yolo

November 21, 2023

1 ECE685 ZC1 Project Weekly Report - Week 2

1.1 Object Detection

1.1.1 Try:

- **Train Faster R-CNN on the SportMoT**:
  - The effect is not really good. The reason may be that we implement it by ourselves and there are some unseen bugs. The next step may be using other mature libraries (Detectron2) to train the model.
- **Yolov3 & Yolov8 on the SportMoT dataset**:
  - Considering the computational complexity and the performance, Yolov8 might be a better choice. However, Yolo can only detect the "Person" object but it is hard for it to recognize the identity of the athlete. Therefore, another identifier is required to determine whether the object is the person we want to track.
- **Try Detectron2 to get the second object detection model**

1.2 Plan for Next Step

1.2.1 Initialize Deep SORT

- **Deep SORT requires the bounding box coordinates and associated feature vectors for each detected object**. You can extract feature vectors using a separate deep learning model (often a CNN trained for re-identification tasks).

1.2.2 Track Objects Across Frames

- **Deep SORT uses these feature vectors along with the bounding box coordinates to track objects across frames**.
- It assigns a unique identifier to each tracked object, which remains consistent across frames as long as the object is being tracked.

1.2.3 Extracting Central Points and Time Series Data

- **For each tracked object, you can calculate the central point of the bounding box**.
- Store these central points along with their unique identifiers and frame numbers to create a time series of object movements.

[ ]:
```
from ultralytics import YOLO
```

```
model = YOLO('yolov8m.pt')

PATH = 'dataset/train/v_-6Os86HzwCs_c009/img1/'

PIC = PATH + '000001.jpg'
```

[ ]:
```
results = model(PIC)

for result in results:
    boxes = result.boxes
    masks = result.masks
    keypoints = result.keypoints
    probs = result.probs
```

```
image 1/1 g:\My Drive\US_used\DUKE\ECE685DL\Project\ECE685-CDV\dataset\train\v_-
6Os86HzwCs_c009\img1\000001.jpg: 384x640 13 persons, 192.0ms
Speed: 2.0ms preprocess, 192.0ms inference, 2.0ms postprocess per image at shape
(1, 3, 384, 640)
```

[ ]:
```
boxes, masks, keypoints, probs
```

[ ]:
```
(ultralytics.engine.results.Boxes object with attributes:

 cls: tensor([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
device='cuda:0')
 conf: tensor([0.8503, 0.8389, 0.8297, 0.7982, 0.7873, 0.7470, 0.7244, 0.7209,
0.7135, 0.7062, 0.5021, 0.3182, 0.2551], device='cuda:0')
 data: tensor([[1.0364e+03, 3.7078e+02, 1.0997e+03, 4.9169e+02, 8.5031e-01,
0.0000e+00],
         [4.5961e+02, 4.9646e+02, 5.1498e+02, 5.9427e+02, 8.3886e-01,
0.0000e+00],
         [2.9584e+02, 4.4958e+02, 3.8646e+02, 5.7867e+02, 8.2973e-01,
0.0000e+00],
         [1.0513e+02, 2.4759e+02, 1.4464e+02, 3.8256e+02, 7.9815e-01,
0.0000e+00],
         [2.2769e+02, 4.0341e+02, 3.0315e+02, 5.3294e+02, 7.8728e-01,
0.0000e+00],
         [8.6094e+02, 2.7767e+02, 8.8958e+02, 3.7341e+02, 7.4696e-01,
0.0000e+00],
         [3.8267e+02, 2.6870e+02, 4.3876e+02, 3.4803e+02, 7.2441e-01,
0.0000e+00],
         [9.5023e+02, 2.5390e+02, 9.7299e+02, 3.2932e+02, 7.2088e-01,
0.0000e+00],
         [1.1111e+03, 2.2882e+02, 1.1442e+03, 3.1772e+02, 7.1354e-01,
0.0000e+00],
         [7.6777e+02, 3.5741e+02, 8.1291e+02, 4.6400e+02, 7.0618e-01,
0.0000e+00],
        [4.1839e+02, 1.8488e+02, 4.4949e+02, 2.7364e+02, 5.0212e-01,
0.0000e+00],
        [9.5095e-02, 1.7810e+02, 2.7172e+01, 2.1638e+02, 3.1820e-01,
0.0000e+00],
        [1.2080e+03, 1.7972e+02, 1.2416e+03, 2.2431e+02, 2.5512e-01,
0.0000e+00]], device='cuda:0')
 id: None
 is_track: False
 orig_shape: (720, 1280)
 shape: torch.Size([13, 6])
 xywh: tensor([[1068.0822,  431.2354,   63.2649,  120.9159],
        [ 487.2960,  545.3634,   55.3694,   97.8098],
        [ 341.1534,  514.1236,   90.6183,  129.0886],
        [ 124.8819,  315.0758,   39.5130,  134.9729],
        [ 265.4161,  468.1788,   75.4611,  129.5307],
        [ 875.2583,  325.5441,   28.6401,   95.7394],
        [ 410.7149,  308.3674,   56.0836,   79.3268],
        [ 961.6139,  291.6058,   22.7593,   75.4194],
        [1127.6462,  273.2710,   33.0645,   88.9014],
        [ 790.3436,  410.7049,   45.1418,  106.5902],
        [ 433.9387,  229.2634,   31.1073,   88.7592],
        [  13.6338,  197.2401,   27.0773,   38.2897],
        [1224.8075,  202.0166,   33.5664,   44.5882]], device='cuda:0')
 xywhn: tensor([[0.8344, 0.5989, 0.0494, 0.1679],
        [0.3807, 0.7574, 0.0433, 0.1358],
        [0.2665, 0.7141, 0.0708, 0.1793],
        [0.0976, 0.4376, 0.0309, 0.1875],
        [0.2074, 0.6502, 0.0590, 0.1799],
        [0.6838, 0.4521, 0.0224, 0.1330],
        [0.3209, 0.4283, 0.0438, 0.1102],
        [0.7513, 0.4050, 0.0178, 0.1047],
        [0.8810, 0.3795, 0.0258, 0.1235],
        [0.6175, 0.5704, 0.0353, 0.1480],
        [0.3390, 0.3184, 0.0243, 0.1233],
        [0.0107, 0.2739, 0.0212, 0.0532],
        [0.9569, 0.2806, 0.0262, 0.0619]], device='cuda:0')
 xyxy: tensor([[1.0364e+03, 3.7078e+02, 1.0997e+03, 4.9169e+02],
        [4.5961e+02, 4.9646e+02, 5.1498e+02, 5.9427e+02],
        [2.9584e+02, 4.4958e+02, 3.8646e+02, 5.7867e+02],
        [1.0513e+02, 2.4759e+02, 1.4464e+02, 3.8256e+02],
        [2.2769e+02, 4.0341e+02, 3.0315e+02, 5.3294e+02],
        [8.6094e+02, 2.7767e+02, 8.8958e+02, 3.7341e+02],
        [3.8267e+02, 2.6870e+02, 4.3876e+02, 3.4803e+02],
        [9.5023e+02, 2.5390e+02, 9.7299e+02, 3.2932e+02],
        [1.1111e+03, 2.2882e+02, 1.1442e+03, 3.1772e+02],
        [7.6777e+02, 3.5741e+02, 8.1291e+02, 4.6400e+02],
        [4.1839e+02, 1.8488e+02, 4.4949e+02, 2.7364e+02],
         [9.5095e-02, 1.7810e+02, 2.7172e+01, 2.1638e+02],
         [1.2080e+03, 1.7972e+02, 1.2416e+03, 2.2431e+02]], device='cuda:0')
 xyxyn: tensor([[8.0973e-01, 5.1497e-01, 8.5915e-01, 6.8291e-01],
         [3.5907e-01, 6.8953e-01, 4.0233e-01, 8.2537e-01],
         [2.3113e-01, 6.2442e-01, 3.0192e-01, 8.0371e-01],
         [8.2129e-02, 3.4387e-01, 1.1300e-01, 5.3134e-01],
         [1.7788e-01, 5.6030e-01, 2.3683e-01, 7.4020e-01],
         [6.7261e-01, 3.8566e-01, 6.9498e-01, 5.1863e-01],
         [2.9896e-01, 3.7320e-01, 3.4278e-01, 4.8338e-01],
         [7.4237e-01, 3.5263e-01, 7.6015e-01, 4.5738e-01],
         [8.6806e-01, 3.1781e-01, 8.9389e-01, 4.4128e-01],
         [5.9982e-01, 4.9640e-01, 6.3509e-01, 6.4444e-01],
         [3.2686e-01, 2.5678e-01, 3.5117e-01, 3.8006e-01],
         [7.4293e-05, 2.4735e-01, 2.1228e-02, 3.0053e-01],
         [9.4377e-01, 2.4961e-01, 9.6999e-01, 3.1154e-01]], device='cuda:0'),
 None,
 None,
 None)
```

[ ]:
```python
from PIL import Image

for r in results:
    im_array = r.plot()  # plot a BGR numpy array of predictions
    im = Image.fromarray(im_array[..., ::-1])  # RGB PIL image
    im.show()
```

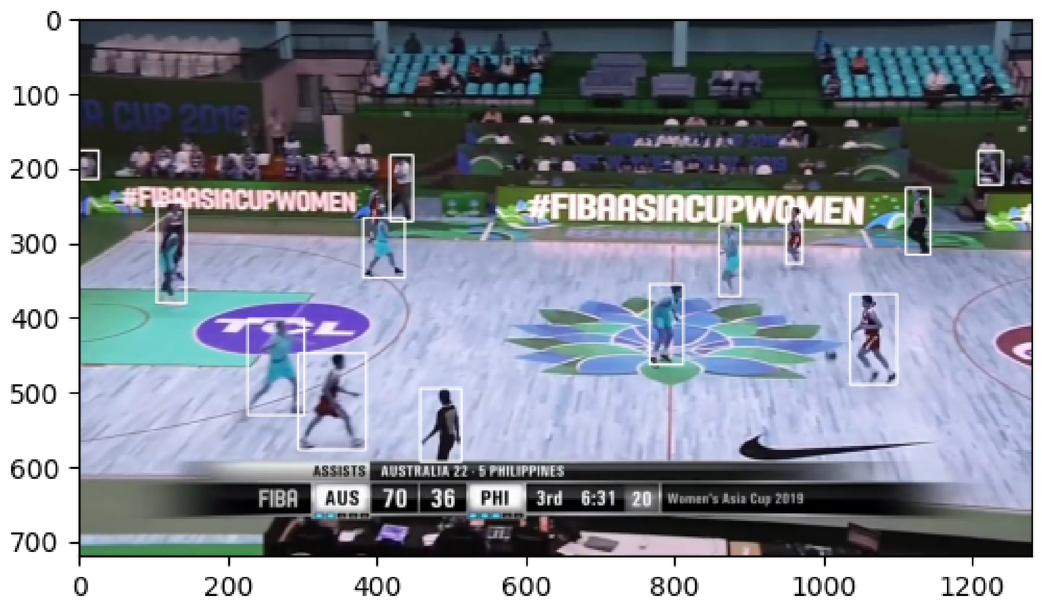
[ ]:
```python
import cv2
import matplotlib.pyplot as plt

for result in results:
    img = cv2.imread(PIC)
    boxes = result.boxes.cpu().numpy()
    for box in boxes:
        r = box.xyxy[0].astype(int)
        print(r)
        cv2.rectangle(img, tuple(r[:2]), tuple(r[2:]), (255, 255, 255), 2)

    # Display the image
    plt.imshow(img)                       # show img
```

```
[1036  370 1099  491]
[459 496 514 594]
[295 449 386 578]
[105 247 144 382]
[227 403 303 532]
[860 277 889 373]
[382 268 438 348]
[950 253 972 329]
```

```
[1111  228 1144  317]
[767 357 812 464]
[418 184 449 273]
[  0 178  27 216]
[1208  179 1241  224]
```



```
[ ]:
```