주차 관리 시스템

Parking Management System Using YOLOv5

김 주 호, 송 귀 석
(Joo Ho Kim[1], Gwi seok Song[1])
[1]School of Mechanical & Control Engineering, Handong Global University

Abstract: 최근 AI의 발달은 사회전반에 적용되며 커다란 영향을 주고 있다. 그 중 딥러닝 기술의 일부인 yolo 모델을 통하여 주차 관리 시스템에 적용해보도록 한다. 적용된 주차장 관리 시스템은 13대의 주차 가능한 공간에서 주차 가능한 공간의 자릿수를 사용자에게 알려주는 기능을 수행한다. 예제로 제공된 영상에 대하여 실험 결과 실사용에 문제없는 성능을 보여줄 것으로 예상된다.

Keywords: yolov5, parking management system

I. 서론

본 논문에서는 주차가능한 공간을 빠르고 정확하게 검출 가능한 시스템을 연구하는 것으로 YOLO v5모델을 사용한 주차장 관제시스템에 대한 연구를 진행하였다. 이 연구는 실시간 카메라를 통해 입력되는 이미지만으로 차량을 인식한 결과를 바탕으로 차량 수를 자동으로 세서 현재 주차장에 남은 주차 가능 공간을 파악함으로써 주차장이용시 빠른 정보제공을 통해 주차장을 효율적이고 편리하게 이용하도록 할 수 있을 것으로 기대된다.

II. 모델 설명 및 구성

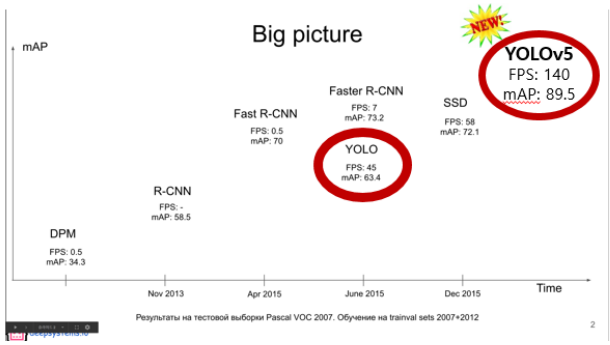

Fig. 1. CNN 모델들의 FPS & mAP 비교.

Fig. 1은 Yolov5의 성능을 다른 모델과 비교하기 위해 기존 자료에 Yolov5의 FPS와 mAP를 표시에 준 것이다. 이전 모델인 Yolov3와 비교하였을 때 FPS와 mAP 모두 더 좋은 성능을 보인다. YOLO v5는 이전모델들과는 다르게 모델의 model depth multiple과 layer width multiple로 구분되어 4가지 모델(s, m, l, x)로 나누어진다.

이번 실험에서는 제공받은 동영상을 통해 이미지 detection을 하기 때문에 너무 무거운 모델을 사용하지 않았으며, 정확도와 속도는 trade-off 관계를 가지므로 무거운 모델을 사용할수록 높은 정확도 대신 처리속도가 느려지므로 사용목적에 초점을 두어 어느정도 높은 정확도를 보여주는 YOLOv5l을 사용하여 실험을 진행하였다. 각 모델의 성능은 아래와 같다.

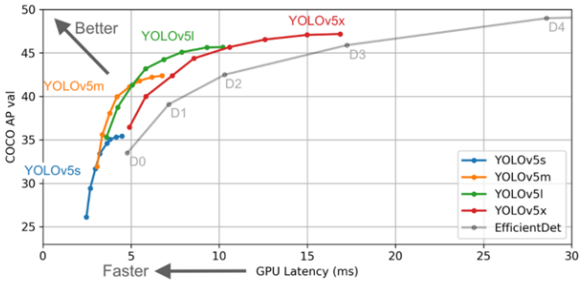

Fig. 2. YOLO v5의 모델 별 성능.

Yolov5가 기존 yolo 시리즈보다 빠른 이유는 Backbone으로 CSPNet을 사용하였으며, Head 부분을 Feature Map을 바탕으로 3개의 스케일의 Anchor Box를 통해 Bounding box를 생성하여 물체의 위치를 찾는다.

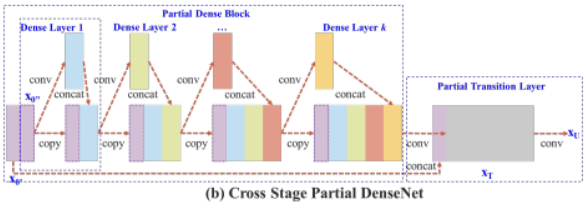

Fig. 3. CSPNet

CSPNet은 base layer를 두개의 파트로 나누어 마지막 cross-stage 계층에서 합침으로써 점점 늘어나는gradient combination의 연산량을 크게 줄여주고, 정확도 향상과 inference time을 줄여주었다.

CSPNet을 통해 기존 ResNet, ResNeXt, DenseNet에서 10~20%에 해당하는 연산량을 감소시킬 수 있었으며, 각 layer의 연산량을 균등하게 분해하여 연산 bottleneck 현상을 없애고 yolov3 모델기준 80%가량 감소시킬 수 있다. 또한, CSPNet에서의 cross-channel pooling을 통해서 기존의 feature pyramid 작업을 압축시켜 메모리 cost를 효과적으로 줄일 수 있다.

III. 실험

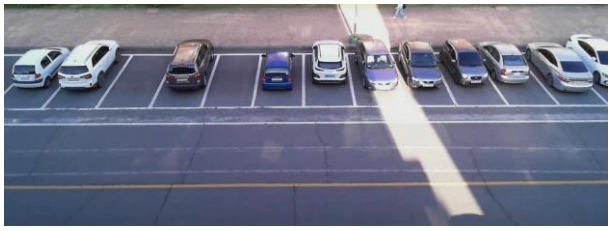

Fig. 4. 실험에 사용된 주차장의 배경.

실험이 진행되는 환경은 한동대학교 올네이션스홀 뒷편의 주차장을 배경으로 설정하였다. 총 13대의 주차 공간을 대상으로 주차장 차량 관제시스템 구축을 진행하였으며, 학습된 모델을 통해 영상에서 관측되는 모든 차량 class에 대해서만 인식하게 하였으며 주차장내 관측되는 모든 차량에 대해서 주차할 의도가 있다고 가정하여 주차장내 남은 공간을 계산해주어 화면에 남은 자리가 표시되도록 시스템을 구성하였다.

- 실험 환경 구성

Table 1. 실험에 사용된 GPU 성능

Model	NVIDIA GeForce GTX 1660 Ti Max-Q
Manufacturer	NVIDIA
Core	1440-1335(Boost) MHz
Memory	12000MHz
Bus	192 Bit
Memory	GDDR6
Max. Memory	6144MB
DirectX	DirectX 12_1
Technology	12nm
Introduced	23.04.2019

- Code 구성

```
if len(det):
    # Rescale boxes from img_size to im0 size
    det[:, :4] = scale_coords(img.shape[2:], det[:, :4], im0.shape).round()

    # Print results
    for c in det[:, -1].unique():
        n = (det[:, -1] == c).sum()  # detections per class 클래스의 총갯수

        s += f"{n} {names[int(c)]}{'s' * (n > 1)}, "  # add to string

    car_count=(det[:, -1] == 2).sum() # car(label의 넘버가2)의 개수를 나타내는 변수
```

Fig. 5. Car의 수를 세기 위한 code

Fig. 6. 다른 class로 detect된 차량

Fig. 6. 은 car를 truck으로 잘못 인식한 결과이다. 본 실험은 차량에 대한 detection만을 필요로 하는 시스템을 만드는 것이 목적이므로 주차장 차량 관제 시스템의 목적을 고려하고 그에 대한 성능을 높이기 위해 'car' class에 대해서만 object detection을 실행하였다. Fig. 5. 에서 'car' 에 해당하는 label number는 '2' 이므로 label number가 2인 객체에 대해서만 인식하여 frame별 차량 counting이 진행되도록 하였다.

```
## 주차장 여분 자리(원래자리 개수인 13 - 현재 주차장 내 차량 수)
car_count1 = f'Left Space: {13-car_count}'
car_count2 = f'{i3}, {car_count}' # i3 번째 프레임에 detection된 차량 수
text1 = open('counting_result.txt', 'a')
text1.write( car_count2+ '\n')
```

Fig. 7. 현재 주차장내 차량 수 저장 코드

현재 주차장내 남은 공간을 표시해주기 위해 총 주차 가능 공간 수인 13에서 frame마다 car의 수를 빼 주어 표시되도록 하였으며, 모델의 정확도를 측정하기위해 각 프레임별로 detection된 차량의 수가 text 파일로 저장되도록 하였다.

```
parser.add_argument('--conf-thres', type=float, default=0.25, help='object confidence threshold')
parser.add_argument('--iou-thres', type=float, default=0.45, help='IOU threshold for NMS')
```

Fig. 8. Confidence threshold와 IOU threshold parameter 설정

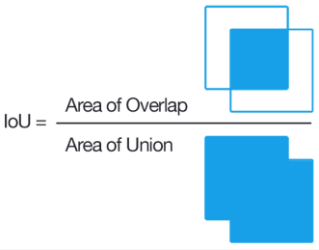

Fig. 9. Equation of IOU

실험에 사용된 모델은 미리 훈련된 open source의 모델을 사용하였다. 모델을 통해 objection detection 실행할 때 detection 결과에 영향을 미치는 주요 parameter로는 IOU threshold와 Confidence threshold가 있어 각 값들을 조절하여 모델의 성능을 조절할 수 있었다.

이번 실험에 사용된 모델의 결과를 평가하는데 있어서 중점적으로 사용될 변수는 IOU, Confidence level 그리고 FPS이다. 위에 해당하는 변수들에 대한 설명은 다음과 같다.

```
# Apply NMS
pred = non_max_suppression(pred, opt.conf_thres, opt.iou_thres, opt.classes, opt.agnostic_nms,
                           max_det=opt.max_det)
t2 = time_synchronized()
```

Fig. 10. NMS (Non-max suppression) code

- NMS (Non-max suppression)

Non-max suppression이란 하나의 class에 대해 bounding box를 그려줄 때 중복 detection이 일어나는 것을 방지하는 방법으로, 하나의 class에 대한 bounding boxes의 목록에서 가장 높은 점수를 가지고 있는 bounding box만을 선택하여 최종 출력할 final box에 추가하고 추가한 bounding box를 final box에 이전에 추가된 다른 bounding box들과 IOU를 비교하여 설정해준 threshold 값보다 높으면 목록에서 제거하는 작업을 bounding boxes 목록에 아무것도 남지 않을 때까지 반복하여 하나의 Class에 대해서 IOU threshold 값을 넘으면 중복 객체를 detect했다고 판단하고 제거하여 하나의 객체에 중복

detection 현상이 나타나는 것을 방지할 수 있다. IOU threshold 값이 낮을수록 bounding box가 겹치지 않게 detection이 되며, 높아질수록, 겹쳐서 detection하는 특성을 가지고 있다.

- **Confidence Threshold**

보통 Confidence threshold의 값은 0.5로 설정한 후 그 이하의 경계박스후보는 배경으로 간주하여 제거한다. Confidence의 값에 따라 Recall과 Precision에 영향을 준다. Confidence 가 높아질수록 object detection의 기준이 보수적으로 되는 반면 낮아지게 될 경우 쉽게 detection 되는 경향이 있다. 즉, Confidence의 값이 커지면 성능면에서 Recall은 낮아지지만, Precision은 높아진다.

- **FPS(Frame Per Second)**

이번 논문에서 진행하는데 있어 중점적인 변수는 아니지만, 실제 시스템에 적용할 때 고려해야할 중요한 변수이다. FPS란 초당 처리가능한 이미지의 장을 의미한다. FPS는 사용되는 하드웨어의 환경과 모델에 따라 달라지므로, 두가지의 환경을 잘 타협하여 실제 환경에서 적용하는 것이 중요하다.

IV. 실험 결과

Table 2. 모델에 따른 정확도와 FPS

모델	모델 성능 (차량 인식률)	FPS
Reference	0.9515	[-]
yolov5s	0.711515	19
yolov5m	0.711515	10
yolov5l	0.993939	6
yolov5x	0.993939	4

yolov5에서 제공하는 모델로는 yolov5의 s, m, l, x 모델이 있다. 각 모델들의 성능을 평가하기 위하여 confidence = 0.25, IOU = 0.45를 default 로 설정한 후 제공된 동영상에서 진행되었다. Table 2는 모델에 대한 성능과 FPS를 나타낸다. 모델 성능면에 있어서 l과 x 모델이 매 프레임에 대해 높은 성능을 보이는 반면 s, m 모델의 경우 70% 로 성능이 좋지 않은 것을 확인할 수 있다. 그러나 FPS를 고려할 경우, 같은 성능 대비 각각 s,l 모델을 사용하는 것이 좋음을 알 수 있다.

실제 주차 관리시스템에서는 높은 FPS을 요하는 실시간 데이터 처리 보다는 주차 가능한 차량의 대수를 정확하게 측정하는 것이 더 중요하다. 따라서 적어도 Reference 모델의 모델 성능보다는 성능이 좋아야 하므로, YOLOv5l, YOLOv5x 모델 중 상대적으로 더 높은 FPS를 갖는 'YOLOv5l'모델을 기준으로 하이퍼 파라미터의 변화에 따른 실험을 진행하였다.

하이퍼 파라미터의 default 환경에서 실험을 진행할 경우 object에 대해서 중복으로 object detection이 되는 문제가 발생하였다. 오차에 대한 분석을 해보면 물체를 잘못인식한 경우는 없었으며, 하나의 객체에 대해 중복 인식을 한 것이 오차의 주요 원인이었다.

Fig. 11. Duplicate Object Detection Error

Fig. 11. 은 미리 훈련된 모델을 통해서 detection된 차량들 중 단일 차량에 대하여 두 개의 Bounding box가 그려져 두 개의 차량으로 잘못 인식한 결과를 보여준다. Object detection의 오류를 해결하기위해 모델의 IOU threshold값과 confidence threshold 값을 조절하여 정확도를 각각 비교하였다.

Table 3. YOLO v5l의 IOU에 따른 정확도

IOU	모델 성능 (차량 인식률)
0.25	0.10303
0.30	0.99333
0.35	0.993939
0.40	0.993939
0.45(default)	0.993939
0.50	0.98606
0.55	0.68969

Table3는 Yolov5l 모델 환경에서 Confidence threshold는 default값인 0.25로 고정한 후 IOU threshold 값의 default 값인 0. 45를 기준으로 0.05단위만큼 키우고 줄임에 따른 모델 성능을 평가하였다. IOU threshold를 낮추는 경우, 쉽게 bounding detection 되지 않아0.25 부근에서 차량 인식률이 현저히 떨어짐을 알 수 있다. 또한 0.55 이상의 높은 IOU threshold 에서는 detection이 쉽게 되어 점진적으로 성능이 떨어지는 것을 확인하였다. 따라서 두 값 사이의 값으로 조절함으로써 실험을 진행하였다. 그러나 여러 개의 bounding detection이 발생하지 않도록 조절하면, 기존의 차량을 검출하지 못하는 문제가 있었다. 따라서, IOU threshold 값의 조절만으로는 하나의 물체에 대해서 여러 개의 물체 검출이 되지 않으면서 모든 물체에 대해 하나의 bounding box로 검출할 수 없었다.

Table 4. YOLO v5l의 confidence에 따른 정확도

confidence	정확도
0.25(default)	0.993939
0.30	1
0.35	0.998788
0.4	0.998182

Table4는 YOLOv5l의 Confidence threshold 값을 0.5 단위로 측정하여 구한 정확도를 나타낸다. Confidence threshold 값이 약 0.3 인 경우 모든 프레임에서 차량을 검출하고, 주차장의 남은 차량의 공간의 자릿수를 정확하게 나타냈다. threshold값이 높아질수록 정확도가 떨어지는 것은 물체 검출의 기준이

보수적으로 되어 차량을 인식하지 못한 것으로 생각되어진다.

IOU threshold값과 Confidence threshold값을 비교 분석을 통해최종적으로 각각 IOU threshold = 0.45, Confidence threshold = 0.33으로 설정하였다. 이를 통해 하나의 물체에 대해 2개의 물체로 인식하는 것을 방지하였으며 기존 정확도인 0.993939에서 100% 정확도로 시스템의 성능을 개선하였다.

V. 결론

본 논문은 주차장으로 사용되어질 수 있는 환경에서 주차 가능한 차량의 공간을 능동적으로 사용할 수 있도록 만드는 것을 제안한다. 주어진 counting reference 결과 비교와 모델들의 성능 평가를 기준으로 주어진 모델들 중 주차 시스템에 사용될 모델로 yolov5l 을 선택하였다. 고려되어진 하이퍼 파라미터는 IOU와 Confidence, FPS 이다. 이 변수들의 default 값에 대하여 실험을 진행 한 결과 default 값 기준, IOU threshold 값의 변화로는 성능을 향상시킬 수 없었으며 Confidence threshold의 값을 올려 IOU threshold =0.45, Confidence threshold = 0.33 에서 100%의 정확도를 보여주는 것을 확인하였다. 그러나 FPS 의 값이 6이므로 실제 적용되어 실시간으로 관측하기 위해서는 GPU의 성능이 더 좋은 것을 사용해야한다.

향후 전국 시내 공용 주차장의 CCTV에 이 모델이 적용된다면 복잡한 시내 공간을 좀더 효율적이고 시민들에게 쾌적한 도시문화를 제공할 수 있을 것으로 기대된다.

참고문헌

[1] YOLOv5 - Introduction (tistory.com)
[2] ultralytics/yolov5: YOLOv5 in PyTorch > ONNX > CoreML > TFLite (github.com)
[3] https://bigdata-analyst.tistory.com/194
[4] https://dyndy.tistory.com/275
[5] Tutorials of Object Detection using Deep Learning How to measure performance of object detection (hoya012.github.io)

Appendix

* Use 1-column for appendix

1. **Trouble Shooting**

하나의 물체에 대해 여러 개로 검출되는 문제 발생하였는데, 하이퍼 파라미터 변수값조절을 통해 해결하였다.

2. **Program Code**

```
3. import argparse
4. import time
5. from pathlib import Path
6. from timeit import default_timer as timer # 타이머 함수
7.
8. import cv2
9. import torch
10.import torch.backends.cudnn as cudnn
11.
12.from models.experimental import attempt_load
13.from utils.datasets import LoadStreams, LoadImages
14.from utils.general import check_img_size, check_requirements, check_imshow,
   non_max_suppression, apply_classifier, \
15.    scale_coords, xyxy2xywh, strip_optimizer, set_logging, increment_path, s
   ave_one_box
16.from utils.plots import colors, plot_one_box
17.from utils.torch_utils import select_device, load_classifier, time_synchroni
   zed
18.

19.@torch.no_grad()
20.def detect(opt):
21.    source, weights, view_img, save_txt, imgsz = opt.source, opt.weights, op
   t.view_img, opt.save_txt, opt.img_size
22.    save_img = not opt.nosave and not source.endswith('.txt')  # save infere
   nce images
23.    webcam = source.isnumeric() or source.endswith('.txt') or source.lower()
   .startswith(
24.        ('rtsp://', 'rtmp://', 'http://', 'https://'))
25.
26.    # Directories
27.    save_dir = increment_path(Path(opt.project) / opt.name, exist_ok=opt.exi
   st_ok)  # increment run
28.    (save_dir / 'labels' if save_txt else save_dir).mkdir(parents=True, exis
   t_ok=True)  # make dir
29.
30.    # Initialize
31.    set_logging()
32.    device = select_device(opt.device)
33.    half = device.type != 'cpu'  # half precision only supported on CUDA
34.
35.    # Load model
36.    model = attempt_load(weights, map_location=device)  # load FP32 model
37.    stride = int(model.stride.max())  # model stride
```

```
38.     imgsz = check_img_size(imgsz, s=stride)  # check img_size
39.     names = model.module.names if hasattr(model, 'module') else model.names
     # get class names
40.     if half:
41.         model.half()  # to FP16
42.
43.     # Second-stage classifier
44.     classify = False
45.     if classify:
46.         modelc = load_classifier(name='resnet101', n=2)  # initialize
47.         modelc.load_state_dict(torch.load('weights/resnet101.pt', map_locati
    on=device)['model']).to(device).eval()
48.
49.     # Set Dataloader
50.     vid_path, vid_writer = None, None
51.     if webcam:
52.         view_img = check_imshow()
53.         cudnn.benchmark = True  # set True to speed up constant image size i
    nference
54.         dataset = LoadStreams(source, img_size=imgsz, stride=stride)
55.     else:
56.         dataset = LoadImages(source, img_size=imgsz, stride=stride)
57.
58.     # Run inference
59.     if device.type != 'cpu':
60.         model(torch.zeros(1, 3, imgsz, imgsz).to(device).type_as(next(model.
    parameters())))  # run once
61.
62.
63.     FPS1=0  # FPS 변수 초기화
64.     t0 = timer() # 프레임 업데이트 전 시간
65.     i3 = 0 # 프레임넘버
66.     for path, img, im0s, vid_cap in dataset:
67.         img = torch.from_numpy(img).to(device)
68.         img = img.half() if half else img.float()  # uint8 to fp16/32
69.         img /= 255.0  # 0 - 255 to 0.0 - 1.0
70.         if img.ndimension() == 3:
71.             img = img.unsqueeze(0)
72.
73.         # Inference
74.         t1 = time_synchronized()
75.         pred = model(img, augment=opt.augment)[0]
76.
77.         # Apply NMS
78.         pred = non_max_suppression(pred, opt.conf_thres, opt.iou_thres, opt.
    classes, opt.agnostic_nms,
79.                                    max_det=opt.max_det)
80.         t2 = time_synchronized()
```

```
81.
82.         # Apply Classifier
83.         if classify:
84.             pred = apply_classifier(pred, modelc, img, im0s)
85.
86.         # Process detections
87.         for i, det in enumerate(pred):  # detections per image
88.             if webcam:  # batch_size >= 1
89.                 p, s, im0, frame = path[i], f'{i}: ', im0s[i].copy(), datase
    t.count
90.             else:
91.                 p, s, im0, frame = path, '', im0s.copy(), getattr(dataset, '
    frame', 0)
92.             p = Path(p)  # to Path
93.             save_path = str(save_dir / p.name)  # img.jpg
94.             txt_path = str(save_dir / 'labels' / p.stem) + ('' if dataset.mo
    de == 'image' else f'_{frame}')  # img.txt
95.             s += '%gx%g ' % img.shape[2:]  # print string
96.             gn = torch.tensor(im0.shape)[[1, 0, 1, 0]]  # normalization gain
     whwh
97.             imc = im0.copy() if opt.save_crop else im0  # for opt.save_crop
98.
99.             if len(det):
100.                         # Rescale boxes from img_size to im0 size
101.                         det[:, :4] = scale_coords(img.shape[2:], det[:, :4],
     im0.shape).round()
102.
103.                         # Print results
104.                         for c in det[:, -1].unique():
105.                             n = (det[:, -
    1] == c).sum()  # detections per class 클래스의 총갯수
106.
107.                             s += f"{n} {names[int(c)]}{'s' * (n > 1)}, "  #
    add to string
108.
109.                         car_count=(det[:, -
    1] == 2).sum() # car(label의 넘버가 2)의 개수를 나타내는 변수
110.                         # Write results
111.                         for *xyxy, conf, cls in reversed(det):
112.                             if save_txt:  # Write to file
113.                                 xywh = (xyxy2xywh(torch.tensor(xyxy).view(1,
     4)) / gn).view(-1).tolist()  # normalized xywh
114.                                 line = (cls, *xywh, conf) if opt.save_conf e
    lse (cls, *xywh)  # label format
115.                                 with open(txt_path + '.txt', 'a') as f:
116.                                     f.write(('%g ' * len(line)).rstrip() % l
    ine + '\n')
117.
```

```
118.                             if save_img or opt.save_crop or view_img:  # Add
    bbox to image
119.                                 c = int(cls)  # integer class
120.                                 label = None if opt.hide_labels else (names[
   c] if opt.hide_conf else f'{names[c]} {conf:.2f}')
121.                                 plot_one_box(xyxy, im0, label=label, color=c
   olors(c, True), line_thickness=opt.line_thickness)
122.                                 if opt.save_crop:
123.                                     save_one_box(xyxy, imc, file=save_dir /
   'crops' / names[c] / f'{p.stem}.jpg', BGR=True)
124.
125.
126.                     ## 주차장 여분 자리(원래자리 개수인 13 -
    현재 주차장 내 차량 수)
127.                     car_count1 = f'Left Space: {13-car_count}'
128.                     car_count2 = f'{i3}, {car_count}' # i3 번째 프레임에 dete
   ction된 차량 수
129.                     # mean average filter
130.
131.                     text1 = open('counting_result.txt', 'a')
132.                     text1.write( car_count2+ '\n')
133.                     i3+=1 # 프레임 수 +1
134.
135.                     ##FPS 계산
136.                     TA=timer() #한 프레임이 업데이트된 시간
137.                     FPS1 = 1/(TA-
   t0)  # 1초를 한 프레임이 업데이트된 시간으로 나누어 FPS 구함
138.                     t0=TA #다음 프레임까지 시간을 측정하기 위해 t0 업데이트
139.                     FPS_print = f'FPS: {FPS1:.0f}'
140.                     print(f'{s}--------FPS: {FPS1:.0f} Done. ({t2 -
    t1:.3f}s)')
141.
142.
143.                     ##영상에 남은자리, FPS 글씨 출력
144.                     cv2.putText(im0, text=car_count1, org=(5, 30), fontFace=
   cv2.FONT_HERSHEY_SIMPLEX,
145.                                 fontScale=1.0, color=(50, 50, 255), thickness=2)
146.                     cv2.putText(im0, text=FPS_print, org=(5, 60), fontFace=c
   v2.FONT_HERSHEY_SIMPLEX,
147.                                 fontScale=1.0, color=(50, 50, 255), thickness=2)

148.




149.                     # Stream results 영상출력
```

```
150.                     if view_img:
151.
152.                         cv2.imshow(str(p), im0)
153.                         cv2.waitKey(1)  # 1 millisecond
154.
155.                     # Save results (image with detections)
156.                     if save_img:
157.                         if dataset.mode == 'image':
158.                             cv2.imwrite(save_path, im0)
159.                         else:  # 'video' or 'stream'
160.                             if vid_path != save_path:  # new video
161.                                 vid_path = save_path
162.                                 if isinstance(vid_writer, cv2.VideoWriter):
163.                                     vid_writer.release()  # release previous
    video writer
164.                                 if vid_cap:  # video
165.                                     fps = vid_cap.get(cv2.CAP_PROP_FPS)
166.                                     w = int(vid_cap.get(cv2.CAP_PROP_FRAME_W
    IDTH))
167.                                     h = int(vid_cap.get(cv2.CAP_PROP_FRAME_H
    EIGHT))
168.                                 else:  # stream
169.                                     fps, w, h = 30, im0.shape[1], im0.shape[
    0]
170.                                     save_path += '0.mp4'
171.                                 vid_writer = cv2.VideoWriter(save_path, cv2.
    VideoWriter_fourcc(*'mp4v'), fps, (w, h))
172.                             vid_writer.write(im0)
173.
174.             if save_txt or save_img:
175.                 s = f"\n{len(list(save_dir.glob('labels/*.txt')))} labels sa
    ved to {save_dir / 'labels'}" if save_txt else ''
176.                 print(f"Results saved to {save_dir}{s}")
177.
178.
179.

180.         if __name__ == '__main__':
181.             parser = argparse.ArgumentParser()
182.             parser.add_argument('--
    weights', nargs='+', type=str, default='yolov5s.pt', help='model.pt path(s)'
    )
183.             parser.add_argument('--
    source', type=str, default='data/images', help='source')  # file/folder, 0 f
    or webcam
184.             parser.add_argument('--img-
    size', type=int, default=640, help='inference size (pixels)')
```

```
185.            parser.add_argument('--conf-
    thres', type=float, default=0.25, help='object confidence threshold')
186.            parser.add_argument('--iou-
    thres', type=float, default=0.45, help='IOU threshold for NMS')
187.            parser.add_argument('--max-
    det', type=int, default=1000, help='maximum number of detections per image')
188.            parser.add_argument('--
    device', default='', help='cuda device, i.e. 0 or 0,1,2,3 or cpu')
189.            parser.add_argument('--view-
    img', action='store_true', help='display results')
190.            parser.add_argument('--save-
    txt', action='store_true', help='save results to *.txt')
191.            parser.add_argument('--save-
    conf', action='store_true', help='save confidences in --save-txt labels')
192.            parser.add_argument('--save-
    crop', action='store_true', help='save cropped prediction boxes')
193.            parser.add_argument('--
    nosave', action='store_true', help='do not save images/videos')
194.            parser.add_argument('--
    classes', nargs='+', type=int, help='filter by class: --class 0, or --
    class 0 2 3')
195.            parser.add_argument('--agnostic-
    nms', action='store_true', help='class-agnostic NMS')
196.            parser.add_argument('--
    augment', action='store_true', help='augmented inference')
197.            parser.add_argument('--
    update', action='store_true', help='update all models')
198.            parser.add_argument('--
    project', default='runs/detect', help='save results to project/name')
199.            parser.add_argument('--
    name', default='exp', help='save results to project/name')
200.            parser.add_argument('--exist-
    ok', action='store_true', help='existing project/name ok, do not increment')
201.            parser.add_argument('--line-
    thickness', default=2, type=int, help='bounding box thickness (pixels)')
202.            parser.add_argument('--hide-
    labels', default=False, action='store_true', help='hide labels')
203.            parser.add_argument('--hide-
    conf', default=False, action='store_true', help='hide confidences')
204.            opt = parser.parse_args()
205.            print(opt)
206.            check_requirements(exclude=('tensorboard', 'pycocotools', 'thop'
    ))
207.
208.            if opt.update:  # update all models (to fix SourceChangeWarning)
209.                for opt.weights in ['yolov5s.pt', 'yolov5m.pt', 'yolov5l.pt'
    , 'yolov5x.pt']:
210.                    detect(opt=opt)
```

```
211.                     strip_optimizer(opt.weights)
212.             else:
213.                 detect(opt=opt)
214.
```