YOLO training code：

|  |
| --- |
| from ultralytics import YOLO  import multiprocessing  # Load a model  model = YOLO("yolov8n.pt") # The folder of train.py will download a file named yolov8n.pt, which is used as the training model. It's recommended to load a pretrained model for training.  if \_\_name\_\_ == '\_\_main\_\_':  multiprocessing.freeze\_support()  results = model.train(  data="2LED\_detect.yaml", # Location of the yaml file: Lib\site-packages\ultralytics\cfg\datasets  epochs=10) # Train the model |

YOLO calculate average brightness ,image sending, and light signal switching code：

|  |
| --- |
| import cv2  import os  from datetime import datetime  import time  from ultralytics import YOLO  from ultralytics.utils.plotting import Annotator  import requests  from pyfirmata2 import Arduino  # Use AUTODETECT to automatically detect the Arduino board  port = 'COM?'  board = Arduino(port)  # Set the pins to be controlled  pin = 11 # Abnormal signal light  pin1 = 10 # Normal signal light  board.digital[pin].write(1)  board.digital[pin1].write(1)  # Add imgwrite function  def imgwrite(img, save\_dir, prefix):  if not os.path.exists(save\_dir):  os.makedirs(save\_dir)  current\_time = datetime.now().strftime("%Y-%m-%d\_%H-%M-%S")  filename = f"{save\_dir}/{prefix}\_{current\_time}.jpg"  cv2.imwrite(filename, img)  return filename  def put\_text\_top\_center(frame, text, font=cv2.FONT\_HERSHEY\_SIMPLEX, font\_scale=0.8, color=(255, 255, 255),  thickness=1):  text\_width, text\_height = cv2.getTextSize(text, font, font\_scale, thickness)[0]  x = (frame.shape[1] - text\_width) // 2  y = frame.shape[0] - 60  cv2.rectangle(frame, (x - 5, y - text\_height - 5), (x + text\_width + 5, y + 5), (0, 0, 0), -1)  cv2.putText(frame, text, (x, y), font, font\_scale, color, thickness)  def put\_text\_bottom\_center(frame, text, font=cv2.FONT\_HERSHEY\_SIMPLEX, font\_scale=0.8, color=(255, 255, 255),  thickness=1):  text\_width, text\_height = cv2.getTextSize(text, font, font\_scale, thickness)[0]  x = (frame.shape[1] - text\_width) // 2  y = frame.shape[0] - text\_height  cv2.rectangle(frame, (x - 5, y - text\_height - 5), (x + text\_width + 5, y + 5), (0, 0, 0), -1)  cv2.putText(frame, text, (x, y), font, font\_scale, color, thickness)  # Open the computer's camera  cap = cv2.VideoCapture(1)  mean\_brightness1 = 0 # Or other appropriate initial value  # Load the YOLOv8 model  model = YOLO('Your model.pt')  names = model.model.names  target\_object\_index = 0  while True:  ret, frame = cap.read()  if ret:  blue\_ratio = frame[:, :, 0].mean() / 255.0  green\_ratio = frame[:, :, 1].mean() / 255.0  red\_ratio = frame[:, :, 2].mean() / 255.0  gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)  mean\_brightness = gray.mean()  # Frame without text  frame\_no\_text = frame.copy()  put\_text\_top\_center(frame, f"Mean Brightness: {mean\_brightness:.2f}")  put\_text\_bottom\_center(frame,  f"B Ratio: {blue\_ratio:.2f}, G Ratio: {green\_ratio:.2f}, R Ratio: {red\_ratio:.2f}")  # Perform YOLO object detection  results = model.predict(frame)  boxes = results[0].boxes.xyxy.cpu().numpy().astype(int)  classes = results[0].boxes.cls.tolist()  annotator = Annotator(frame, line\_width=2, example=str(names))  for box, cls in zip(boxes, classes):  if cls == target\_object\_index:  annotator.box\_label(box, "Target Object Detected", (255, 42, 4))  start\_x, start\_y, end\_x, end\_y = box  roi = frame\_no\_text[start\_y:end\_y, start\_x:end\_x]  roi1 = frame\_no\_text[start\_y:end\_y, start\_x:end\_x]  gray\_roi1 = cv2.cvtColor(frame\_no\_text, cv2.COLOR\_BGR2GRAY)  color\_map\_roi1 = cv2.applyColorMap(roi1, cv2.COLORMAP\_JET)  gray\_roi = cv2.cvtColor(frame\_no\_text, cv2.COLOR\_BGR2GRAY)  color\_map\_roi = cv2.applyColorMap(gray\_roi, cv2.COLORMAP\_JET)  b\_ratio = roi[:, :, 0].mean() / 255.0  g\_ratio = roi[:, :, 1].mean() / 255.0  r\_ratio = roi[:, :, 2].mean() / 255.0  gray = cv2.cvtColor(roi, cv2.COLOR\_BGR2GRAY)  mean\_brightness1 = gray.mean()  # RGB changes for each frame where an object is detected  put\_text\_top\_center(color\_map\_roi, f"Detect Object Brightness and RGB: {mean\_brightness1:.2f}")  put\_text\_bottom\_center(color\_map\_roi,  f"R Ratio: {r\_ratio:.2f}, G Ratio: {g\_ratio:.2f}, B Ratio: {b\_ratio:.2f}")  cv2.imshow('Color Map ROI', color\_map\_roi)  cv2.imshow('frame', frame)  if mean\_brightness1 < 50:  current\_time = datetime.now().strftime("%Y-%m-%d\_%H-%M-%S")  # Save the image to the specified folders  os.makedirs('original', exist\_ok=True)  os.makedirs('roi', exist\_ok=True)  os.makedirs('intensity', exist\_ok=True)  filename = f"original/{current\_time}\_low\_brightness\_screenshot.jpg"  roi\_filename = f"roi/roi\_{current\_time}.png"  intensity\_filename = f"intensity/intensity\_{current\_time}.png"  cv2.imwrite(filename, frame\_no\_text)  cv2.imwrite(roi\_filename, roi)  cv2.imwrite(intensity\_filename, color\_map\_roi1)  # Save the image to the webcam\_photos folder  webcam\_photos\_directory = 'webcam\_photos'  imgwrite(frame\_no\_text, webcam\_photos\_directory, 'original')  imgwrite(roi, webcam\_photos\_directory, 'roi')  imgwrite(color\_map\_roi1, webcam\_photos\_directory, 'intensity')  board.digital[pin].write(1) # Abnormal signal light  board.digital[pin1].write(0) # Normal signal light  # Line notification code  token = 'Your Token type here' # Line Token can be adjusted to your own setting  headers = {'Authorization': 'Bearer ' + token}  files\_frame\_ = {'imageFile': open(roi\_filename, 'rb'), 'imageFile2': open(filename, 'rb'),  'imageFile3': open(intensity\_filename, 'rb')}  message\_frame\_ = f"Low brightness detected, Mean Brightness: {mean\_brightness1:.2f}"  payload\_frame\_ = {'message': message\_frame\_}  r\_frame\_ = requests.post('https://notify-api.line.me/api/notify', headers=headers,  params=payload\_frame\_, files=files\_frame\_)  time\_delay\_seconds\_frame\_ = 5  time.sleep(time\_delay\_seconds\_frame\_)  print(r\_frame\_.status\_code)  print(r\_frame\_.text)  time.sleep(3)  else:  board.digital[pin].write(0) # Off  board.digital[pin1].write(1) # On  if cv2.waitKey(1) & 0xFF == ord('q'):  break  cap.release() |