yolo

- yolov8□□□□
- yolo□□□□□□□&cv

yolov8□□□□

□□□□

https://docs.ultralytics.com/

□□

```
# □□□□□□□□□□
# https://pypi.tuna.tsinghua.edu.cn/simple/

# □□□□□□□□□□
pip list

# □□□□
pip uninstall <package-Name>

# □□opencv
pip install python-opencv -i https://pypi.tuna.tsinghua.edu.cn/simple/

# □□opencv □□
pip install opencv-contrib-python  -i https://pypi.tuna.tsinghua.edu.cn/simple/

# □□pytorch
# https://pytorch.org/get-started/locally/
pip install torch torchvision torchaudio -i https://pypi.tuna.tsinghua.edu.cn/simple/

# □□yolov8
# https://docs.ultralytics.com/quickstart/
pip install ultralytics -i https://pypi.tuna.tsinghua.edu.cn/simple/
```

□□□□

```
# □□□□□
yolo TASK MODE ARGS
```

```
# TASK[ detect : [], [segment [], [classify [], [pose []
# MODE [ train: [], [ val[], [ predict[]/[], [ export: [], [ track: []
```

```
# []yolov8n.pt []
yolo detect predict model=". /yolo/yolov8n.pt" source=". /images/1.jpg"
```




```
# []yolov8n-seg.pt []
yolo segment predict model=". /yolo/yolov8n-seg.pt" source=". /images/1.jpg"
```



□□□□

```
# □□coco128□□□□□□□□□
yolo detect train data=./yolo/coco128.yaml model=./yolo/yolov8n.pt epochs=100 imgsz=640
```

```
# □□□□□□□□yaml□□□□
# □□□□□https://github.com/ultralytics/ultralytics/tree/main/ultralytics/cfg/datasets
# coco128□□□

path: ../datasets/coco128 # dataset root dir
train: images/train2017 # train images (relative to 'path') 128 images
val: images/train2017 # val images (relative to 'path') 128 images
test: # test images (optional)

# Classes
names:
  0: person
  1: bicycle

download: https://ultralytics.com/assets/coco128.zip
```

python□□

```
from ultralytics import YOLO

model = YOLO('./yolo/yolov8n.pt')  # load a pretrained model (recommended for training)
results = model.train(data='./images/coco128.yaml', epochs=100, imgsz=640)
```

yolo读取视频并检测&cv

这是一段示例代码，用于跑通基础验证，不可用于生产，除非你的视频路数少于4路

```
from time import sleep
import cv2 as cv, torch
from ultralytics import YOLO
import logging
import time

logging.basicConfig(
    level=logging.INFO, format="%(asctime)s - %(name)s - %(levelname)s - %(message)s"
)
log = logging.getLogger(__name__)

# log.info("-- OpenCV Info --")
# log.info(cv.getBuildInformation())
# log.info("-" * 50)

# device = "cuda:0" if torch.cuda.is_available() else "cpu"
# log.info(f"Using device: {device}")

yolo_model = YOLO("./yolo_model/yolo11s.pt")
image_size = 640
yolo_model.overrides["imgsz"] = image_size

cls_map = {
    0: "person",
    1: "bicycle",
    2: "car",
    3: "motorcycle",
    5: "bus",
    7: "truck",
}

# 帧率
```

```
FPS_TARGET = 12
# □□□
T_INTERVAL = 1.0 / FPS_TARGET
#  □□□□□
last_t = time.time()
# □□□□
detect_every = 3
# □□□□
detect_count = 0

cap = cv.VideoCapture(
    "https://smart.saas.vppark.cn/oss/1.mp4",
)



while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break
    # if time.time() - last_t < T_INTERVAL:
    #     continue
    # last_t = time.time()



    detect_count += 1
    if detect_count % detect_every == 0 or True:
        # □□ resize
        h0, w0 = frame.shape[:2]
        scale = min(image_size / h0, image_size / w0)
        h1, w1 = int(h0 * scale), int(w0 * scale)
        frame_resize = cv.resize(frame, (w1, h1), interpolation=cv.INTER_LINEAR)

        # □□□□
        results = yolo_model(
            frame_resize, imgsz=image_size, classes=list(cls_map.keys())
        )
        # □□□□□□□
        for result in results:
            boxes_data = result.boxes.data.clone()
            boxes_data[..., :4] /= scale
            result.boxes.data = boxes_data
```

```
            react_frame = result.plot(img=frame, line_width=2)
            cv.imshow("frame", react_frame)
    else:
        cv.imshow("frame", frame)
    # if cv.waitKey(int(1000 / FPS_TARGET)) & 0xFF == ord("q"):
    if cv.waitKey(1) & 0xFF == ord("q"):
        break

cap.release()
cv.destroyAllWindows()
```