Object Detection and Tracking with Event Vision-Based Sensors Application on AMD Kria™ KV260 Vision AI Starter Kit

Getting Started Guide

1. Overview

This Getting Started Guide describes the setup steps for running event camera-based machine learning inference on the AMD Kria™ KV260 Vision AI Starter Kit. The example in this guide uses an event camera from PROPHESEE and performs recorded event file-based and camera-based ML inference.

The detailed implementation of this application can be found at GitHub: https://github.com/LogicTronixInc/Kria-Prophesee-Event-VitisAI. The GitHub site consists of downloadable files to get the app running as well as steps to redesign and customize the app itself. This getting started document focuses on how to run the Kria KV260 Vision AI Starter Kit app.

This app supports both the Ubuntu and PetaLinux operating systems.

2. Prerequisites

The following are prerequisites for the app:

- AMD Kria KV260 Vision AI Starter Kit and power supply
- Ethernet cable
- USB cable (for serial connection to host machine)
- HDMI monitor or DP monitor and cable
- PROPHESEE CCAM5 (optional if using event file-based ML data)
- Mouse and keyboard
- SD card (32 GB)

3. Set Up the Ubuntu SD Card for the Kria KV260 Vision AI Starter Kit

- Get Ubuntu 22.04 from here.
- Create an SD card using the balena-etcher tool and prepare the Kria KV260 Vision AI Starter Kit for the SD card.

- Connect the accessories (keyboard, mouse, MIPI camera*, and Ethernet cable) and monitor to the Kria KV260 Vision AI Starter Kit.
- Boot the SD card and log into the Ubuntu desktop environment of the Kria KV260 Vision AI Starter Kit.
- Run a terminal in the Ubuntu home folder.
- **Run the steps that follow in the Ubuntu desktop terminal with the keyboard and mouse attached to the Kria KV260 Vision AI Starter Kit.**

* Camera is optional if using event file-based ML data.

4. Get the Ubuntu Firmware and Applications

From the Ubuntu home directory, get the Ubuntu firmware and necessary installation scripts from the following repository: https://github.com/LogicTronixInc/Kria-Prophesee-Event-VitisAI.

```
ubuntu@kria:~$ git clone https://github.com/LogicTronixInc/Kria-Prophesee-Event-VitisAI.git
```

After cloning the repository, run the scripts described in the steps that follow to update the Ubuntu kernel followed by the PROPHESEE driver installation.

5. Update the Ubuntu Kernel

Run the following script to update the Ubuntu kernel:

```
ubuntu@kria:~$ Kria-Prophesee-Event-VitisAI/Kria-Ubuntu/scripts/install_updated_linux_kernel.sh
```

6. Set Up the PROPHESEE Camera Drivers

Run the following script to install the PROPHESEE camera drivers:

```
ubuntu@kria:~$ Kria-PROPHESEE-Event-VitisAI/Kria-Ubuntu/scripts/install_psee_drivers.sh
```

7. Set Up the Hardware Firmware Overlays and vart Config File

Run the following script to install the hardware overlays:

```
ubuntu@kria:~$ Kria-PROPHESEE-Event-VitisAI/Kria-Ubuntu/scripts/install_hardware_overlays.sh
```

8. Get the Raw File for Test Input

For testing file-based input in the ML application, a raw event file is needed. Get a sample raw file by running the following command in the **home directory**:

```
ubuntu@kria:~$ wget https://logictronix.com/wp-
content/uploads/2024/07/traffic_cam_day_0007.zip

ubuntu@kria:~$ unzip traffic_cam_day_0007.zip
```

9. Prepare the Docker-Based Application

First, install Docker by following the instructions from Step 6 in Booting the Kria KV260 Vision AI Starter Kit Linux.

Pull the Docker image:

```
docker pull logictronixinc/PROPHESEE-ml-kria:0.3
```

It may take approximately 10–15 minutes to download the Docker image, depending on the internet speed.

View the available images using the Docker command:

```
docker images
```

```
ubuntu@kria:~$ docker pull logictronixinc/prophesee-ml-kria:0.3
0.3: Pulling from logictronixinc/prophesee-ml-kria
00f50047d606: Pull complete
d7951c234d55: Pull complete
05265a2d1f35: Pull complete
90b46a25b424: Pull complete
80e164c37cc5: Pull complete
3d8f42a1f194: Pull complete
b98fe3f03a5b: Pull complete
59a6d05de11d: Pull complete
c3201d2e9455: Pull complete
5a86aa1eda97: Pull complete
1c16e9132328: Pull complete
d5655ba163b7: Pull complete
40715a8038d7: Pull complete
f7a8afbba997: Pull complete
186feb8768ae: Pull complete
8eb6ae37a33d: Pull complete
915401969a52: Pull complete
4fb877479c5c: Pull complete
Digest: sha256:a18aa21beebb34edf349445a112841b1c1c2a01c82e6e2d22e1686eb0393081f
Status: Downloaded newer image for logictronixinc/prophesee-ml-kria:0.3
docker.io/logictronixinc/prophesee-ml-kria:0.3
ubuntu@kria:~$ docker images
REPOSITORY                          TAG       IMAGE ID       CREATED        SIZE
logictronixinc/prophesee-ml-kria    0.3       e1b54e9c305c   4 hours ago    2.41GB
ubuntu@kria:~$
```

Load the hardware overlay using xmutil:

```
sudo xmutil unloadapp
sudo xmutil loadapp psee-vitis-mipi-dpu
```

```
ubuntu@kria:~$ sudo xmutil listapps
[sudo] password for ubuntu:
                  Accelerator          Accel_type                          Base          Base_type      #slots(PL+AIE)        Active_slot

           psee-vitis-mipi-dpu            XRT_FLAT           psee-vitis-mipi-dpu            XRT_FLAT             (0+0)                 -1
         prophesee-kv260-imx636           XRT_FLAT         prophesee-kv260-imx636           XRT_FLAT             (0+0)                 -1
               kv260-smartcam             XRT_FLAT               kv260-smartcam             XRT_FLAT             (0+0)                 -1
          kv260-benchmark-b4096           XRT_FLAT          kv260-benchmark-b4096           XRT_FLAT             (0+0)                 -1
               k26-starter-kits           XRT_FLAT               k26-starter-kits           XRT_FLAT             (0+0)                 0,
                 kv260-psee-dpu           XRT_FLAT                 kv260-psee-dpu           XRT_FLAT             (0+0)                 -1
ubuntu@kria:~$ sudo xmutil unloadapp
remove from slot 0 returns: 0 (Ok)
ubuntu@kria:~$ sudo xmutil loadapp
-load expects a package name. Try again.
ubuntu@kria:~$ sudo xmutil loadapp psee-vitis-mipi-dpu
psee-vitis-mipi-dpu: loaded to slot 0
ubuntu@kria:~$
```

To get the GUI to obtain output from the application running in the Docker container, run the following command:

```
xhost local:docker
```

Then run the docker run command as below or run the **docker_run.sh** file present in the Kria-Prophesee-Event-VitisAI/Kria-Ubuntu/scripts folder:

```
docker run \
-e DISPLAY=unix$DISPLAY \
-h "xlnx-docker" \
--env="XDG_SESSION_TYPE" \
--net=host \
--privileged \
-v /tmp:/tmp \
-v /dev:/dev \
-v /sys:/sys \
-v /etc/vart.conf:/etc/vart.conf \
-v /lib/firmware/xilinx:/lib/firmware/xilinx \
-v /run:/run \
-v /home/ubuntu:/ubuntu \
-it logictronixinc/PROPHESEE-ml-kria:0.3 bash
```

This will start the Docker container and open a Docker shell terminal as shown below:

```
ubuntu@kria:~$ docker run \
-e DISPLAY=unix$DISPLAY \
-h "xlnx-docker" \
--env="XDG_SESSION_TYPE" \
--net=host \
--privileged \
-v /tmp:/tmp \
-v /dev:/dev \
-v /sys:/sys \
-v /etc/vart.conf:/etc/vart.conf \
-v /lib/firmware/xilinx:/lib/firmware/xilinx \
-v /run:/run \
-v /home/ubuntu:/ubuntu \
-it logictronixinc/prophesee-ml-kria:0.3 bash

==========================================


 _  __     _          ____    ___  __   __
| |/ /_ __(_) __ _  / ___|  / _ \|  \/  |
| ' /| '__| |/ _` | \___ \| | | | |\/| |
| . \| |  | | (_| |  ___) | |_| | |  | |
|_|\_\_|  |_|\__,_| |____/ \___/|_|  |_|


==========================================

Build Date: 2022/09/26 15:21
root@xlnx-docker:/#
```

Note: In the above docker run command, the Ubuntu home directory **/home/ubuntu** is mounted as the **/ubuntu** volume, so you can access the files and folders located in the Ubuntu home directory.

Set up the environment variables for running the AMD Vitis™ AI-based application in Docker:
In the Docker terminal, run the following command to set up the environment variables to run AMD Vitis AI tests and applications:

```
root@xlnx-docker:/# cd /ubuntu
root@xlnx-docker:/ubuntu# source /ubuntu/Kria-PROPHESEE-Event-VitisAI/Kria-Ubuntu/scripts/setup_dpu_env.sh
root@xlnx-docker:/ubuntu#
```

Things to check before running the application:

- Camera test (**optional** if the PROPHESEE CCAM5 hardware is connected to the board):
  - Run the following command in the console to get the v4l2 media graph and sensor device number:

```
# media-ctl -p
```

```
----------------------
driver          psee-video
model           Prophesee Video Pipeline
serial
bus info
hw revision     0x0
driver version  5.15.136

Device topology
- entity 1: ps_host_if output 0 (1 pad, 1 link)
            type Node subtype V4L flags 0
            device node name /dev/video0
        pad0: Sink
                <- "a0050000.event_stream_smart_tra":1 [ENABLED]

- entity 5: a0010000.mipi_csi2_rx_subsystem (2 pads, 2 links)
            type V4L2 subdev subtype Unknown flags 0
            device node name /dev/v4l-subdev0
        pad0: Sink
                [fmt:unknown/1280x720 field:none colorspace:raw xfer:none]
                <- "imx636 6-003c":0 [ENABLED]
        pad1: Source
                [fmt:unknown/1280x720 field:none colorspace:raw xfer:none]
                -> "a0040000.axis_tkeep_handler":0 [ENABLED]

- entity 8: imx636 6-003c (1 pad, 1 link)
            type V4L2 subdev subtype Sensor flags 0
            device node name /dev/v4l-subdev1
        pad0: Source
                [fmt:unknown/1280x720 field:none colorspace:raw xfer:none]
                -> "a0010000.mipi_csi2_rx_subsystem":0 [ENABLED]

- entity 10: a0040000.axis_tkeep_handler (2 pads, 2 links)
             type V4L2 subdev subtype Unknown flags 0
             device node name /dev/v4l-subdev2
        pad0: Sink
                [fmt:unknown/1280x720 field:none colorspace:raw xfer:none]
                <- "a0010000.mipi_csi2_rx_subsystem":1 [ENABLED]
        pad1: Source
                [fmt:unknown/1280x720 field:none colorspace:raw xfer:none]
                -> "a0050000.event_stream_smart_tra":0 [ENABLED]

- entity 13: a0050000.event_stream_smart_tra (2 pads, 2 links)
             type V4L2 subdev subtype Unknown flags 0
             device node name /dev/v4l-subdev3
        pad0: Sink
                [fmt:unknown/1280x720 field:none colorspace:raw xfer:none]
                <- "a0040000.axis_tkeep_handler":1 [ENABLED]
        pad1: Source
                [fmt:unknown/1280x720 field:none colorspace:raw xfer:none]
                -> "ps_host_if output 0":0 [ENABLED]

root@xlnx-docker:/ubuntu#
```

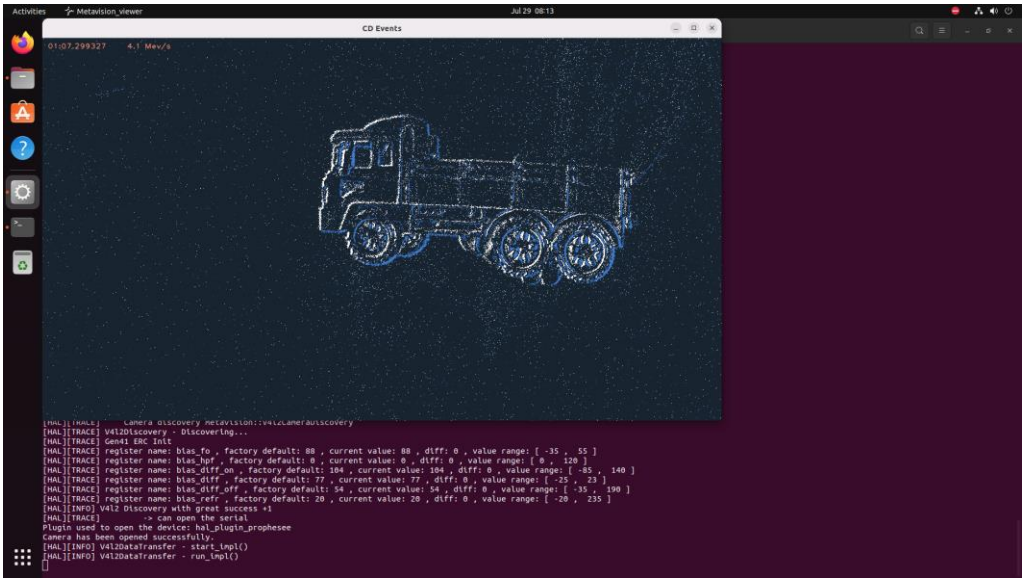
In the above media graph imx636, the device node is **/dev/v4l-subdev1**. This information is required to enable and run the application.

Before running the event camera ML application, you can check if the camera data pipeline is working or not by running the default metavision-viewer application. To run this test application, run the following commands:

```
# source /ubuntu/Kria-Prophesee-Event-VitisAI/Kria-Ubuntu/scripts/load.sh
# echo on > /sys/class/video4linux/v4l-subdev<X>/device/power/control
# MV_LOG_LEVEL=TRACE V4L2_HEAP=reserved  V4L2_SENSOR_PATH=/dev/v4l-
subdev<X> metavision_viewer
```

Here, the v4l-subdev<X> value replaces the <X> value according to the v4l2 device number for the imx636 device in the media graph obtained from the camera test step.

This will display the camera output as shown below:



If the camera data pipeline is stuck or not working, then the above command will result in a blank output. In this scenario, restart the application or review the **Known Issues and Solutions** section of this guide.

10. Run Event File-based YoloV4-tiny

If you are using file input, get the raw event file located in the /home/ubuntu directory that was downloaded in "Step 8: Get the Raw File for Test Input".

Next, run the event ML app **metavision_histo_viewer_yolov4** executable located at: Kria-Prophesee-Event-VitisAI/Kria-Ubuntu/event-cam-ml-app-Yolov4-tiny/build.

Note:
If you cloned the repository in the /home/ubuntu directory as mentioned in Step 4: Get the Ubuntu Firmware and Applications, then from the Docker console you can access the cloned repository at the /ubuntu path by running the "cd /ubuntu" command.
Run the command below from the user directory, which contains the cloned repository. In this case, it is the /ubuntu directory, which is mapped to the /home/ubuntu directory.

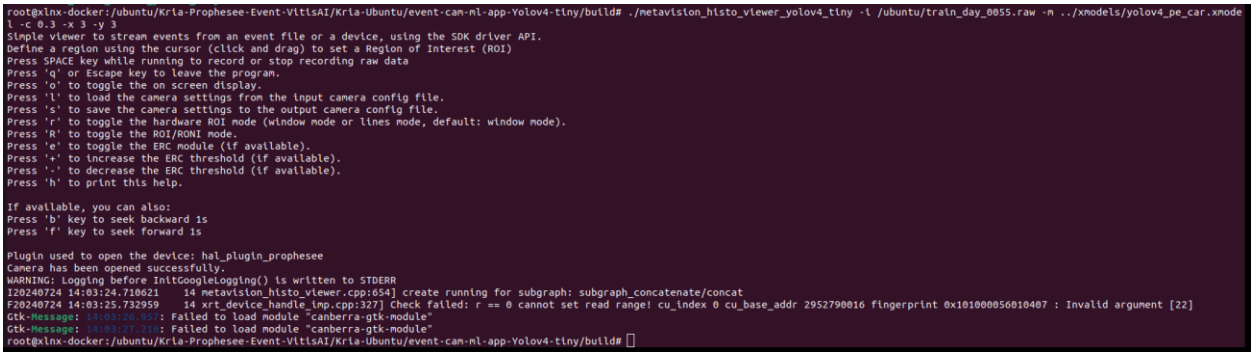
Here are the options for the metavision_histo_viewer_yolov4_tiny application:

- -i : input file
- -m : ML model path
- -c : confidence level threshold
- -x : channel 1 mean level
- -y : channel 2 mean level

```
# cd /ubuntu/Kria-Prophesee-Event-VitisAI/Kria-Ubuntu/event-cam-ml-app-Yolov4-tiny/build/

# ./metavision_histo_viewer_yolov4_tiny -i /ubuntu/train_day_0007.raw -m
../xmodels/yolov4_pe_car.xmodel -c 0.3 -x 3 -y 3
```

Here is the console output after running the above command:

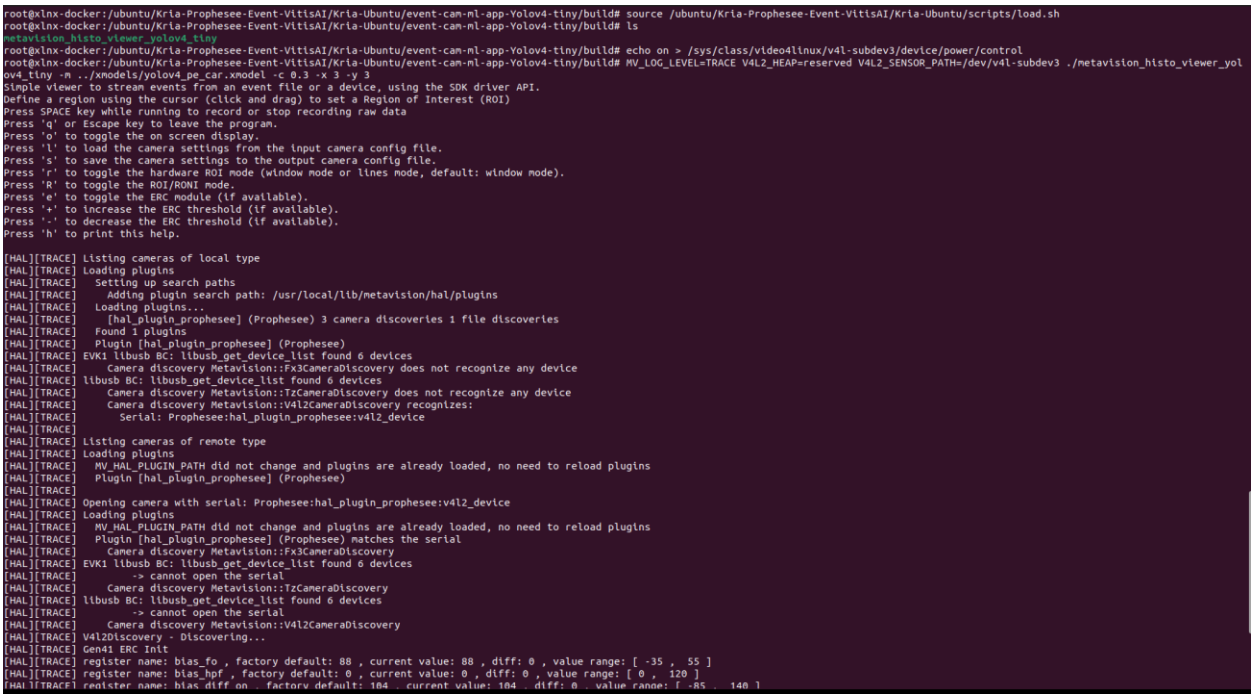


11. Run Event Camera-based YoloV4-tiny

If you are using camera input, change the build directory in the cloned repository as described in Step 10 and run the commands that follow. First, initialize the camera and then run the application:

```
# source /ubuntu/Kria-Prophesee-Event-VitisAI/Kria-Ubuntu/scripts/load.sh
# echo on > /sys/class/video4linux/v4l-subdev<X>/device/power/control
# MV_LOG_LEVEL=TRACE V4L2_HEAP=reserved  V4L2_SENSOR_PATH=/dev/v4l-
subdev<X> ./metavision_histo_viewer_yolov4 -m
../xmodels/yolov4_pe_car.xmodel -c 0.3 -x 3 -y 3
```

Here, the v4l-subdev<X> value replaces the <X> value according to the v4l2 device number for the imx636 device in the media graph obtained from the camera test step.



12. Run Event File-based YoloV7-tiny

If you are using file input, get the raw event file located in the /home/ubuntu directory that was downloaded in "Step 8: Get the Raw File for Test Input".

Next, run the event ML app **metavision_histo_viewer_yolov7_tiny** executable located at: Kria-Prophesee-Event-VitisAI/Kria-Ubuntu/event-cam-ml-app-Yolov7-tiny/build.

Note:
If you cloned the repository in the /home/ubuntu directory as mentioned in Step 4: Get the Ubuntu Firmware and Applications, then from the Docker console you can access the cloned repository at the /ubuntu path by running the "cd /ubuntu" command.

Here are the options for the metavision_histo_viewer_yolov7_tiny application:

- -i : input file
- -m : ML model path
- -c : confidence level threshold
- -x : channel 1 mean level
- -y : channel 2 mean level

```
# cd /ubuntu/Kria-Prophesee-Event-VitisAI/Kria-Ubuntu/event-cam-ml-app-Yolov7-tiny/build/
# ./metavision_histo_viewer_yolov7_tiny -i /ubuntu/train_day_0007.raw -m
../xmodel/Yolov7_tiny_320.xmodel -c 0.3 -x 3 -y 3
```

13. Run Event Camera-based YoloV7-tiny

If you are using camera input, run the following commands to first initialize the camera and then run the application:

```
# cd /ubuntu/Kria-Prophesee-Event-VitisAI/Kria-Ubuntu/event-cam-ml-app-Yolov7-tiny/build
# source /ubuntu/Kria-Prophesee-Event-VitisAI/Kria-Ubuntu/scripts/load.sh
# echo on > /sys/class/video4linux/v4l-subdev<X>/device/power/control
# MV_LOG_LEVEL=TRACE V4L2_HEAP=reserved  V4L2_SENSOR_PATH=/dev/v4l-subdev<X>
./metavision_histo_viewer_yolov7_tiny -m ../xmodel/Yolov7_tiny_320.xmodel -c 0.3 -x 3 -y 3
```

Here, the v4l-subdev<X> value replaces the <X> value according to the v4l2 device number for the imx636 device in the media graph obtained from the camera test step.

14. Run Event File-based YoloV7

If you are using file input, get the raw event file located in the /home/ubuntu directory that was downloaded in "Step 8: Get the Raw File for Test Input".

Next, run the event ML app **metavision_histo_viewer_yolov7** executable located at: Kria-Prophesee-Event-VitisAI/Kria-Ubuntu/event-cam-ml-app-Yolov7/build.

Note:
If you cloned the repository in the /home/ubuntu directory as mentioned in Step 4: Get the Ubuntu Firmware and Applications, then from the Docker console you can access the cloned repository at the /ubuntu path by running the "cd /ubuntu" command.

Here are the options for the metavision_histo_viewer_yolov7_tiny application:

- -i : input file
- -m : ML model path
- -c : confidence level threshold
- -x : channel 1 mean level
- -y : channel 2 mean level

```
# cd /ubuntu/Kria-Prophesee-Event-VitisAI/Kria-Ubuntu/event-cam-ml-app-Yolov7/build/
# ./metavision_histo_viewer_yolov7 -i /ubuntu/train_day_0007.raw -m
../xmodel/Yolov7_320.xmodel -c 0.3 -x 3 -y 3
```

15. Run Event Camera-based YoloV7-tiny

If you are using camera input, run the following commands to first initialize the camera and then run the application:

```
# cd /ubuntu/Kria-Prophesee-Event-VitisAI/Kria-Ubuntu/event-cam-ml-app-Yolov7/build
# source /ubuntu/Kria-Prophesee-Event-VitisAI/Kria-Ubuntu/scripts/load.sh
# echo on > /sys/class/video4linux/v4l-subdev<X>/device/power/control
# MV_LOG_LEVEL=TRACE V4L2_HEAP=reserved  V4L2_SENSOR_PATH=/dev/v4l-subdev<X>
./metavision_histo_viewer_yolov7 -m ../xmodel/Yolov7_320.xmodel -c 0.3 -x 3 -y 3
```

Here, the v4l-subdev<X> value replaces the <X> value according to the v4l2 device number for the imx636 device in the media graph obtained from the camera test step.

16. Run Yolov7 with a USB Webcam (RGB Input) - Optional

Please follow the instructions at this link to test the Yolov7 app with a USB webcam.

17. Stop the Docker Container and Application

Run the **`exit`** command in the Docker shell console to close and stop the Docker container. You can also shut down the board by running the **`sudo shutdown -h now`** command in the terminal.

18. Known Issues and Solutions

a. When testing camera input, objects in motion should be kept at least 1 meter away from the camera. If an object in motion is too close to the camera, the application stops or starts with no display.
b. When running an application with camera input, if there is no display, then rerun the application after a few seconds.
c. When the camera is running, avoid touching it with bare hands as the camera can be prone to static electricity and may stop running.
d. The following warning can be ignored when running the application using a DPU:

```
WARNING: Logging before InitGoogleLogging() is written to STDERR
F20240726 13:09:26.295787    18 xrt_device_handle_imp.cpp:327] Check failed: r == 0 cannot set read
range! cu_index 0 cu_base_addr 2952790016 fingerprint 0x101000056010407 : Invalid argument [22]
device_core_id=0 device= 0 core = 0 fingerprint = 0x101000056010407 batch = 1
full_cu_name=DPUCZDX8G:DPUCZDX8G_1
```

***

Document Revision History

Date	Version	Revision	Author
14/08/2024	1.3	Updated the overview	Krishna Gaihre
29/07/2024	1.2	Updated for public release	Sanam Shakya, Krishna Gaihre
08/07/2024	1.0	Updated for closed release	Sanam Shakya