Documentation for Camera Detection Proof of Concept (PoC)

Introduction:

The Camera Detection Proof of Concept (PoC) focuses on real-time camera detection using computer vision techniques. Its purpose is to accurately identify cameras in a webcam stream, serving to safeguard confidential documents and ensure privacy.

Approach:

The development process began by training the YOLOv7 model on a custom dataset consisting of 300 annotated images containing classes such as phone, camera, and GoPro. However, the model's performance did not meet the desired expectations, prompting the exploration of alternative approaches.

Pretrained models were investigated to leverage their existing knowledge and achieve better results. Detectron2, a model pre-trained on 21K classes, was implemented. Although it yielded satisfactory results, the main challenge was its slow processing speed, which hindered real-time camera detection.

To address the speed issue, YOLO9000 was considered, as it exhibited faster performance. However, its accuracy fell below the required standards. Consequently, the OpenImages and Roboflow datasets were discovered. These datasets contained a large number of camera and mobile phone images, annotated in the darknet (YOLO) format.

The YOLOv8 model was trained on the OpenImages and Roboflow datasets for approximately 100 epochs. The results obtained were promising and met the desired accuracy levels. Additionally, pretrained weights from the Object365 dataset, which included camera and mobile phone classes, were applied to YOLOv5. After testing both models in different scenarios, YOLOv5 was selected for its optimal performance.

User Interface:

A StreamIt application was developed to facilitate the deployment of the camera detection system. The application opens the webcam and continuously analyzes the video stream to detect cameras. Upon camera detection, it captures a screenshot of the screen and a snapshot from the webcam. These images are then saved in a designated folder for further analysis or storage.

Dockerization:

- **Dockerfile**
    1. Define the Base Image
       Start by specifying the base image as "python:3.9-slim-buster" in the Dockerfile.
       FROM python:3.9-slim-buster

    2. Install Dependencies
       Install the necessary dependencies using the following commands:
       **RUN apt-get update && **
       **apt-get install -y --no-install-recommends **
       **build-essential **
       **libjpeg-dev **
       **libpng-dev **
       **libfreetype6-dev **
       **libblas-dev **
       **liblapack-dev **
       **gfortran **
       **git && **
       rm -rf /var/lib/apt/lists/*

    3. Upgrade pip
       Upgrade pip to the latest version using the following command:
       RUN python -m pip install --upgrade pip

    4. Set Working Directory
       Set the working directory to "/app" using the following command:
       WORKDIR /app

5. Copy Requirements File
   Copy the "requirements.txt" file from the host to the working directory in the container using the following command:
   COPY requirements.txt .

6. Clone YOLOv5 Repository
   Clone the YOLOv5 repository from GitHub using the following command:
   RUN git clone https://github.com/ultralytics/yolov5.git

7. Install Additional Dependencies
   Install additional dependencies required by the application using the following command:
   RUN apt-get update && apt-get install ffmpeg libsm6 libxext6 -y

8. Install Python Packages
   Install Python packages specified in the "requirements.txt" file using the following command:
   RUN pip install --no-cache-dir -r requirements.txt

9. Copy Application Files
   Copy all the files from the host to the working directory in the container using the following command:
   COPY . .

10. Define the Default Command
    Set the default command to run the application using Streamlit by executing the following command:
    CMD ["streamlit","run","app_ui_yolov5_with_video.py"]

- **Docker Compose**
  1. Create a Docker Compose File
     Create a new file named "docker-compose.yml" and open it for editing.

  2. Step 2: Specify Version
     In the first line of the file, specify the version of Docker Compose being used. For example, use version '3'.
     version: '3'

  3. Define Services
     Under the 'services' section, specify the services you want to build and run. In this case, we will define a service named 'app'.
     services:
     App:

4. Specify Build Configuration
   Under the 'app' service, define the build configuration by specifying the build context. Use a dot ('.') to represent the current directory where the Dockerfile is located.
   build: .

5. Specify Port Mapping
   Specify the port mapping to expose the application's port. In this case, we wil map port 8501 of the host to port 8501 of the container.
   ports:
   - "8501:8501"

Deployment:

To deploy the Camera Detection PoC on AWS, a ***t2 medium*** instance with ***16GiB*** storage was created. The steps required to set up a local instance of the "Camera Detection PoC" using Docker and Docker Compose.

1. Update Package Repository

Before installing any new packages, it's always a good idea to update the package repository to ensure that you have the latest versions of all packages.
Run the following command to update the package repository:
sudo apt-get update

2. Install Python 3 pip Package

Python 3 pip is a package manager for Python. It is used to install and manage packages and dependencies for Python projects. Run the following command to install the Python 3 pip package:
sudo apt-get install python3-pip

3. Install Docker Compose

Docker Compose is a tool that allows you to define and run multi-container Docker applications. Run the following command to install Docker Compose:
sudo apt install docker-compose

4. Install Docker.io

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Run the following command to install Docker:

sudo apt install docker.io

5. Clone the Project Repository

The "MMM_poc" project code is hosted on GitHub. Clone the project repository using the following command:

git clone https://github.com/ayushpatel-srijan/camera_detection_yolov5.git

6. Change Directory to the Project Directory

Change the current working directory to the “camera_detection_yolov5" project directory using the following command:

***cd* camera_detection_yolov5**

7. Start the Docker Compose Build Process

Run the following command to start the Docker Compose build process:

sudo docker-compose up --build

This will download all the necessary dependencies and build the project. Once the build process is complete, you can access the project by navigating to exposed ports in your web browser.