Project Report

Course: DLBAIPCV01 Project: Computer Vision

Task #2: Recognizing Objects in Video Sequences

Student: Dmitrii Shevchuk, Matriculation: 9213737

Contents

Introduction

The task:

“Develop a computer vision system that takes as input a video sequence and returns the same showing the position, shape and name of the objects present in the video. “

Process:

1. Choose a publicly available video database with object labels and bounding boxes.
2. Identify three publicly available state of the art detection and segmentation models to be tested and present a qualitative analysis of the chosen models.
3. Evaluate each model on the dataset with respect to validity, reliability and objectivity on the chosen dataset.
4. Present the best system in a video sequence. The link to the video should be provided in the report.

My plan is simple. I will use Internet reviews for choosing three models and one dataset. Then use OpenCV Python library to extract frames from videos, pass them to each of three DSM’s and compare the result with ground truth classification and bounding box. Based on the result, the base model will be chosen and the final sequence video produced.

For this process I will use my personal laptop with Intel Core i5 8250U, 8GB DDR. This will place limits on the database and each model version.

Implementation

1. Database

I found only two video databases with object labels and bounding boxes available to the public. Models are usually trained on image databases, like COCO, ImageNet, Open Images Dataset and the like. But we need a video database, and here there are much less options.

First is YouTube-BoundingBoxes Dataset[1]:

“The data set consists of approximately 380,000 15-20s video segments extracted from 240,000 different publicly visible YouTube videos, automatically selected to feature objects in natural settings without editing or post-processing, with a recording quality often akin to that of a hand-held cell phone camera. All these video segments were human-annotated with high precision classifications and bounding boxes at 1 frame per second.”

Second is an extension of the ImageNet dataset called ImageNet Video Dataset. It is managed by Stanford University, and there is an application process to get access. It takes time to get approved, so I chose YouTube-BoundingBoxes Dataset because of the simple access.

Database includes 625K frames, so it is suitable for training and validation. I will use the first 650 frames due to limited compute resources and no requirement on the dataset size to be used.

2. Choosing and comparing models

Based on several reviews on the Internet I’ve chosen three models below.

1. **YOLO (You Only Look Once)** version 9 released in 2024. It is a fast model (only look once) and provides decent quality. Below is the comparative table for different versions of YOLOv9. So far only YOLOv9c and YOLOv9e are publicly available, I chose YOLOv9c as it demands less computing power.

[1] https://research.google.com/youtube-bb/index.html

Table 1. Comparison of different versions of YOLOv9 on MS-COCO dataset[2].

Model	**size** (pixels)	**mAP$^{val}_{50-95}$**	**mAP$^{val}_{50}$**	**params** (M)	**FLOPs** (B)
YOLOv9t	640	38.3	53.1	2.0	7.7
YOLOv9s	640	46.8	63.4	7.2	26.7
YOLOv9m	640	51.4	68.1	20.1	76.8
YOLOv9c	640	53.0	70.2	25.5	102.8
YOLOv9e	640	55.6	72.8	58.1	192.5

2. **RetinaNet** was developed by Facebook AI Research in 2017, and is still a state of the art system. I will use the RetinaNet model with a ResNet-50-FPN backbone available in torchvision.models.detection module. It has 34 million parameters and requires 151 billion FLOPs, which makes it about 50% “heavier” than YOLOv9c[3].

3. **Faster Region-based Convolutional Neural Networks (Faster R-CNN)** was developed in 2015 by the Microsoft Research team. I will use the Faster R-CNN model with a ResNet-50-FPN backbone available in torchvision.models.detection module. It has 42 million parameters and requires 134 billion FLOPs.[4]

Most reviews note that YOLO is simple and fast, but less accurate, struggling with small objects and inference when only parts of the object are visible. RetinaNet and Faster R-CNN are more accurate, but slower.

There are many other interesting models, like Mask Region-based Convolutional Neural Networks (Mask R-CNN) providing fine-grained pixel-level bounds. It would be not easy to compute Intersection over Union for such bounds with ground truth bounding boxes in the dataset.

EfficientDet from Google Research team is another state of the art model, but it returns no bonding boxes, only classification.

[2] https://docs.ultralytics.com/models/yolov9/#performance-on-ms-coco-dataset
[3]https://pytorch.org/vision/main/models/generated/torchvision.models.detection.retinanet_resnet50_fpn.html

[4]https://pytorch.org/vision/main/models/generated/torchvision.models.detection.fasterrcnn_resnet50_fpn.html

3. Evaluation process

To evaluate models we need metrics. Based on the textbook for Computer Vision course I will measure:

Precision = True Positive / (True Positive + False Positive)

Recall = True Positive / (True Positive + False Negative)

F1 = 2 x Precision x Recall / (Precision + Recall)

Accuracy = (True Positive + True Negative) / Total number of predictions

IoU (Intersection over Union) = Area of Intersection / Area of Union

IoU is measured for the true and predicted bounding boxes. I will also measure the time spent for inference with each model. This does not include pre-processing of the video frame.

YouTube-BoundingBoxes Dataset labeled with 23 object classes, models were trained on COCO dataset with 107 classes, including those 23. So evaluation will be relevant for 23 classes only.

YouTube-BoundingBoxes Dataset provides following information, below is an example of one line:

ACDc6tGnXXQ - Youtube video ID
27000 - frame time in msec
20 - class id
Elephant - class name
0 - object_id (not used in our process)
Present - object ‘present’, ‘absent’ or ‘uncertain’
0.546   0.856   0.48666668   0.67 - relative coordinates of the bounding box (xmin, xmax, ymin, ymax - should be changed to xmin, ymin, xmax, ymax delivered by models).

‘Uncertain’ was treated as ‘absent’ because in both cases no bounding box is given.

Database provides only one object present or absent on a frame. But models usually provide many detected objects with different confidence probabilities. This means we need additional limits on model predictions to qualify them. Here they are:

1. Only top-10 by confidence probabilities objects will be considered in each model's output.
2. True Negative = object is marked as ‘absent’ in the database, and the model does not detect this object, but other objects can be detected.

3. False Positive = object is marked as ‘present’ in the database, and the model does not detect this particular object, or it is detected, but bounding box has no intersection with the true bounding box (IoU = 0).
4. If the model detected several same true class objects with different bounding boxes, I chose the bos with maximum IoU. Hence True Positive with max IoU are reported.
5. False Negative = object is marked as ‘absent’, but model detects the object class which is marked ‘absent’. If it detects other objects this is True Negative.

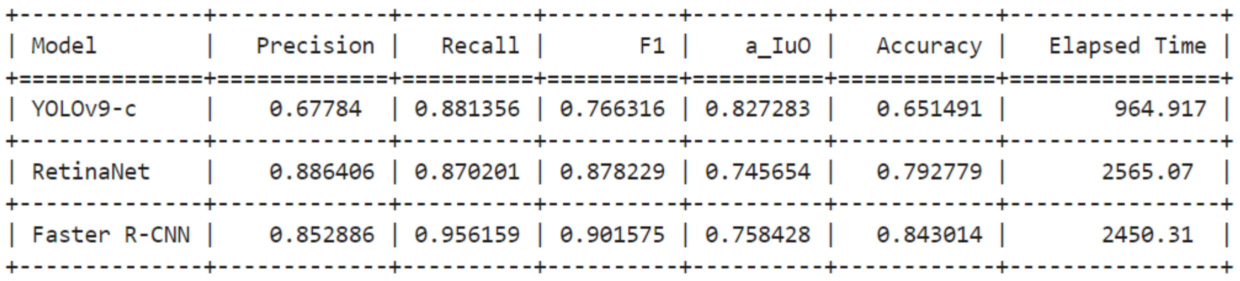
Python script can be found here: https://github.com/shevchukum/cimputer_vision_project.
I used ultralytics module to import YOLOv9 model. But I could not manage to feed it with images as arrays or tensors, it might be possible only for commercial license users. But it did process image files. So the pipeline looks like this:

1. Read a line from a csv database file. Get Youtube ID of the video and timestamp.
2. Read the frame from the video and save it as RGB jpg file.
3. For each model read the file and provide output: top-10 labels and bounding boxes arrays.
4. Check labels and boxes against ground truth label and box to classify the result and calculate IoU.
5. Calculate metrics.

4. Evaluation results

Below is the results table for 637 sample frames, run time 2h 26m.

Table 2. Comparison of three models performance.



```
+-------------+-------------+----------+----------+----------+------------+----------------+
| Model       |   Precision |   Recall |       F1 |    a_IuO |   Accuracy |   Elapsed Time |
+=============+=============+==========+==========+==========+============+================+
| YOLOv9-c    |    0.67784  | 0.881356 | 0.766316 | 0.827283 |   0.651491 |        964.917 |
+-------------+-------------+----------+----------+----------+------------+----------------+
| RetinaNet   |    0.886406 | 0.870201 | 0.878229 | 0.745654 |   0.792779 |       2565.07  |
+-------------+-------------+----------+----------+----------+------------+----------------+
| Faster R-CNN|    0.852886 | 0.956159 | 0.901575 | 0.758428 |   0.843014 |       2450.31  |
+-------------+-------------+----------+----------+----------+------------+----------------+
```

As expected, the YOLOv9 model is much faster compared to the two other models. It is also more precise in bounding boxes. Considering the decent quality of precision and accuracy for streaming video I would try it first.

For off-line video processing two other models are significantly better in accuracy and precision, although slightly worse in bounding boxes. As I am going to process my short video off-line, I choose Faster R-CNN based on slightly better F1 and accuracy.

In general all three models showed very high precision and accuracy for such low quality images (home video frames). Also through random checking I found that about 10% of the frames were wrongly labeled (object is present, but labeled absent and vice versa). This makes the result even more impressive.

One of the reasons for the high precision and accuracy is very small dataset used for evaluation - 637 frames out of 1 million. There was one video with 95 frames showing a woman on a stage with several potted plants. Label was a ‘potted plant’. Of course, all three models found a potted plant in almost 100% of frames.

Another reason is that I gave each model 10 trials to guess the annotated object. It would be much better if we had a database where all objects from our class list are labeled if they are present on the image. This would allow rigorous recognition testing.

A short 10 sec video I made on a street outside my home supports this idea. I used Faster R-CNN to detect and classify objects with a probability threshold 20%. Each frame was processed, 300 frames in total. OpnenCV was used to draw green boxes. Here is the resulting video link: https://youtube.com/shorts/G0HLOH2I-ro

Mainly there are cars and people on bikes. But the model found additionally a clock, cell phone, tv, bottle, traffic light, microwave, cup, remote, bench, bird and sports ball. Huge space for improvement.

Conclusion

As far as I can see the test of the project is complete: three state of the art models are tested against a dataset and compared in standard metrics. Best model is chosen and applied to a short video.

I found this project quite interesting and useful to see what best publicly available models are capable of. I have noticed that it is harder for models to recognize an object when it is shown from an unusual angle or on the background of another big object or only some part of an object is shown. It is still possible to recognize the object for a human, but not for models yet. I would guess that one possible solution might be teaching models to predict possible views of the rotated object from basic knowledge of geometry.

I would also expect that models trained for video detection and classification might use information from previous frames in analysis of the next frames. Models considered in this project consider each frame as totally new, which is expected as they are trained on images.

And finally: current models are able to recognize several objects in the picture, but it seems they do not reason in any way about different objects fitting together on the same image. Like it is harder to expect an elephant on top of the bike, or giant cell phone near a tiny bench. Such relational reasoning might also improve performance.