Preface
I chose this project because it fits my skills. Sports ball can have many types, but I only chose football to be my example target object. However, the method can be extended to any other ball games.

Introduction
Pseudo-labelling uses trained models to predict labels from our custom data. The models have not been trained on our data so they are weak labellers; in other word, they do not produce ground truth labels, but the label quality can be "close to ground truth". Thus, we can use them to produce semi-decent, massive amount of data, at very low cost (since there is no human labour involved).

In this project, I use Segment Anything Model 2 (Ravi et al., 2024) to produce labels. It has a powerful image encoder that the model can take in prompts from the users and produce corresponding masks in the proceeding frames. The prompts can be pixel locations, bounding boxes, or masks in the first frame of the videos.

The object detector I used is Yolov7 since it is fast and light for real-time detection. My object detection pipeline inspired by (Ferreira et al., 2023) is as followed:

- Step 1. Use SAM2 to produce two groups of data,

(1) The first group is train and validation set; their labels are verified by human eyes.
(2) The second group is pseudo-annotated set. Although SAM2 has produced its labels, they are not verified by human eyes.

- Step 2. Train and Validate Yolov7
- Step 3. Use trained Yolo to predict confidence for (2)
- Step 4. Add high-confidence samples from (2) to the train set in (1). Retrain model on the bigger train set and check if validation results are improved.
- Step 5. Iterative Step 3 and 4 to get an improved model.

Note that SAM2 can fail to predict correct labels. We tell SAM2 the coordinates of the ball in the first frame, and it detects it in all subsequent frames. However, some of the videos fade between different shots, and we would need to re-input the coordinates of the ball in the new shot or SAM2 will make poor predictions.

Experiment setup and results
I installed two software from the following repositories:
- SAM2: https://github.com/facebookresearch/sam2/tree/main
- Yolov7: https://github.com/WongKinYiu/yolov7

Firstly, I used SAM2 to create [a custom dataset] which holds train/validation/test and pseudo-annotated folders. Labels in the train_val_test/ dataset are efficiently verified by human eye by using this [GUI] which is custom-made. Therefore, they are treated as ground truth. Then, I moved the custom dataset to Yolov7 folder. There are a few files which I modified to train the model:
(1) Download a pretrained [weight], place it in yolov7/
(2) Create [dataset.yaml] and put it to custom_data/ if it is not there
(3) In the terminal, we train 20 epochs with a pretrained weight yolov7.pt:

```
python train.py --img-size 640 --cfg cfg/training/yolov7.yaml --hyp
data/hyp.scratch.custom.yaml --batch 16 --epochs 20 --data
custom_data/dataset.yaml --weights yolov7.pt --workers 24 --name
yolo_ball_det
```

After training, we can visit this [notebook] to iteratively add more data to train set and improve our benchmark through the next-run, or future-run trainings. For reproducing without training, download [first run best.pt], [second run best.pt], [third run best.pt] and [third+thres run best.pt]. Then, we can run the following command line to visualise our object detector performance:

```
python detect.py --weights runs/train/yolo_ball_det_2/weights/best.pt
--conf 0.25 --img-size 640 --source association_football_4.mp4
```

Testing results in two tables are shown as follows, with major improvements in bold:

Testing results in confidence distribution

run	detected	mean	std	min	25%	50%	75%	max	Add data
1	**28**	**0.31**	0.05	0.25	0.26	0.30	0.34	0.41	-
2	**84**	**0.63**	0.11	0.26	0.56	0.61	0.70	0.85	274
3	84	0.67	0.11	0.31	0.61	0.66	0.73	0.87	1046
3+thres	84	**0.74**	0.09	0.4	0.68	0.74	0.8	0.9	**141**

Testing results in precision, recall, mean Average Precision

run	P	R	mAP@.5	mAP@.5:.95
1	0.918	0.918	0.932	**0.698**
2	0.962	0.905	0.947	**0.788**
3	0.974	0.894	0.945	0.825
3+thres	0.974	0.906	0.951	0.82

From 1st run to 2nd:

1. Number of detected objects increase by 67% (total test sample: 85)
2. Mean confidence grows double from 0.31 to 0.63
3. mAP increases by around 10%.

From 2nd run to 3rd and from 2nd to 3rd with a confidence threshold (for adding high-confidence data to the train set):

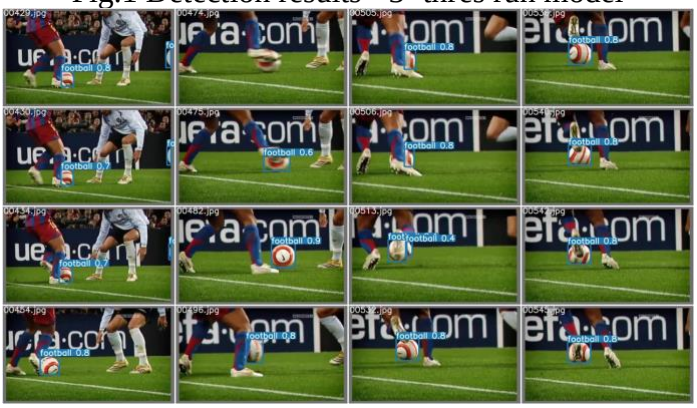
1. Mean confidence grows incrementally from 0.63 to 0.7+-
2. Not much change on mAP@.5 and mAP@.5:.95
3. The mAP performance between 3rd and 3rd with threshold are similar but the latter is the result of adding much less data (-85%) to the train set

This has shown that adding more data does not guarantee a performance boost, especially when pseudo-labelled data isn't verified by a human, so I used a threshold to add only high-quality data. In addition, we must be aware of training on our custom model with small data for many times, as it might lose generalisation ability from the Yolov7 pretrained weight.

Qualitative analysis

The below picture is from testing the first batch of the test set using our 3+thres model.

Fig.1 Detection results - 3+thres run model



The below picture is from testing the first batch of the test set using our first run model.

Fig.2 Detection results – 1st run model

We can see the improvement of confidence as well as the detection rate on occluded targets. However, in Fig.1 the 1st row 2nd photo was not detected at all. In addition, in Fig.1 in the first column, some irrelevant white area was detected as football although we cannot see their confidence. We might be able to improve the model if we collect more occluded samples as well as more diverse types of footballs and put them in the train set. Real-time visualisation results in video can be download [here].

Further improvement and other interesting research papers

Further improvement

There are few tasks to think about for future improvement.

1. Get more ground truth data by verifying pseudo-annotated labels with human eye, add it to the current train set.
2. Since val and test sets are small, they cannot represent real-world data distribution. Therefore, we can always add more val and test data whenever new ground truth data is accessible. By doing so, our model can produce more convincing validation and testing results.
3. Because SAM2 can fail to predict correct labels when background drastically change in the frame, it took me a while to identify and [group images] from similar video shots. We can consider retrain SAM2 to get better mask/bounding-box detection ability so it can deal with background shift better.

4. Change the object detector to a later version like yolov9 or yolov10 as they have smaller trainable parameters and better detection performance.
5. Write a script to automate training sessions after collecting sufficient data
6. Write a script to make data generation more efficient

Other interesting research papers

1. SAMURAI (Yang et al., 2024) [github]: just came out two days ago, it features zero-shot, robust in occlusion object tracking method. Zero-shot means there is no need for training any labelled sample. The method is leveraged by SAM2 pretrained-weights, and it adds a motion modelling part for motion tracking as well as matching object visual features from historical frames to current frame.

   However, for local inferencing, a RTX4090 GPU is needed, and it might fail tracking object if the object disappears for too long which its information is no longer in the memory bank. Currently it only supports single object tracking and does not seem there is a way to retrain it.

2. CoTracker (Karaev et al., 2024) [github]: suggests it can track large number of pixels in a very long time and very robust with occlusion. The work uses CNN with Transformer to extract and improve tracking performance. It is computational expensive to train the model but can be very powerful tracker. Moreover, since it is a point cloud tracking method, applying 3D reconstruction is plausible for future application.

Bibliography

Ferreira, R. E. P., Lee, Y. J., & Dórea, J. R. R. (2023). Using pseudo-labeling to improve performance of deep neural networks for animal identification. *Scientific Reports*, *13*(1), 13875. https://doi.org/10.1038/s41598-023-40977-x

Karaev, N., Rocco, I., Graham, B., Neverova, N., Vedaldi, A., & Rupprecht, C. (2024). *CoTracker: It is Better to Track Together*. https://arxiv.org/abs/2307.07635

Ravi, N., Gabeur, V., Hu, Y.-T., Hu, R., Ryali, C., Ma, T., Khedr, H., Rädle, R., Rolland, C., Gustafson, L., Mintun, E., Pan, J., Alwala, K. V., Carion, N., Wu, C.-Y., Girshick, R., Dollár, P., & Feichtenhofer, C. (2024). *SAM 2: Segment Anything in Images and Videos*.

Yang, C.-Y., Huang, H.-W., Chai, W., Jiang, Z., & Hwang, J.-N. (2024). *SAMURAI: Adapting Segment Anything Model for Zero-Shot Visual Tracking with Motion-Aware Memory*. https://arxiv.org/abs/2411.11922