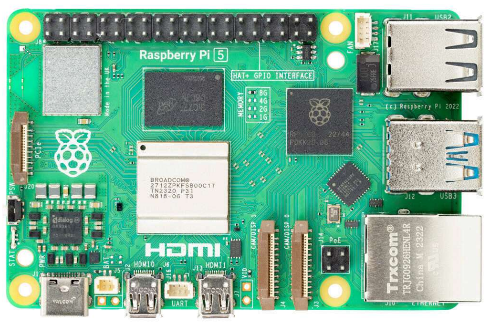
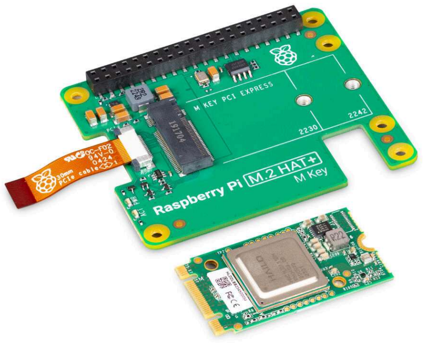
Tutorial of AI Kit with Raspberry Pi 5 about YOLOv8n object detection

Introduction

YOLOv8 (You Only Look Once version 8) is the popular most YOLO series of real-time pose estimation and object detection models. It builds upon the strengths of its predecessors by introducing several advancements in speed, accuracy, and flexibility. The Raspberry-pi-AI-kit is used to accelerate inference speed, featuring a 13 tera-operations per second (TOPS) neural network inference accelerator built around the Hailo-8L chip.

This wiki will guide you on how to use YOLOv8n for object detection with AI Kit on Raspberry Pi 5, from training to deployment.

Prepare Hardware

Raspberry Pi5 8GB	Raspberry Pi AI Kit
	

Get One Now	Get One Now

Install Hardware

Please refer to this

On Host Computer

NOTE

We will install hailo software, make sure you have a hailo account.

Install Ultralytics and train model:

Install python3.11

```
sudo apt install python3.11
```

Create yolo_env as your virtual environment

```
python3.11 -m venv yolo_env
```

Activate the environment

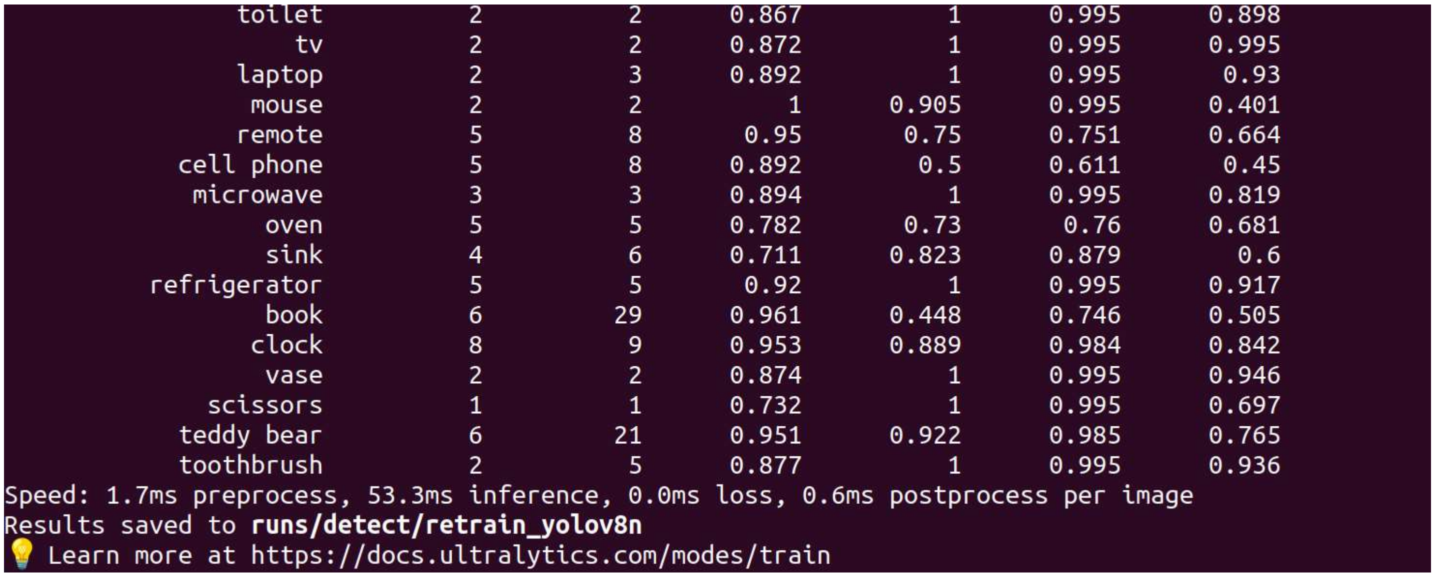
```
source yolo_env/bin/activate
```

Install ultralytics

```
pip install ultralytics
```

Train YOLOv8n using the COCO dataset. If you want to train your own dataset, you can refer to this for instructions on how to do so.

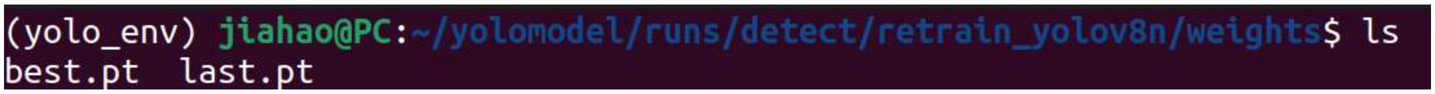
```
mkdir yolomodel && cd yolomodel
yolo detect train data=coco128.yaml model=yolov8n.pt name=retrain_yolov8n epochs=100
batch=16
```



```
                  toilet          2          2      0.867          1      0.995      0.898
                      tv          2          2      0.872          1      0.995      0.995
                  laptop          2          3      0.892          1      0.995       0.93
                   mouse          2          2          1      0.905      0.995      0.401
                  remote          5          8       0.95       0.75      0.751      0.664
              cell phone          5          8      0.892        0.5      0.611       0.45
               microwave          3          3      0.894          1      0.995      0.819
                    oven          5          5      0.782       0.73       0.76      0.681
                    sink          4          6      0.711      0.823      0.879        0.6
            refrigerator          5          5       0.92          1      0.995      0.917
                    book          6         29      0.961      0.448      0.746      0.505
                   clock          8          9      0.953      0.889      0.984      0.842
                    vase          2          2      0.874          1      0.995      0.946
                scissors          1          1      0.732          1      0.995      0.697
              teddy bear          6         21      0.951      0.922      0.985      0.765
              toothbrush          2          5      0.877          1      0.995      0.936
Speed: 1.7ms preprocess, 53.3ms inference, 0.0ms loss, 0.6ms postprocess per image
Results saved to runs/detect/retrain_yolov8n
💡 Learn more at https://docs.ultralytics.com/modes/train
```

You will get the `best.pt` model after your training, as shown below:

```
cd ./runs/detect/retrain_yolov8n/weights/
ls
```

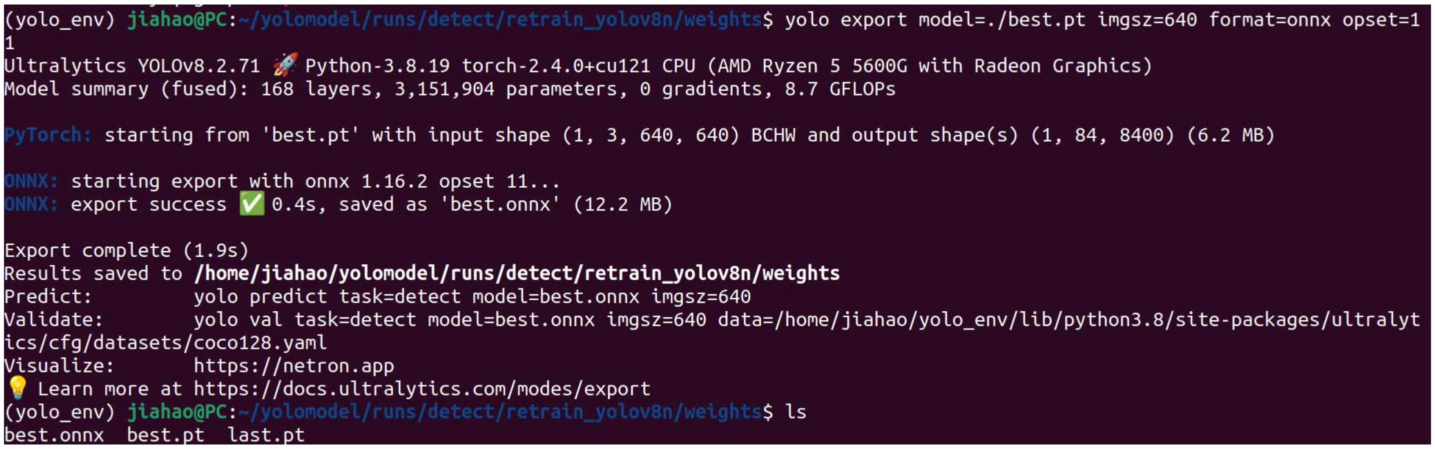


```
(yolo_env) jiahao@PC:~/yolomodel/runs/detect/retrain_yolov8n/weights$ ls
best.pt  last.pt
```

Convert the `.pt` model to `.onnx`.

```
yolo export model=./best.pt imgsz=640 format=onnx opset=11
```

Result like below:



```
(yolo_env) jiahao@PC:~/yolomodel/runs/detect/retrain_yolov8n/weights$ yolo export model=./best.pt imgsz=640 format=onnx opset=1
1
Ultralytics YOLOv8.2.71 🚀 Python-3.8.19 torch-2.4.0+cu121 CPU (AMD Ryzen 5 5600G with Radeon Graphics)
Model summary (fused): 168 layers, 3,151,904 parameters, 0 gradients, 8.7 GFLOPs

PyTorch: starting from 'best.pt' with input shape (1, 3, 640, 640) BCHW and output shape(s) (1, 84, 8400) (6.2 MB)

ONNX: starting export with onnx 1.16.2 opset 11...
ONNX: export success ✅ 0.4s, saved as 'best.onnx' (12.2 MB)

Export complete (1.9s)
Results saved to /home/jiahao/yolomodel/runs/detect/retrain_yolov8n/weights
Predict:         yolo predict task=detect model=best.onnx imgsz=640
Validate:        yolo val task=detect model=best.onnx imgsz=640 data=/home/jiahao/yolo_env/lib/python3.8/site-packages/ultralyt
ics/cfg/datasets/coco128.yaml
Visualize:       https://netron.app
💡 Learn more at https://docs.ultralytics.com/modes/export
(yolo_env) jiahao@PC:~/yolomodel/runs/detect/retrain_yolov8n/weights$ ls
best.onnx  best.pt  last.pt
```

Install hailo software:

Install python 3.8

```
cd ~
sudo apt install python3.8
```

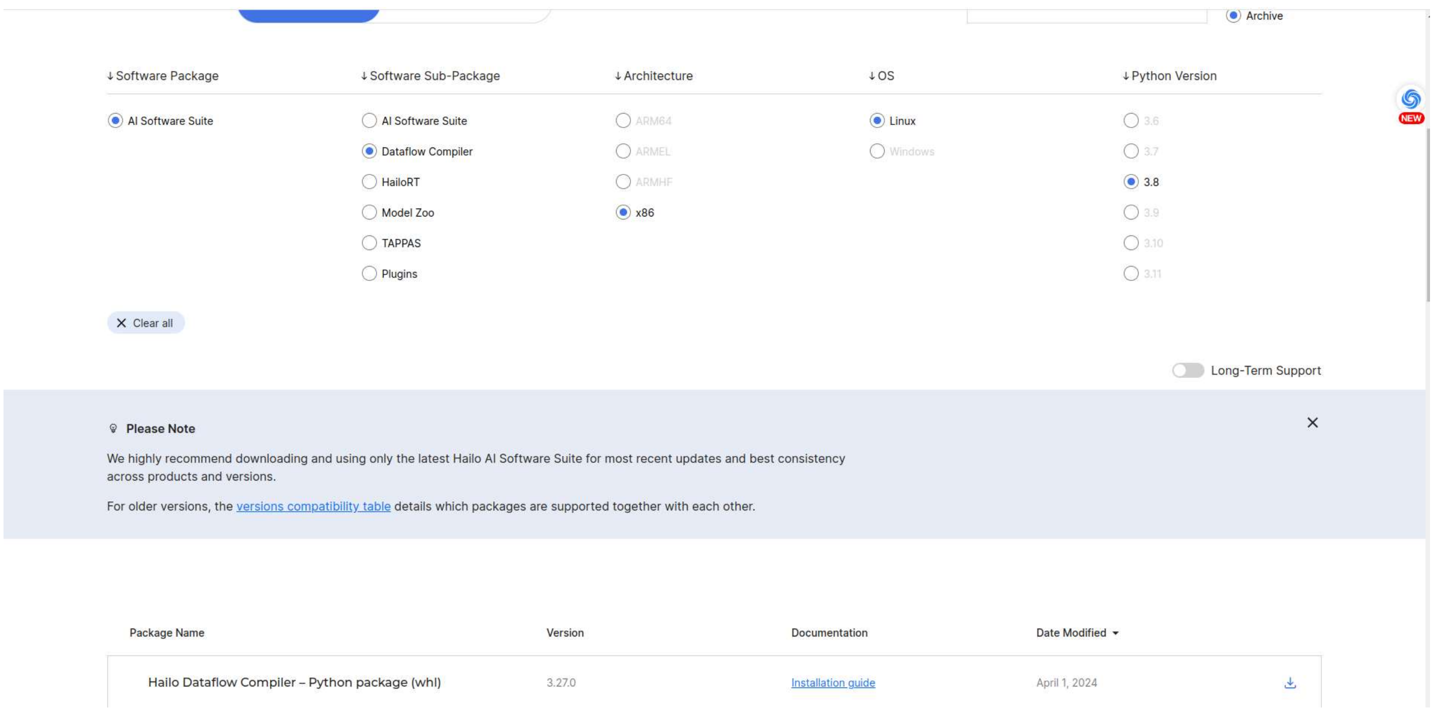
Creat hailo_env as your virtual environment

```
python3.8 -m venv hailo_env
```

Activate the environment

```
source hailo_env/bin/activate
```

Install Hailo Dataflow Compiler 3.27, here you need to register Hailo and login, and download the software.


[✓ Archive]

↓Software Package	↓Software Sub-Package	↓Architecture	↓OS	↓Python Version
[✓ AI Software Suite]	AI Software Suite	ARM64	[✓ Linux]	3.6
	[✓ Dataflow Compiler]	ARMEL	Windows	3.7
	HailoRT	ARMHF		[✓ 3.8]
	Model Zoo	[✓ x86]		3.9
	TAPPAS			3.10
	Plugins			3.11

Clear all

Long-Term Support

Please Note

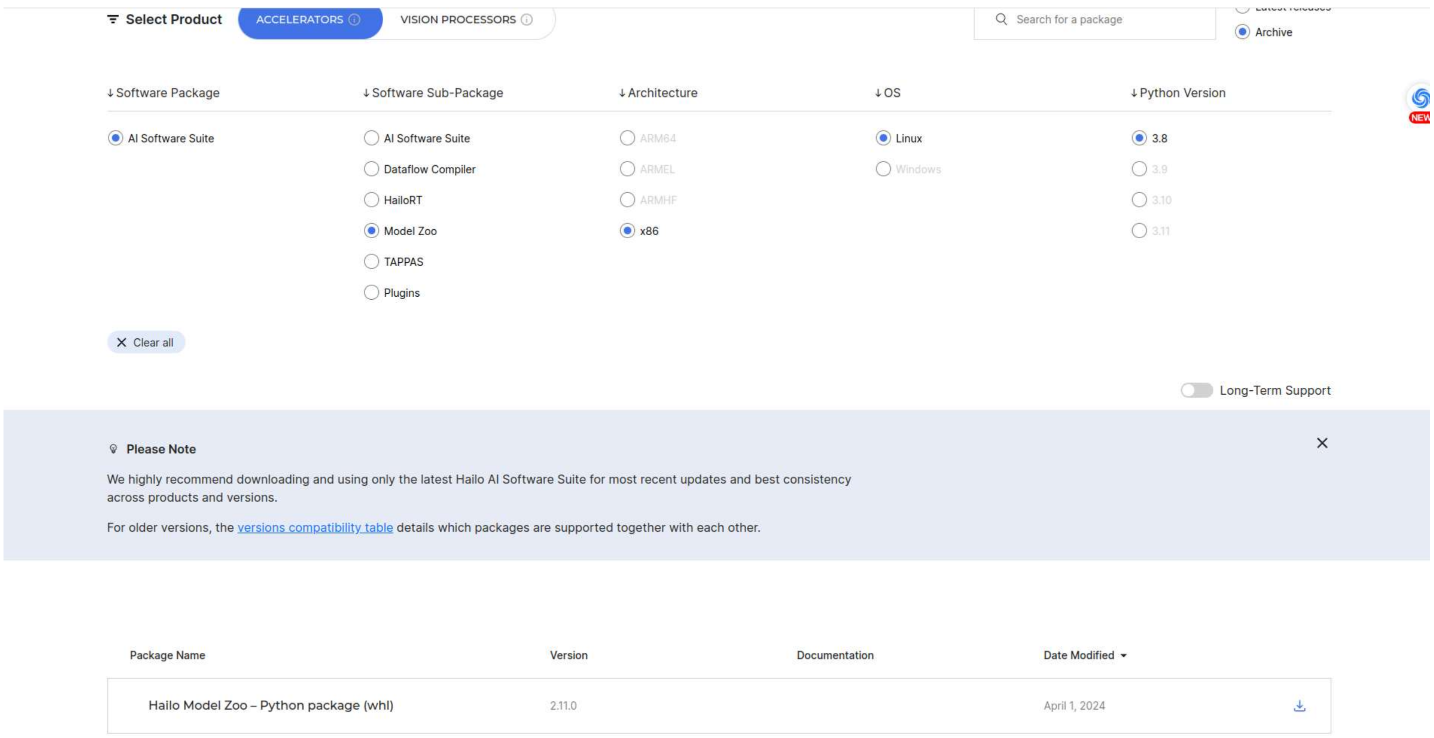
We highly recommend downloading and using only the latest Hailo AI Software Suite for most recent updates and best consistency across products and versions.

For older versions, the versions compatibility table details which packages are supported together with each other.

Package Name	Version	Documentation	Date Modified
Hailo Dataflow Compiler – Python package (whl)	3.27.0	Installation guide	April 1, 2024

```
pip install hailo_dataflow_compiler-3.27.0-py3-none-linux_x86_64.whl
```

Install Model zoo, here you need to register Hailo and login, and download the software.


Select Product [✓ ACCELERATORS] VISION PROCESSORS

Search for a package

[✓ Archive]

↓Software Package	↓Software Sub-Package	↓Architecture	↓OS	↓Python Version
[✓ AI Software Suite]	AI Software Suite	ARM64	[✓ Linux]	[✓ 3.8]
	Dataflow Compiler	ARMEL	Windows	3.9
	HailoRT	ARMHF		3.10
	[✓ Model Zoo]	[✓ x86]		3.11
	TAPPAS			
	Plugins			

Clear all

Long-Term Support

Please Note

We highly recommend downloading and using only the latest Hailo AI Software Suite for most recent updates and best consistency across products and versions.

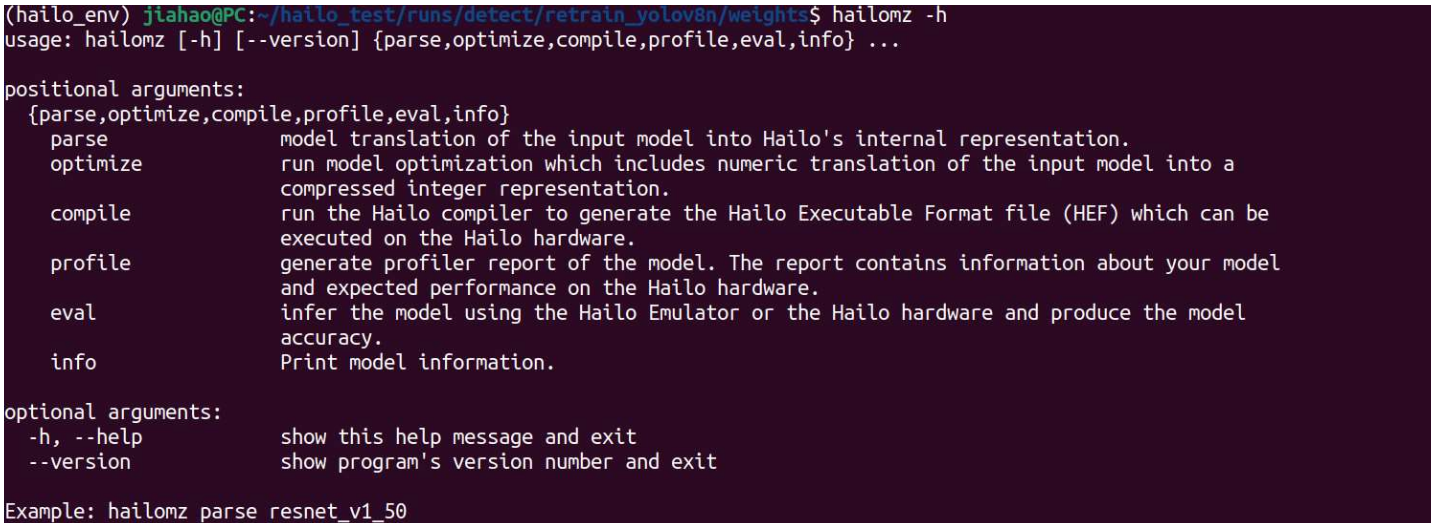
For older versions, the versions compatibility table details which packages are supported together with each other.

Package Name	Version	Documentation	Date Modified
Hailo Model Zoo – Python package (whl)	2.11.0		April 1, 2024

```
pip install hailo_model_zoo-2.11.0-py3-none-any.whl
```

Test whether `hailo_model_zoo` is functioning correctly.

```
hailomz -h
```



```
(hailo_env) jiahao@PC:~/hailo_test/runs/detect/retrain_yolov8n/weights$ hailomz -h
usage: hailomz [-h] [--version] {parse,optimize,compile,profile,eval,info} ...

positional arguments:
  {parse,optimize,compile,profile,eval,info}
    parse               model translation of the input model into Hailo's internal representation.
    optimize            run model optimization which includes numeric translation of the input model into a
                        compressed integer representation.
    compile             run the Hailo compiler to generate the Hailo Executable Format file (HEF) which can be
                        executed on the Hailo hardware.
    profile             generate profiler report of the model. The report contains information about your model
                        and expected performance on the Hailo hardware.
    eval                infer the model using the Hailo Emulator or the Hailo hardware and produce the model
                        accuracy.
    info                Print model information.

optional arguments:
  -h, --help            show this help message and exit
  --version             show program's version number and exit

Example: hailomz parse resnet_v1_50
```

Install `hailo_model_zoo` github file

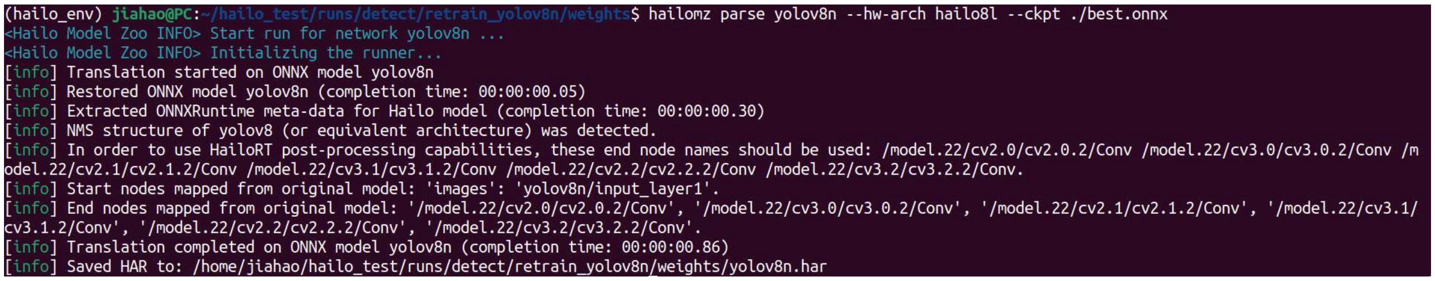
```
cd yolomodel/runs/detect/retrain_yolov8n/weights

git clone https://github.com/hailo-ai/hailo_model_zoo.git
```

Install coco dataset for evaluate/optimize/compile the yolov8n model

```
python hailo_model_zoo/datasets/create_coco_tfrecord.py val2017
python hailo_model_zoo/datasets/create_coco_tfrecord.py calib2017
```

Use `hailo_model_zoo` to parse the model:

```
hailomz parse --hw-arch hailo8l --ckpt ./best.onnx yolov8n
```

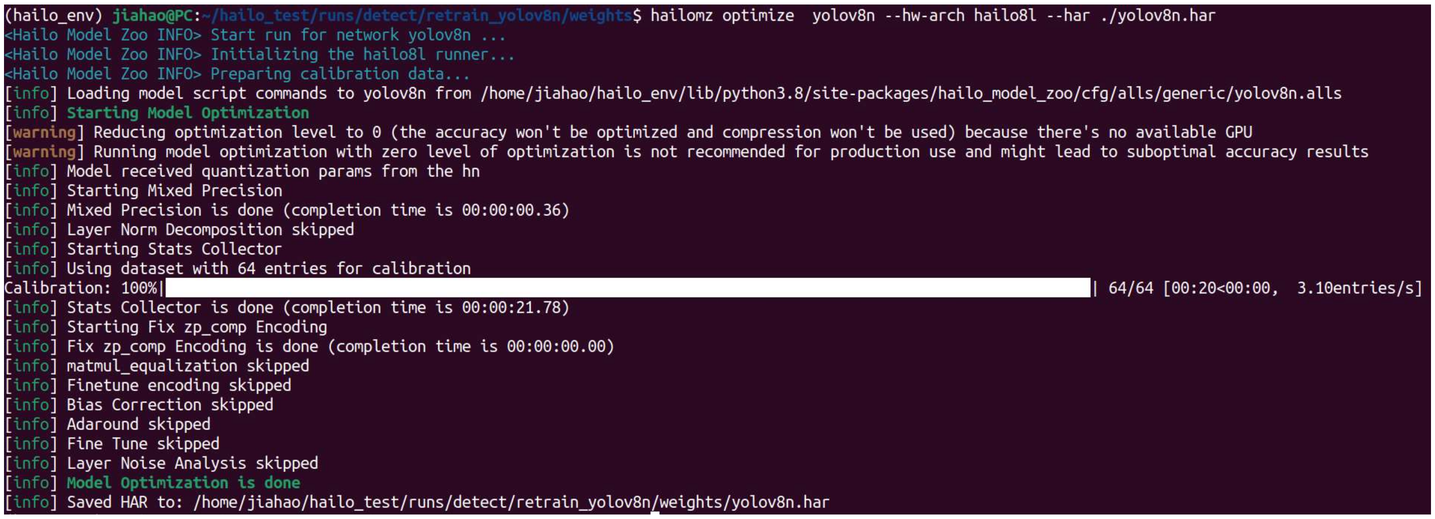


```
(hailo_env) jiahao@PC:~/hailo_test/runs/detect/retrain_yolov8n/weights$ hailomz parse yolov8n --hw-arch hailo8l --ckpt ./best.onnx
<Hailo Model Zoo INFO> Start run for network yolov8n ...
<Hailo Model Zoo INFO> Initializing the runner...
[info] Translation started on ONNX model yolov8n
[info] Restored ONNX model yolov8n (completion time: 00:00:00.05)
[info] Extracted ONNXRuntime meta-data for Hailo model (completion time: 00:00:00.30)
[info] NMS structure of yolov8 (or equivalent architecture) was detected.
[info] In order to use HailoRT post-processing capabilities, these end node names should be used: /model.22/cv2.0/cv2.0.2/Conv /model.22/cv3.0/cv3.0.2/Conv /m
odel.22/cv2.1/cv2.1.2/Conv /model.22/cv3.1/cv3.1.2/Conv /model.22/cv2.2/cv2.2.2/Conv /model.22/cv3.2/cv3.2.2/Conv.
[info] Start nodes mapped from original model: 'images': 'yolov8n/input_layer1'.
[info] End nodes mapped from original model: '/model.22/cv2.0/cv2.0.2/Conv', '/model.22/cv3.0/cv3.0.2/Conv', '/model.22/cv2.1/cv2.1.2/Conv', '/model.22/cv3.1/
cv3.1.2/Conv', '/model.22/cv2.2/cv2.2.2/Conv', '/model.22/cv3.2/cv3.2.2/Conv'.
[info] Translation completed on ONNX model yolov8n (completion time: 00:00:00.86)
[info] Saved HAR to: /home/jiahao/hailo_test/runs/detect/retrain_yolov8n/weights/yolov8n.har
```

Use `hailo_model_zoo` to optimize the model:

> **NOTE**
>
> If you execute the following command, you may encounter some errors. However, you can copy `./hailo_model_zoo/hailo_model_zoo` to your local library. If the error indicates that the dataset cannot be found, you can navigate to `~/.hailomz`, locate the dataset, and copy it to your local dataset directory.

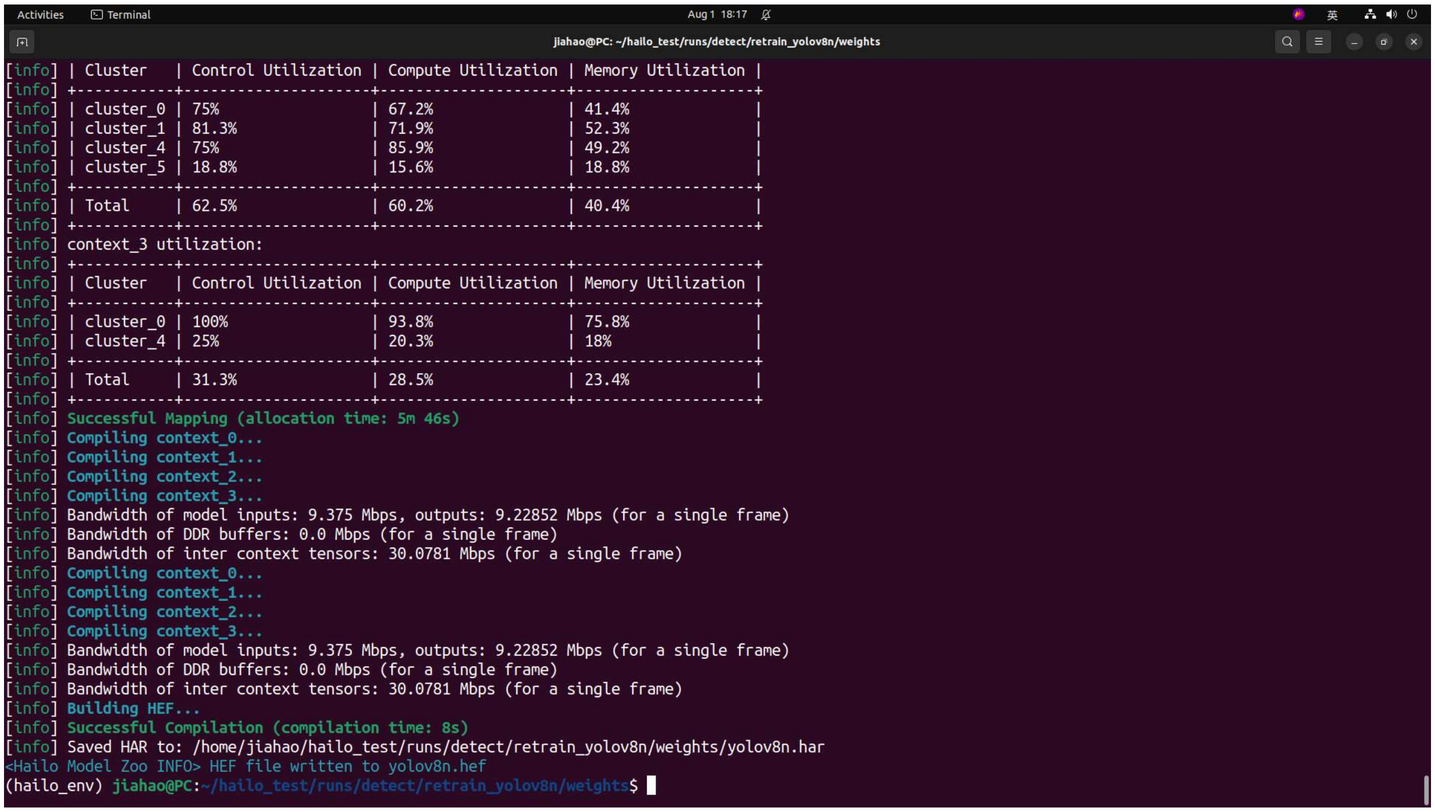
```
hailomz optimize --hw-arch hailo8l --har ./yolov8n.har yolov8n
```



```
(hailo_env) jiahao@PC:~/hailo_test/runs/detect/retrain_yolov8n/weights$ hailomz optimize  yolov8n --hw-arch hailo8l --har ./yolov8n.har
<Hailo Model Zoo INFO> Start run for network yolov8n ...
<Hailo Model Zoo INFO> Initializing the hailo8l runner...
<Hailo Model Zoo INFO> Preparing calibration data...
[info] Loading model script commands to yolov8n from /home/jiahao/hailo_env/lib/python3.8/site-packages/hailo_model_zoo/cfg/alls/generic/yolov8n.alls
[info] Starting Model Optimization
[warning] Reducing optimization level to 0 (the accuracy won't be optimized and compression won't be used) because there's no available GPU
[warning] Running model optimization with zero level of optimization is not recommended for production use and might lead to suboptimal accuracy results
[info] Model received quantization params from the hn
[info] Starting Mixed Precision
[info] Mixed Precision is done (completion time is 00:00:00.36)
[info] Layer Norm Decomposition skipped
[info] Starting Stats Collector
[info] Using dataset with 64 entries for calibration
Calibration: 100%|████████████████████████████████████████| 64/64 [00:20<00:00,  3.10entries/s]
[info] Stats Collector is done (completion time is 00:00:21.78)
[info] Starting Fix zp_comp Encoding
[info] Fix zp_comp Encoding is done (completion time is 00:00:00.00)
[info] matmul_equalization skipped
[info] Finetune encoding skipped
[info] Bias Correction skipped
[info] Adaround skipped
[info] Fine Tune skipped
[info] Layer Noise Analysis skipped
[info] Model Optimization is done
[info] Saved HAR to: /home/jiahao/hailo_test/runs/detect/retrain_yolov8n/weights/yolov8n.har
```

Use `hailo_model_zoo` to compile the model:

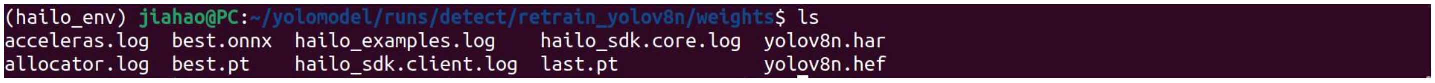
```
hailomz compile  yolov8n --hw-arch hailo8l --har ./yolov8n.har
```



```
[info] | Cluster   | Control Utilization | Compute Utilization | Memory Utilization |
[info] +-----------+---------------------+---------------------+--------------------+
[info] | cluster_0 | 75%                 | 67.2%               | 41.4%              |
[info] | cluster_1 | 81.3%               | 71.9%               | 52.3%              |
[info] | cluster_4 | 75%                 | 85.9%               | 49.2%              |
[info] | cluster_5 | 18.8%               | 15.6%               | 18.8%              |
[info] +-----------+---------------------+---------------------+--------------------+
[info] | Total     | 62.5%               | 60.2%               | 40.4%              |
[info] +-----------+---------------------+---------------------+--------------------+
[info] context_3 utilization:
[info] +-----------+---------------------+---------------------+--------------------+
[info] | Cluster   | Control Utilization | Compute Utilization | Memory Utilization |
[info] +-----------+---------------------+---------------------+--------------------+
[info] | cluster_0 | 100%                | 93.8%               | 75.8%              |
[info] | cluster_4 | 25%                 | 20.3%               | 18%                |
[info] +-----------+---------------------+---------------------+--------------------+
[info] | Total     | 31.3%               | 28.5%               | 23.4%              |
[info] +-----------+---------------------+---------------------+--------------------+
[info] Successful Mapping (allocation time: 5m 46s)
[info] Compiling context_0...
[info] Compiling context_1...
[info] Compiling context_2...
[info] Compiling context_3...
[info] Bandwidth of model inputs: 9.375 Mbps, outputs: 9.22852 Mbps (for a single frame)
[info] Bandwidth of DDR buffers: 0.0 Mbps (for a single frame)
[info] Bandwidth of inter context tensors: 30.0781 Mbps (for a single frame)
[info] Compiling context_0...
[info] Compiling context_1...
[info] Compiling context_2...
[info] Compiling context_3...
[info] Bandwidth of model inputs: 9.375 Mbps, outputs: 9.22852 Mbps (for a single frame)
[info] Bandwidth of DDR buffers: 0.0 Mbps (for a single frame)
[info] Bandwidth of inter context tensors: 30.0781 Mbps (for a single frame)
[info] Building HEF...
[info] Successful Compilation (compilation time: 8s)
[info] Saved HAR to: /home/jiahao/hailo_test/runs/detect/retrain_yolov8n/weights/yolov8n.har
<Hailo Model Zoo INFO> HEF file written to yolov8n.hef
(hailo_env) jiahao@PC:~/hailo_test/runs/detect/retrain_yolov8n/weights$
```

After all you will get a `hef` model, you can use it to deploy on raspberry pi5 with AI kit

```
ls
```



```
(hailo_env) jiahao@PC:~/yolomodel/runs/detect/retrain_yolov8n/weights$ ls
acceleras.log  best.onnx  hailo_examples.log    hailo_sdk.core.log  yolov8n.har
allocator.log  best.pt    hailo_sdk.client.log  last.pt             yolov8n.hef
```

On Raspberry Pi5

update the system:

```
sudo apt update
sudo apt full-upgrade
```

Set pcie to gen2/gen3(gen3 is faster than gen2):

Add following text to `/boot/firmware/config.txt`

```
#Enable the PCIe external connector
```

```
dtparam=pciex1

#Force Gen 3.0 speeds

dtparam=pciex1_gen=3
```

> **NOTE**
>
> If you want to use `gen2`, please comment `dtparam=pciex1_gen=3`

Install hailo-all and reboot:

Open the terminal on the Raspberry Pi 5 and enter the following command to install the Hailo software

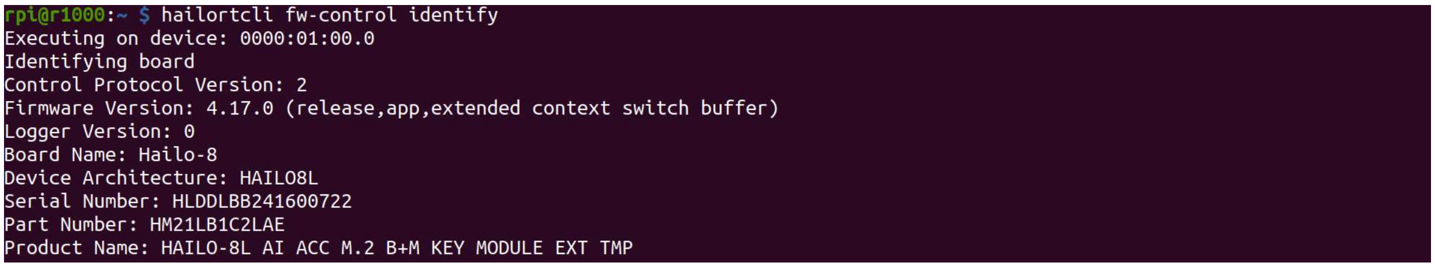
```
sudo apt install hailo-all
sudo reboot
```

Check Software and Hardware:

Open terminal on the Raspberry Pi5, and input command as follows to check if hailo-all have been installed.

```
hailortcli fw-control identify
```

The right result show as bellow:

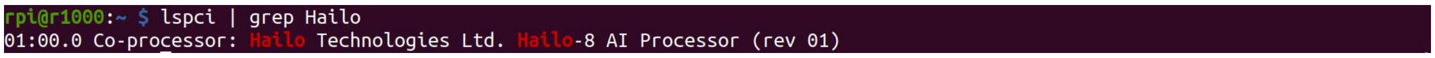


```
rpi@r1000:~ $ hailortcli fw-control identify
Executing on device: 0000:01:00.0
Identifying board
Control Protocol Version: 2
Firmware Version: 4.17.0 (release,app,extended context switch buffer)
Logger Version: 0
Board Name: Hailo-8
Device Architecture: HAILO8L
Serial Number: HLDDLBB241600722
Part Number: HM21LB1C2LAE
Product Name: HAILO-8L AI ACC M.2 B+M KEY MODULE EXT TMP
```

Open terminal on the Raspberry Pi5, and input command as follows to check if hailo-8L have been connected.

```
lspci | grep Hailo
```

The right result show as bellow:



```
rpi@r1000:~ $ lspci | grep Hailo
01:00.0 Co-processor: Hailo Technologies Ltd. Hailo-8 AI Processor (rev 01)
```

Clone the project:

```
git clone https://github.com/Seeed-Projects/Benchmarking-YOLOv8-on-Raspberry-PI-
reComputer-r1000-and-AIkit-Hailo-8L.git
cd Benchmarking-YOLOv8-on-Raspberry-PI-reComputer-r1000-and-AIkit-Hailo-8L
```

Copy your model to the raspberry pi5:

Make a directory named `hailomodel`

```
mkdir hailomodel
```

> **NOTE**
>
> The command below should be run on your host computer, not your Raspberry Pi 5. Ensure that both your host computer and Raspberry Pi 5 are connected to the same network.

```
scp -r ./yolomodel/runs/detect/retrain_yolov8n/weights/yolov8n.hef username@ip
/home/pi/Benchmarking-YOLOv8-on-Raspberry-PI-reComputer-r1000-and-AIkit-Hailo-
8L/hailomodel/
```

Change code

Find line 105 and 106 in `object-detection-hailo.py`, and change the code like below:

```
        elif args.network == "yolov8n":
            self.hef_path = os.path.join(self.current_path, './hailomodel/yolov8n.hef')
```

Find line 172 in `object-detection-hailo.py`, and change the code like below:

```
    parser.add_argument("--network", default="yolov8n", choices=['yolov6n', 'yolov8s',
'yolox_s_leaky'], help="Which Network to use, defult is yolov6n")
```

Run the code:

```
bash run.sh object-detection-hailo
```