Addis Ababa University

Master's in Artificial Intelligence

Computer Vision (CV) Laboratory Manual

Prepared by: Dr. Natnael Argaw Wondimu

Environment: OpenCV and Visual Studio

Preface

This manual is prepared in a workbook style for AI Master's students at Addis Ababa University. It provides a hands-on introduction to Computer vision with OpenCV, covering advanced topics. Each lab includes objectives, theoretical background, procedures, OpenCV code, checkpoints, try-it-yourself prompts, and collaborative assignments. The manual is intended to serve as a base for advanced studies in Computer Vision.

Chapter 11: Real-Time Object Detection (YOLOv8, SSD, MobileNet – OpenCV DNN)

Objective

To implement real-time object detection using pre-trained deep learning models like YOLOv8, SSD, and MobileNet-SSD with OpenCV's DNN module. This lab demonstrates how to load models, process input, and visualize detections using OpenCV.

1. What is Real-Time Object Detection?

Description: Real-time object detection involves identifying objects in images or video streams with low latency. Models like YOLO, SSD, and MobileNet-SSD are optimized for speed and accuracy.

2. Requirements

pip install opencv-python ultralytics numpy

- **OpenCV** for real-time video and DNN handling
- **Ultralytics** package for running YOLOv8 (or use exported ONNX model)

3. YOLOv8 with Ultralytics

3.1 Load and Run YOLOv8 Model

```
from ultralytics import YOLO
import cv2

model = YOLO('yolov8n.pt')  # Or yolov8s.pt for higher accuracy
cap = cv2.VideoCapture(0)

while True:
  ret, frame = cap.read()
  results = model.predict(source=frame, show=True, conf=0.5)
  if cv2.waitKey(1) & 0xFF == ord('q'):
    break

cap.release()
cv2.destroyAllWindows()
```

Output Description: Live object detection using YOLOv8 with bounding boxes and class labels.

4. SSD with OpenCV DNN Module

4.1 Load Pretrained SSD Model

```
net = cv2.dnn.readNetFromCaffe('deploy.prototxt', 'mobilenet_iter_73000.caffemodel')
cap = cv2.VideoCapture(0)

while True:
  ret, frame = cap.read()
```

```
    h, w = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(frame, 0.007843, (300, 300), 127.5)
    net.setInput(blob)
    detections = net.forward()

    for i in range(detections.shape[2]):
        confidence = detections[0, 0, i, 2]
        if confidence > 0.5:
            idx = int(detections[0, 0, i, 1])
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (x1, y1, x2, y2) = box.astype("int")
            cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)

    cv2.imshow("SSD Detection", frame)
    if cv2.waitKey(1) == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Output Description: Runs MobileNet-SSD object detection on a live webcam feed with bounding boxes.

5. Using YOLOv8 with ONNX in OpenCV

5.1 Export YOLOv8 to ONNX and Load with OpenCV

```
net = cv2.dnn.readNetFromONNX('yolov8n.onnx')
blob = cv2.dnn.blobFromImage(frame, 1/255.0, (640, 640), swapRB=True, crop=False)
net.setInput(blob)
out = net.forward()
# Post-process results (non-max suppression, label drawing)
```

Note: ONNX support allows OpenCV integration for deployment on CPU/GPU.

6. Summary

- **YOLOv8**: State-of-the-art performance, ultrafast and highly accurate.
- **SSD + MobileNet**: Lightweight and easy to integrate with OpenCV.
- **OpenCV DNN**: Enables hardware-independent deep learning inference.

Suggested Exercises

1. Replace YOLOv8 with YOLOv5 or YOLOv7.
2. Benchmark SSD and YOLOv8 FPS on your device.
3. Train a YOLO model on a small custom dataset and test in real-time.
4. Use OpenCV DNN to run a COCO-trained ONNX model without Ultralytics.