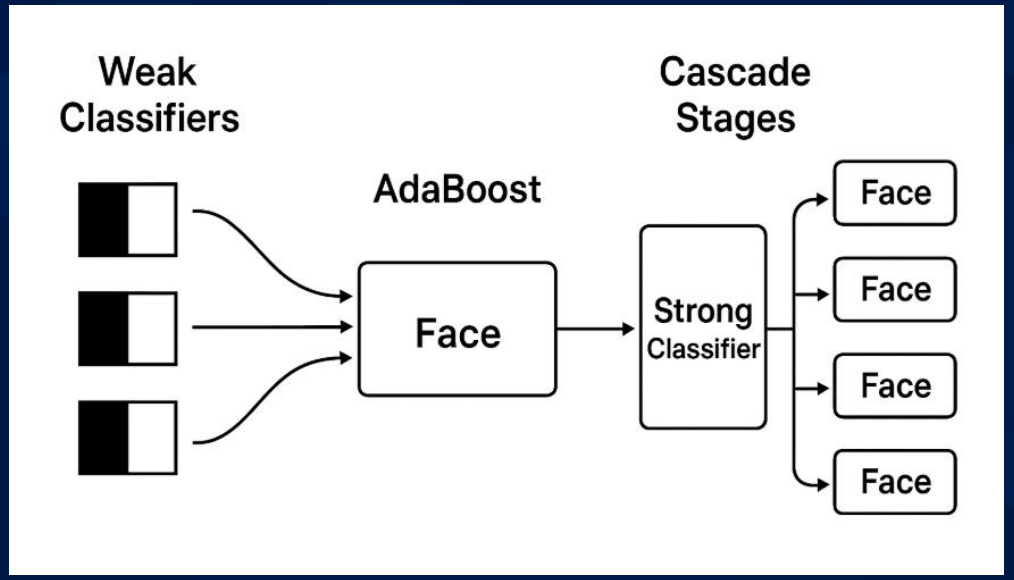
1. Haar Cascade Classifier:

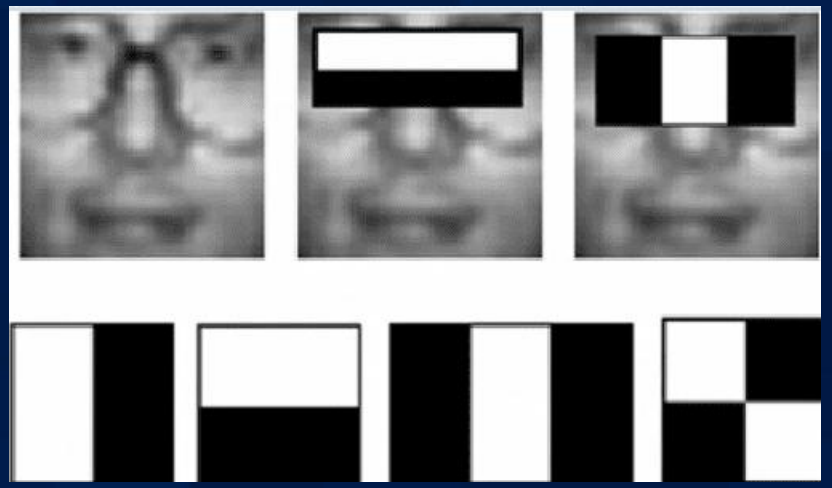
- A **machine learning-based** approach proposed by Viola and Jones.
- Uses **features** like edges, lines, and rectangles.
- Lightweight and **real-time on CPU**.

How it works:

- Uses a series of **Haar-like features** (simple rectangular patterns).
- Applies a **cascade of classifiers** trained via **AdaBoost**.
- Fast because it rejects non-face regions quickly.

Example :


Weak
Classifiers
AdaBoost
Face
Cascade
Stages
Strong
Classifier
Face
Face
Face
Face



2.2 methods used for face detection :

2. DNN-Based Detection (Deep Neural Networks):

- Caffe-based OpenCV DNN face detector (res10_300x300_ssd_iter_140000.caffemodel).
- MTCNN (Multi-task Cascaded Convolutional Networks).
- RetinaFace, BlazeFace, YOLO-based face detectors.

How it works:

- Uses **deep convolutional neural networks (CNNs)** to learn complex patterns.
- Detects faces more robustly under varied conditions (lighting, angle, size).

2.3 Key Parameters for Face Detection:

1. scaleFactor:

Purpose: Controls how the image is resized during detection to handle faces of different sizes.

Example Values:

- **1.1:** Resizes the image by 10% at each scale. Higher accuracy, but slower.
- **1.5:** Resizes by 50% at each scale. Faster but less precise.

Effect: Lower values detect smaller faces but take more time.

2.3 Key Parameters for Face Detection:

2. minNeighbors:

Purpose: Specifies how many overlapping rectangles are required to confirm a face.

Example Values:

- 3: Less strict, detects more faces but may include false positives.
- 5: Stricter, fewer false positives but may miss some faces.

Effect: Higher values reduce false positives but might miss some faces.

2.3 Key Parameters for Face Detection:

3. minSize :

Purpose: Sets the minimum size (width, height) of the detected face. Smaller objects will be ignored.

Example Values:

- (30, 30): Ignores faces smaller than 30x30 pixels. Helps improve speed and reduce false detections.

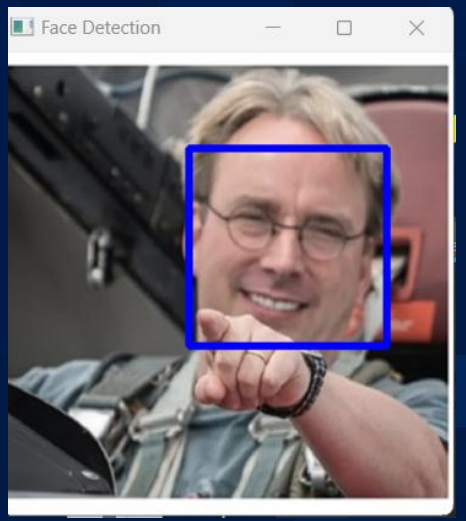
Effect: Filtering out small regions can make detection faster and more accurate, especially in high-resolution images.

2.4 Setting Up Haar Cascade Classifier :

..py | facedetection.py | merge.py | roi.py | new.png | test.py

```python
import cv2

# Load the pre-trained Haar cascade classifier for face detection
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
image = cv2.imread('new.png')

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Detect faces in the image
faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

# Draw rectangles around detected faces
for (x, y, w, h) in faces:
    cv2.rectangle(image, (x, y), (x + w, y + h), (255, 0, 0), 3)  # Blue rectangle

cv2.imshow( winname: 'Face Detection', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

The Output :



Pixels
EVERY PIXEL IS KNOWLEDGE

Task 1:

1. Detect a face from one image **Using OpenCV's Haar Cascade classifier .**
2. Crop it .
3. Resize and rotate it .
4. Paste it onto a white image (like placing a sticker) .

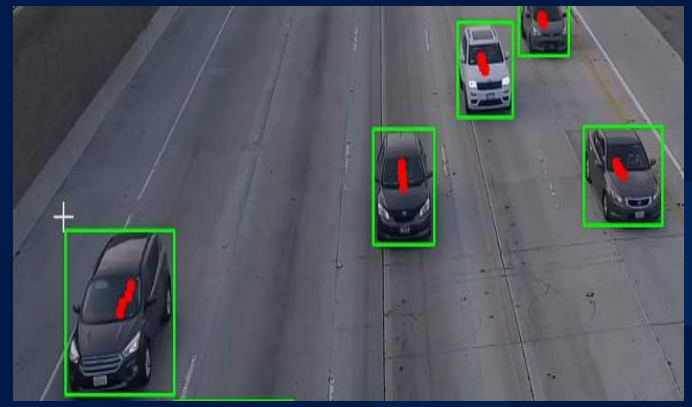
3. Object Tracking and Color-Based Tracking :

1. What is Object Tracking?

- Tracks an object **after it's been detected** in the first frame.
- Doesn't re-detect the object in future frames—just estimates movement based on the previous position.

How to Create a Tracker ?

1. cv2.legacy.TrackerMIL_create()
2. cv2.legacy.TrackerKCF_create()
3. cv2.legacy.TrackerTLD_create()
4. cv2.legacy.TrackerMOSSE_create()
5. cv2.legacy.TrackerCSRT_create()



3.1 Which OpenCV Tracker Should You Use?

Feature	CSRT	KCF	MOSSE
Accuracy	Very high	Medium to good	Low
Speed	Slower	Faster than CSRT	Fastest
Scale Adaptation	Supports scale changes	Does **not** support scale	Does **not** support scale
Handles Deformation	Very good	Average	Poor
Best Use Case	Precise tracking in complex scenes	Balance between speed and accuracy	Real-time, simple tracking
Ideal for	When object size or shape changes	When object is stable	When you need instant speed

3.2 Setting Up Object Tracking:

1.install Required Package : pip install opencv-contrib-python

Example :

```
# Initialize tracker
tracker = cv2.TrackerCSRT_create()
tracker.init(frame, bbox)

# Start tracking
while True:
    ret, frame = cap.read()
    if not ret:
        break

    success, bbox = tracker.update(frame)

    if success:
        x, y, w, h = [int(v) for v in bbox]
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
        cv2.putText(frame, "Tracking", (20, 50),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2)
    else:
        cv2.putText(frame, "Lost", (20, 50),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

    cv2.imshow("Object Tracking", frame)

    if cv2.waitKey(1) & 0xFF == ord("q"):
        break

cap.release()
cv2.destroyAllWindows()
```

3.3 COLOR-BASED TRACKING (Using HSV Color Filtering):

1. What is Color-Based Tracking?

- Tracks objects **based on a specific color** rather than shape or texture.
- Works well for **bright, distinct-colored objects** like a red ball, green gloves, etc.

2. Why Use HSV Instead of RGB?

- RGB is sensitive to lighting.
- **HSV (Hue, Saturation, Value)** allows better control of color filtering.

Pixels
EVERY PIXEL IS KNOWLEDGE

Example :

```python
import cv2
import numpy as np

cap = cv2.VideoCapture(0)
while True:
    ret, frame = cap.read()
    if not ret:
        break

    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    # Track red color
    lower_red = np.array([0, 120, 70])
    upper_red = np.array([10, 255, 255])
    mask = cv2.inRange(hsv, lower_red, upper_red)

    # Find contours
    contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    for contour in contours:
        if cv2.contourArea(contour) > 300:
            x, y, w, h = cv2.boundingRect(contour)
            cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
            cv2.putText(frame, text: "Red Object", org: (x, y - 10),
                        cv2.FONT_HERSHEY_SIMPLEX, fontScale: 0.6, color: (0, 255, 0), thickness: 2)

    cv2.imshow( winname: "Mask + Drawing", frame)
```

Pixels
EVERY PIXEL IS KNOWLEDGE

Task 2:

1- Track an object of a specific color
(like blue or red) in real-time using your webcam.

2- draw a circle around it on the live video feed .

4. Masking and Drawing on Video:

1. Masking :

It is the process of using a **binary image** (mask) to define which parts of an image to show or hide.

- A **mask** is usually black and white (0s and 255s).

How do we apply it to the image?

We apply a mask to an image using **bitwise operations**, specifically **cv2.bitwise_and()** in OpenCV.

Pixels
EVERY PIXEL IS KNOWLEDGE

4.1 Bitwise Operations on Images :

Common Operations :

I. cv2.bitwise_and(img1, img2) : Performs AND operation .

I. cv2.bitwise_or(img1, img2) : Performs OR operation .

II. cv2.bitwise_xor(img1, img2) : Performs XOR operation .

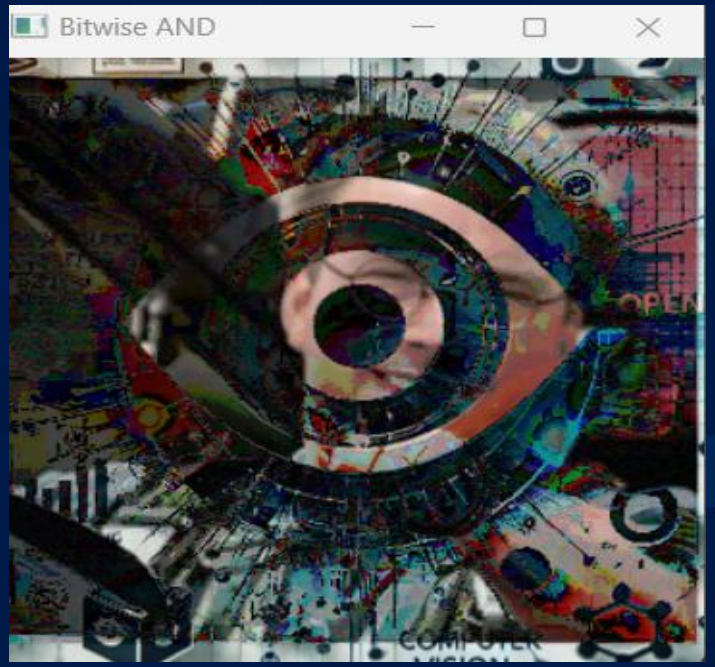
III. cv2.bitwise_not(img) : Performs NOT operation .

Example on Common Operations :

..py | facedetection.py | operation.py | merge.py | roi.py | new.png | test.py

```python
import cv2

img1 = cv2.imread('new.png')
img2 = cv2.imread("C:\\Users\\Basma Ahmed\\Pictures\\Screenshots\\Screenshot 2024-12-07 065433.png")

# Resize both images to the same size
img2 = cv2.resize(img2, dsize: (img1.shape[1], img1.shape[0]))

bitwise_and = cv2.bitwise_and(img1, img2)

cv2.imshow( winname: 'Image 1', img1)
cv2.imshow( winname: 'Image 2', img2)
cv2.imshow( winname: 'Bitwise AND', bitwise_and)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

The Output :



Pixels
EVERY PIXEL IS KNOWLEDGE

Task 3:

Combine two images using cv2.bitwise_and, cv2.bitwise_or, cv2.bitwise_xor, and cv2.bitwise_not, and visualize the differences.

4.2 Masking in Real-time Video :

```python
cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    if not ret:
        break

    # Convert to HSV color space
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    # Define color range for blue
    lower_blue = np.array([100, 150, 50])
    upper_blue = np.array([140, 255, 255])

    # Create mask
    mask = cv2.inRange(hsv, lower_blue, upper_blue)

    # Apply mask
    masked_output = cv2.bitwise_and(frame, frame, mask=mask)

    # Show both original and masked output
    cv2.imshow( winname: "Original", frame)
    cv2.imshow( winname: "Masked (Blue Only)", masked_output)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

4.3 Drawing on Video Frames :

```python
import cv2
cap = cv2.VideoCapture(0)
if not cap.isOpened():
    print("Error: Could not open webcam.")
    exit()
x = 0
direction = 1
while True:
    ret, frame = cap.read()
    if not ret:
        break

    height, width = frame.shape[:2]
    center = (x, height // 2)
    cv2.circle(frame, center, radius: 50, color: (255, 0, 0), thickness: 3)
    cv2.line(frame, pt1: (0, 0), center, color: (0, 0, 255), thickness: 2)

    x += 5 * direction
    if x >= width or x <= 0:
        direction *= -1  # change direction

    cv2.imshow( winname: "Drawing on Video", frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Pixels
EVERY PIXEL IS KNOWLEDGE

BreakTime

فيما أمضاه

5. Using Pretrained AI Models in OpenCV:

Recap:

A **pretrained model** is a machine learning model that has already been trained on a large dataset (like ImageNet, COCO, etc.) and can be reused for similar tasks without retraining.

2. Examples of pretrained models used with OpenCV:

- Face detectors (based on DNN or Haar Cascades).
- Object detection (e.g., MobileNet + SSD).
- Pose estimation.
- Age & gender prediction.

5.2 How does OpenCV use pretrained models?

OpenCV provides a tool called cv2.dnn (DNN = Deep Neural Network), which lets us:

1. **Load** the pretrained model into our code.

2. **Feed images or video frames** into it.

3. **Get predictions** like: “There’s a face here, at this location.”

Example:

Think of the pretrained model like a **scanner**:

1. You give it a picture.
2. It looks at it.
3. It tells you what it sees: “I see a face in this part of the image.”

5.3 Face Detection Using Pretrained DNN:

OpenCV supports a pre-trained model called **Res10 SSD Face Detector**, trained using a framework called **Caffe**.

Steps	Description
1. Load the model files	- deploy.prototxt.txt: Defines the model architecture - res10_300x300_ssd_iter_140000.caffemodel: Contains the pre-trained weights
2. Capture a frame	Read a frame from the webcam using cv2.VideoCapture()
3. Convert to blob	Use cv2.dnn.blobFromImage() to preprocess the frame for the neural network
4. Set input to the model	Feed the blob into the model using net.setInput()
5. Forward pass	Run net.forward() to get detection predictions
6. Draw results	Use OpenCV drawing functions (e.g., cv2.rectangle) to display detected faces

Pixels
EVERY PIXEL IS KNOWLEDGE

Example :

```python
# Load the pretrained model (Caffe format)
model_file = "res10_300x300_ssd_iter_140000.caffemodel"
config_file = "deploy.prototxt.txt"

net = cv2.dnn.readNetFromCaffe(config_file, model_file)
# Start video capture from webcam
cap = cv2.VideoCapture(0)
while True:
    ret, frame = cap.read()
    if not ret:
        break

    # Prepare the frame as input blob for the model
    blob = cv2.dnn.blobFromImage(frame, scalefactor: 1.0, size: (300, 300), mean: (104, 117, 123))
    net.setInput(blob)
    detections = net.forward()
    h, w = frame.shape[:2]
```

```python
    # Loop through detections
    for i in range(detections.shape[2]):
        confidence = detections[0, 0, i, 2]

        if confidence > 0.5:
            # Get coordinates of the bounding box
            box = detections[0, 0, i, 3:7] * [w, h, w, h]
            x1, y1, x2, y2 = box.astype(int)

            # Draw the bounding box
            cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
            label = f"{confidence * 100:.1f}%"
            cv2.putText(frame, label, org: (x1, y1 - 10),
                        cv2.FONT_HERSHEY_SIMPLEX, fontScale: 0.6, color: (0, 255, 0), thickness: 2)

    # Display the result
    cv2.imshow( winname: "Face Detection (DNN)", frame)

    # Press 'q' to quit
    if cv2.waitKey(1) & 0xFF == ord("q"):
        break

cap.release()
cv2.destroyAllWindows()
```

6. Real-Time Detection with MobileNet-SSD :

1. What is MobileNet-SSD?

- **MobileNet** is a lightweight, efficient deep learning model — designed to run fast on devices like laptops or even phones.

- **SSD** stands for **Single Shot Multibox Detector** — a type of object detection method that detects multiple objects **in a single pass**, in **real-time**.

6. Real-Time Detection with MobileNet-SSD :

MobileNet-SSD is a model that can **quickly and accurately detect multiple objects** in an image or video stream — like people, cars, bottles, dogs, etc.

Model Files:

- MobileNetSSD_deploy.prototxt → model structure .
- MobileNetSSD_deploy.caffemodel → pretrained weights.

Pixels
EVERY PIXEL IS KNOWLEDGE

6.2 Steps to Use MobileNet-SSD :

1. **Load the model** using OpenCV's cv2.dnn.readNetFromCaffe().
2. **Capture video frames** from webcam or a video.
3. **Convert each frame into a blob** (input format for the model).
4. **Feed the blob into the network** using net.setInput().
5. **Run a forward pass** to get detections using net.forward().
6. **Draw boxes and labels** for detected objects.

Pixels
EVERY PIXEL IS KNOWLEDGE

Example :

```python
import cv2
import numpy as np

# Load class labels MobileNet SSD was trained on
classNames = ["background", "aeroplane", "bicycle", "bird", "boat",
              "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
              "dog", "horse", "motorbike", "person", "pottedplant",
              "sheep", "sofa", "train", "tvmonitor"]

# Load model files
net = cv2.dnn.readNetFromCaffe("MobileNetSSD_deploy.prototxt",
                               "MobileNetSSD_deploy.caffemodel")

cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    if not ret:
        break

    h, w = frame.shape[:2]

    # Convert to blob
    blob = cv2.dnn.blobFromImage(frame, scalefactor: 0.007843, size: (300, 300), mean: 127.5)
    net.setInput(blob)
    detections = net.forward()
```

```python
    # Loop through detections
    for i in range(detections.shape[2]):
        confidence = detections[0, 0, i, 2]
        if confidence > 0.5:
            class_id = int(detections[0, 0, i, 1])
            class_name = classNames[class_id]
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            x1, y1, x2, y2 = box.astype("int")

            cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
            cv2.putText(frame, text: f"{class_name}: {confidence*100:.1f}%",
                        org: (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, fontScale: 0.5,
                        color: (0, 255, 0), thickness: 2)

    cv2.imshow( winname: "MobileNet-SSD Real-Time Detection", frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Hands-On Practice:

Notebook:

https://colab.research.google.com/drive/1pkAAQF3ikl1Iwh_ylFzIQjcCuswW4hSi?usp=sharing

Pixels
EVERY PIXEL IS KNOWLEDGE

Any questions

فيما أفناه

Pixels
EVERY PIXEL IS KNOWLEDGE

THANK YOU .