COMPUTER VISION HOMEWORK 2022

2D OBJECT DETECTION

By Saif Albuhayder

AGENDA

- Introduction`
- Detection method
- Code implementation
- Evaluation metrics
- Results
- Future improvement
- Conclusion

INTRODUCTION

The task is developing a custom 2D object detection algorithm and thereafter train and test the object detector which should detect the objects in the scene

DETECTION METHOD

Detectron2 library that provides state-of-the-art detection and segmentation algorithms.

I used two-stage object detection algorithm based on COCO pretrained model and Fine-tuned Faster R-CNN on CARLA Simulator dataset and trained our custom dataset.

CODE IMPLEMENTATION

- Annotate all images using annotation tool
- Split the data set into train – validation
- Back up dataset in github and Google drive
- Use Google Colab IDE to develop the code
- Install Detectron2 and import required libraries
- Register the data-set to COCO Format
- Choose the detection model and start train
- Show images using test dataset and Inspect the results
- Evaluate the training using evaluation metrics

ANNOTATE ALL IMAGES USING LABELME

Total annotated images is 1200

Total instances is 3826



SPLIT THE DATA SET INTO TRAIN – VALIDATION

75% train　　　　　　25% val.

8 > import os.py > ...

```python
import os
import re
import json
os.listdir('D:\l')
files = [f for f in os.listdir('.') if re.match(r'[0-9]+.*\.json', f)]
files.sort()
len(files)
# print(files)
c = 89;
my_counter = 0
for i in files:
    my_counter += 1;
    c += 1
    a = f"{c}"
    with open(i, 'r') as file:
        print(i)
        json_data = json.load(file)
        json_data['imagePath'] = f"1 ({a}).jpg"
        output_filename = f'{i}'# + '_output.json'
    with open(output_filename, 'w') as file:
        json.dump(json_data, file, indent=2)
# print(my_counter)
```

REGISTER DATASET TO COCO FORMAT

```
    record = {}
    record["image_id"] = id        #          solved

    filename = os.path.join(directory, img_anns["imagePath"])
    height, width = cv2.imread(filename).shape[:2]  # make it easier for
    record["file_name"] = filename
    record["height"] = height
    record["width"] = width

    print(record["image_id"])
    annos = img_anns["shapes"]
    objs = []
    for anno in annos:
        px = [a[0] for a in anno['points']]  # x coord
        py = [a[1] for a in anno['points']]  # y-coord
        poly = [(x, y) for x, y in zip(px, py)]  # poly for segmentation
        poly = [p for x in poly for p in x]

        obj = {
            "bbox": [np.min(px), np.min(py), np.max(px), np.max(py)],
            "bbox_mode": BoxMode.XYXY_ABS,
            "segmentation": [poly],
            "category_id": classes.index(anno['label']),
            "iscrowd": 0
```

CHOOSE DETECTION ALGORITHM

Faster R-CNN:

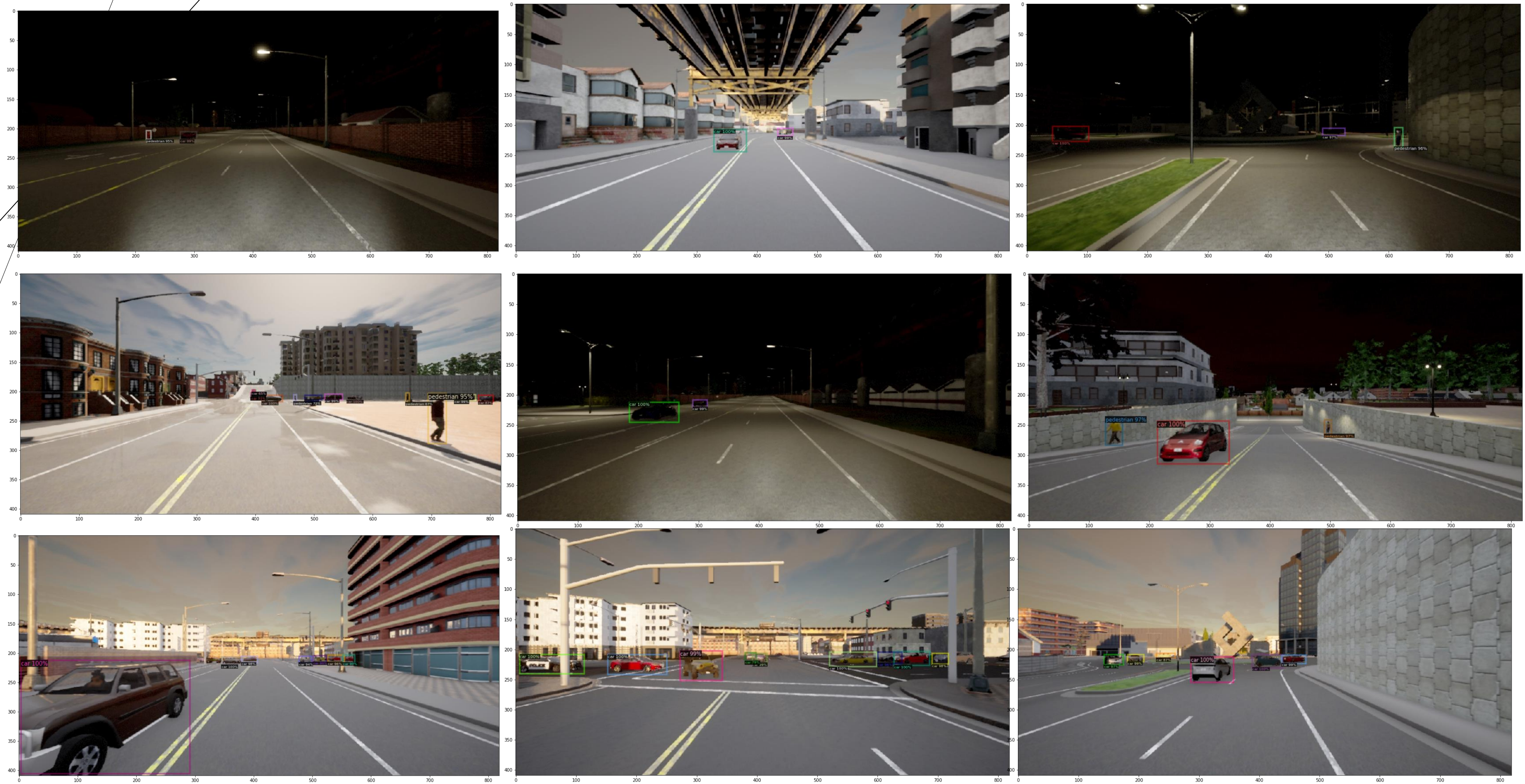
Name	lr sched	train time (s/iter)	inference time (s/im)	train mem (GB)	box AP	model id	download
R50-C4	1x	0.551	0.102	4.8	35.7	137257644	model | metrics
R50-DC5	1x	0.380	0.068	5.0	37.3	137847829	model | metrics
R50-FPN	1x	0.210	0.038	3.0	37.9	137257794	model | metrics
R50-C4	3x	0.543	0.104	4.8	38.4	137849393	model | metrics
R50-DC5	3x	0.378	0.070	5.0	39.0	137849425	model | metrics
R50-FPN	3x	0.209	0.038	3.0	40.2	137849458	model | metrics
R101-C4	3x	0.619	0.139	5.9	41.1	138204752	model | metrics
R101-DC5	3x	0.452	0.086	6.1	40.6	138204841	model | metrics
R101-FPN	3x	0.286	0.051	4.1	42.0	137851257	model | metrics
X101-FPN	3x	0.638	0.098	6.7	43.0	139173657	model | metrics

```
# Create a configuration and set up the model and datasets
cfg = get_cfg()
cfg.merge_from_file(model_zoo.get_config_file("COCO-Detection/faster_rcnn_R_101_FPN_3x.yaml"))
cfg.DATASETS.TRAIN = ("category_train",)
# cfg.DATASETS.TEST = ("category_train",)
cfg.DATASETS.TEST = ()
cfg.DATALOADER.NUM_WORKERS = 2
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-Detection/faster_rcnn_R_101_FPN_3x.yaml")
```

TRAIN THE MODEL

```
[05/24 20:25:15 d2.utils.events]:  eta: 0:00:58  iter: 4919  total_loss: 0.245  loss_cls: 0.06413  l
[05/24 20:25:30 d2.utils.events]:  eta: 0:00:44  iter: 4939  total_loss: 0.194  loss_cls: 0.06282  l
[05/24 20:25:45 d2.utils.events]:  eta: 0:00:29  iter: 4959  total_loss: 0.2075  loss_cls: 0.05598
[05/24 20:25:59 d2.utils.events]:  eta: 0:00:14  iter: 4979  total_loss: 0.2154  loss_cls: 0.06319
[05/24 20:26:16 d2.utils.events]:  eta: 0:00:00  iter: 4999  total_loss: 0.2279  loss_cls: 0.06457
[05/24 20:26:16 d2.engine.hooks]: Overall training speed: 4998 iterations in 1:00:47 (0.7299 s / it)
[05/24 20:26:16 d2.engine.hooks]: Total training time: 1:00:53 (0:00:05 on hooks)
```

SHOW IMAGES USING TEST DATASET AND INSPECT THE RESULTS




car 65%
car 100%
pedestrian 72%
car 99%
car 100%
pedestrian 67%
pedestrian 95%
car 99%
0
50
100
150
200
250
300
350
400
0
100
200
300
400
500
600
700
800

EVALUATION METRICS

Pascal VOC metric defines the mAP metric using a single IoU threshold of 0.5

COCO metrics defines several mAP metrics using different thresholds

Average Precision (AP)

- mAP@IoU= .50:.05:.95
- Primary challenge metric
- mAP@IoU= .50
- PASCAL VOC metric
- mAP@IoU= .75
- Strict metric

Average Recall (AR)

- mAR@max=1
- 1 detection per image
- mAR@max=10
- 10 detections per image
- mAR@max=100
- 100 detections per image

RESULTS

We did the training on all videos and then then compared it to signal video results

Average Precision	000	001	002	003	004	005	006	007	All data
mAP@IoU =.50:.05:.95	0.532	0.578	0.707	0.582	0.493	0.533	0.472	0.471	0.506
mAP@IoU = .50	0.886	0.990	0.965	0.937	0.847	0.990	0.834	1.000	0.864
mAP@IoU = .75	0.567	0.702	0.832	0.645	0.544	0.554	0.528	0.432	0.551
Average Recall									
mAR@max=1	0.302	0.575	0.340	0.250	0.291	0.536	0.420	0.435	0.308
mAR@max=10	0.579	0.653	0.742	0.640	0.552	0.620	0.551	0.537	0.563
mAR@max=100	0.579	0.653	0.742	0.641	0.553	0.620	0.551	0.537	0.565

DISTRIBUTION OF TEST INSTANCES

category	#instances
car	676
pedestrian	122
bicycle	0
motorcycle	0
truck	0
Total	820

DISTRIBUTION OF TRAIN INSTANCE

category	#instances
car	2409
pedestrian	597
bicycle	0
motorcycle	0
truck	0
Total	3006

Final results

Precision	0.506
Recall	0.565
Car bbox AP	59.306
Pedestrian bbox AP	41.922
Overall training speed	0.7299 s / it
Class accuracy	0.975

1 Implementation in real-time detection

2 Larger dataset

3 Use more accurate dataset annotation

4 Improve annotation process to save time and human resources

FUTURE IMPROVEMENT

CONCLUSION

Object detection is an important research topic. Due to the need of making self-driving cars and autonomous vehicles safer and more reliable. We hope that we participate in developing that interesting future.

THANK YOU

Saif Albuhayder

https://github.com/sluvd/CV_homework