인공지능 응용시스템
차량 번호판 추출기

팀원 : 김동민, 김동현, 박재원

역할

김동현 : 프로젝트 총괄, Yolo v3 및 CRNN 사용 및 학습 이외의 딥러닝 방법 담당

김동민 : 디버깅, CRNN 성능 개선, OpenCV를 이용한 방식 및 전처리 담당

박재원 : 데이터 어노테이션, 멀티 GPU를 위한 서버 구축 담당

목차

1. 문제 정의
2. 문제 해결 계획
   a. OpenCV way
   b. Yolo v3
   c. CRNN
3. 문제 해결 과정
   a. OpenCV
   b. 데이터
   c. SSD
   d. 모델 학습
4. 결과 및 평가

문제 정의

Parking : Grayscale 된 주어진 이미지에서 차량 번호판의 내용을 출력

CCTV : 컬러이미지에서 차량들에 대한 차량 번호판을 찾은 뒤 가장 가까운 번호판의 내용을 출력

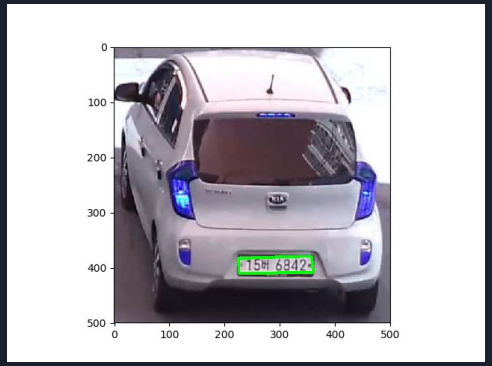
디텍션과 문자 인식을 모두 요구하는 문제

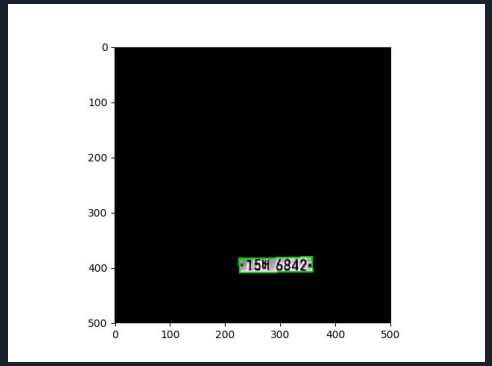
문제 해결 계획 - OpenCV

번호판 인식 알고리즘

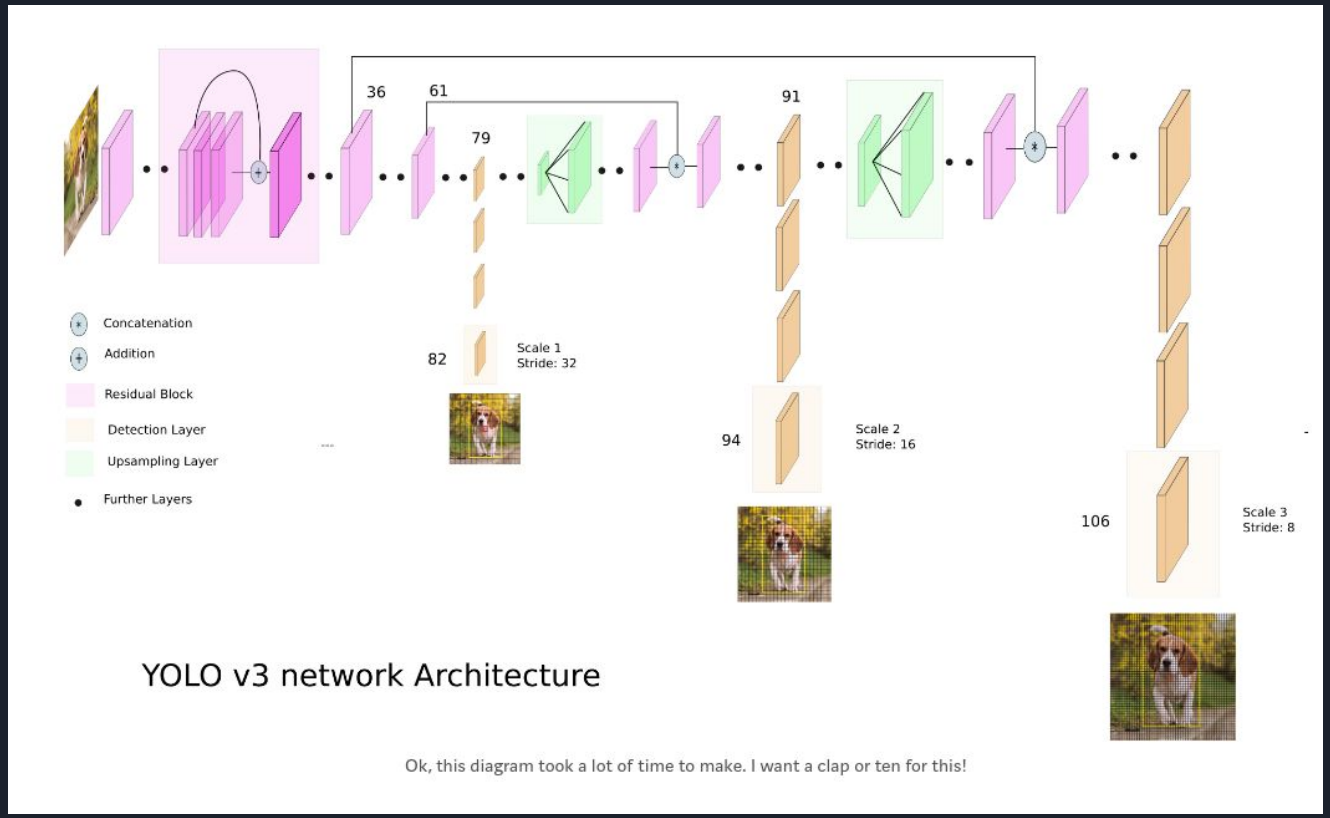
1. 방법

Image Grayscale -> noise removed image(using bilateral filter) -> histogram equalization -> morphological opening -> subtraction image(3 - 4) -> binarization image -> canny edge -> Using dilation -> detect contour -> enhanced image

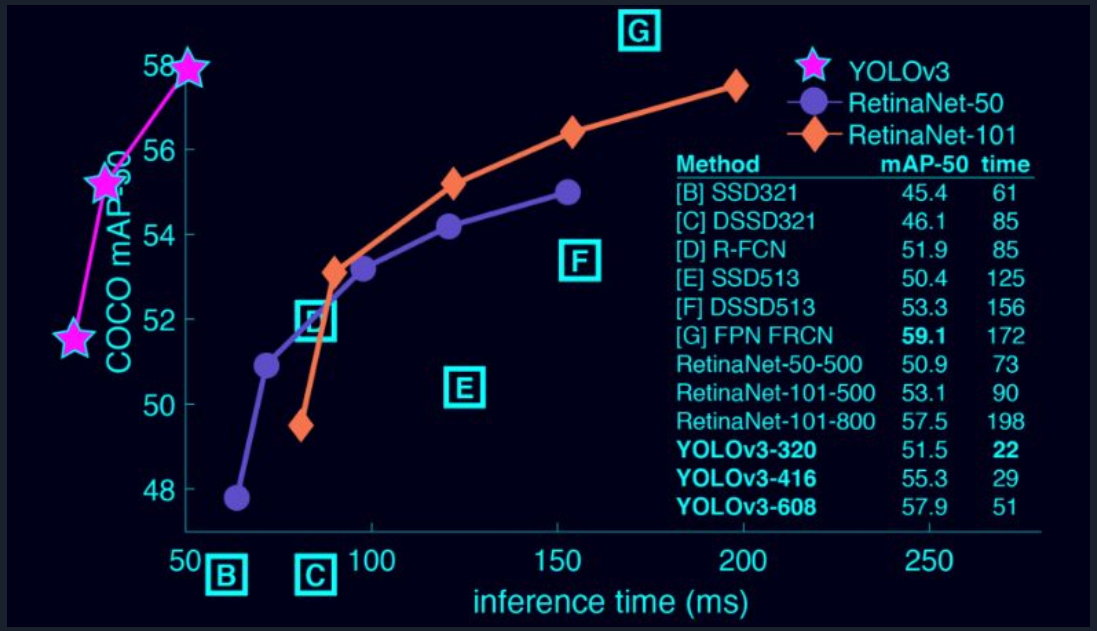




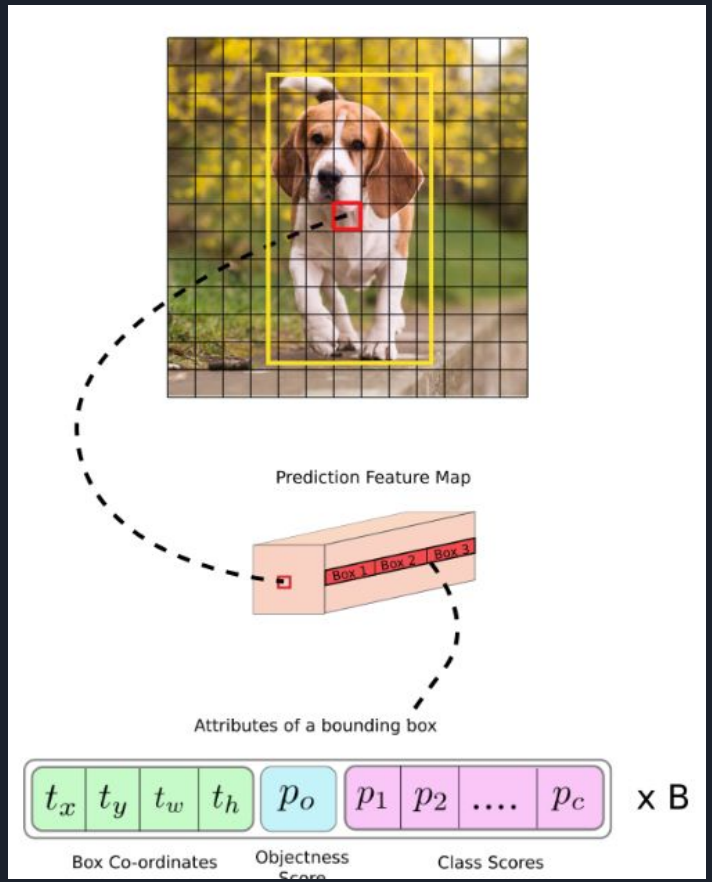
문제 해결 계획 - Yolo V3



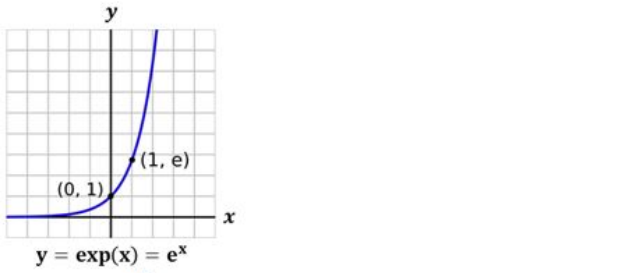
문제 해결 계획 - Yolo V3

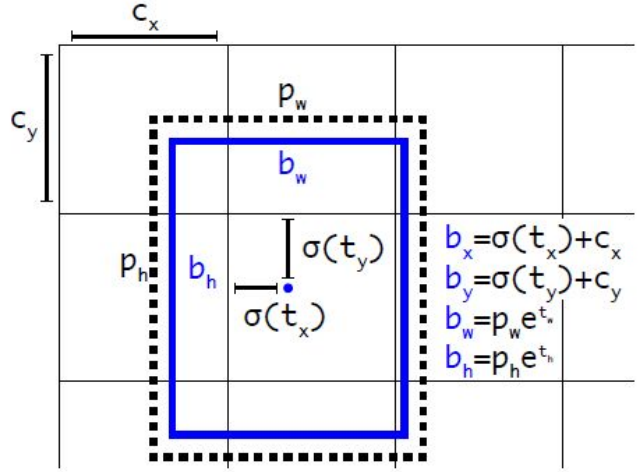


Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51



문제 해결 계획 - Yolo V3





$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(x_i - \hat{x}_i\right)^2 + \left(y_i - \hat{y}_i\right)^2$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2$$

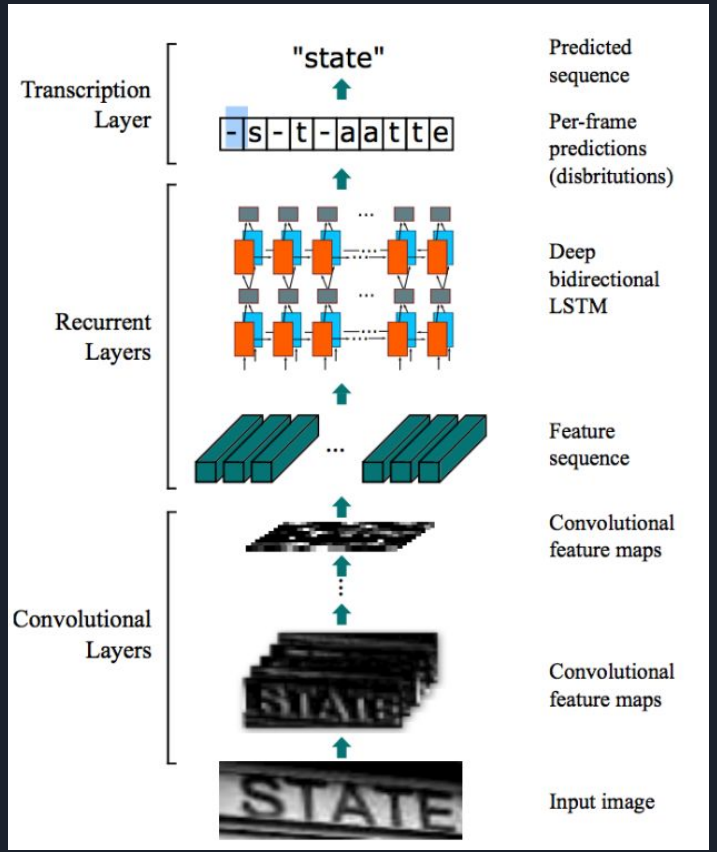
$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i\right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i\right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c)\right)^2 \qquad (3)$$

문제 해결 계획 - CRNN

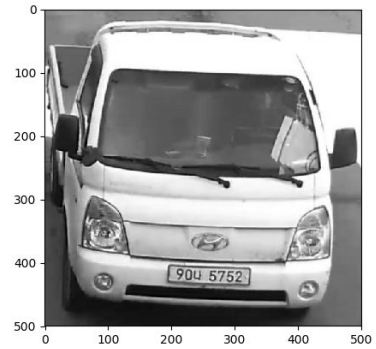
Type	Configurations
Transcription	-
Bidirectional-LSTM	#hidden units:256
Bidirectional-LSTM	#hidden units:256
Map-to-Sequence	-
Convolution	#maps:512, k:2 × 2, s:1, p:0
MaxPooling	Window:1 × 2, s:2
BatchNormalization	-
Convolution	#maps:512, k:3 × 3, s:1, p:1
BatchNormalization	-
Convolution	#maps:512, k:3 × 3, s:1, p:1
MaxPooling	Window:1 × 2, s:2
Convolution	#maps:256, k:3 × 3, s:1, p:1
Convolution	#maps:256, k:3 × 3, s:1, p:1
MaxPooling	Window:2 × 2, s:2
Convolution	#maps:128, k:3 × 3, s:1, p:1
MaxPooling	Window:2 × 2, s:2
Convolution	#maps:64, k:3 × 3, s:1, p:1
Input	$W \times 32$ gray-scale image

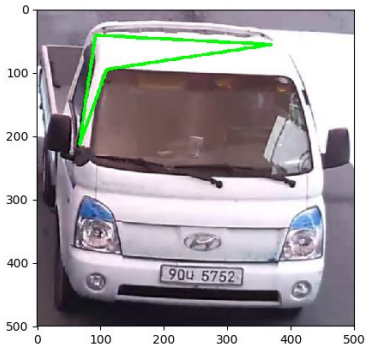


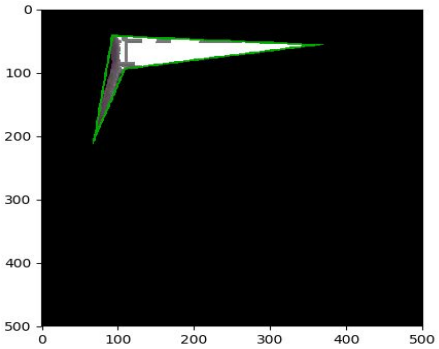
문제 해결 과정 - OpenCV

장점 : 빠른 속도를 가진다.

문제점 : cctv데이터에서는 비교적 작은 차량들에 대해서 알고리즘을 적용해야해서 정확도가 떨어지고 작은 이미지를 크롭해서 크게 리사이즈 할 경우 차량을 먼저 찾아야하기 때문에 거기에서 또 비용이 들어가게 된다. 무엇보다도 각종 영상처리 알고리즘의 threshold에 맞지 않는 이미지가 들어오면 완전 다른 부분을 마킹해버린다.







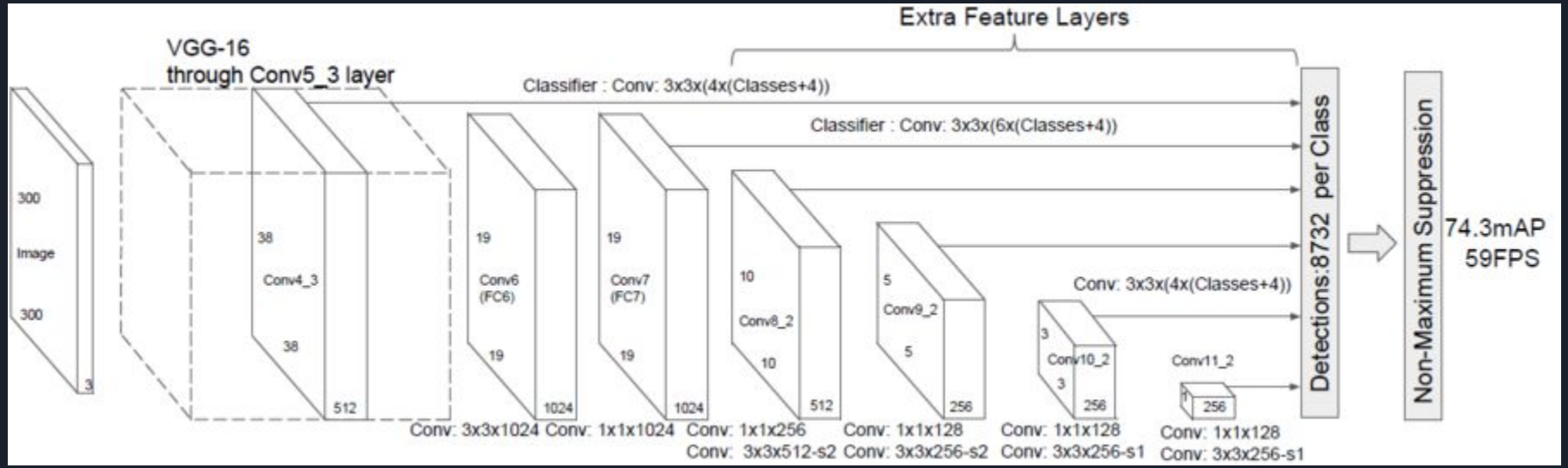
문제 해결 과정 - 데이터

문제점 : 데이터 어노테이션이 제대로 되어있지 않아있고 모든 자동차 번호판에 바운딩 박스가 그려지지않아 데이터의 질이 떨어진다고 판단하였다.

해결 : cctv 데이터 약 2000장을 모두 다시 어노테이션을 진행하였다. 더 나은 학습을 위해 기존의 자동차 번호판에 포함되지 않았던 주차된 부분이나 반대 차선에 있는 번호판 모두를 표시하고 그부분은 번호판 내용을 제외한 오로지 classify를 위한 p1 ~ p5로만 표현하여 제작 하였다.

모델 해결 과정 - SSD

문제점 : SSD는 parking 데이터에서는 준수한 성능을 보였지만 cctv같은 디텍팅할 물체가 대부분 작기 때문에 제대로 학습이 되지않았다. SSD는 작은 물체 디텍션에는 성능이 좋지않은 모델이다.

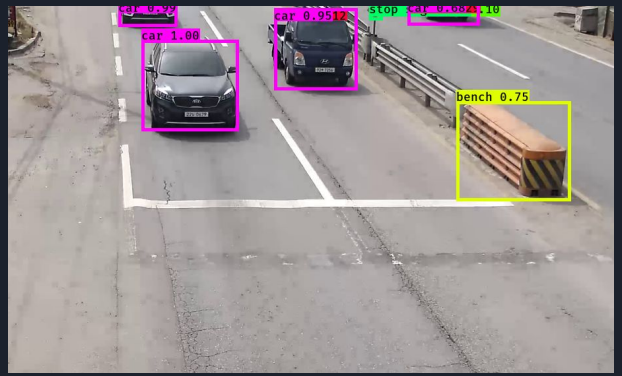


모델 해결 과정 - 학습

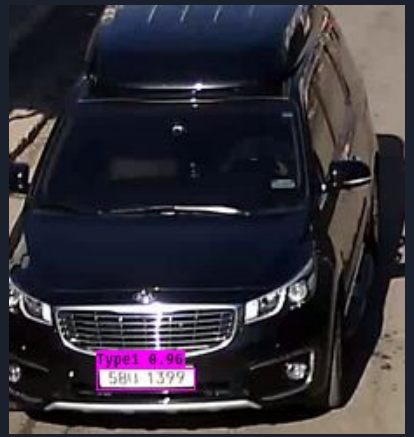
SSD, YOLO V3 결과 모두 까마득. No bounding box output ..

너무 작은 object라 one stage detector로는 학습을 시키기 힘들다 판단.

Faster_rcnn 은 속도가 느린데..



Detection 2번 수행!



모델 해결 과정 - 학습

CRNN

1. 기존 CRNN논문과 다르게 구현한 crnn은 lstm input 전에 fully connected layer를 하나 더 거치게 되고 인풋도 32가 아닌 64를 받아왔다.
2. 이것이 좀더 공간적인 정보를 잘 통합하여 세로로 2줄이 있는 번호판을 더 잘읽게 된다.
3. 위 아이디어는 Keras 공식 튜토리얼의 OCR프로젝트를 참고하면서 얻었다.

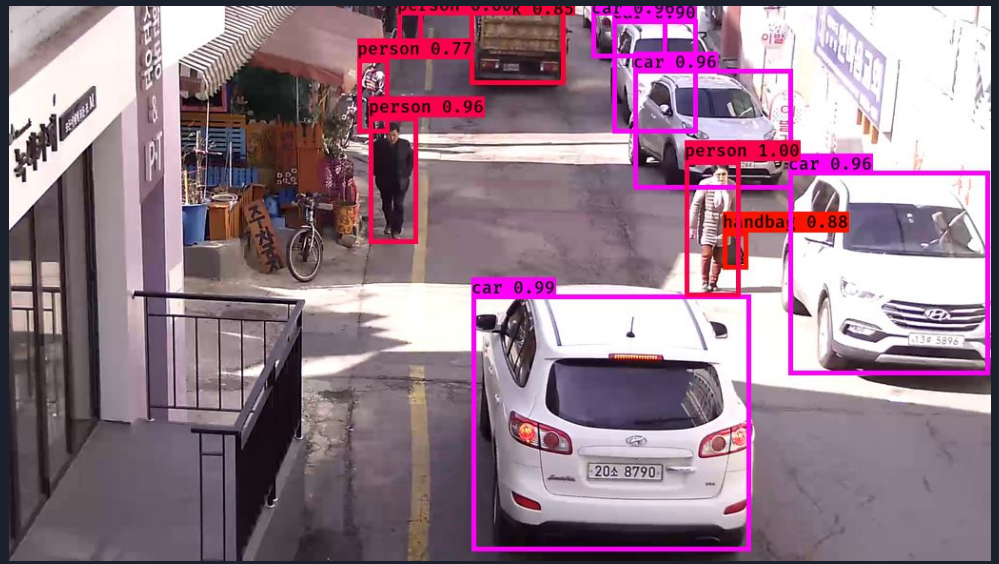
input_img (InputLayer)	(None, 128, 64, 1)
conv1 (Conv2D)	(None, 128, 64, 64)
batch_normalization_1 (BatchNor	(None, 128, 64, 64)
activation_1 (Activation)	(None, 128, 64, 64)
max1 (MaxPooling2D)	(None, 64, 32, 64)
conv2 (Conv2D)	(None, 64, 32, 128)
batch_normalization_2 (BatchNor	(None, 64, 32, 128)
activation_2 (Activation)	(None, 64, 32, 128)
max2 (MaxPooling2D)	(None, 32, 16, 128)
conv3 (Conv2D)	(None, 32, 16, 256)
batch_normalization_3 (BatchNor	(None, 32, 16, 256)
activation_3 (Activation)	(None, 32, 16, 256)
conv4 (Conv2D)	(None, 32, 16, 256)
batch_normalization_4 (BatchNor	(None, 32, 16, 256)
activation_4 (Activation)	(None, 32, 16, 256)
max3 (MaxPooling2D)	(None, 32, 8, 256)
conv5 (Conv2D)	(None, 32, 8, 512)
batch_normalization_5 (BatchNor	(None, 32, 8, 512)
activation_5 (Activation)	(None, 32, 8, 512)
conv6 (Conv2D)	(None, 32, 8, 512)
batch_normalization_6 (BatchNor	(None, 32, 8, 512)
activation_6 (Activation)	(None, 32, 8, 512)
max4 (MaxPooling2D)	(None, 32, 4, 512)
con7 (Conv2D)	(None, 32, 4, 512)
batch_normalization_7 (BatchNor	(None, 32, 4, 512)
activation_7 (Activation)	(None, 32, 4, 512)
reshape (Reshape)	(None, 32, 2048)
dense1 (Dense)	(None, 32, 64)
lstm1 (LSTM)	(None, 32, 256)
lstm1_b (LSTM)	(None, 32, 256)
add_1 (Add)	(None, 32, 256)
batch_normalization_8 (BatchNor	(None, 32, 256)
lstm2 (LSTM)	(None, 32, 256)
lstm2_b (LSTM)	(None, 32, 256)
concatenate_1 (Concatenate)	(None, 32, 512)
dense2 (Dense)	(None, 32, 50)
softmax (Activation)	(None, 32, 50)

결과 및 평가

결과 - Parking DATA(Detection)

결과 및 평가

결과 - cctv DATA(Detection)



결과 및 평가

결과 - CRNN


```
[7, 49, 49, 49, 49, 49, 49, 49, 49, 49, 49, 49, 49, 49, 49, 49, 49, 49, 25, 24, 38, 38, 27, 21, 24, 24, 18, 49,
24]
[7, 49, 25, 24, 38, 27, 21, 24, 18, 49, 24]
경기87바4717
```

결과 및 평가

평가

Parking 모델에대한 Test

```
num_bbox_examples,451
num_bbox_corrects,318
bbox_accuracy,70.51
num_rec_examples,436
num_rec_corrects,210
rec_accuracy,48.17
avg_pt,164.56
score,112.22
```

CCTV 모델에대한 Test

```
num_bbox_examples,285
num_bbox_corrects,209
bbox_accuracy,73.33
num_rec_examples,285
num_rec_corrects,128
rec_accuracy,44.91
avg_pt,75.08
score,120.74
```