# LAB # 12

## Real-time Object Classification

### OBJECTIVE

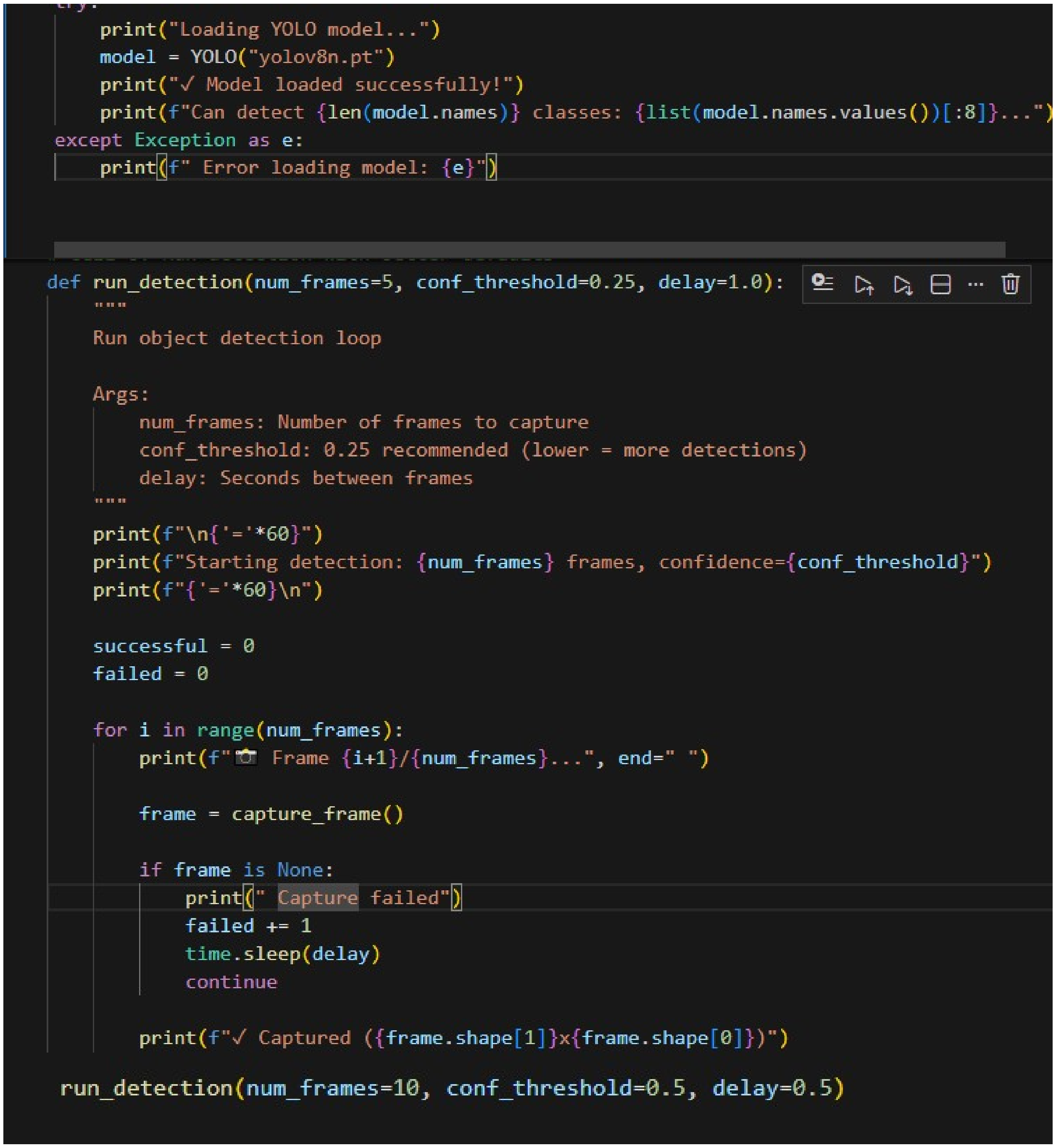
To demonstrate how a Machine Learning model can process the real-world camera feed in an AR environment and augment the user's view with intelligent visual feedback (like identifying objects or labeling surroundings).

## Code :

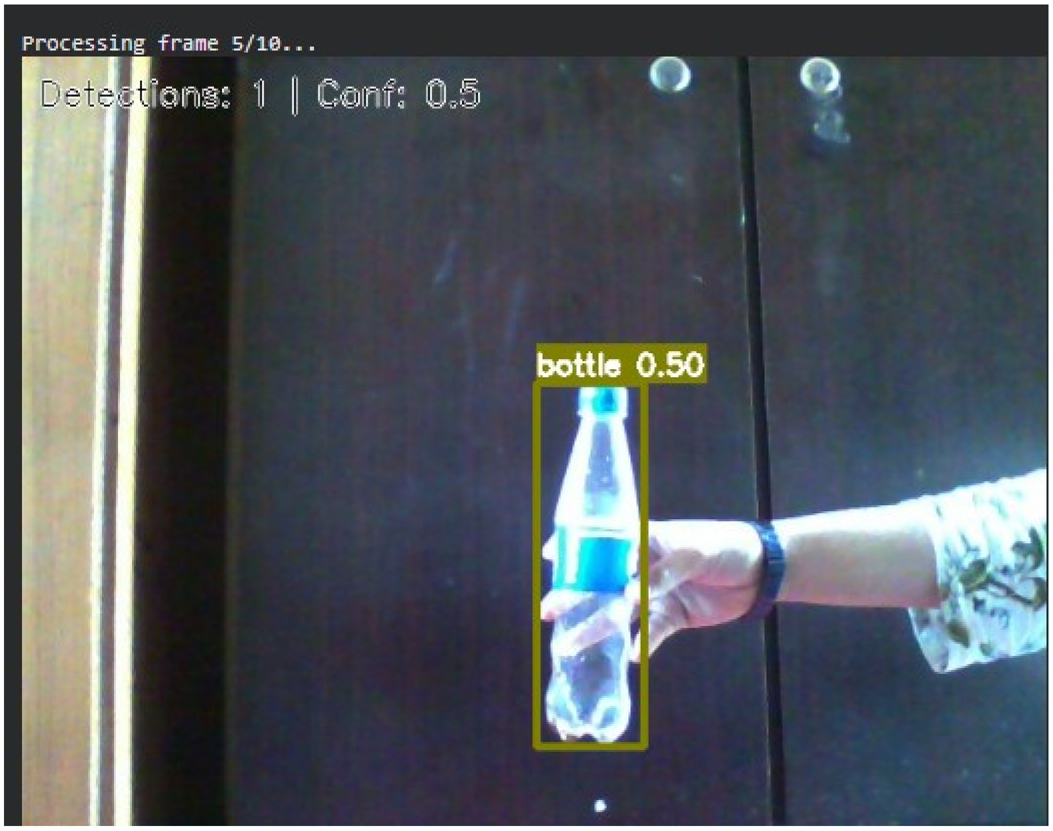
*Main highlights colab didn't support streaming that why I suppose to detect by frames*

```
# Cell 2: Imports
import cv2
import base64
import numpy as np
import time
from ultralytics import YOLO
from google.colab.output import eval_js
from google.colab.patches import cv2_imshow
from IPython.display import display, Javascript, HTML
```

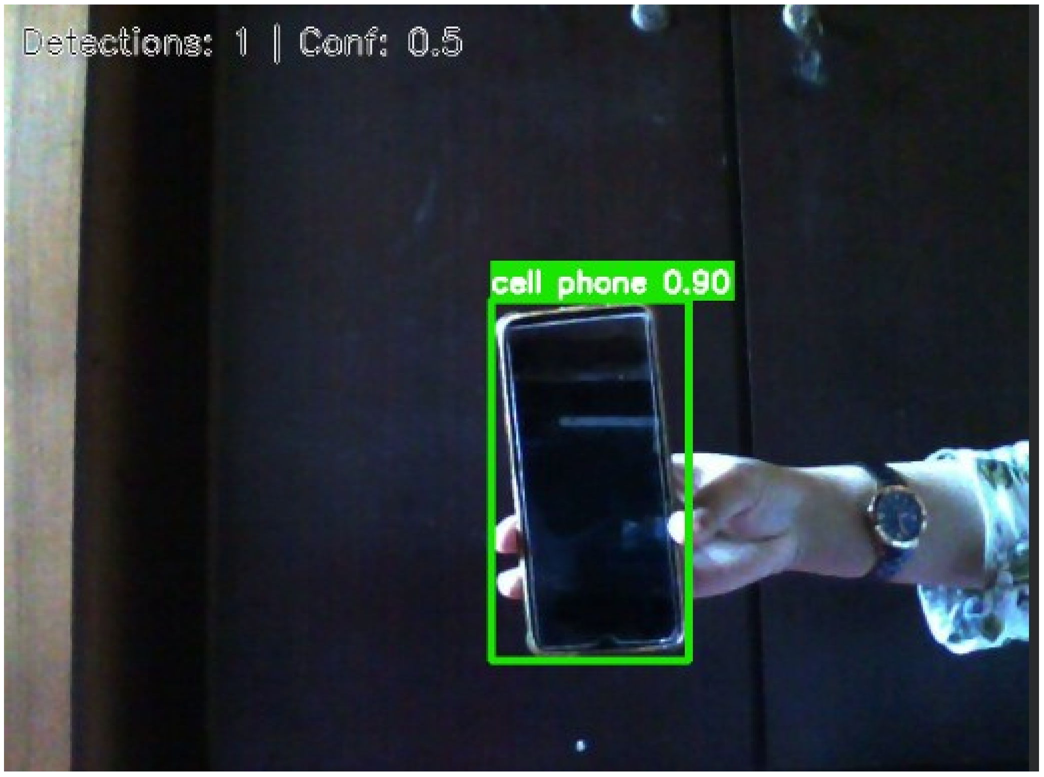
```
# Cell 3: Initialize webcam (RUN THIS FIRST!)
def setup_webcam():
    """Initialize webcam access - must run this cell first!"""
    js_code = Javascript('''
    async function setupCamera() {
        const div = document.createElement('div');
        div.innerHTML = `
            <div style="padding: 20px; background: #f0f0f0; border-radius:
10px;">
                <h3>□ Webcam Setup</h3>
                <video id="webcam" autoplay playsinline
style="display:none;"></video>
                <p id="status">Initializing camera...</p>
            </div>
        `;
        document.body.appendChild(div);


        try {
            const stream = await navigator.mediaDevices.getUserMedia({
                video: { width: 640, height: 480 }
            });
            const video = document.getElementById('webcam');
            video.srcObject = stream;
            await video.play();
            document.getElementById('status').innerHTML = '✓ Camera ready!
You can now run detection.';
            document.getElementById('status').style.color = 'green';
        } catch(err) {
            document.getElementById('status').innerHTML = '✗ Error: ' +
err.message;
            document.getElementById('status').style.color = 'red';
        }
    }
    setupCamera();
    ''')
    display(js_code)
    print("Please allow camera access when prompted!")
    print("Wait for 'Camera ready' message before proceeding")
```

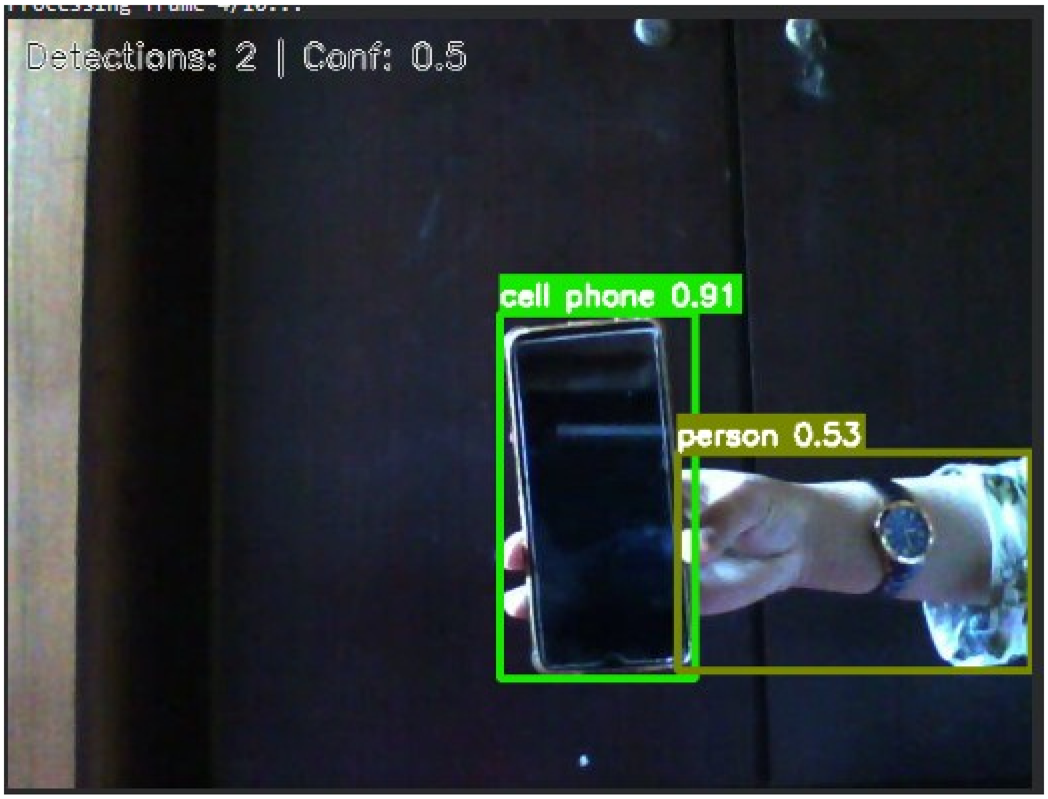


```
    print("Loading YOLO model...")
    model = YOLO("yolov8n.pt")
    print("✓ Model loaded successfully!")
    print(f"Can detect {len(model.names)} classes: {list(model.names.values())[:8]}...")
except Exception as e:
    print(f" Error loading model: {e}")


def run_detection(num_frames=5, conf_threshold=0.25, delay=1.0):
    """
    Run object detection loop

    Args:
        num_frames: Number of frames to capture
        conf_threshold: 0.25 recommended (lower = more detections)
        delay: Seconds between frames
    """
    print(f"\n{'='*60}")
    print(f"Starting detection: {num_frames} frames, confidence={conf_threshold}")
    print(f"{'='*60}\n")

    successful = 0
    failed = 0

    for i in range(num_frames):
        print(f"📷 Frame {i+1}/{num_frames}...", end=" ")

        frame = capture_frame()

        if frame is None:
            print(" Capture failed")
            failed += 1
            time.sleep(delay)
            continue

        print(f"✓ Captured ({frame.shape[1]}x{frame.shape[0]})")

run_detection(num_frames=10, conf_threshold=0.5, delay=0.5)
```

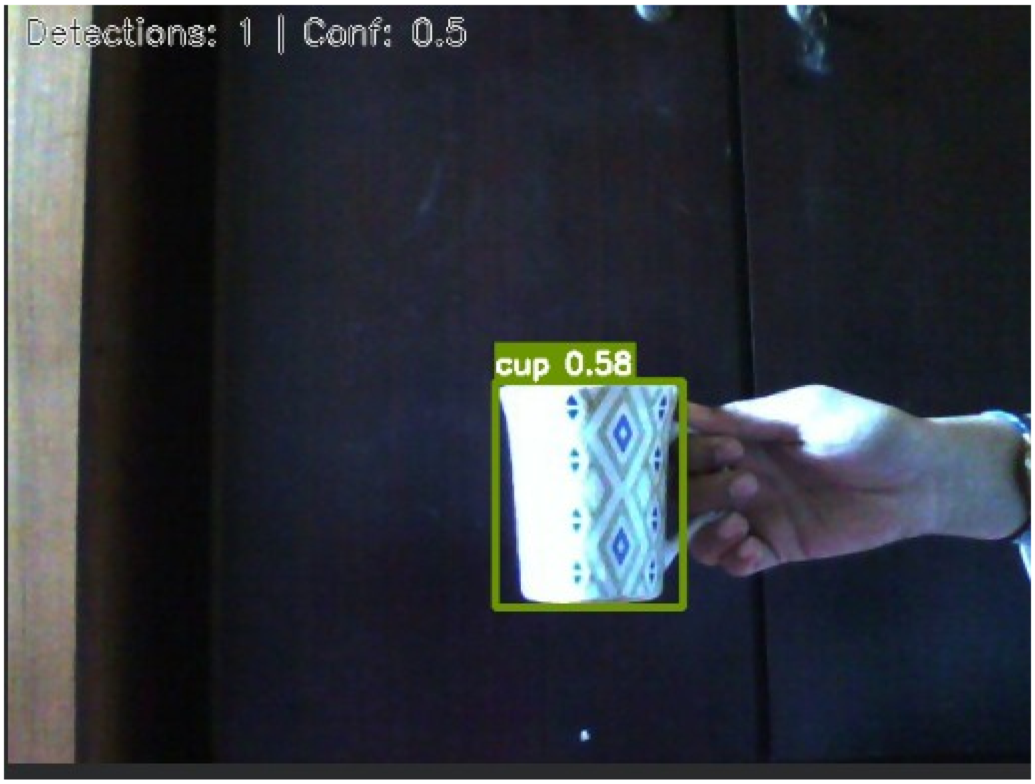


```
Processing frame 5/10...
```





```
Processing frame 4/10...
```



Detections: 1 | Conf: 0.5
cup 0.58