

Local deployment of yolo training environment

Local deployment of yolo training environment

Environment Configuration

Download and install Yolo v5 environment dependencies

Preparing training data

Use YOLOv5 to train a fruit classification model

Convert the trained model to a kmodel that can be recognized by K230

Download the conversion tool

Other considerations

Environment Configuration

In this section, we will demonstrate how to deploy the Yolo v5 environment locally and train a classification (recognition) model.

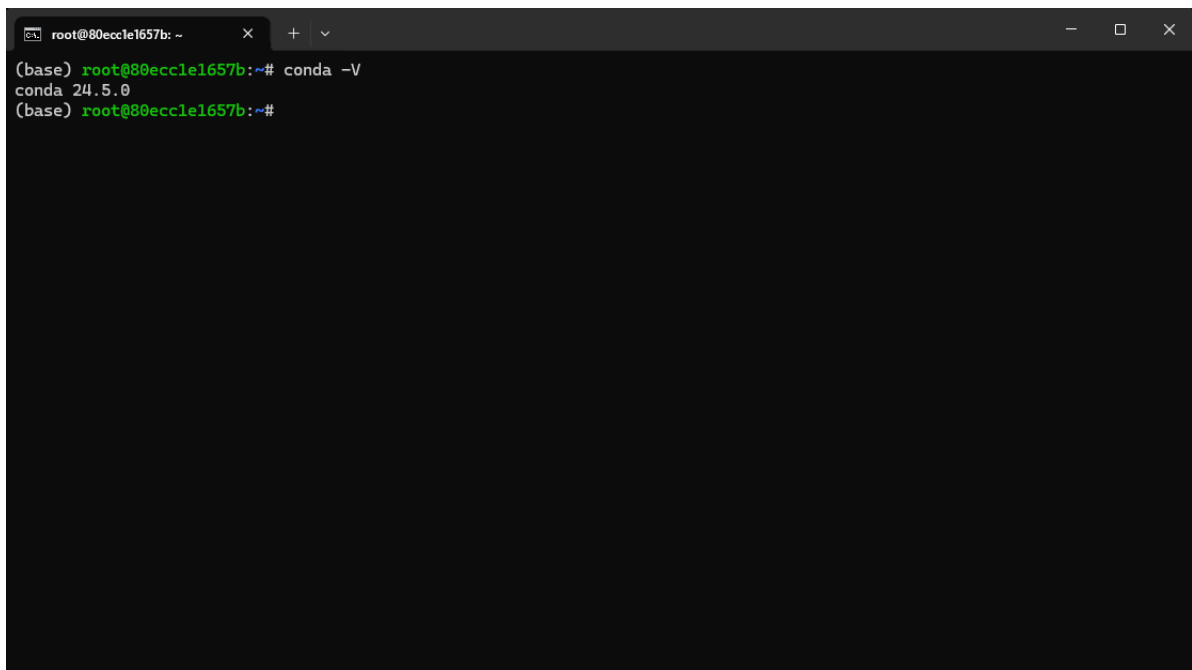
The operating environment we demonstrate is as follows:

System: Ubuntu 22.04

GPU: 4090D*1

CUDA version: 12.4

The system has pre-installed Miniconda3_3.10

A terminal window with a dark background and light green text. The window title is 'root@80ecc1e1657b: ~'. The terminal shows the command 'conda -V' being executed, which returns 'conda 24.5.0'. The prompt '(base) root@80ecc1e1657b:~#' is visible at the bottom.

```
root@80ecc1e1657b: ~  
(base) root@80ecc1e1657b:~# conda -V  
conda 24.5.0  
(base) root@80ecc1e1657b:~#
```

Download and install Yolo v5 environment dependencies

Enter the command

```
git clone https://github.com/ultralytics/yolov5.git
```

```
root@80ecc1e1657b: ~  
(base) root@80ecc1e1657b:~# conda -V  
conda 24.5.0  
(base) root@80ecc1e1657b:~# git clone https://github.com/ultralytics/yolov5.git  
正克隆到 'yolov5'...  
remote: Enumerating objects: 17360, done.  
remote: Counting objects: 100% (52/52), done.  
remote: Compressing objects: 100% (34/34), done.  
remote: Total 17360 (delta 36), reused 18 (delta 18), pack-reused 17308 (from 2)  
接收对象中: 100% (17360/17360), 16.25 MiB | 12.48 MiB/s, 完成。  
处理 delta 中: 100% (11901/11901), 完成。  
(base) root@80ecc1e1657b:~# |
```

implement

```
cd yolov5  
pip install -r requirements.txt
```

This part of the file contains a lot of content, please be patient

```
root@80ecc1e1657b: ~/yolov5  
Downloading ultralytics_thop-2.0.14-py3-none-any.whl (26 kB)  
Downloading filelock-3.18.0-py3-none-any.whl (16 kB)  
Downloading fsspec-2025.3.2-py3-none-any.whl (194 kB)  
194.4/194.4 kB 4.8 MB/s eta 0:00:00  
Downloading networkx-3.4.2-py3-none-any.whl (1.7 MB)  
1.7/1.7 MB 3.3 MB/s eta 0:00:00  
Downloading py_cpuinfo-9.0.0-py3-none-any.whl (22 kB)  
Downloading mpmath-1.3.0-py3-none-any.whl (536 kB)  
536.2/536.2 kB 2.5 MB/s eta 0:00:00  
Downloading smmap-5.0.2-py3-none-any.whl (24 kB)  
Installing collected packages: triton, pytz, py-cpuinfo, nvidia-cusparse-cu12, mpmath, tzdata, sympy, smmap, setuptools, pyparsing, pillow, nvidia-nvtx-cu12, nvidia-nvjitlink-cu12, nvidia-nccl-cu12, nvidia-curand-cu12, nvidia-cufft-cu12, nvidia-cuda-runtime-cu12, nvidia-cuda-nvrtc-cu12, nvidia-cuda-cupti-cu12, nvidia-cublas-cu12, numpy, networkx, kiwisolver, fsspec, fonttools, filelock, cycler, scipy, pandas, opencv-python, nvidia-cusparselt-cu12, nvidia-cudnn-cu12, gitdb, contourpy, nvidia-cusolver-cu12, matplotlib, gitpython, torch, seaborn, ultralytics-thop, torchvision, thop, ultralytics  
Attempting uninstall: setuptools  
Found existing installation: setuptools 69.5.1  
Uninstalling setuptools-69.5.1:  
Successfully uninstalled setuptools-69.5.1  
Successfully installed contourpy-1.3.1 cycler-0.12.1 filelock-3.18.0 fonttools-4.57.0 fsspec-2025.3.2 gitdb-4.0.12 gitpython-3.1.44 kiwisolver-1.4.8 matplotlib-3.10.1 mpmath-1.3.0 networkx-3.4.2 numpy-2.1.1 nvidia-cublas-cu12-12.4.5.8 nvidia-cuda-cupti-cu12-12.4.127 nvidia-cuda-nvrtc-cu12-12.4.127 nvidia-cuda-runtime-cu12-12.4.127 nvidia-cudnn-cu12-9.1.0.70 nvidia-cufft-cu12-11.2.1.3 nvidia-curand-cu12-10.3.5.147 nvidia-cusolver-cu12-11.6.1.9 nvidia-cusparselt-cu12-12.3.1.170 nvidia-cusparse-cu12-12.4.127 nvidia-nccl-cu12-2.21.5 nvidia-nvjitlink-cu12-12.4.127 nvidia-nvtx-cu12-12.4.127 opencv-python-4.11.0.86 pandas-2.2.3 pillow-11.1.0 py-cpuinfo-9.0.0 pyparsing-3.2.3 pytz-2025.2 scipy-1.15.2 seaborn-0.13.2 setuptools-78.1.0 smmap-5.0.2 sympy-1.13.1 thop-0.1.1.post2209072238 torch-2.6.0 torchvision-0.21.0 triton-3.2.0 tzdata-2025.2 ultralytics-8.3.103 ultralytics-thop-2.0.14  
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv  
(base) root@80ecc1e1657b:~/yolov5# |
```

Preparing training data

Here we use the training set officially provided by Canaan Technology as sample data

Please download the sample dataset provided. The sample dataset contains classification, detection and segmentation datasets for three types of fruits (apple, banana and orange). Unzip the dataset to `yolo5` the directory and use it `fruits_cls` as the dataset for the fruit classification task.

We execute the following code

```
wget https://kendryte-download.canaan-creative.com/developer/k230/yolo_files/datasets.zip
```

```
root@80ecc1e1657b: ~/yolo5
(base) root@80ecc1e1657b:~# ls
Desktop  Documents  Downloads  miniconda3  Music  Pictures  Public  Templates  Videos  yolo5
(base) root@80ecc1e1657b:~# cd yolo5/
(base) root@80ecc1e1657b:~/yolo5# ls
benchmarks.py  CONTRIBUTING.md  export.py  models  README.zh-CN.md  train.py  val.py
CITATION.cff  data  hubconf.py  pyproject.toml  requirements.txt  tutorial.ipynb
classify  detect.py  LICENSE  README.md  segment  utils
(base) root@80ecc1e1657b:~/yolo5# wget https://kendryte-download.canaan-creative.com/developer/k230/yolo_files/datasets
.zip
--2025-04-07 09:30:48-- https://kendryte-download.canaan-creative.com/developer/k230/yolo_files/datasets.zip
正在解析主机 kendryte-download.canaan-creative.com (kendryte-download.canaan-creative.com)... 27.155.113.144, 124.238.25
1.141
正在连接 kendryte-download.canaan-creative.com (kendryte-download.canaan-creative.com)|27.155.113.144|:443... 已连接。
已发出 HTTP 请求, 正在等待回应... 200 OK
长度: 64746262 (62M) [application/zip]
正在保存至: 'datasets.zip'

datasets.zip          100%[=====] 61.75M  57.6MB/s   用时 1.1s

2025-04-07 09:30:50 (57.6 MB/s) - 已保存 'datasets.zip' [64746262/64746262]

(base) root@80ecc1e1657b:~/yolo5#
```

```
unzip datasets.zip
```

```
root@80ecc1e1657b: ~/yolo5
inflating: datasets/fruits_yolo/labels/val/orange_24.txt
inflating: datasets/fruits_yolo/labels/val/orange_27.txt
inflating: datasets/fruits_yolo/labels/val/orange_29.txt
inflating: datasets/fruits_yolo/labels/val/orange_3.txt
extracting: datasets/fruits_yolo/labels/val/orange_30.txt
inflating: datasets/fruits_yolo/labels/val/orange_32.txt
inflating: datasets/fruits_yolo/labels/val/orange_34.txt
extracting: datasets/fruits_yolo/labels/val/orange_35.txt
inflating: datasets/fruits_yolo/labels/val/orange_36.txt
extracting: datasets/fruits_yolo/labels/val/orange_4.txt
extracting: datasets/fruits_yolo/labels/val/orange_47.txt
extracting: datasets/fruits_yolo/labels/val/orange_49.txt
inflating: datasets/fruits_yolo/labels/val/orange_5.txt
extracting: datasets/fruits_yolo/labels/val/orange_54.txt
inflating: datasets/fruits_yolo/labels/val/orange_57.txt
extracting: datasets/fruits_yolo/labels/val/orange_58.txt
inflating: datasets/fruits_yolo/labels/val/orange_62.txt
extracting: datasets/fruits_yolo/labels/val/orange_67.txt
extracting: datasets/fruits_yolo/labels/val/orange_71.txt
extracting: datasets/fruits_yolo/labels/val/orange_73.txt
extracting: datasets/fruits_yolo/labels/val/orange_74.txt
inflating: datasets/fruits_yolo/labels/val/orange_8.txt
extracting: datasets/fruits_yolo/labels/val/orange_85.txt
extracting: datasets/fruits_yolo/labels/val/orange_9.txt
inflating: datasets/fruits_yolo/labels/val/orange_93.txt
extracting: datasets/fruits_yolo/labels/val/orange_95.txt
inflating: datasets/fruits_yolo/test.txt
inflating: datasets/fruits_yolo/train.txt
inflating: datasets/fruits_yolo/val.txt
(base) root@80ecc1e1657b:~/yolo5#
```

Use YOLOv5 to train a fruit classification model

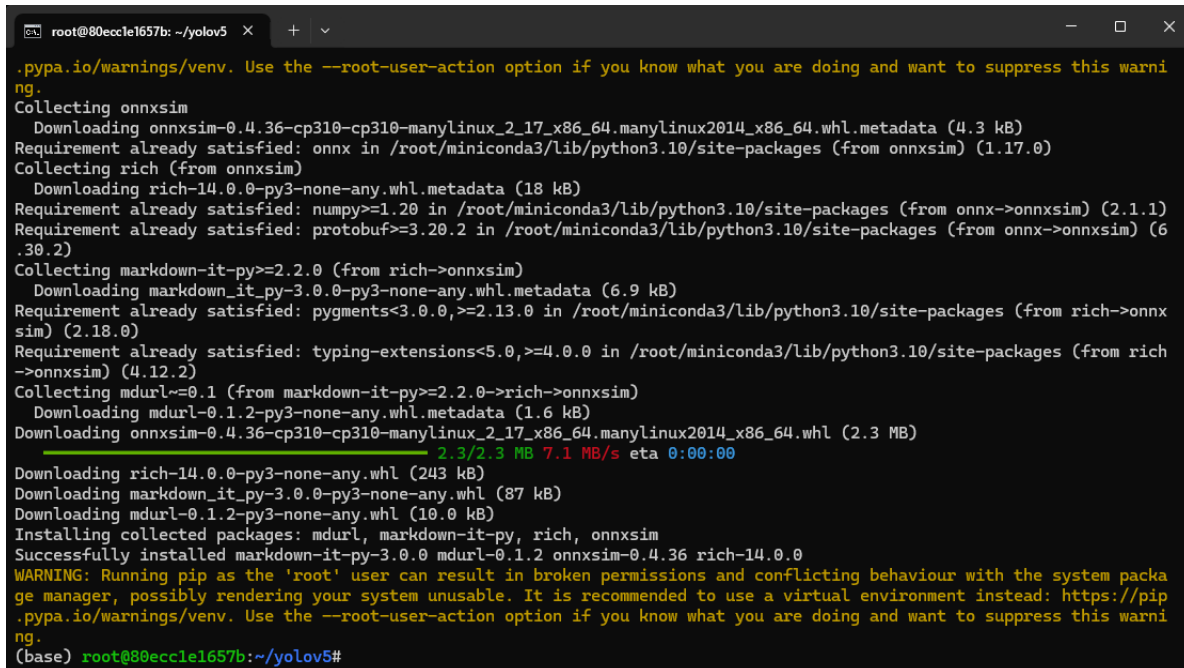
`yolo5` Execute the command in the directory and use `yolo5` to train three types of fruit classification models:

In order to use it on K230, we need to convert it into a model in kmodel format

Download the conversion tool

Execute the following code to install the required tools

```
sudo apt-get install -y dotnet-sdk-7.0
pip install --upgrade pip
pip install nncase==2.9.0
pip install nncase-kpu==2.9.0
pip install onnx
pip install onnxruntime
pip install onnxsim
```

A terminal window with a dark background and light-colored text. The prompt is 'root@80ecc1e1657b: ~/yolov5'. The command '.pypa.io/warnings/venv. Use the --root-user-action option if you know what you are doing and want to suppress this warning.' is executed. The output shows the installation of onnxsim and its dependencies: rich, markdown-it-py, and mdurl. The progress bar for onnxsim shows 2.3/2.3 MB at 7.1 MB/s. The prompt changes to '(base) root@80ecc1e1657b: ~/yolov5#'.

```
root@80ecc1e1657b: ~/yolov5 X + v
.pypa.io/warnings/venv. Use the --root-user-action option if you know what you are doing and want to suppress this warning.
Collecting onnxsim
  Downloading onnxsim-0.4.36-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (4.3 kB)
Requirement already satisfied: onnx in /root/.miniconda3/lib/python3.10/site-packages (from onnxsim) (1.17.0)
Collecting rich (from onnxsim)
  Downloading rich-14.0.0-py3-none-any.whl.metadata (18 kB)
Requirement already satisfied: numpy>=1.20 in /root/.miniconda3/lib/python3.10/site-packages (from onnx->onnxsim) (2.1.1)
Requirement already satisfied: protobuf>=3.20.2 in /root/.miniconda3/lib/python3.10/site-packages (from onnx->onnxsim) (6.30.2)
Collecting markdown-it-py>=2.2.0 (from rich->onnxsim)
  Downloading markdown_it_py-3.0.0-py3-none-any.whl.metadata (6.9 kB)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /root/.miniconda3/lib/python3.10/site-packages (from rich->onnxsim) (2.18.0)
Requirement already satisfied: typing-extensions<5.0,>=4.0.0 in /root/.miniconda3/lib/python3.10/site-packages (from rich->onnxsim) (4.12.2)
Collecting mdurl~0.1 (from markdown-it-py>=2.2.0->rich->onnxsim)
  Downloading mdurl-0.1.2-py3-none-any.whl.metadata (1.6 kB)
Download onnxsim-0.4.36-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (2.3 MB)
  2.3/2.3 MB 7.1 MB/s eta 0:00:00
Download rich-14.0.0-py3-none-any.whl (243 kB)
Download markdown_it_py-3.0.0-py3-none-any.whl (87 kB)
Download mdurl-0.1.2-py3-none-any.whl (10.0 kB)
Installing collected packages: mdurl, markdown-it-py, rich, onnxsim
Successfully installed markdown-it-py-3.0.0 mdurl-0.1.2 onnxsim-0.4.36 rich-14.0.0
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager, possibly rendering your system unusable. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv. Use the --root-user-action option if you know what you are doing and want to suppress this warning.
(base) root@80ecc1e1657b: ~/yolov5#
```

Download the script tool and `test_yolov5.zip` unzip the model conversion script tool to `yolov5` the directory;

```
wget https://kendryte-download.canaan-creative.com/developer/k230/yolo_files/test_yolov5.zip
unzip test_yolov5.zip
```

```
root@80ecc1e1657b: ~/yolov5
inflating: test_yolov5/test/banana_31.jpg
inflating: test_yolov5/test/banana_32.jpg
inflating: test_yolov5/test/banana_39.jpg
inflating: test_yolov5/test/banana_43.jpg
inflating: test_yolov5/test/banana_5.jpg
inflating: test_yolov5/test/banana_55.jpg
inflating: test_yolov5/test/banana_65.jpg
inflating: test_yolov5/test/banana_66.jpg
inflating: test_yolov5/test/banana_74.jpg
inflating: test_yolov5/test/banana_8.jpg
inflating: test_yolov5/test/mixed_2.jpg
inflating: test_yolov5/test/mixed_22.jpg
inflating: test_yolov5/test/orange_10.jpg
inflating: test_yolov5/test/orange_23.jpg
inflating: test_yolov5/test/orange_26.jpg
inflating: test_yolov5/test/orange_38.jpg
inflating: test_yolov5/test/orange_42.jpg
inflating: test_yolov5/test/orange_52.jpg
inflating: test_yolov5/test/orange_59.jpg
inflating: test_yolov5/test/orange_6.jpg
inflating: test_yolov5/test/orange_76.jpg
inflating: test_yolov5/test/orange_77.jpg
inflating: test_yolov5/test/orange_82.jpg
inflating: test_yolov5/test/orange_94.jpg
creating: test_yolov5/test_images/
inflating: test_yolov5/test_images/test.jpg
extracting: test_yolov5/test_images/test_apple.jpg
inflating: test_yolov5/test_images/test_banana.jpg
inflating: test_yolov5/test_images/test_orange.jpg
(base) root@80ecc1e1657b:~/yolov5#
```

Start model format conversion

Step 1: Convert pt format to onnx format

```
python export.py --weight runs/train-cls/exp/weights/best.pt --imgsz 224 --batch 1 --include onnx
```

Note that "runs/train-cls/exp/weights/best.pt" here is the path of the pt model we trained.

If the previous steps are done exactly as the tutorial, then the path here does not need to be modified

Step 2: Enter the test_yolov5/classify directory

```
cd test_yolov5/classify
```

```
root@80ecc1e1657b: ~/yolov5
include onnx
export: data=coco128.yaml, weights=['runs/train-cls/exp/weights/best.pt'], imgsz=[224], batch_size=1, device=cpu, half=False, inplace=False, keras=False, optimize=False, int8=False, per_tensor=False, dynamic=False, cache=False, simplify=False, mmodel=False, opset=17, verbose=False, workspace=4, nms=False, agnostic_nms=False, topk_per_class=100, topk_all=100, iou_thres=0.45, conf_thres=0.25, include=['onnx']
YOLOv5 v7.0-411-gf4d8a84c Python-3.10.14 torch-2.6.0+cu124 CPU

Fusing layers...
Model summary: 117 layers, 1212307 parameters, 0 gradients, 2.9 GFLOPs

PyTorch: starting from runs/train-cls/exp/weights/best.pt with output shape (1, 3) (2.4 MB)

ONNX: starting export with onnx 1.17.0...
ONNX: export success 0.9s, saved as runs/train-cls/exp/weights/best.onnx (4.6 MB)

Export complete (1.4s)
Results saved to /root/yolov5/runs/train-cls/exp/weights
Detect: python classify/predict.py --weights runs/train-cls/exp/weights/best.onnx
Validate: python classify/val.py --weights runs/train-cls/exp/weights/best.onnx
PyTorch Hub: model = torch.hub.load('ultralytics/yolov5', 'custom', 'runs/train-cls/exp/weights/best.onnx') # WARN!
NG ⚠ ClassificationModel not yet supported for PyTorch Hub AutoShape inference
Visualize: https://netron.app
(base) root@80ecc1e1657b:~/yolov5# ls
benchmarks.py  data  export.py  pyproject.toml  runs  train.py  yolov5n-cls.pt
CITATION.cff  datasets  hubconf.py  README.md  segment  tutorial.ipynb
classify  datasets.zip  LICENSE  README.zh-CN.md  test_yolov5  utils
CONTRIBUTING.md  detect.py  models  requirements.txt  test_yolov5.zip  val.py
(base) root@80ecc1e1657b:~/yolov5# cd test_yolov5/
(base) root@80ecc1e1657b:~/yolov5/test_yolov5# cd classify/
(base) root@80ecc1e1657b:~/yolov5/test_yolov5/classify#
```

Execute model conversion instructions

```
python to_kmodel.py --target k230 --model ../../runs/train-cls/exp/weights/best.onnx --dataset ../test --input_width 224 --input_height 224 --ptq_option 0
```

Note: An error may occur at this time: RuntimeError:

```
/root/miniconda3/bin/./lib/libstdc++.so.6: version `GLIBCXX_3.4.30' not found (required by
/usr/lib/dotnet/host/fxr/7.0.19/libhostfxr.so): Success
```

Solution:

Check if GLIBCXX_3.4.30 exists

```
strings /usr/lib/x86_64-linux-gnu/libstdc++.so.6 | grep GLIBCXX
```

If there is one in the output, it means that the file is not really missing, but the version of libstdc++ in the virtual environment is lower than the version of the libstdc++ library in the system.

```
root@80ecc1e1657b: ~/yolov5 × + -
GLIBCXX_3.4.3
GLIBCXX_3.4.4
GLIBCXX_3.4.5
GLIBCXX_3.4.6
GLIBCXX_3.4.7
GLIBCXX_3.4.8
GLIBCXX_3.4.9
GLIBCXX_3.4.10
GLIBCXX_3.4.11
GLIBCXX_3.4.12
GLIBCXX_3.4.13
GLIBCXX_3.4.14
GLIBCXX_3.4.15
GLIBCXX_3.4.16
GLIBCXX_3.4.17
GLIBCXX_3.4.18
GLIBCXX_3.4.19
GLIBCXX_3.4.20
GLIBCXX_3.4.21
GLIBCXX_3.4.22
GLIBCXX_3.4.23
GLIBCXX_3.4.24
GLIBCXX_3.4.25
GLIBCXX_3.4.26
GLIBCXX_3.4.27
GLIBCXX_3.4.28
GLIBCXX_3.4.29
GLIBCXX_3.4.30
GLIBCXX_DEBUG_MESSAGE_LENGTH
(base) root@80ecc1e1657b: ~/yolov5/test_yolov5/classify#
```

At this point we only need to execute the following command

Delete the `libstdc++.so.6` file in the virtual environment (or rename it to another name, here we rename it to `libstdc++.so.6.old`)

```
mv /root/miniconda3/bin/./lib/libstdc++.so.6
/root/miniconda3/bin/./lib/libstdc++.so.6.old
```

Then create a soft link and let the `libstdc++.so.6` called by the virtual environment point to the `libstdc++.so.6` that comes with the system.

```
ln -s /usr/lib/x86_64-linux-gnu/libstdc++.so.6 libstdc++.so.6
```

After the conversion command is completed, the output is as follows

```
(base) root@80ecc1657b:~/yolov5/test_yolov5/classify# python to_kmodel.py --target k230 --model ../../runs/train-cls/exp/weights/best.onnx --dataset ../test --input_width 224 --input_height 224 --ptq_option 0
warn: Nncase.Hosting.PluginLoader[0]
      NNCASE_PLUGIN_PATH is not set.
/root/yolov5/test_yolov5/classify/to_kmodel.py:25: DeprecationWarning: `mapping.TENSOR_TYPE_TO_NP_TYPE` is now deprecated and will be removed in a future release.To silence this warning, please use `helper.tensor_dtype_to_np_dtype` instead.
  input_dict['dtype'] = onnx.mapping.TENSOR_TYPE_TO_NP_TYPE[onnx_type.elem_type]
WARNING: The argument `input_shapes` is deprecated. Please use `overwrite_input_shapes` and/or `test_input_shapes` instead. An error will be raised in the future.
```

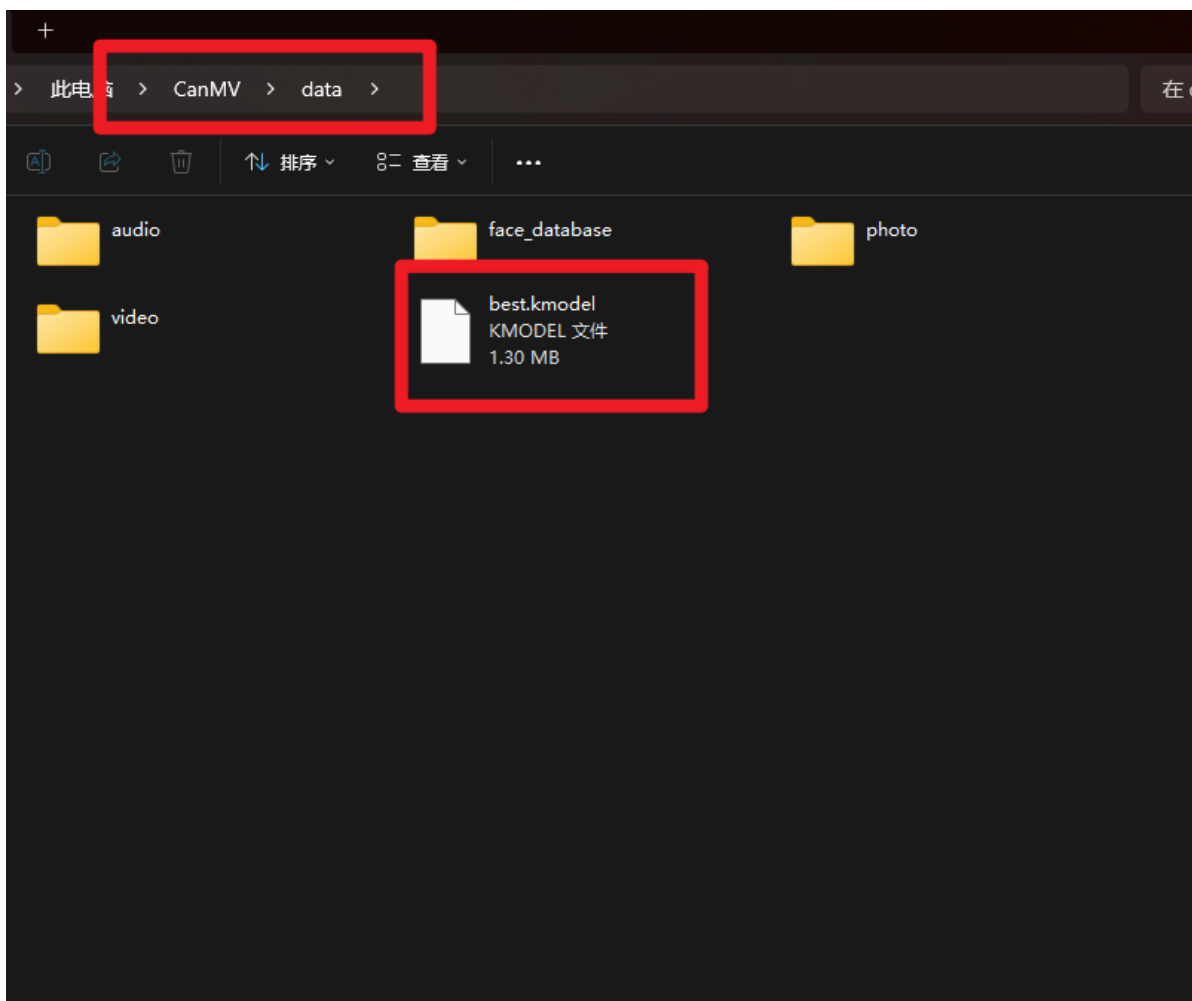

WARNING can be ignored here

The generated model is in the directory ~/yolov5/runs/train-cls/exp/weights

```
in ../Lib/ldc++.so.6.0
(base) root@80ecc1e1657b:~/yolov5/test_yolov5/classify# ln -s /usr/lib/x86_64-linux-gnu/libstdc++.so.6 libstdc++.so.6
(base) root@80ecc1e1657b:~/yolov5/test_yolov5/classify# python to_kmodel.py --target k230 --model ../../runs/train-cls/exp/weights/best.onnx --dataset ../test --input_width 224 --input_height 224 --ptq_option 0
warn: Nncase.Hosting.PluginLoader[0]
Nncase_PLUGIN_PATH is not set.
/root/yolov5/test_yolov5/classify/to_kmodel.py:25: DeprecationWarning: 'mapping.TENSOR_TYPE_TO_NP_TYPE' is now deprecated and will be removed in a future release. To silence this warning, please use 'helper.tensor_dtype_to_np_dtype' instead.
  input_dict['dtype'] = onnx.mapping.TENSOR_TYPE_TO_NP_TYPE[onnx_type.elem_type]
WARNING: The argument 'input_shapes' is deprecated. Please use 'overwrite_input_shapes' and/or 'test_input_shapes' instead. An error will be raised in the future.
(base) root@80ecc1e1657b:~/yolov5/test_yolov5/classify# cd ../../
(base) root@80ecc1e1657b:~/yolov5# ls
benchmarks.py  data  export.py  pyproject.toml  runs  train.py  yolov5n-cls.pt
CITATION.cff  datasets  hubconf.py  README.md  segment  tutorial.ipynb
classify  datasets.zip  LICENSE  README.zh-CN.md  test_yolov5  utils
CONTRIBUTING.md  detect.py  models  requirements.txt  test_yolov5.zip  val.py
(base) root@80ecc1e1657b:~/yolov5# cd runs/
(base) root@80ecc1e1657b:~/yolov5/runs# ls
train-cls
(base) root@80ecc1e1657b:~/yolov5/runs# cd train-cls/
(base) root@80ecc1e1657b:~/yolov5/runs/train-cls# ls
exp
(base) root@80ecc1e1657b:~/yolov5/runs/train-cls# cd exp/
(base) root@80ecc1e1657b:~/yolov5/runs/train-cls/exp# ls
opt.yaml  results.csv  test_images.jpg  train_images.jpg  weights
(base) root@80ecc1e1657b:~/yolov5/runs/train-cls/exp# cd weights/
(base) root@80ecc1e1657b:~/yolov5/runs/train-cls/exp/weights# ls
best.kmodel  best.onnx  best.pt  last.pt
(base) root@80ecc1e1657b:~/yolov5/runs/train-cls/exp/weights#
```

The name is best.kmodel

We copy this best.kmodel to the data directory of our K230



Then open CanMV IDE, copy the following code and click Run

```
from libs.PipeLine import PipeLine, ScopedTiming
```



```

from libs.YOLO import YOLOv5
import os,sys,gc
import ulab.numpy as np
import image

if __name__=="__main__":

    rgb888p_size=[1280,720]
    display_size=[640,480]
    # 模型路径 Model path
    kmodel_path="/data/best.kmodel"
    labels = ["apple","banana","orange"]
    confidence_threshold = 0.5
    model_input_size=[224,224]
    # 初始化Pipeline Initialize Pipeline

    pl=Pipeline(rgb888p_size=rgb888p_size,display_size=display_size,display_mode="1
cd")
    pl.create()
    # 初始化YOLOv5实例 Initialize the YOLOv5 instance

    yolo=YOLOv5(task_type="classify",mode="video",kmodel_path=kmodel_path,labels=la
bels,rgb888p_size=rgb888p_size,model_input_size=model_input_size,display_size=di
splay_size,conf_thresh=confidence_threshold,debug_mode=0)
    yolo.config_preprocess()
    try:
        while True:
            os.exitpoint()
            with ScopedTiming("total",1):
                # 逐帧推理 Frame-by-frame reasoning
                img = pl . get_frame ()
                res = yolo . run ( img )
                yolo . draw_result ( res , pl . osd_img )
                pl . show_image ()
                gc . collect ()
    except Exception as e :
        sys.print_exception ( e )
    finally :
        yolo.deinit ( )
        pl . destroy ()

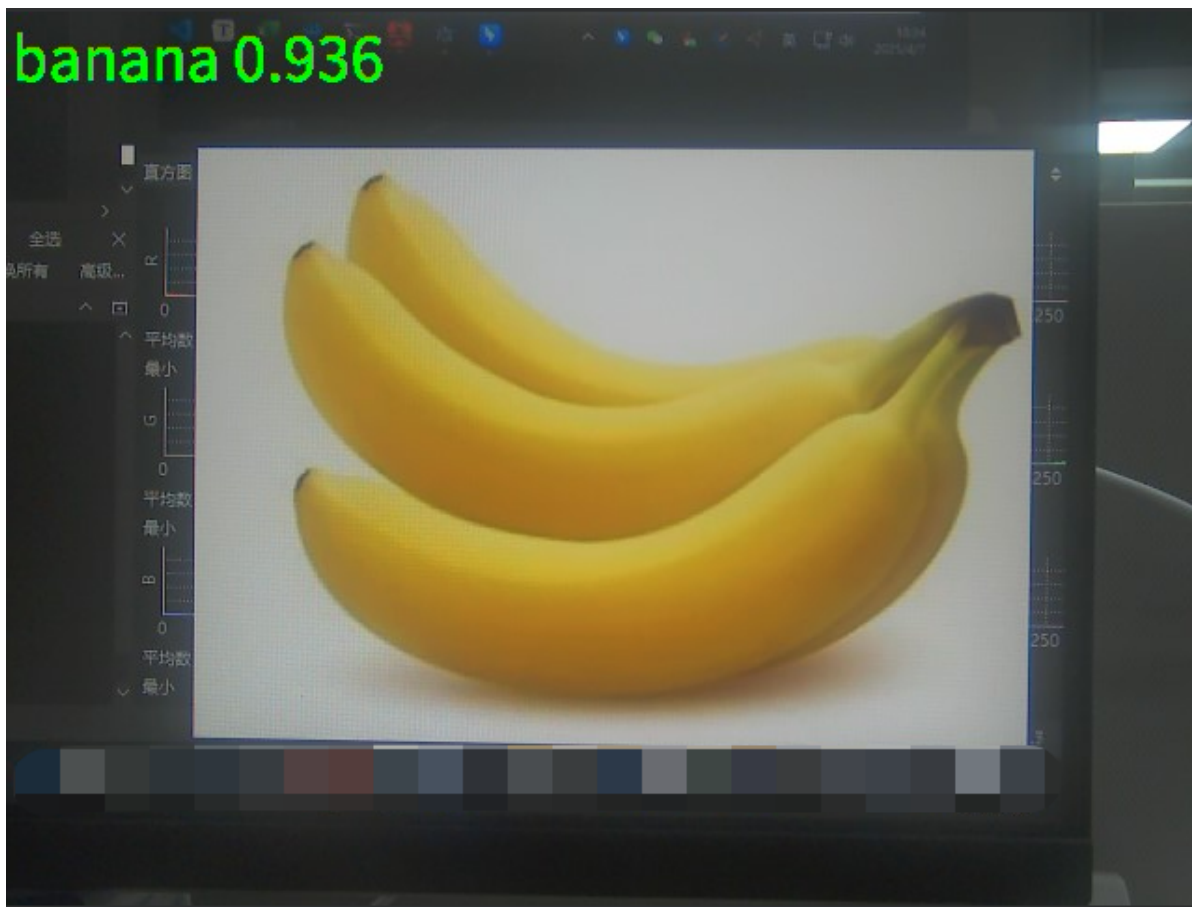
```

Let's test the effect

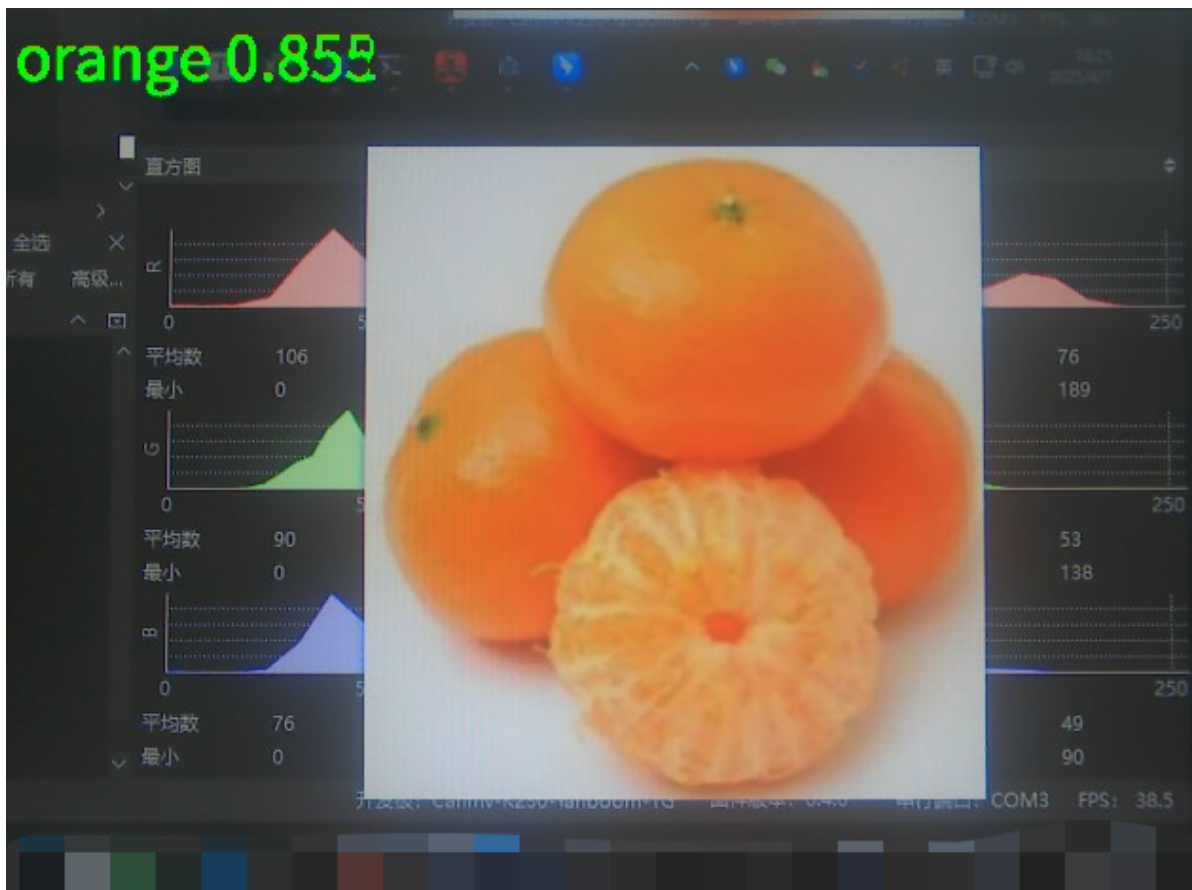
Identify Apples



Identify bananas



Identify oranges



You can see that this routine can run successfully.

Other considerations

1. Environment configuration, YOLOv5 environment dependencies, and model conversion tools only need to be downloaded once, and no re-download is required for subsequent training
2. If the speed of pip download is much lower than the actual network bandwidth, please check whether the pip mirror source is configured.
3. The model trained with the current dataset may identify non-banana objects other than oranges/apples as bananas. This is normal. If a more accurate model is needed, a training set with a larger amount of data can be prepared for training.