Downloading and Loading Kria-Prophesee-Event-VitisAI Application Firmware in KV260 Board

Getting Started Guide

1. Overview of Kria-App

This Kria-App consists of setup steps for running the event camera based Machine learning inference in Kria KV260 Board. Here we are taking an event camera from Prophesee and also performing recorded event file based ML inference.

This Kria-App has Kria-Ubuntu and Kria-Petalinux both OS support.

2. Prerequisites

Following are the prerequisites of this Kria-App:

- KV260 board 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)
- Mouse and Keyboard
- SD card (32 GB)

3. Setup Kria Ubuntu SD card for KV260:

Get the Kria Ubuntu 22.04 from - Link
Create SD card using balena-etcher tool and prepare the KV260 for SD card. Connect accessories (keyboard, mouse, mipi camera* and ethernet) and monitor with KV260 board. Now, boot the SD card and log into Ubuntu Desktop environment of Kria KV260 board. Then run terminal in the Ubuntu Home folder. **Then run the following steps in the Ubuntu Desktop terminal with keyboard and mouse attached to Kria KV260 board.**

* - Camera is optional if using event file based ML

4. Getting the Kria Ubuntu Firmwares and applications

Get the Kria Ubuntu firmware and necessary installation scripts from following repository in the Kria ubuntu home directory: 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 following scripts to update Ubuntu Kernel followed by Prophesee driver installation.

5. Update Kria Ubuntu kernel:

Run scripts to update the Ubuntu kernel:

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

6. Setup Prophesee camera drivers :

Run following script to install Prophesee camera drivers:

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

7. Setup hardware firmware overlay and vart config file

Run following script to install hardware overlays:

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

8. Get Raw file for test input:

For testing file based input in ML application a raw event file is needed. Get a sample raw file running following command in **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. Docker-Based Application Preparation

First install the docker following the instruction in Booting the Kria Starter Kit Linux in step 6.

Pull the docker image:

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

It takes around 10-15 minutes to download the docker image, which depends upon internet speed.

View the available images using 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:~$
```

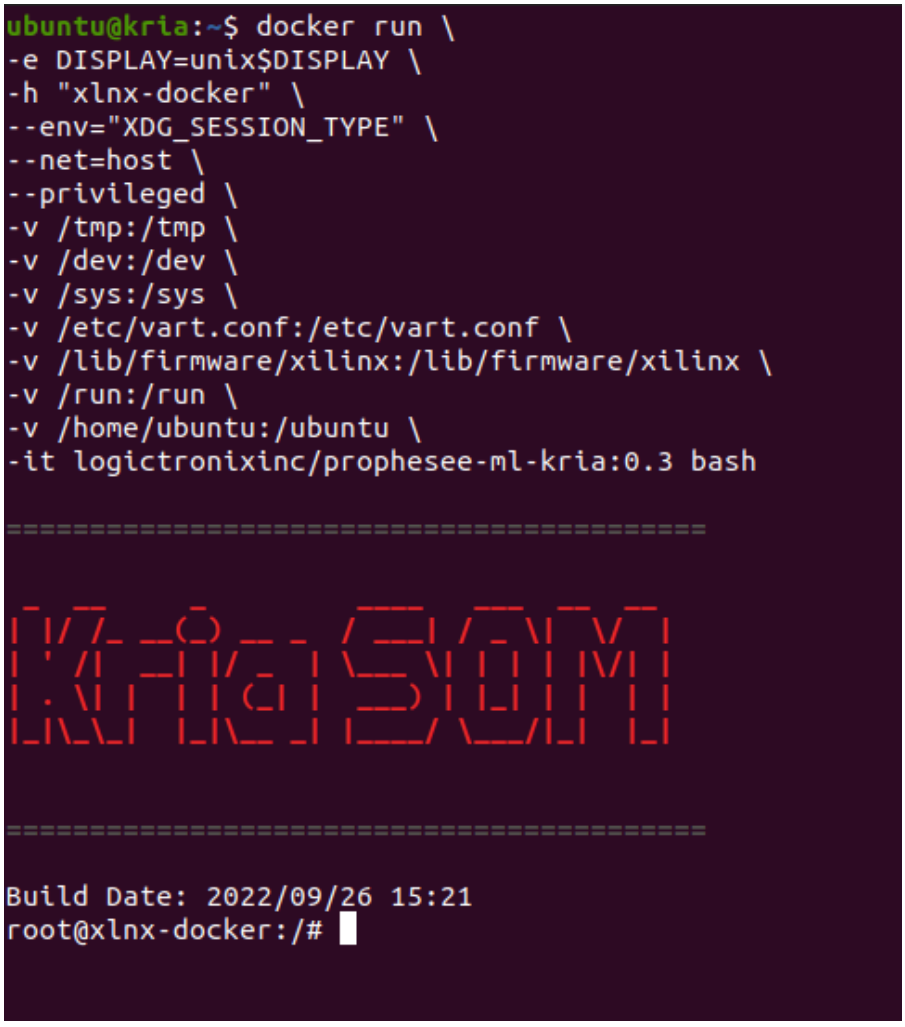
For GUI to get the output from application running in docker container, run following command:

```
xhost local:docker
```

Then run the docker run command as below or run **docker_run.sh** file present in 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 docker shell terminal as shown below:



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

Setup the environment variable for running Vitis AI based application in docker:
In docker terminal run the following command to setup the environment variables to run 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 application:

- Camera Test (**Optional** if Prophesee CCAM5 Hardware is connected to board)
  - Run following command in console to get the v4l2 media graph and sensor device number:

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

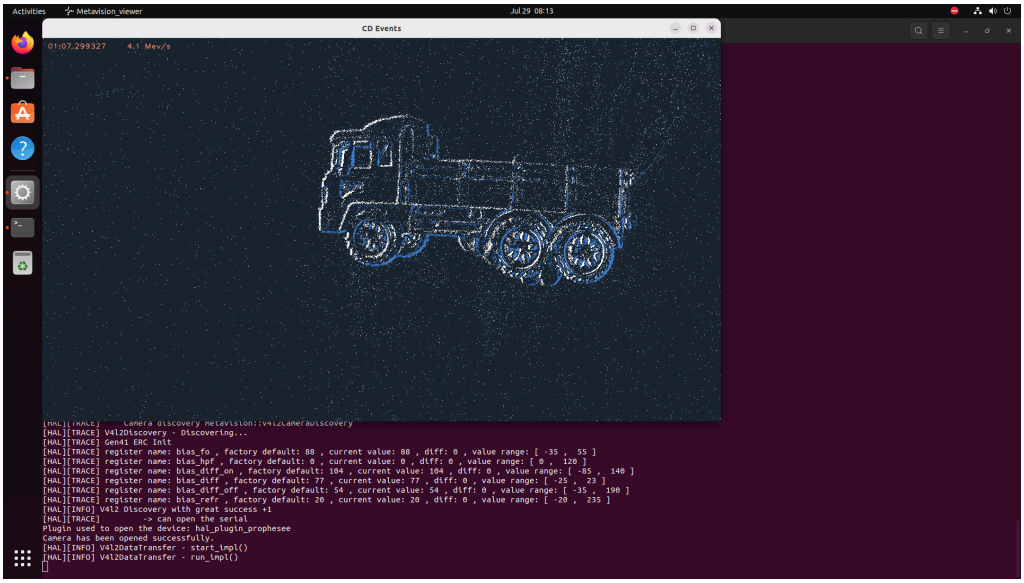
```
----------------------
driver          psee-video
model           Prophesee Video Pipeline
serial
bus info
hw revision     0x0
driver version  5.15.136

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 above media graph imx636 device node is **/dev/v4l-subdev1**, which is required to enable and running the application.

Before running the event camera ML applications, one can check if the camera data pipeline is working or not, by running the default ‘metavision-viewer’ application. For running 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 in v4l-subdev<X> value replaces <X> value according to the v4l2 device number for imx636 device in the media graph obtained in camera test step.
This will display the camera output as shown below:



If camera data pipeline is stuck or not working than above command with throw blank output. In this scenario follow restart the application after a while or review **Known Issues**.

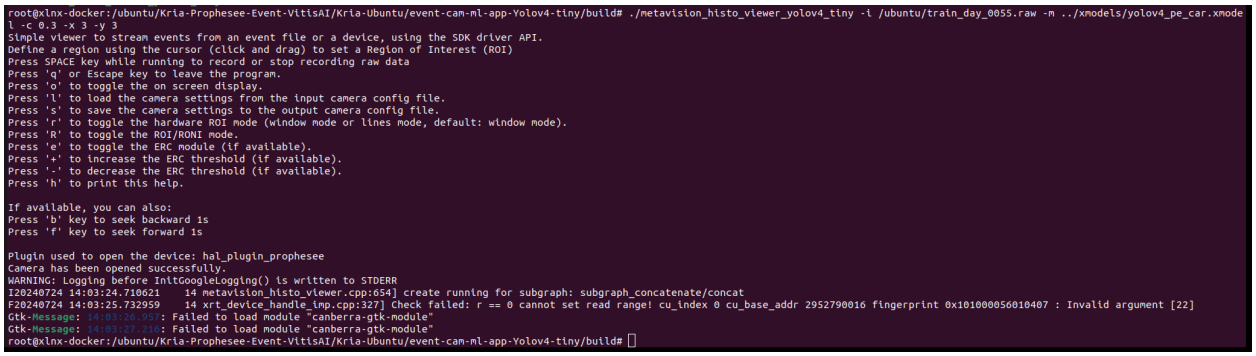
10. Run Event File based YoloV4-tiny

For file input, get the raw event file in home/ubuntu directory downloaded in "Step 8: Get Raw file for test input step".
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 have cloned the repository at /home/ubuntu directory as mentioned in Step 4 : Getting the Kria Ubuntu Firmwares and applications. Then in docker console one can access the cloned repository at /ubuntu path, which you can reach by running "cd /ubuntu" command. Run the command below from the user directory, which contains the cloned repository in this case it is /ubuntu directory, which is mapped to /home/ubuntu directory.

Here are the options for 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 for running above command:



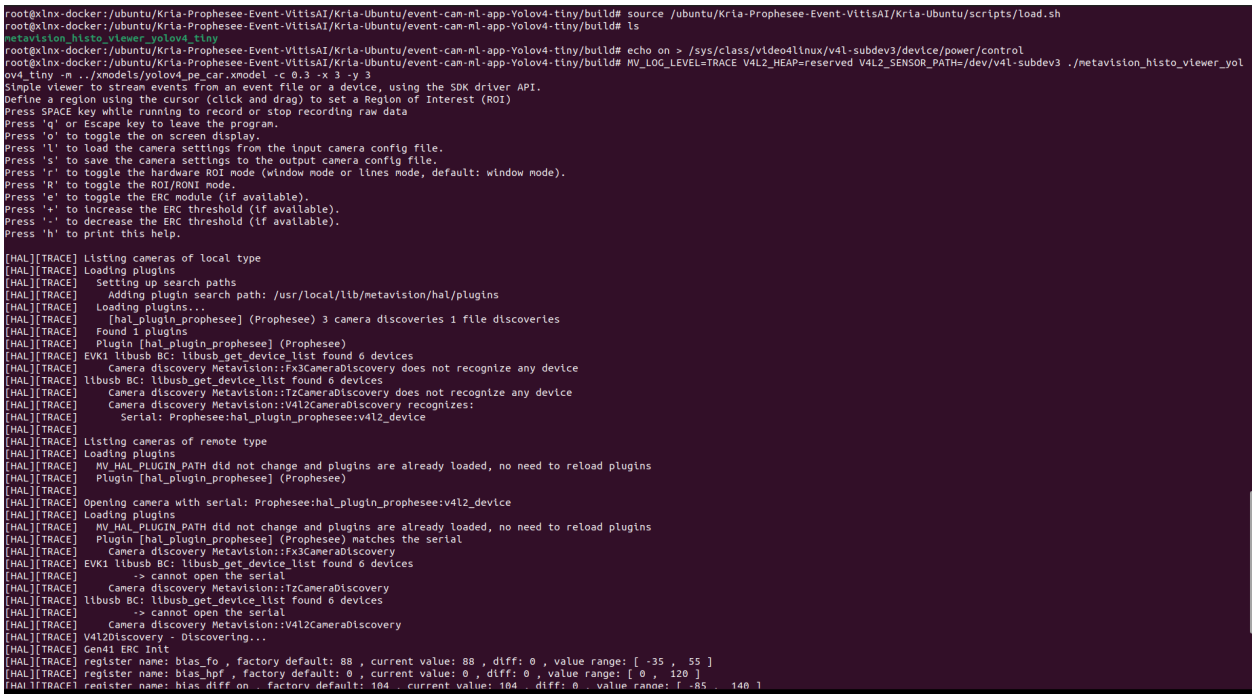
11. Run Event camera based YoloV4-tiny

For camera input, change the directory build directory in the cloned repository as in Step 10 and run following commands. First initialize the camera and run the application as below:

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



12. Run Event File based YoloV7-tiny

For file input, get the raw event file in home/ubuntu directory downloaded in "Step 8: Get Raw file for test input step".
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 have cloned the repository at /home/ubuntu directory as mentioned in Step 4 : Getting the Kria Ubuntu Firmwares and applications. Then in docker console one can access the cloned repository at /ubuntu path, which you can reach by running "cd /ubuntu" command.

Here are the options for 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

For camera input, run following commands to first initialize the camera and run the application as below:

```
# 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 in v4l-subdev<X> value replace <X> value according to the v4l2 device number for imx636 device in the media graph obtained in camera test step.

14. Run Event File based YoloV7

For file input, get the raw event file in home/ubuntu directory downloaded in "Step 8: Get Raw file for test input step".
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 have cloned the repository at /home/ubuntu directory as mentioned in Step 4 : Getting the Kria Ubuntu Firmwares and applications. Then in docker console one can access the cloned repository at /ubuntu path, which you can reach by running "cd /ubuntu" command.

Here are the options for 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

For camera input, run following commands to first initialize the camera and run the application as below:

```
# 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 in v4l-subdev<X> value replace <X> value according to the v4l2 device number for imx636 device in the media graph obtained in camera test step.

16. Running Yolov7 with USB webcam (RGB input) - Optional

Please follow the instructions at the following link to test Yolov7 app in USB webcam.

17. Stopping Docker container and application

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

18. Known Issues and Solutions

a. For testing camera input, objects in motion should be kept at least 1 meter away from the camera. If an object in motion is near to the camera, the application stops or starts with no display.
b. While running an application with camera input, if there is no display then rerun the application after a few seconds.
c. Also the camera is prone to static electricity and stops running if touched with bare hands. So when the camera is running don't touch it with bare hands.
d. Following warning is printed while running application using DPU. This warning can be ignored:

```
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
29/07/2024	1.2	Updated for public release	Sanam Shakya, Krishna Gaihre
08/07/2024	1.0	Updated for closed release	Sanam Shakya