Lab 3: Learn to Containerize Application

Objectives

- Containerize YOLO app.

Deliverables

Compulsory (10%)

- Dockerfile (container image manifest) for YOLO darknet program.
- Dockerhub repo link for the uploaded container image, and screenshot of the repo.
- Container image for the built container. You can upload to GoogleDrive or any other shareable link.

Prerequisites

- Create a user account on the public container registry Dockerhub https://hub.docker.com/

Run YOLO application (using CPU)

Step 1: Create Dockerfile for YOLO

1. **Select a container image base with CUDA and cuDNN. Choose from https://hub.docker.com/r/nvidia/cuda**
2. **Include the following environment variables to skip OpenCV geographic location prompt during install**

```
ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ >
/etc/timezone
```

3. **Install prerequisite program (opencv) to compile darknet. I.e., libopencv-dev python3-opencv wget git build-essential**
4. **Clone darknet program from https://github.com/AlexeyAB/darknet**
5. **Change the working directory to the yolo app folder inside the container.**
6. **Download model weight copy to the container app path, i.e. https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.weights**
7. **Download YOLO config from https://raw.githubusercontent.com/AlexeyAB/darknet/master/cfg/yolov4.cfg**
8. **Copy video file from /opt/videos/traffic.mp4 to the container.**
9. **Expose port 8070 and 8090 for YOLO app. Port 8070 stream out JSON object for the detected object boundary and label, port 8090 is MJPEG stream.**
10. **Compile darknet with CPU flag, OpenCV and CUDNN**

```
make -j6 GPU=0 OPENCV=1 CUDNN_HALF=0 CUDNN=0
```

11. **Specify the command that is executed when startup the container as the last line.**

```
./darknet detector demo ./cfg/coco.data ./cfg/yolov4-custom.cfg
/opt/yolov4.weights /opt/videos/traffic.mp4 -json_port 8070 -
mjpeg_port 8090 -ext_out
```

Step 2: Build container image

1. **Build the container image and tag it as docker-yolo-cuda-cudnn:v1.0-<STUDENT_ID>. (Hint: docker build)**

Step 3: Run the container

1. **Run the container image. (Hint: docker run).**

With CPU

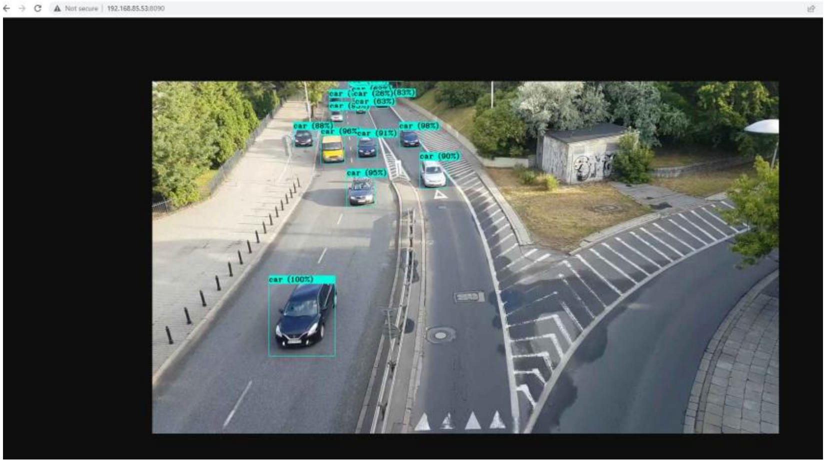
```
docker run --publish 8070:8070 --publish 8090:8090 <IMAGE_ID>
```

With GPU (No need to run in this lab)

```
docker run --runtime nvidia --gpus all --publish 8070:8070 --publish
8090:8090 <IMAGE_ID>
```

2. **Open VLC Player and view the MJPEG stream, i.e. http://XAVIER-NX-IP:8090**

Example output from VLC Player



Step 4: Upload YOLO container image to Dockerhub

1. **Create an account on Dockerhub**
2. **Login to docker (Hint: docker login)**
3. **Push the container image tagged as docker-yolo-cuda-cudnn:v1.0 to your repo, i.e. sxxxxxx/docker-yolo-cuda-cudnn:v1.0 (Hint: docker push)**

Step 5: Export the container image

1. **Export the container image tagged as docker-yolo-cuda-cudnn:v1.0 and save to docker-yolo-cuda-cudnn_v1.0.tar. (Hint: Use docker save)**

Extra – Utilize multi-stage build to slim down YOLO

Refer to https://docs.docker.com/develop/develop-images/multistage-build/

1. **Modify the Dockerfile from Step 1.**
2. **Copy only the compiled darknet to a new container.**
3. **Build and push the image as docker-yolo-cuda-cudnn-multistage:v1.0**
4. **Compare the size of docker-yolo-cuda-cudnn:v1.0 and docker-yolo-cuda-cudnn-multistage:v1.0 (Hint: docker image ls). Provide screenshot.**

References

1. Docker (OCI Container Image) multi stage build: https://docs.docker.com/develop/develop-images/multistage-build/
2. Best practices to write Dockerfile: https://docs.docker.com/develop/develop-images/dockerfile_best-practices/