Training using the online platform

Introduction

In this section, we will introduce the online cloud platform provided by CanMV for model training. We take the commonly used detection model as an example to let K230 recognize two items, my bracelet and screwdriver

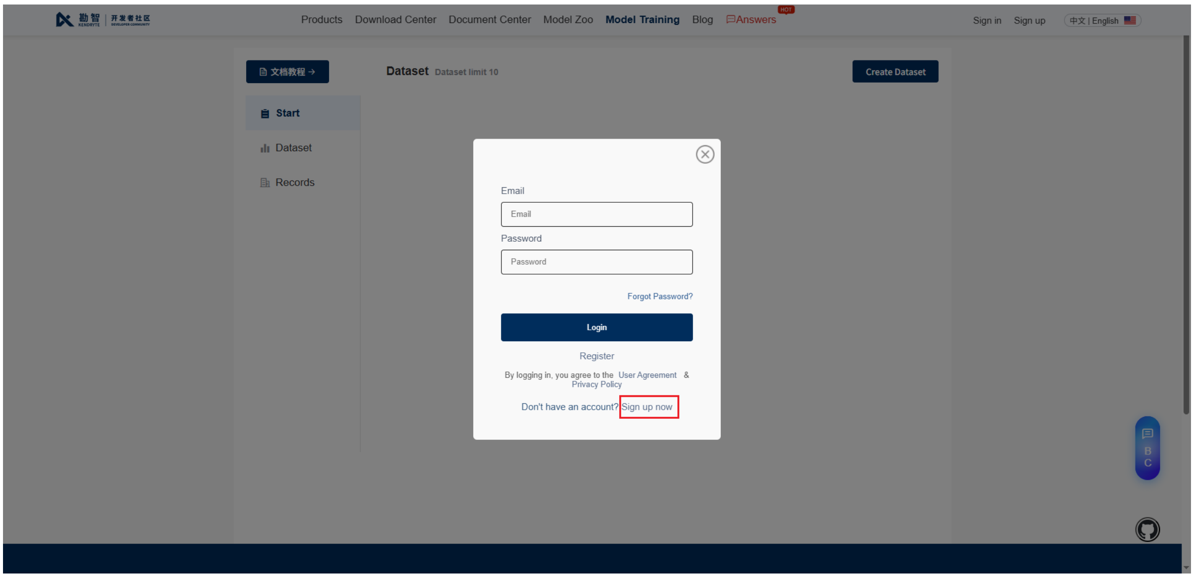
1. Prepare training data

We use the routine code in the [2.Basic Tutorial - 19.Camera] section to take pictures. For specific operations, please refer to the tutorial in the [2.Basic Tutorial - 19.Camera] section

2. Train the model on the training platform

Online training platform address: https://www.kendryte.com/en/training/dataset

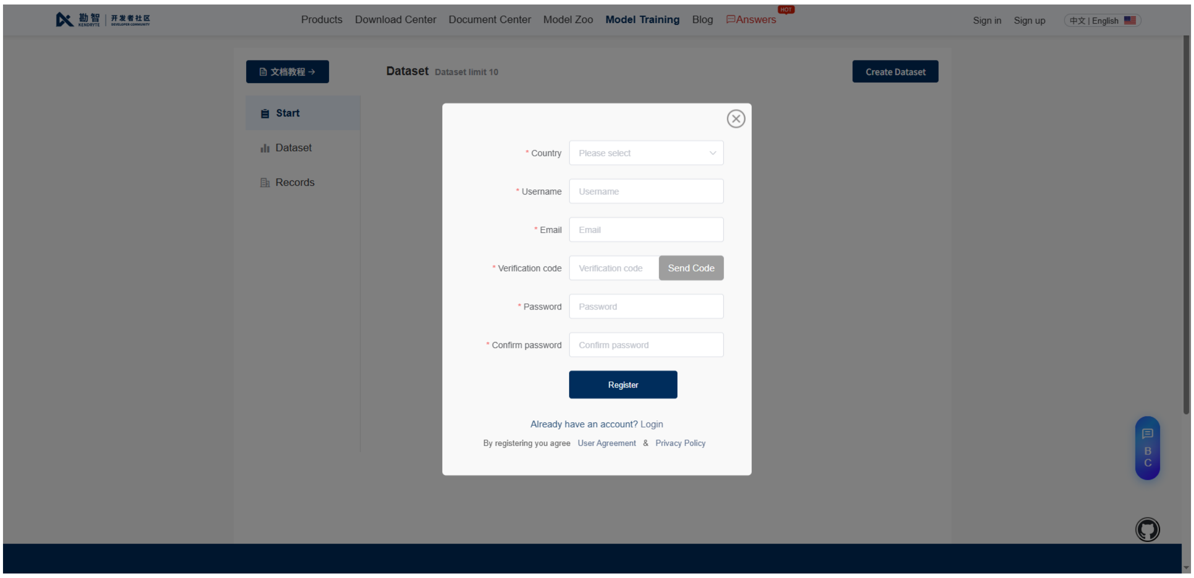
Click the Register button



By logging in, you agree to the User Agreement & Privacy Policy

Don't have an account? Sign up now

Fill in your personal information and click Register



* Country: Please select

* Username: Username

* Email: Email

* Verification code: Verification code Send Code

* Password: Password

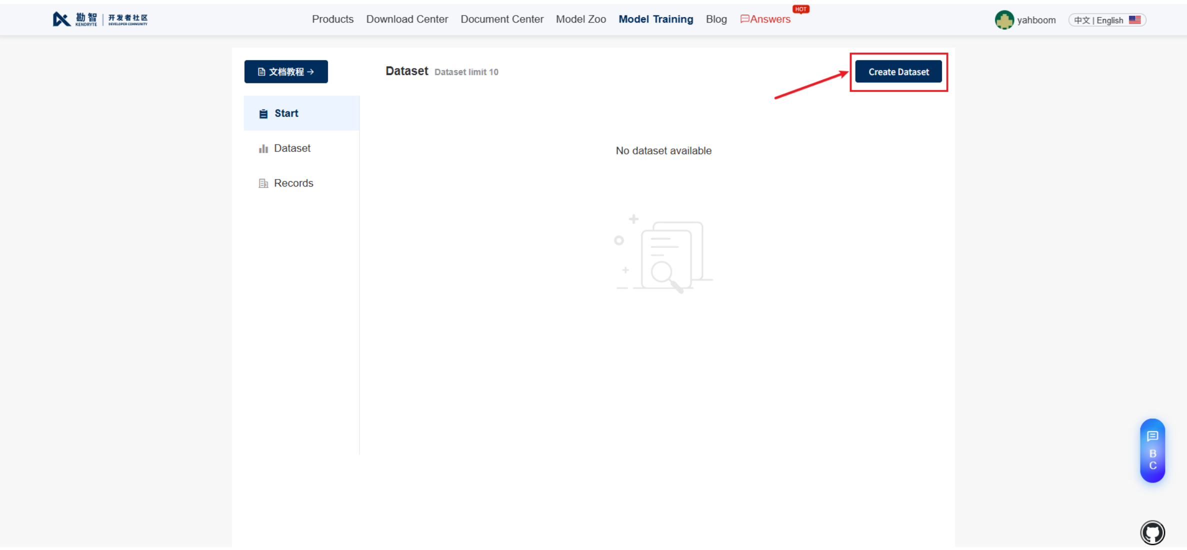
* Confirm password: Confirm password

Register

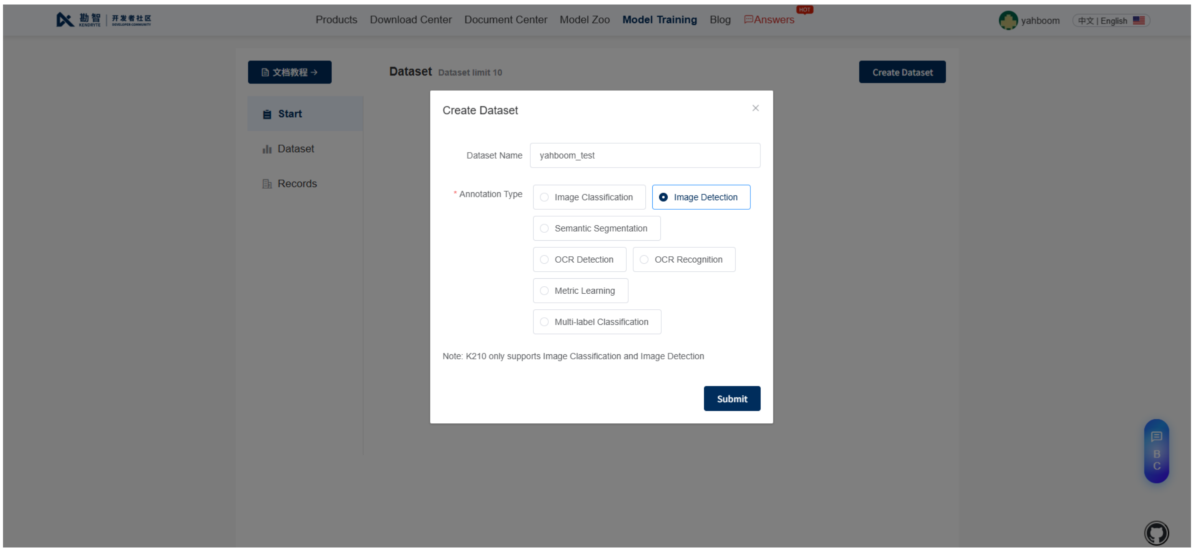
Already have an account? Login

By registering you agree User Agreement & Privacy Policy

After successful registration, you will be automatically logged in. We click "Create training set"



No dataset available



Create Dataset

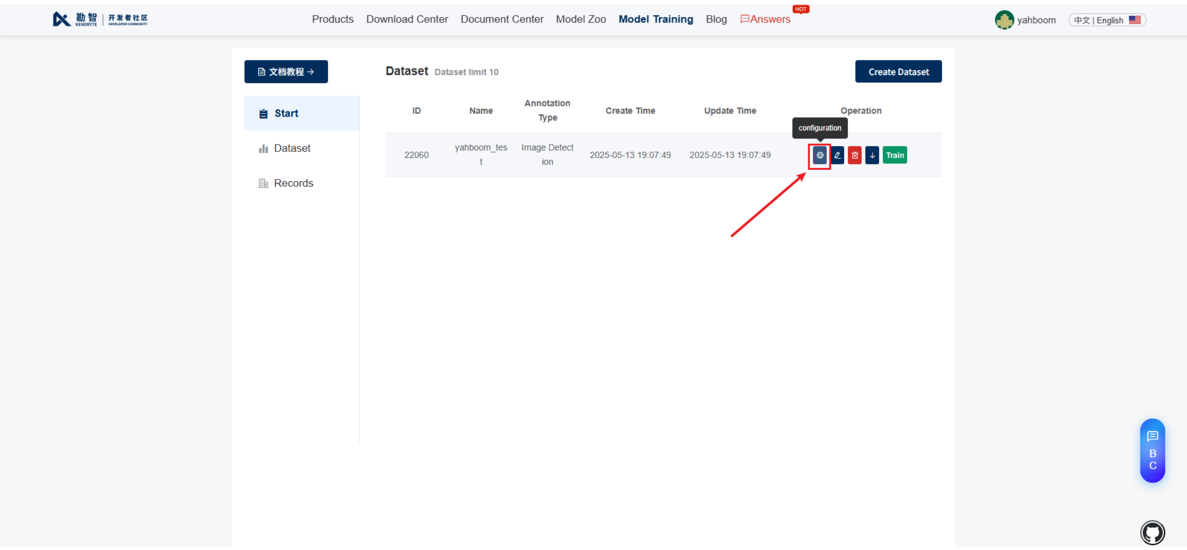
Dataset Name: yahboom_test

* Annotation Type: ○ Image Classification ● Image Detection ○ Semantic Segmentation ○ OCR Detection ○ OCR Recognition ○ Metric Learning ○ Multi-label Classification

Note: K210 only supports Image Classification and Image Detection

Submit

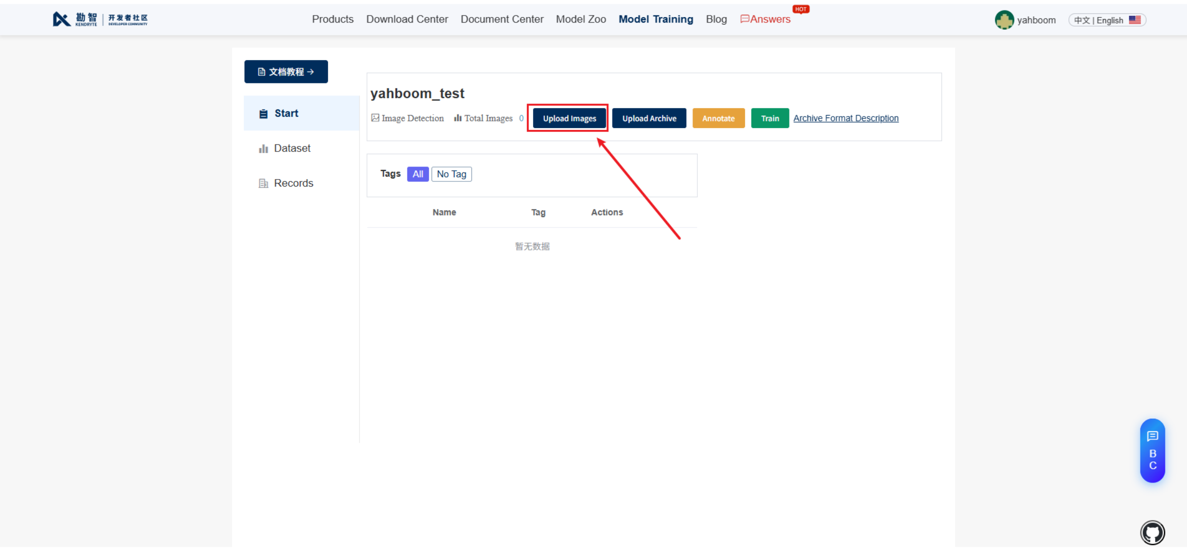
Click Edit



Dataset Dataset limit 10

ID	Name	Annotation Type	Create Time	Update Time	Operation
22060	yahboom_test	Image Detection	2025-05-13 19:07:49	2025-05-13 19:07:49	Train

Click to upload image

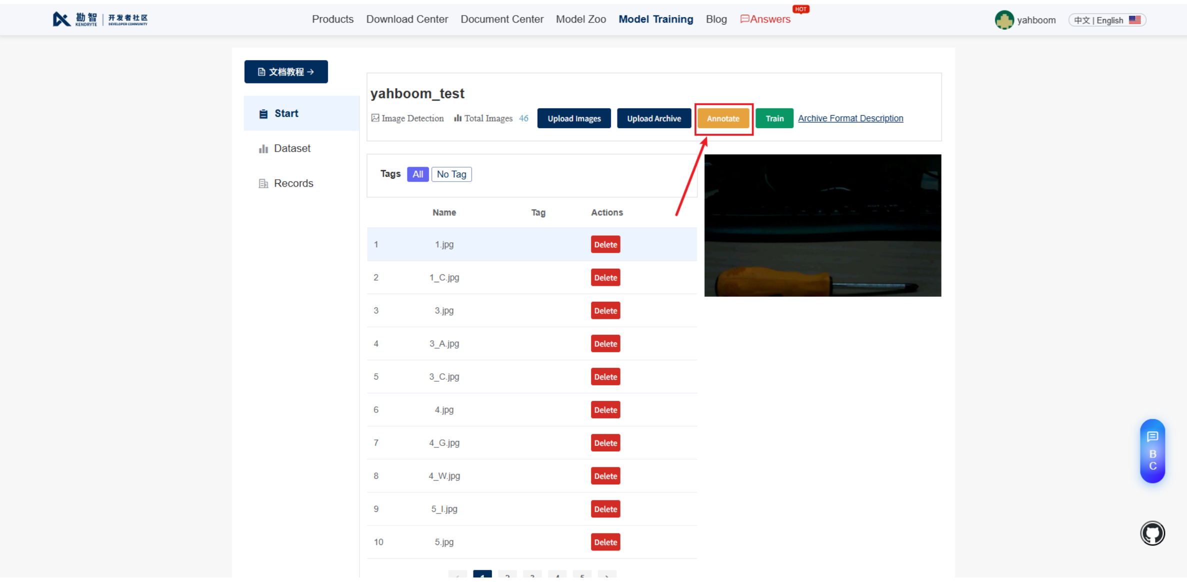


yahboom_test

Image Detection Total Images 0

暂无数据

After uploading the picture, we click the Annotate button

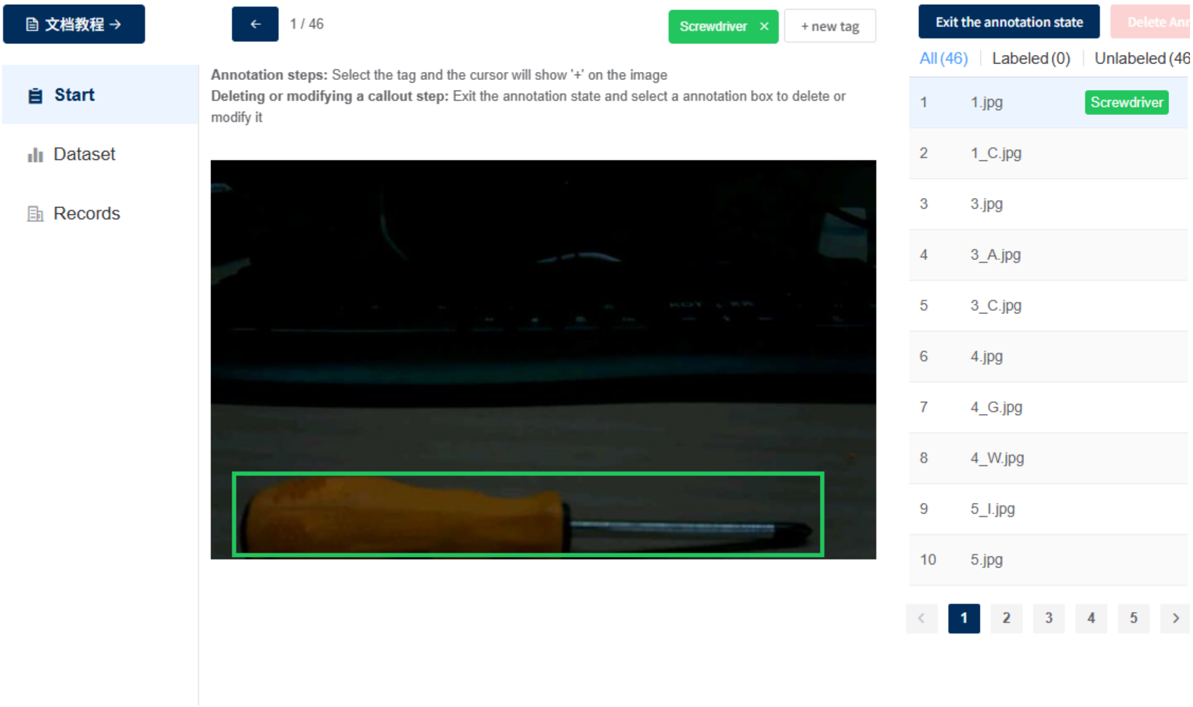


yahboom_test

Image Detection Total Images 46

	Name	Tag	Actions
1	1.jpg		Delete
2	1_C.jpg		Delete
3	3.jpg		Delete
4	3_A.jpg		Delete
5	3_C.jpg		Delete
6	4.jpg		Delete
7	4_G.jpg		Delete
8	4_W.jpg		Delete
9	5_I.jpg		Delete
10	5.jpg		Delete

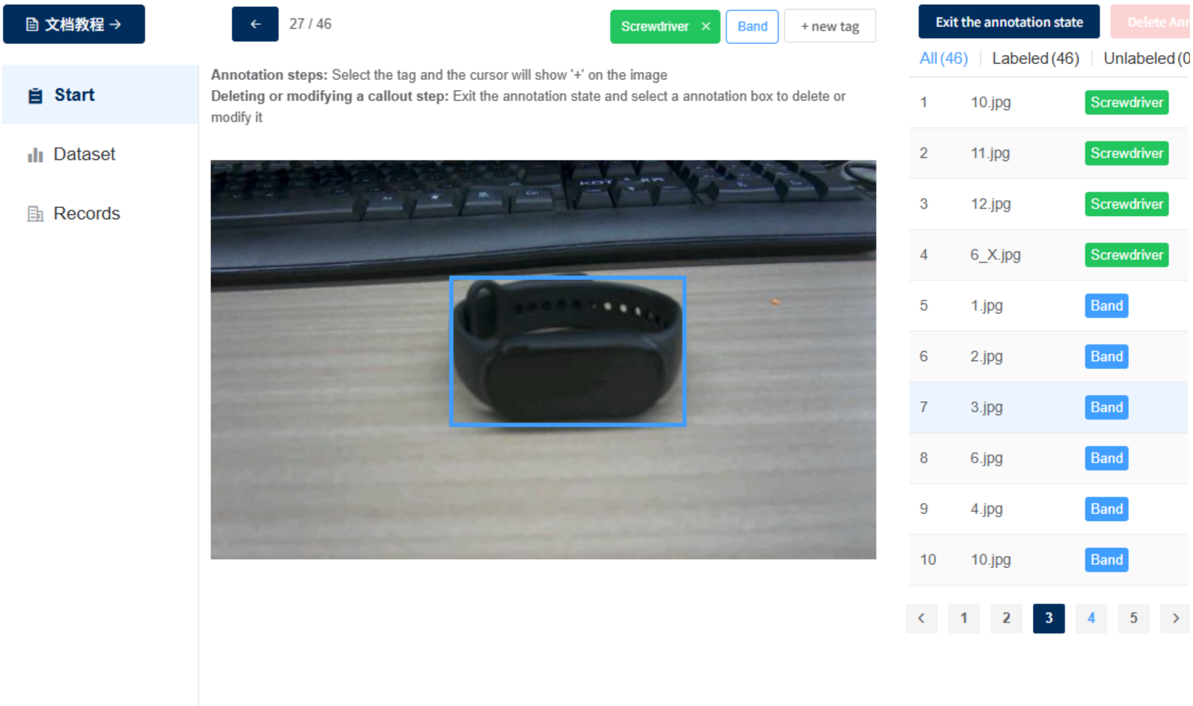
1. We first select the label box, enter a label name, and then press Enter.
2. Then we can use the mouse to mark the object to be detected on the picture.
3.



Annotation steps: Select the tag and the cursor will show '+' on the image

Deleting or modifying a callout step: Exit the annotation state and select a annotation box to delete or modify it

1	1.jpg	Screwdriver
2	1_C.jpg	
3	3.jpg	
4	3_A.jpg	
5	3_C.jpg	
6	4.jpg	
7	4_G.jpg	
8	4_W.jpg	
9	5_I.jpg	
10	5.jpg	

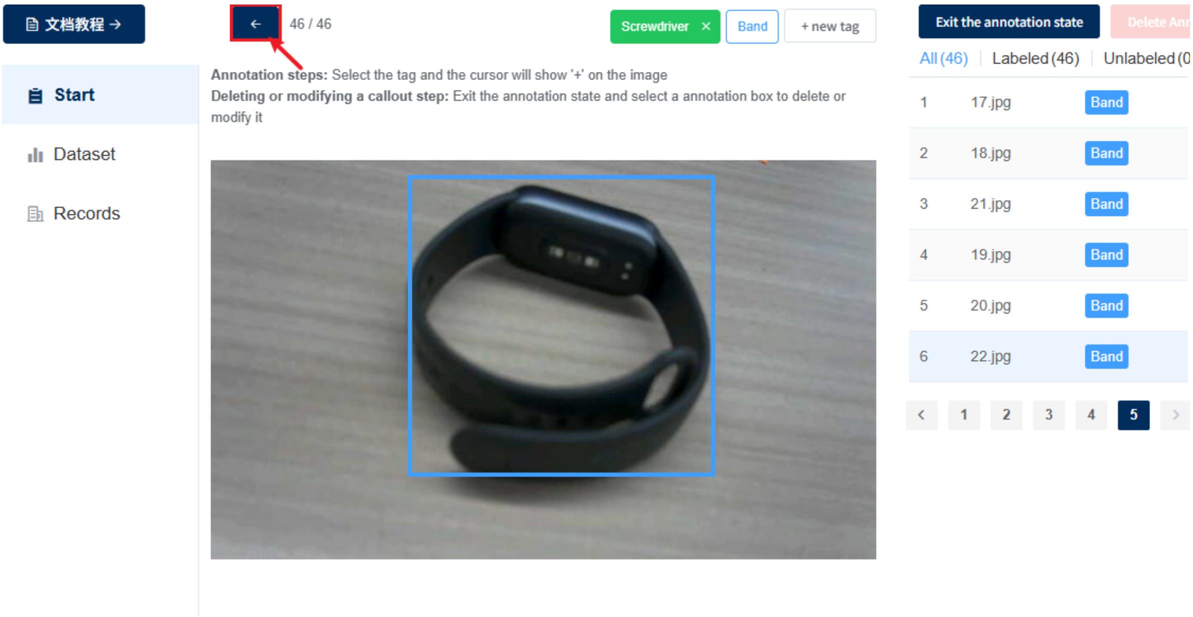


Annotation steps: Select the tag and the cursor will show '+' on the image

Deleting or modifying a callout step: Exit the annotation state and select a annotation box to delete or modify it

1	10.jpg	Screwdriver
2	11.jpg	Screwdriver
3	12.jpg	Screwdriver
4	6_X.jpg	Screwdriver
5	1.jpg	Band
6	2.jpg	Band
7	3.jpg	Band
8	6.jpg	Band
9	4.jpg	Band
10	10.jpg	Band

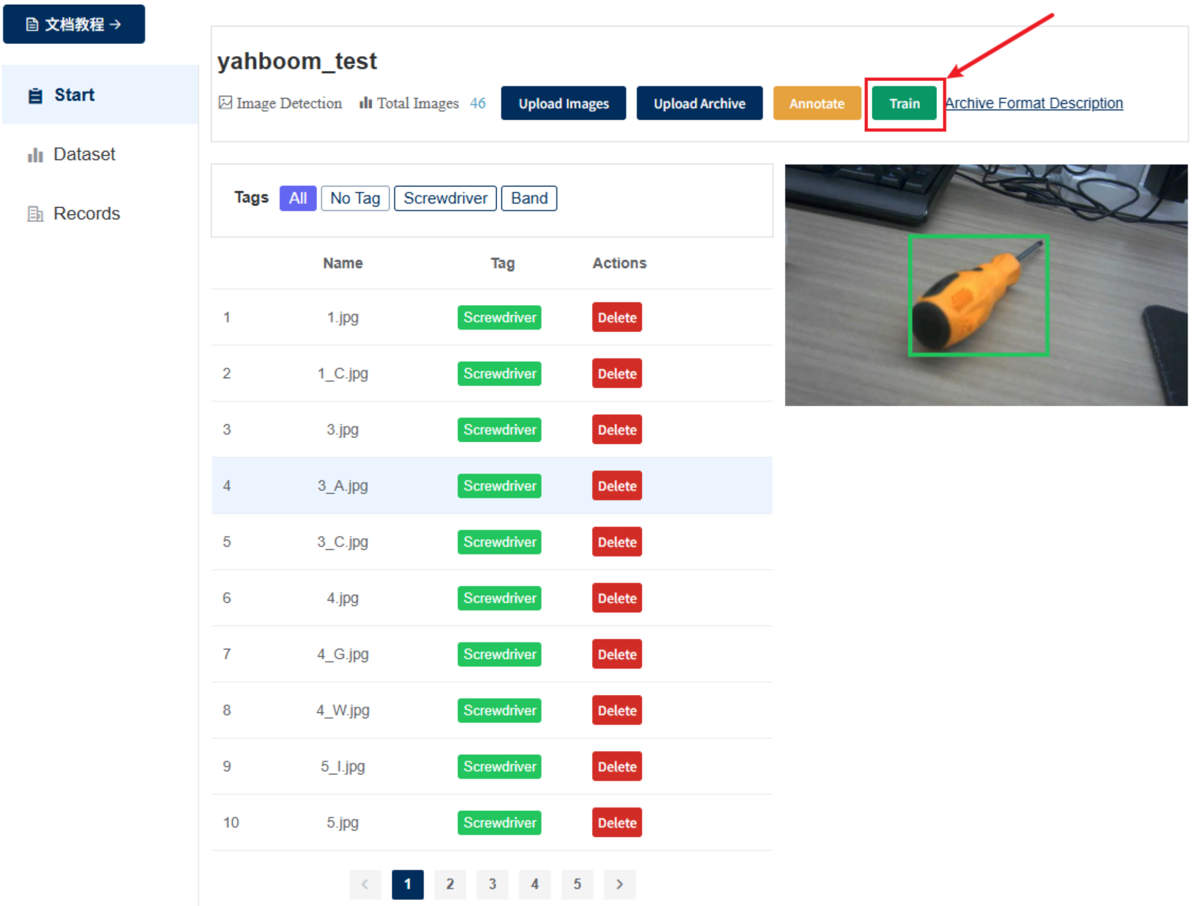
After all the markings are completed, we click the return button



Annotation steps: Select the tag and the cursor will show '+' on the image
Deleting or modifying a callout step: Exit the annotation state and select a annotation box to delete or modify it

1	17.jpg	Band
2	18.jpg	Band
3	21.jpg	Band
4	19.jpg	Band
5	20.jpg	Band
6	22.jpg	Band

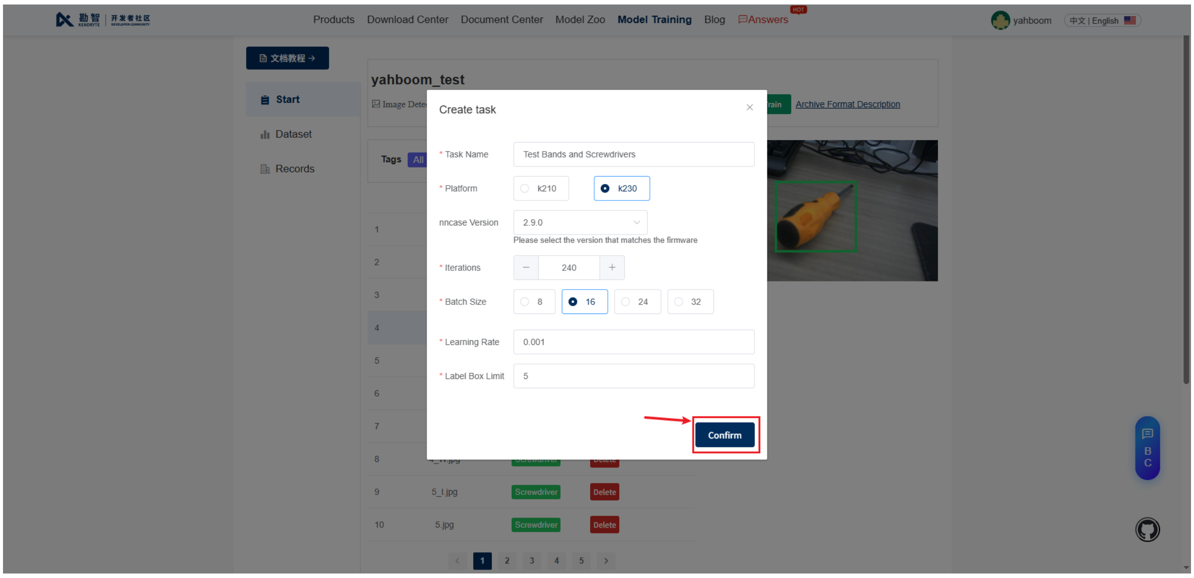
We click the training button



yahboom_test

	Name	Tag	Actions
1	1.jpg	Screwdriver	Delete
2	1_C.jpg	Screwdriver	Delete
3	3.jpg	Screwdriver	Delete
4	3_A.jpg	Screwdriver	Delete
5	3_C.jpg	Screwdriver	Delete
6	4.jpg	Screwdriver	Delete
7	4_G.jpg	Screwdriver	Delete
8	4_W.jpg	Screwdriver	Delete
9	5_I.jpg	Screwdriver	Delete
10	5.jpg	Screwdriver	Delete

Fill in the content as shown in the picture and click OK

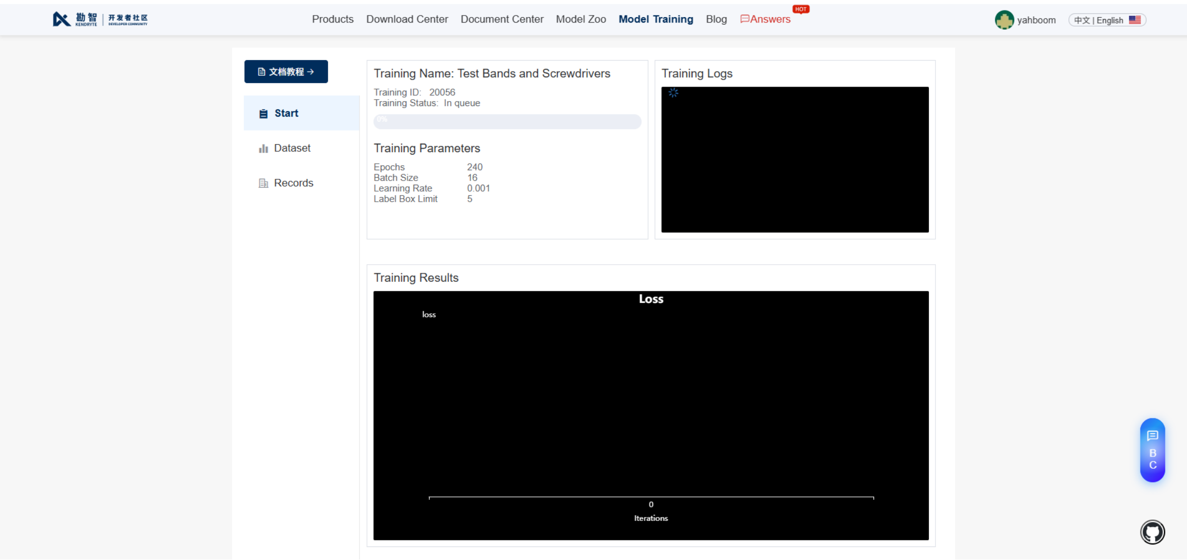


Create task

- * Task Name: Test Bands and Screwdrivers
- * Platform: ○ k210 ● k230
- nncase Version: 2.9.0
  Please select the version that matches the firmware
- * Iterations: 240
- * Batch Size: ○ 8 ● 16 ○ 24 ○ 32
- * Learning Rate: 0.001
- * Label Box Limit: 5

Confirm

Waiting for training (need to queue if there are many people)



Training Name: Test Bands and Screwdrivers

Training ID: 20056
Training Status: In queue

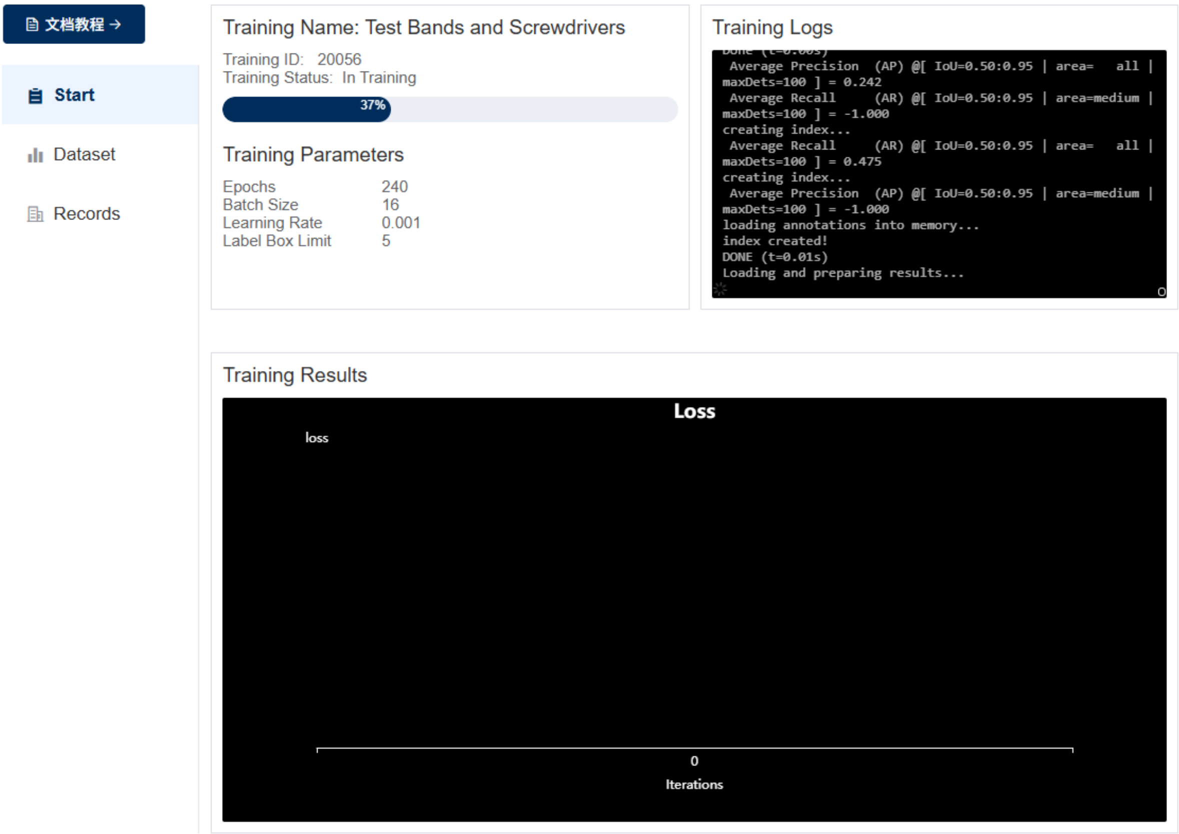
Training Parameters

```
Epochs           240
Batch Size       16
Learning Rate    0.001
Label Box Limit  5
```

Training Logs

Training Results

The training takes a long time and it is uncertain how long it will take. Closing the webpage or turning off the computer will not affect the training. You can do other things first.



Training Name: Test Bands and Screwdrivers

Training ID: 20056
Training Status: In Training

Training Parameters

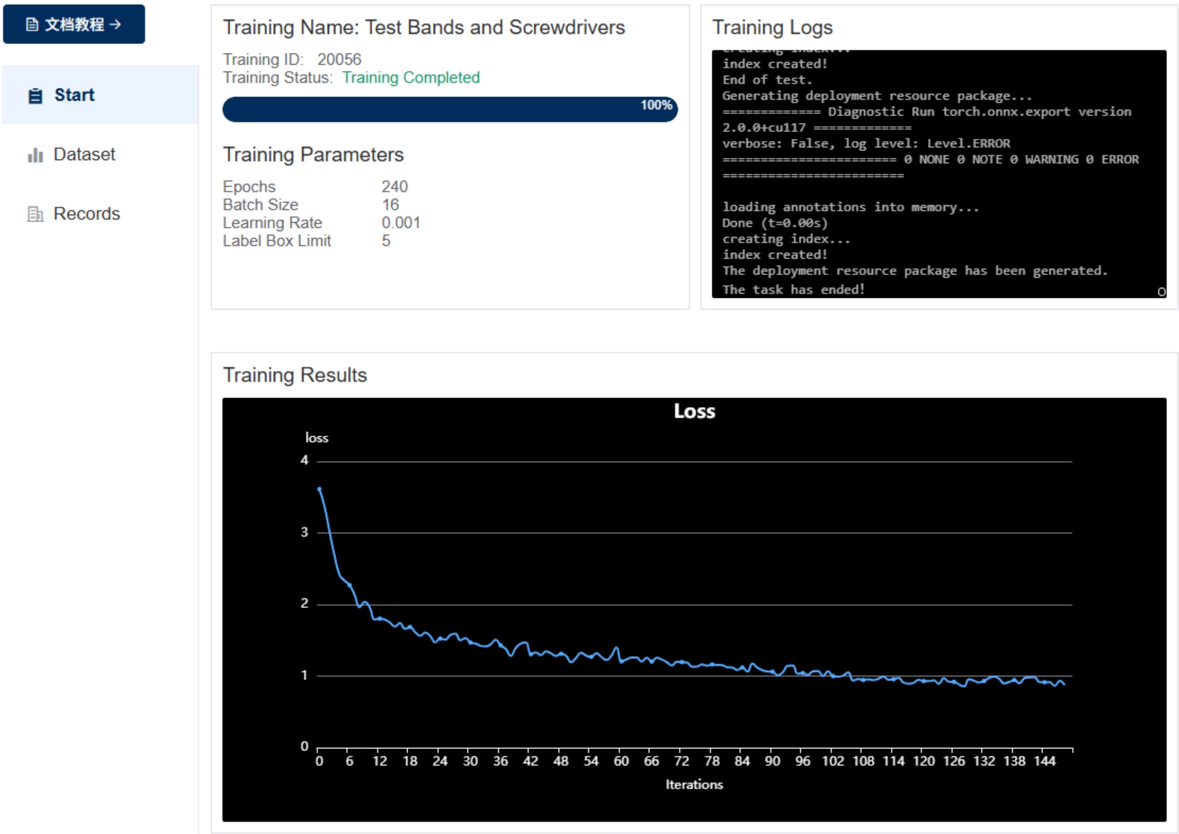
Epochs	240
Batch Size	16
Learning Rate	0.001
Label Box Limit	5

Training Logs

```
Done (t=0.00s)
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all |
maxDets=100 ] = 0.242
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium |
maxDets=100 ] = -1.000
creating index...
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all |
maxDets=100 ] = 0.475
creating index...
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium |
maxDets=100 ] = -1.000
loading annotations into memory...
index created!
DONE (t=0.01s)
Loading and preparing results...
```

Training Results

After waiting for the training to end, the interface is as follows



Training Name: Test Bands and Screwdrivers

Training ID: 20056
Training Status: Training Completed

Training Parameters

Epochs	240
Batch Size	16
Learning Rate	0.001
Label Box Limit	5

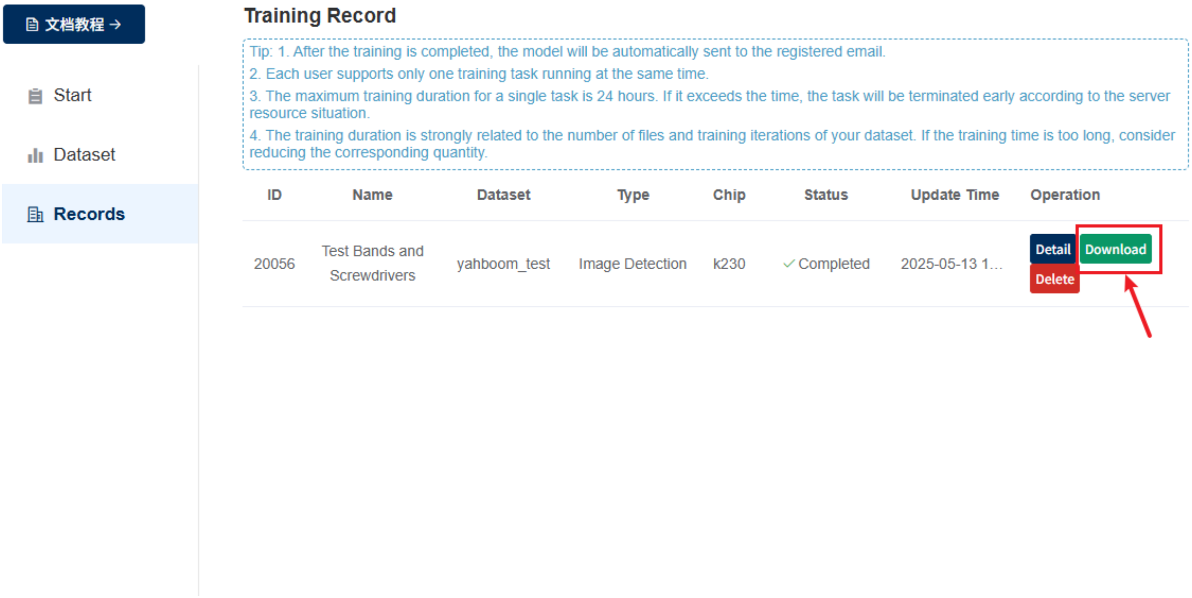
Training Logs

```
creating index...
index created!
End of test.
Generating deployment resource package...
============= Diagnostic Run torch.onnx.export version
2.0.0+cu117 =============
verbose: False, log level: Level.ERROR
======================= 0 NONE 0 NOTE 0 WARNING 0 ERROR
========================

loading annotations into memory...
Done (t=0.00s)
creating index...
index created!
The deployment resource package has been generated.
The task has ended!
```

Training Results

We return to the place before starting training, find the training record, and click Download



Training Record

Tip: 1. After the training is completed, the model will be automatically sent to the registered email.
2. Each user supports only one training task running at the same time.
3. The maximum training duration for a single task is 24 hours. If it exceeds the time, the task will be terminated early according to the server resource situation.
4. The training duration is strongly related to the number of files and training iterations of your dataset. If the training time is too long, consider reducing the corresponding quantity.

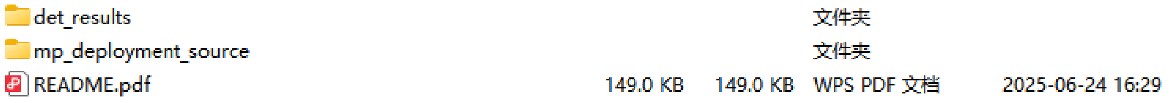
ID	Name	Dataset	Type	Chip	Status	Update Time	Operation
20056	Test Bands and Screwdrivers	yahboom_test	Image Detection	k230	✓ Completed	2025-05-13 1...	Detail Download Delete

3. Use the code and model we downloaded

The following is the latest version of the tutorial on 2025.06.10

> Note: If you are using firmware 1.3.3 or earlier, please update to the latest version. The latest version of the firmware can be downloaded from our official website tutorial page

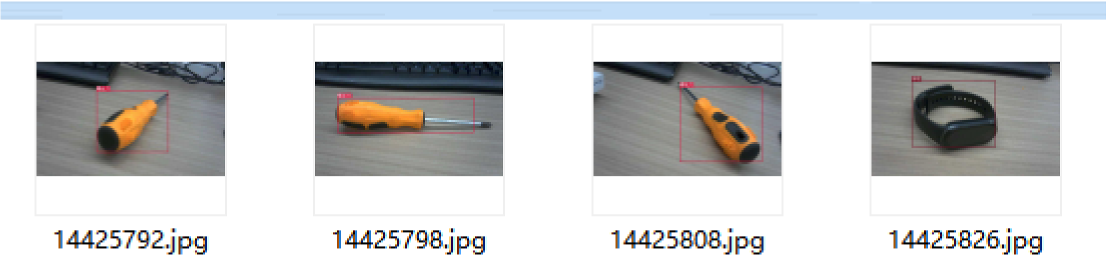
The downloaded compressed package is as follows



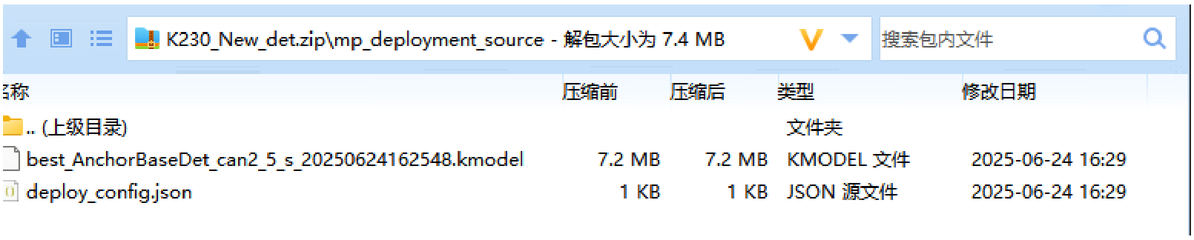
det_results			文件夹	
mp_deployment_source			文件夹	
README.pdf	149.0 KB	149.0 KB	WPS PDF 文档	2025-06-24 16:29

The [det_results] directory contains the results of our test verification, we can open it and take a look.

Here are the results of the verification test. If the object in the picture is correctly framed, then basically there is no big problem with the trained model



Then we open the [mp_deployment_source] folder, which contains our model file and configuration file

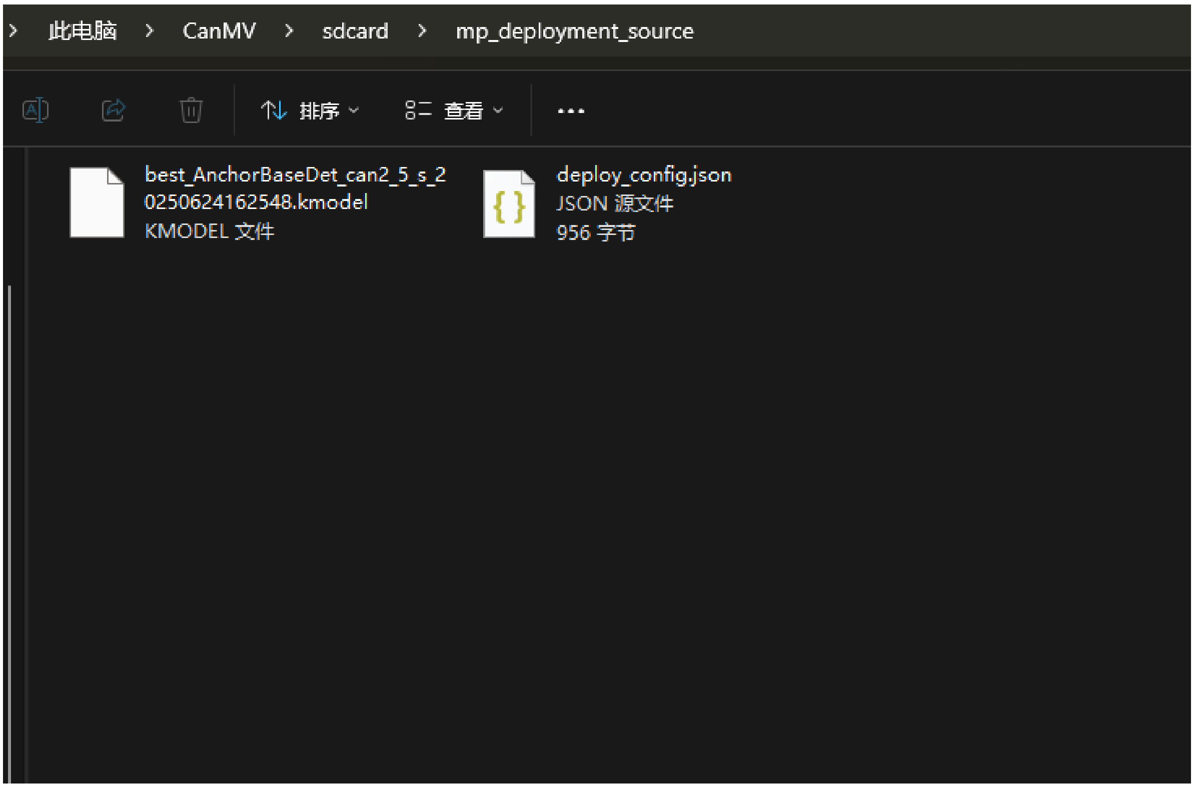


K230_New_det.zip\mp_deployment_source - 解包大小为 7.4 MB

称	压缩前	压缩后	类型	修改日期
.. (上级目录)			文件夹	
best_AnchorBaseDet_can2_5_s_20250624162548.kmodel	7.2 MB	7.2 MB	KMODEL 文件	2025-06-24 16:29
deploy_config.json	1 KB	1 KB	JSON 源文件	2025-06-24 16:29

Then we return to the sdcard directory of CanMV and create a new folder named [mp_deployment_source]

Put this in the compressed package Unzip the two files to this folder (first unzip to your computer, then copy to the mp_deployment_source folder)

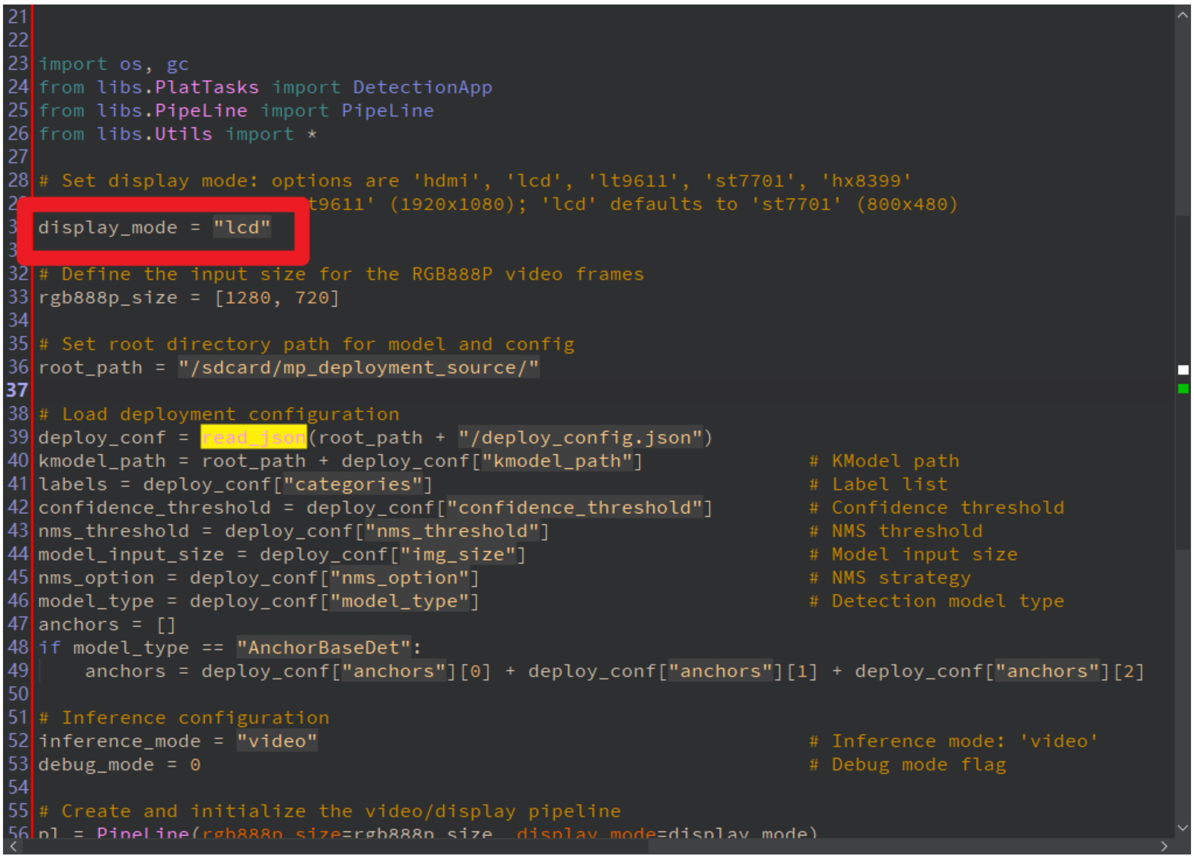


best_AnchorBaseDet_can2_5_s_20250624162548.kmodel
KMODEL 文件

deploy_config.json
JSON 源文件
956 字节

Then open the example source code summary [13.training/deploy_det_video.py]

Change display_mode to "lcd"



```


import os, gc
from libs.PlatTasks import DetectionApp
from libs.PipeLine import PipeLine
from libs.Utils import *

# Set display mode: options are 'hdmi', 'lcd', 'lt9611', 'st7701', 'hx8399'
                          t9611' (1920x1080); 'lcd' defaults to 'st7701' (800x480)
display_mode = "lcd"

# Define the input size for the RGB888P video frames
rgb888p_size = [1280, 720]

# Set root directory path for model and config
root_path = "/sdcard/mp_deployment_source/"

# Load deployment configuration
deploy_conf = read_json(root_path + "/deploy_config.json")
kmodel_path = root_path + deploy_conf["kmodel_path"]              # KModel path
labels = deploy_conf["categories"]                                # Label list
confidence_threshold = deploy_conf["confidence_threshold"]        # Confidence threshold
nms_threshold = deploy_conf["nms_threshold"]                      # NMS threshold
model_input_size = deploy_conf["img_size"]                        # Model input size
nms_option = deploy_conf["nms_option"]                            # NMS strategy
model_type = deploy_conf["model_type"]                            # Detection model type
anchors = []
if model_type == "AnchorBaseDet":
    anchors = deploy_conf["anchors"][0] + deploy_conf["anchors"][1] + deploy_conf["anchors"][2]

# Inference configuration
inference_mode = "video"                                          # Inference mode: 'video'
debug_mode = 0                                                    # Debug mode flag

# Create and initialize the video/display pipeline
pl = PipeLine(rgb888p_size=rgb888p_size, display_mode=display_mode)
```

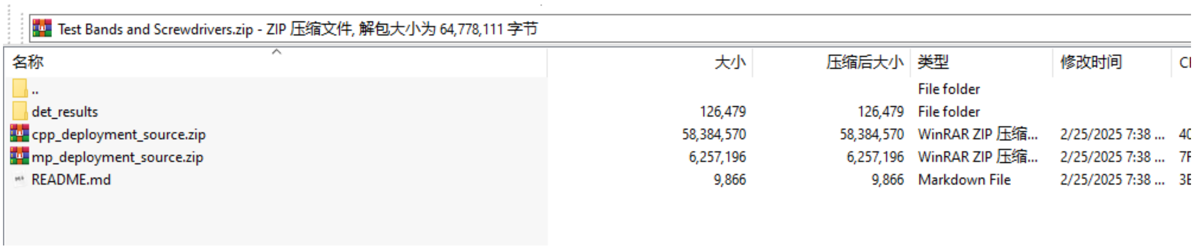
After the modification, click the run button in the lower left corner of the IDE to see the effect of the case running.

> Note: After testing, the new version of the tutorial may drop frames after deployment. At this time, you can try to use the old version of the tutorial to deploy.

> The det_video.py file required by the old version of the tutorial has been placed in [Source Code Summary / 13.training / det_video.py]

The following is an old version tutorial, which is only used for reference and backup.

The downloaded file is a compressed file with the following contents



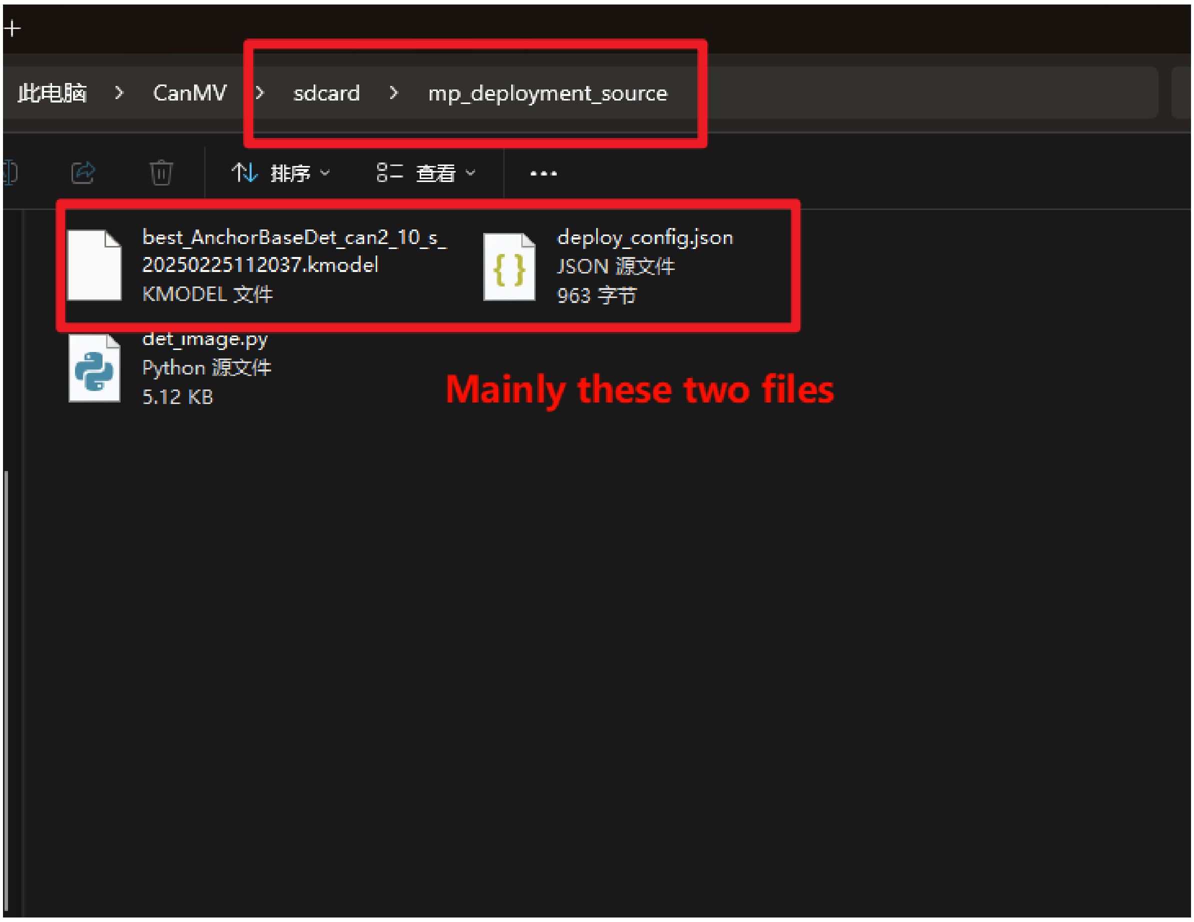
Test Bands and Screwdrivers.zip - ZIP 压缩文件, 解包大小为 64,778,111 字节

名称	大小	压缩后大小	类型	修改时间	C
..			File folder		
det_results	126,479	126,479	File folder		
cpp_deployment_source.zip	58,384,570	58,384,570	WinRAR ZIP 压缩...	2/25/2025 7:38 ...	4(
mp_deployment_source.zip	6,257,196	6,257,196	WinRAR ZIP 压缩...	2/25/2025 7:38 ...	7F
README.md	9,866	9,866	Markdown File	2/25/2025 7:38 ...	3E

We can first look at the det_results folder, which contains the results of the verification test. If the objects in the picture are correctly framed, then basically there is no big problem with the trained model.

Next, we unzip mp_deployment_source and create a mp_deployment_source folder in the /sdcard/ directory of K230, and put the unzipped contents into it.

(Mainly put the model file ending with .kmodel and the configuration file ending with .json in it. The remaining two py files are used for running and can be left out)



此电脑 > CanMV > sdcard > mp_deployment_source

best_AnchorBaseDet_can2_10_s_20250225112037.kmodel
KMODEL 文件

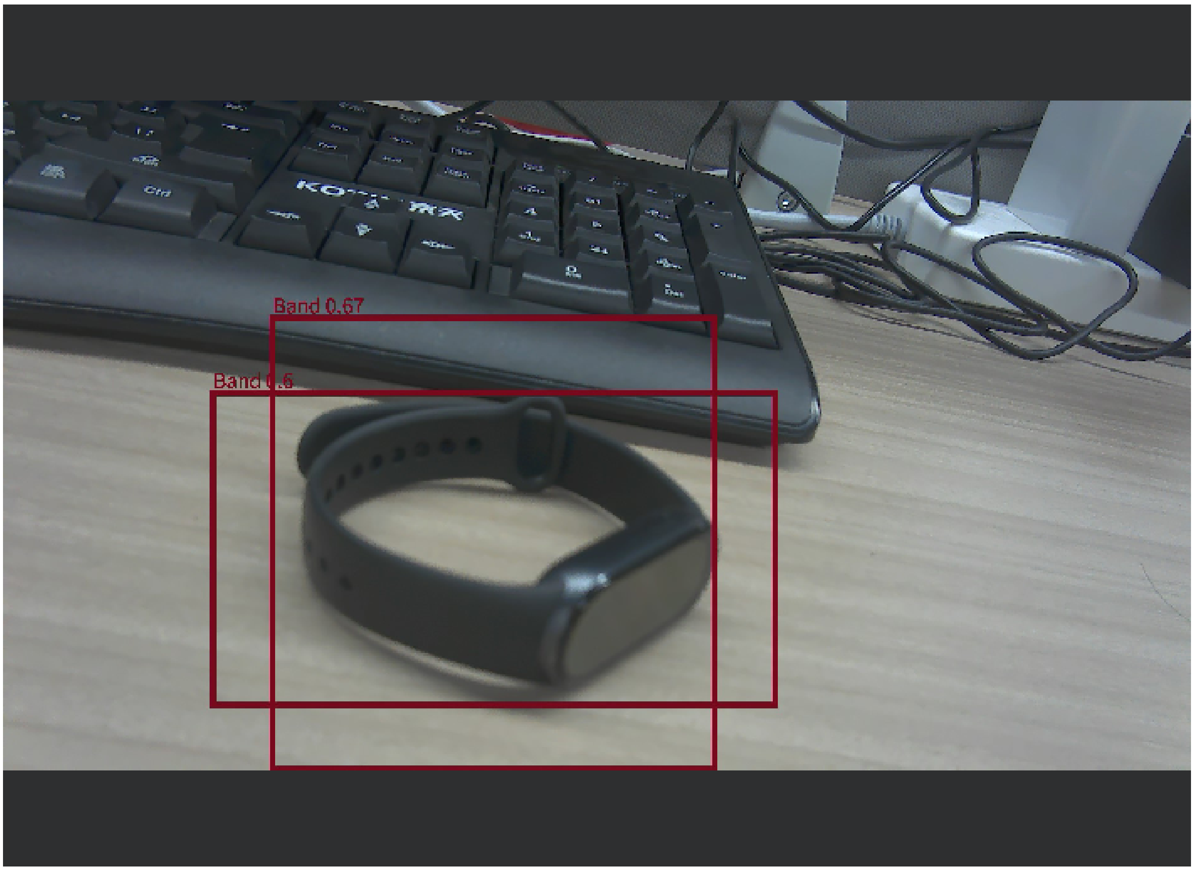
deploy_config.json
JSON 源文件
963 字节

det_image.py
Python 源文件
5.12 KB

Then we open CanMV IDE, copy the code of det_video.py and run it, and we can see that K230 can detect the objects we just marked.

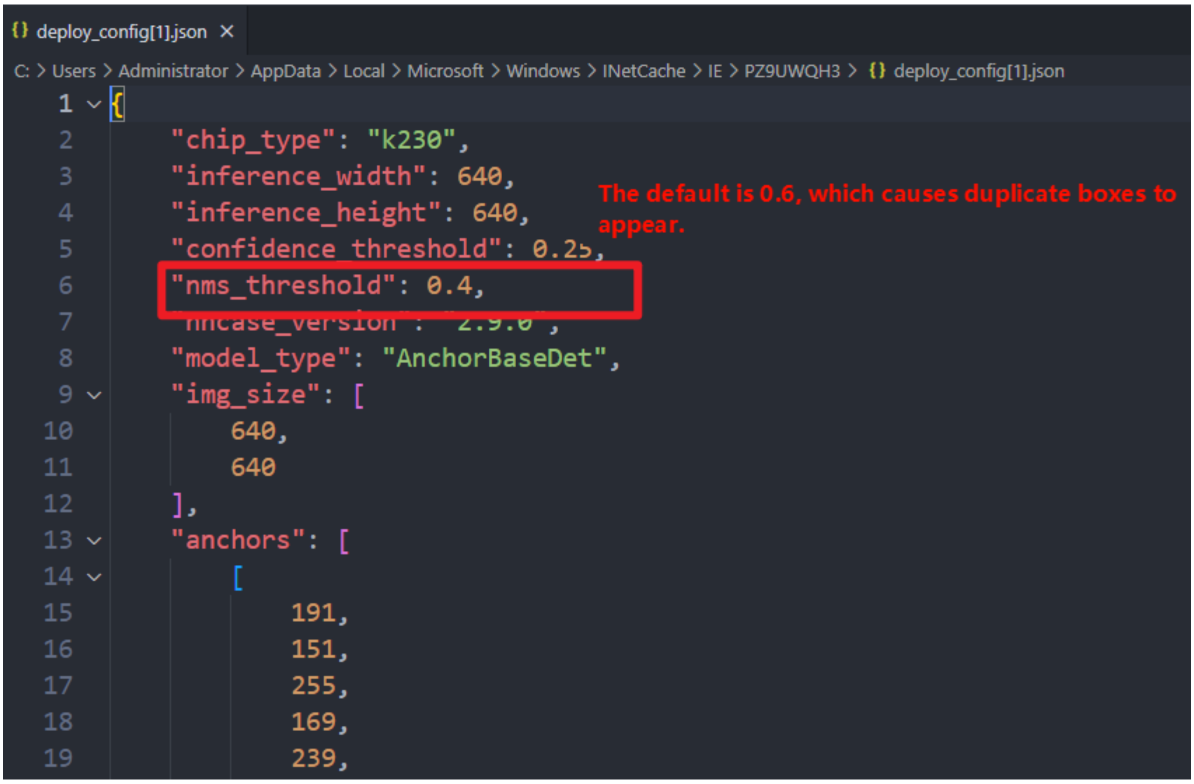
> Here we need to make some modifications to det_video.py:
>
> The original code is as follows:

> ```
> display_mode="hdmi"
> if display_mode=="lcd":
> DISPLAY_WIDTH = ALIGN_UP(800, 16)
> DISPLAY_HEIGHT = 480
> else:
> DISPLAY_WIDTH = ALIGN_UP(1920, 16)
> DISPLAY_HEIGHT = 1080
>
> OUT_RGB888P_WIDTH = ALIGN_UP(1280, 16)
> OUT_RGB888P_HEIGH = 720
> ```
>
> Modified to:
>
> ```
> display_mode="lcd"
> if display_mode=="lcd":
> DISPLAY_WIDTH = ALIGN_UP(640, 16)
> DISPLAY_HEIGHT = 480
> else:
> DISPLAY_WIDTH = ALIGN_UP(1920, 16)
> DISPLAY_HEIGHT = 1080
>
> OUT_RGB888P_WIDTH = ALIGN_UP(1280, 16)
> OUT_RGB888P_HEIGH = 720
> ```



4. Effect Optimization

At this time, we find that there are too many overlapping frames, so we can try to reduce the nms threshold in deploy_config, so that there will be fewer duplicate frames.



deploy_config[1].json ×

C: > Users > Administrator > AppData > Local > Microsoft > Windows > INetCache > IE > PZ9UWQH3 > deploy_config[1].json

```
{
    "chip_type": "k230",
    "inference_width": 640,
    "inference_height": 640,
    "confidence_threshold": 0.25,
    "nms_threshold": 0.4,
    "nncase_version": "2.9.0",
    "model_type": "AnchorBaseDet",
    "img_size": [
        640,
        640
    ],
    "anchors": [
        [
            191,
            151,
            255,
            169,
            239,
```

The default is 0.6, which causes duplicate boxes to appear.