

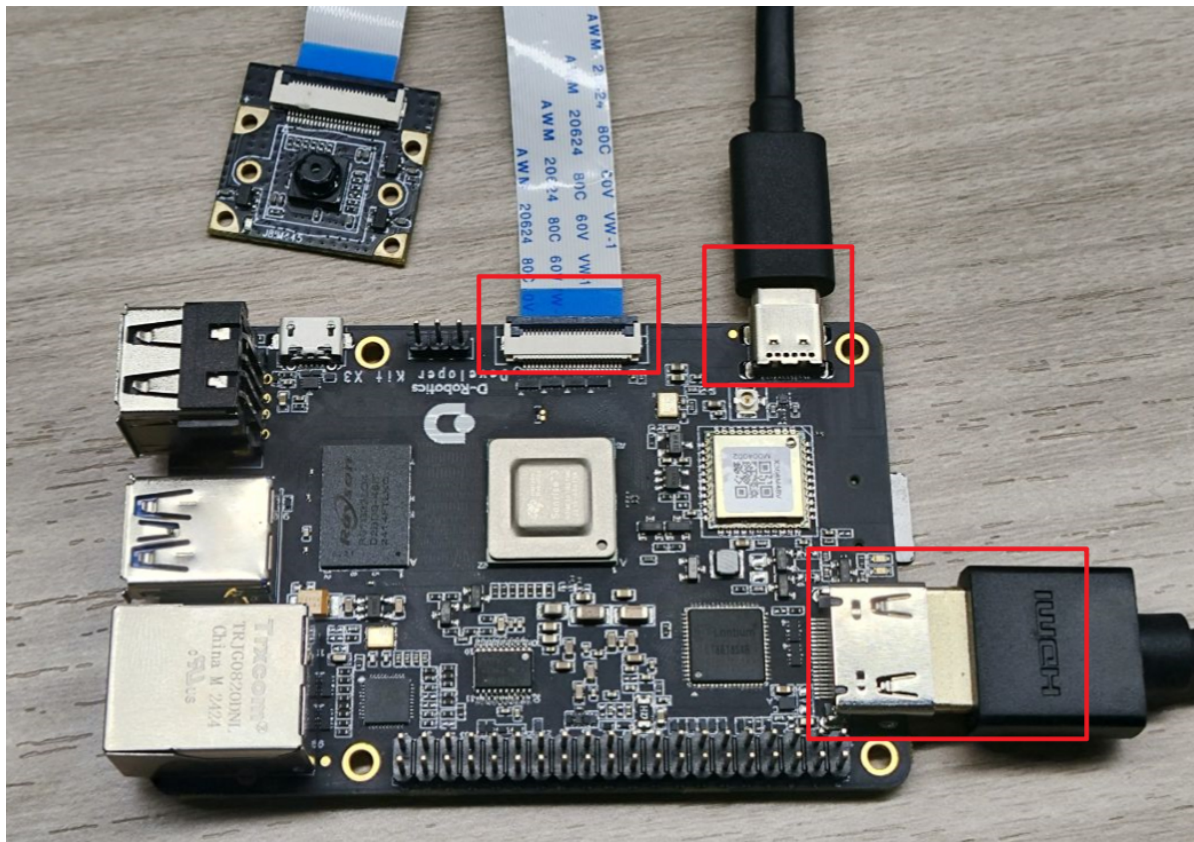
# Object detection algorithm—YOLOv5

## Object detection algorithm—YOLOv5

1. Preparation
2. Running method
3. Result

## 1. Preparation

This example implements the camera target detection algorithm based on the YOLOv5 model. Users can preview the detection results on the display.



- When the development board is powered off, connect the MIPI camera to the development board
- Connect the development board and the monitor via an HDMI cable
- Power on the development board and log in via the command line

## 2. Running method

The sample code is provided in source code form and needs to be compiled and run using the `make` command.

The steps are as follows.

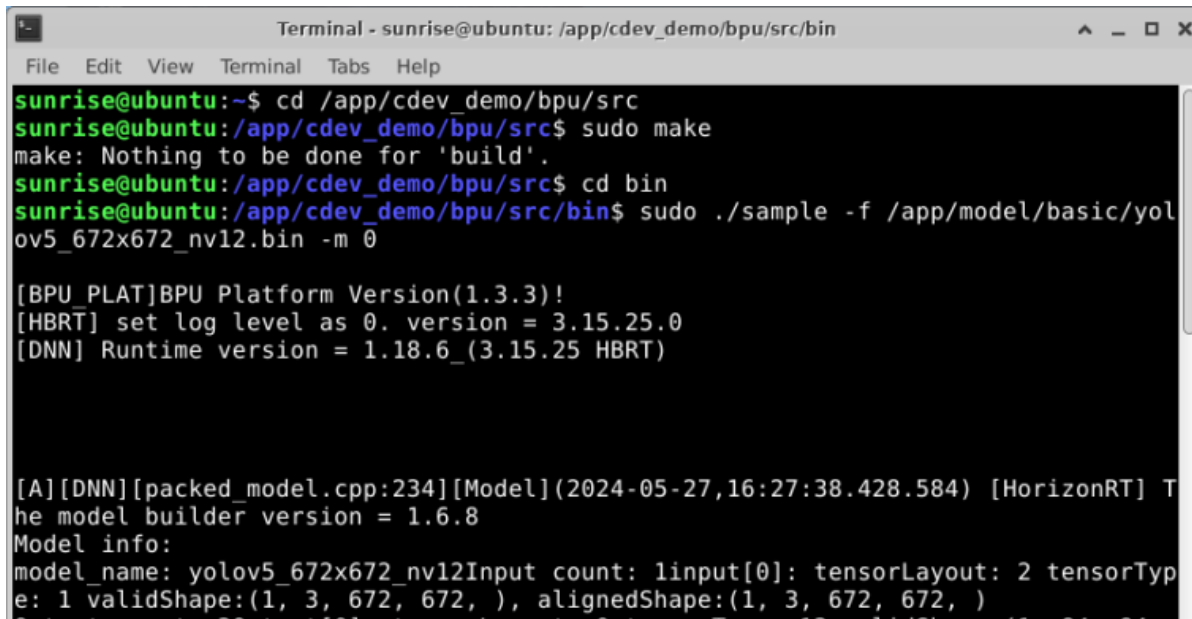
```
sunrise@ubuntu:~$ cd /app/cdev_demo/bpu/src
sunrise@ubuntu:/app/cdev_demo/bpu/src$ sudo make
sunrise@ubuntu:/app/cdev_demo/bpu/src$ cd bin
sunrise@ubuntu:/app/cdev_demo/bpu/src/bin$ sudo ./sample -f
/app/model/basic/yolov5_672x672_nv12.bin -m 0
```

#### Parameter configuration:

- -f: model path
- -m: model type, default is 0

## 3. Result

After the program runs correctly

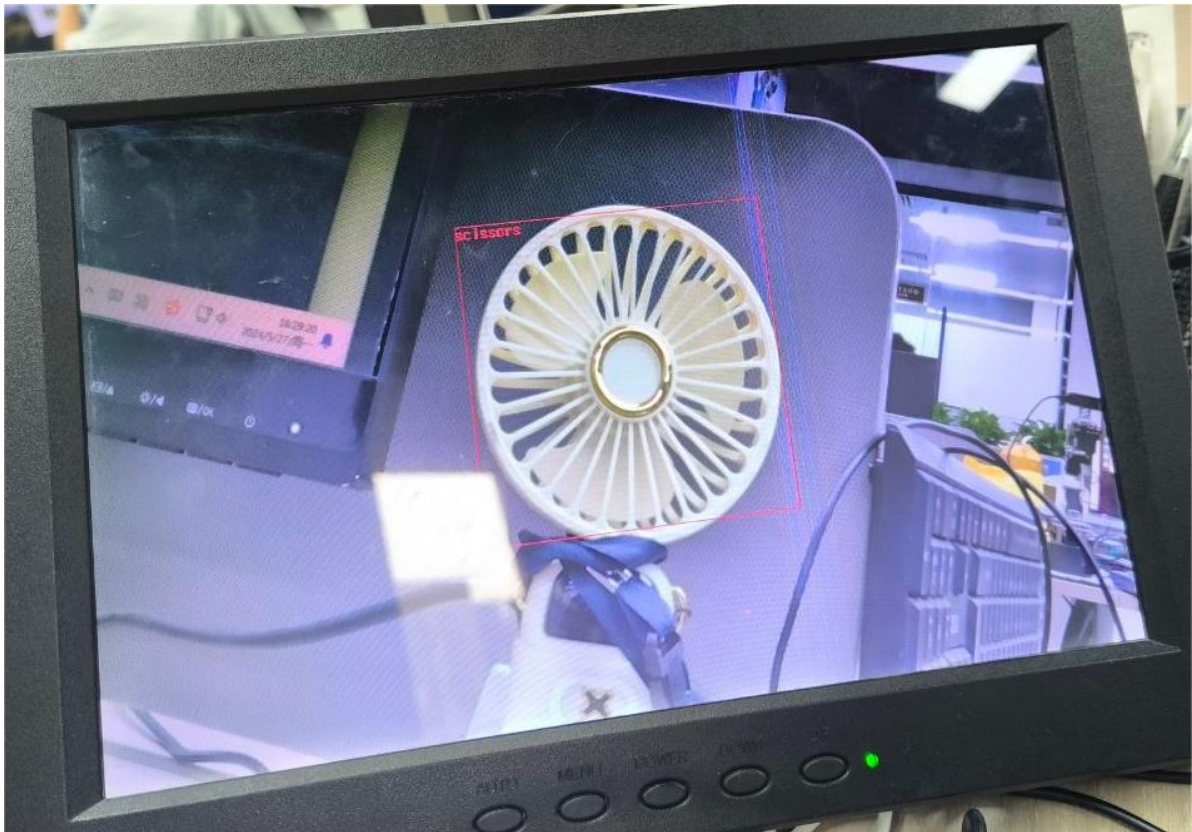
A terminal window titled "Terminal - sunrise@ubuntu: /app/cdev\_demo/bpu/src/bin" showing the execution of the sample program. The user navigates to the src directory, runs 'make', then 'cd bin', and finally runs './sample -f /app/model/basic/yolov5\_672x672\_nv12.bin -m 0'. The output shows version information for BPU Platform (1.3.3), HBRT (3.15.25.0), and DNN (1.18.6). It also displays model information: model\_name: yolov5\_672x672\_nv12, input count: 1, tensor layout: 2, tensor type: 1, valid shape: (1, 3, 672, 672), and aligned shape: (1, 3, 672, 672).

```
sunrise@ubuntu:~$ cd /app/cdev_demo/bpu/src
sunrise@ubuntu:/app/cdev_demo/bpu/src$ sudo make
make: Nothing to be done for 'build'.
sunrise@ubuntu:/app/cdev_demo/bpu/src$ cd bin
sunrise@ubuntu:/app/cdev_demo/bpu/src/bin$ sudo ./sample -f /app/model/basic/yolov5_672x672_nv12.bin -m 0

[BPU_PLAT]BPU Platform Version(1.3.3)!
[HBRT] set log level as 0. version = 3.15.25.0
[DNN] Runtime version = 1.18.6_(3.15.25 HBRT)

[A][DNN][packed_model.cpp:234][Model](2024-05-27,16:27:38.428.584) [HorizonRT] The model builder version = 1.6.8
Model info:
model_name: yolov5_672x672_nv12Input count: 1input[0]: tensorLayout: 2 tensorType: 1 validShape:(1, 3, 672, 672, ), alignedShape:(1, 3, 672, 672, )
```

The video and the image rendered by the algorithm detection will be output through the `HDMI` interface, and the user can preview it through the monitor.



The running log is as follows.

```
sunrise@ubuntu:/app/cdev_demo/bpu/src/bin$ sudo ./sample -f
/app/model/basic/yolov5_672x672_nv12.bin -m 0
[BPU_PLAT]BPU Platform Version(1.3.1)!
[HBRT] set log level as 0. version = 3.14.5
[DNN] Runtime version = 1.9.7_(3.14.5 HBRT)
Model info:
model_name: yolov5_672x672_nv12Input count: 1input[0]: tensorLayout: 2
tensorType: 1 validShape:(1, 3, 672, 672, ), alignedShape:(1, 3, 672, 672, )
Output count: 3Output[0]: tensorLayout: 0 tensorType: 13 validShape:(1, 84, 84,
255, ), alignedShape:(1, 84, 84, 255, )
Output[1]: tensorLayout: 0 tensorType: 13 validShape:(1, 42, 42, 255, ),
alignedShape:(1, 42, 42, 255, )
Output[2]: tensorLayout: 0 tensorType: 13 validShape:(1, 21, 21, 255, ),
alignedShape:(1, 21, 21, 255, )
Setting VPS channel-1: src_w:1920, src_h:1080; dst_w:672, dst_h:672;
Setting VPS channel-3: src_w:1920, src_h:1080; dst_w:1024, dst_h:600;
Setting VPS channel-2: src_w:1920, src_h:1080; dst_w:1920, dst_h:1080;
start linear mode, sensor_name f37, setting_size = 3
libiar: hb_disp_set_timing done!
yolov5_do_post fps:11.627907,processing time :86
```