# 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-ethcer 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.

* - 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

```
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:
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:
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:
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
```

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 or run docker_run.sh file 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 \
--volume="/home/ubuntu/.Xauthority:/root/dot.Xauthority:rw" \
-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 shell terminal of docker container.

```
ubuntu@kria:~$ docker run \
-e DISPLAY=unix$DISPLAY \
-h "xlnx-docker" \
--env="XDG_SESSION_TYPE" \
--net=host \
--privileged \
--volume="/home/ubuntu/.Xauthority:/root/dot.Xauthority:rw" \
-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:/#
root@xlnx-docker:/#
```

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:

```
cd /ubuntu
source Kria-Prophesee-Event-VitisAI/Kria-Ubuntu/scripts/setup_dpu_env.sh
```

Things to check before running application:

- Camera Test (**Optional** if Prophesee CCAM5 Hardware is connected to board)
    - `**media-ctl -p**` to get the v4l2 media graph and sensor device number

```
----------------------
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.

## 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 executable located at :
Kria-Prophesee-Event-VitisAI/Kria-Ubuntu/event-cam-ml-app-Yolov4-tiny/build/metavision_histo_viewer_yolov4_tiny.
If you have cloned the repository at /home/ubuntu directory as mentioned in Step 4 : Getting the Kria Ubuntu Firmwares and applications. Than in docker console one can access clones repository at /ubuntu path, which you can reach by running "cd /ubuntu" command.
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
Kria-Prophesee-Event-VitisAI/Kria-Ubuntu/event-cam-ml-app-Yolov4/build/
# ./metavision_histo_viewer_yolov4_tiny -i /ubuntu/train_day_0055.raw -m
../xmodels/yolov4_pe_car.xmodel -c 0.3 -x 3 -y 3
```

Here is the console output for running above command:

```
root@xlnx-docker:/ubuntu# ./metavision_histo_viewer -i train_day_0055.raw -m yolov4_tiny_jun30_person_vehicle.xmodel -c 0.3 -x 3 -y 3
Simple viewer to stream events from an event file or a device, using the SDK driver API.
Define a region using the cursor (click and drag) to set a Region of Interest (ROI)
Press SPACE key while running to record or stop recording raw data
Press 'q' or Escape key to leave the program.
Press 'o' to toggle the on screen display.
Press 'l' to load the camera settings from the input camera config file.
Press 's' to save the camera settings to the output camera config file.
Press 'r' to toggle the hardware ROI mode (window mode or lines mode, default: window mode).
Press 'R' to toggle the ROI/RONI mode.
Press 'e' to toggle the ERC module (if available).
Press '+' to increase the ERC threshold (if available).
Press '-' to decrease the ERC threshold (if available).
Press 'h' to print this help.

If available, you can also:
Press 'b' key to seek backward 1s
Press 'f' key to seek forward 1s

Plugin used to open the device: hal_plugin_prophesee
Camera has been opened successfully.
```

## 11. Run Event camera based YoloV4-tiny

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

```
# source 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 replace <X> value according to the v4l2 device number for imx636 device in 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 executable located at :
Kria-Prophesee-Event-VitisAI/Kria-Ubuntu/event-cam-ml-app-Yolov7-tiny/build/metavision_histo_viewer_yolov7_tiny

If you have cloned the repository at /home/ubuntu directory as mentioned in Step 4 : Getting the Kria Ubuntu Firmwares and applications. Than in docker console one can access clones 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
Kria-Prophesee-Event-VitisAI/Kria-Ubuntu/event-cam-ml-app-Yolov7-tiny/build
/
# ./metavision_histo_viewer_yolov7_tiny -i /ubuntu/train_day_0055.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
Kria-Prophesee-Event-VitisAI/Kria-Ubuntu/event-cam-ml-app-Yolov7-tiny/build
# source 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 ../xmodels/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 media graph obtained in camera test step.

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

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

## 15. Stopping Docker container and application

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

## 16. Known Issues and Solutions

a. For testing camera input, object in motion should be kept at least 1 meter away from camera. If object in motion is near to the camera, application stops or start with no display.
b. While running application with camera input, if there is no display then rerun the application after few seconds.
c. Also camera is prone to static electricity and stops running if touched with bare hands. So when camera is running don't touch with bare hands.

***

## Document Revision History

| Date | Version | Revision |
|---|---|---|
| 08/07/2024 | 1.0 | Updated for closed release |
| | | |
| | | |