**Real-Time Artifact: Live Webcam Object Detection with Notifications**

**Project Overview**

**Skill Showcased**: Real-time computer vision, model deployment, and edge AI.

Build a system that uses your laptop/webcam to detect objects in real-time and sends notifications (e.g., logs or sounds) when specific objects (e.g., "cell phone," "book") are detected.

**Step-by-Step Implementation**

**1. Tools & Technologies**

* **Framework**: Python + PyTorch/YOLO (for real-time detection)
* **Libraries**: OpenCV (webcam feed), pyttsx3 (text-to-speech alerts)
* **Deployment**: Docker (optional for edge deployment), Flask (for API endpoints)

**2. Code Snippets**

**a. Real-Time Object Detection Script**

import cv2

import torch

from pyttsx3 import init

# Load YOLOv5 model (pre-trained on COCO dataset)

model = torch.hub.load('ultralytics/yolov5', 'yolov5s', pretrained=True)

# Initialize text-to-speech engine

engine = init()

# Open webcam

cap = cv2.VideoCapture(0)

while True:

ret, frame = cap.read()

if not ret:

break

# Perform inference

results = model(frame)

# Parse results

labels = results.pandas().xyxy[0]['name'].tolist()

# Alert if specific objects are detected (e.g., "cell phone")

if "cell phone" in labels:

engine.say("Cell phone detected!")

engine.runAndWait()

# Display bounding boxes

cv2.imshow('Live Object Detection', np.squeeze(results.render()))

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

break

cap.release()

cv2.destroyAllWindows()

**b. Dockerfile for Deployment**

FROM python:3.9-slim

WORKDIR /app

COPY requirements.txt .

RUN pip install -r requirements.txt

COPY . .

CMD ["python", "app.py"]

**c. Requirements.txt**

torch

opencv-python

pyttsx3

ultralytics

**Portfolio Artifact Documentation**

**Title**

**"Real-Time Object Detection System with Alerting"**

**Introduction**

A live webcam application that identifies objects using YOLOv5 and triggers audio alerts for specific items. Demonstrates edge AI deployment and real-time processing.

**Objective**

* Highlight proficiency in real-time computer vision.
* Showcase end-to-end deployment (Docker) and integration with hardware (webcam).

**Process**

1. **Model Selection**: Chose YOLOv5 for speed/accuracy trade-off.
2. **Alert Logic**: Added conditional checks for target objects (e.g., "cell phone").
3. **Deployment**: Containerized the app for reproducibility.

**Tools & Technologies**

* **Python, PyTorch, OpenCV**
* **Docker, YOLOv5, pyttsx3**

**Value Proposition**

* **Unique Value**: Combines real-time inference with user notifications, mimicking industrial IoT use cases.
* **Relevance**: Useful for security systems, retail analytics, or accessibility tools.

**GitHub Repo**

<https://github.com/samikhan81/2.5-Professional-Portfolio-Artifact-2/tree/aad17fbbea5e78a16836fa1e113e1b39dfd666f7>

**Reflection Template**

**Customization for Audience**

* Added a **"Business Impact"** section in the portfolio to explain how this could reduce theft in retail stores.
* Simplified the GitHub README for recruiters (e.g., added a "Quick Start" guide).

**Lessons Learned**

* Real-time systems require balancing accuracy and speed (YOLOv5 processes ~30 FPS).
* Docker simplified deployment but required troubleshooting webcam permissions.