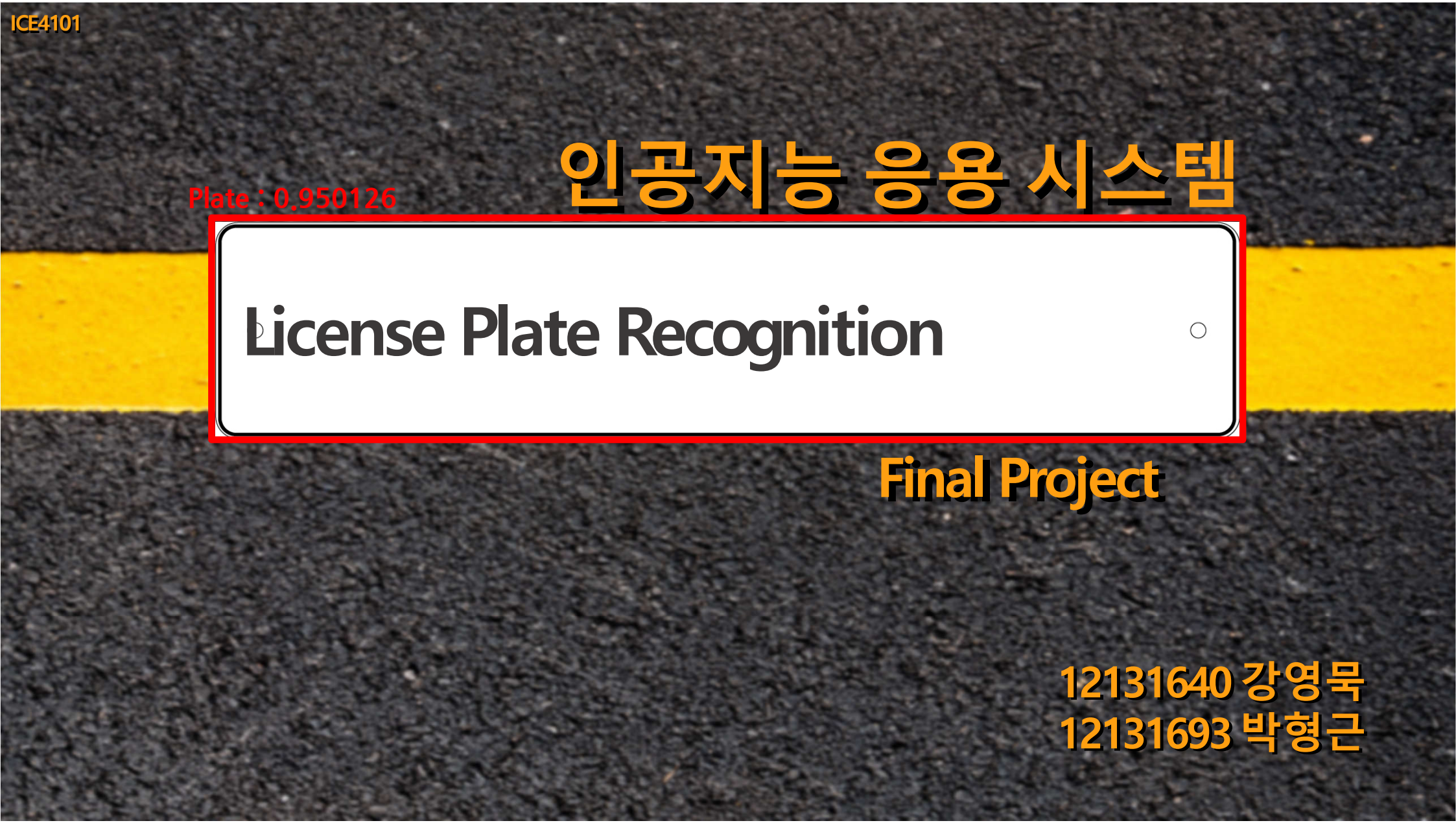


인공지능 응용 시스템

License Plate Recognition

Final Project

12131640 강영묵
12131693 박형근

목차

1.접근 방법

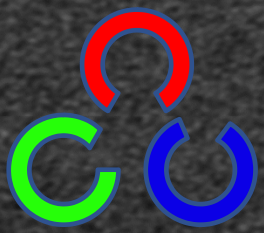
1.접근 방법

+

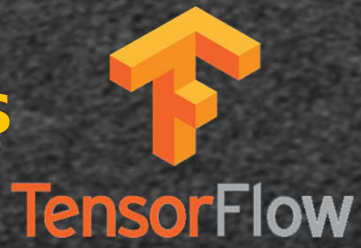
Object Detection

Object Classification

1.접근 방법



VS

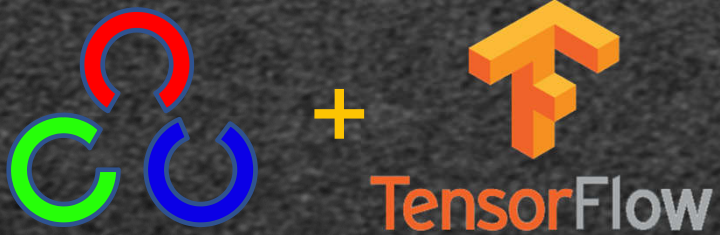


Object Detection

우선, 입력 image에 ROI(Region Of interest)인 번호판 영역을 추출해 내기 위하여, 영상처리 기법(OpenCV)을 이용하거나, CNN(Convolution Neural Network)을 이용할 수 있습니다.

저희는 그 중에서 CNN을 이용하여 번호판 영역을
검출하기로 결정하였습니다.

1.접근 방법

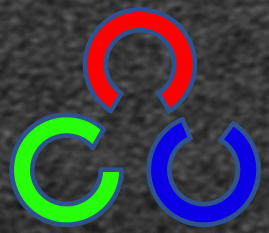


Object Detection

하지만, 오로지 CNN을 이용하는 것이 아닌, Parking Dataset과 같이 몹시 어두운 환경을 고려하여,OpenCV로 image pre-processing을 한 image를 Object Detector에 입력으로 넣어 주게 됩니다.

이 때, pre-processing으로는
Gamma correction(r=0.5)을 해주었습니다.

1.접근 방법



+

TensorFlow

Object Classification

Object Detector를 통해 얻은 번호판 영역을 OpenCV를 통해 각 번호 및 지역 영역에 대해 crop을 해주고, 각 crop된 image에 대해 classification을 진행하도록 하였습니다.

2.Object Detection

2.Object Detection

- SSD(Single Shot Multibox) modol 사용

1번의 네트워크 forwarding 사용하여 Object들의 Boundary box를 찾고 클래스를 인식

입력 이미지에 대한 CNN을 실행하고 여러 크기의 featur map을 이용하여 feature scale을 잘 사용할 수 있도록 모델링

후보 영역 추출 과정과 resampling 과정을 제거한 방식을 이용하여 높은 정확성과 빠른 속도를 얻음

2.Object Detection

구현 환경

CPU : Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz

GPU : NVIDIA GeForce 940MX

RAM : 8.0GB

Python : 3.6

Tensorflow : 1.8.0

트레이닝

Initial learning rate : 0.0001

Batch size : 4

Number of step : 30000

2.Object Detection

Data

트레이닝 이미지 개수 : Total Image : 21061
Train : 18949
Test : 2112

2.Object Detection

Parking

2.Object Detection

OpenCV 활용

OpenCV 이용. YCbCr의 감마값을 조정하여 pre-processing 작업을 하였다.
OpenCV를 추가하면 image 1 장당 16~18 ms 정도 느려지지만 성능은 향상

2.Object Detection

감마 값에 따른 인식률

Random 104 DB

r = 0.65

0.8705

r = 0.35

0.8875

r = 0.5

0.9042

2.Object Detection

성능평가 - 주차장

num_bbox_examples	285
num_bbox_corrects	275
bbox_accuracy	96.49
num_rec_examples	285
num_rec_corrects	0
rec_accuracy	0
avg_pt	1106.94
score	-4.2

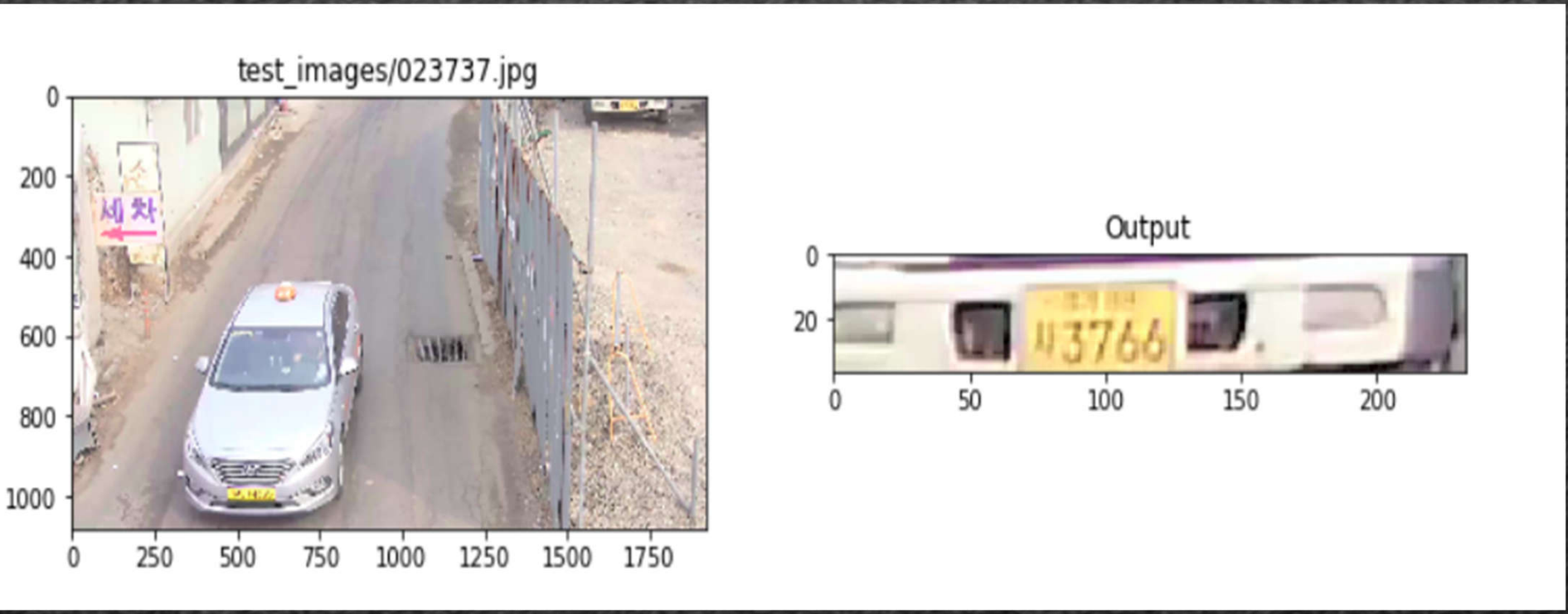
GT = 285

True Positive = 275

Accuracy = 96.49 %

2.Object Detection

CCTV



2.Object Detection

CCTV

2.Object Detection

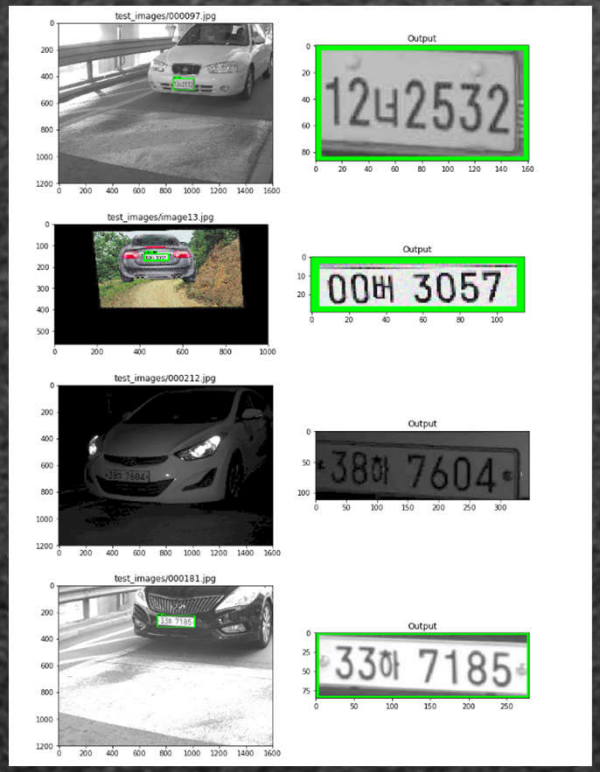
CCTV

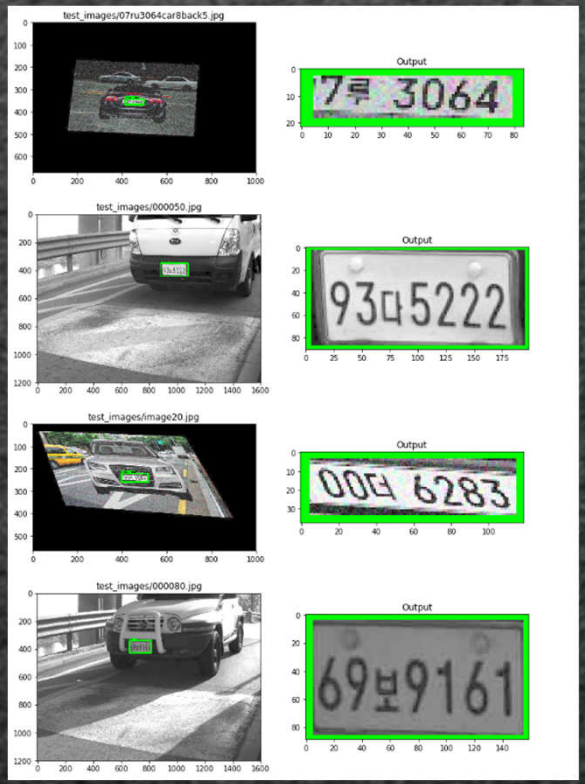
2.Object Detection

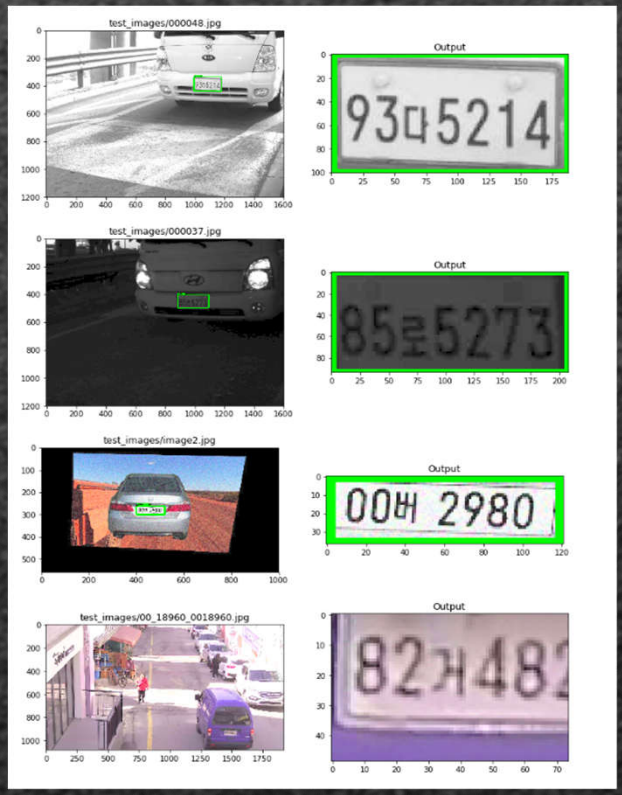
CCTV

2.Object Detection

Detection Output

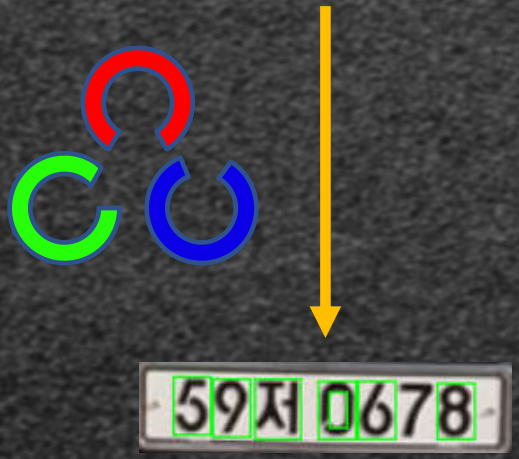






3.Object classification

3.Object Classification



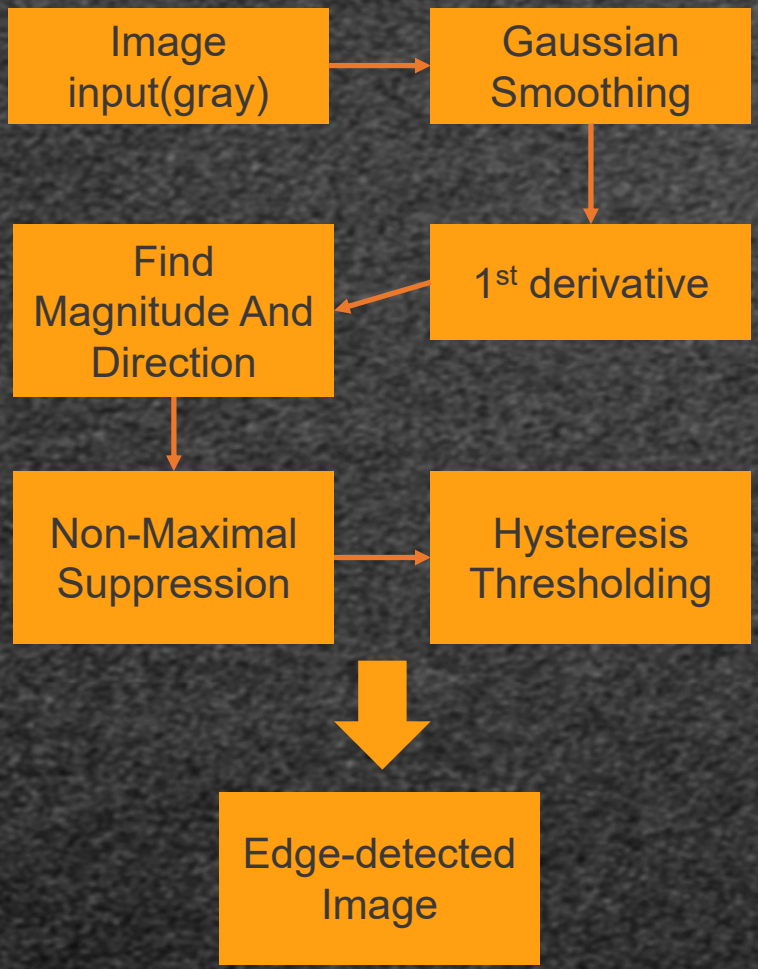
Blob with OpenCV

입력 image에서 추출해낸 번호판 영역에서 각각의 문자에 대해 Object Classification을 진행하게 됩니다.

그러기 위해서 각 문자에 대해서만 Crop을 해야 하는데, 이를 진행하기
위해서 OpenCV로 영상처리를 하여 edge를 통하여 각 문자의 영역을 추출해 냈습니다.

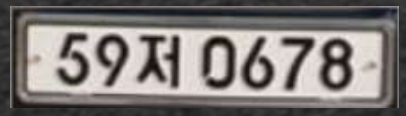
각 문자의 edge를 추출하기 위하여 시도한 방법으로는 Canny edge detection, DoG(Difference of Gaussian)였습니다.

3.Object Classification



Canny Edge Detection

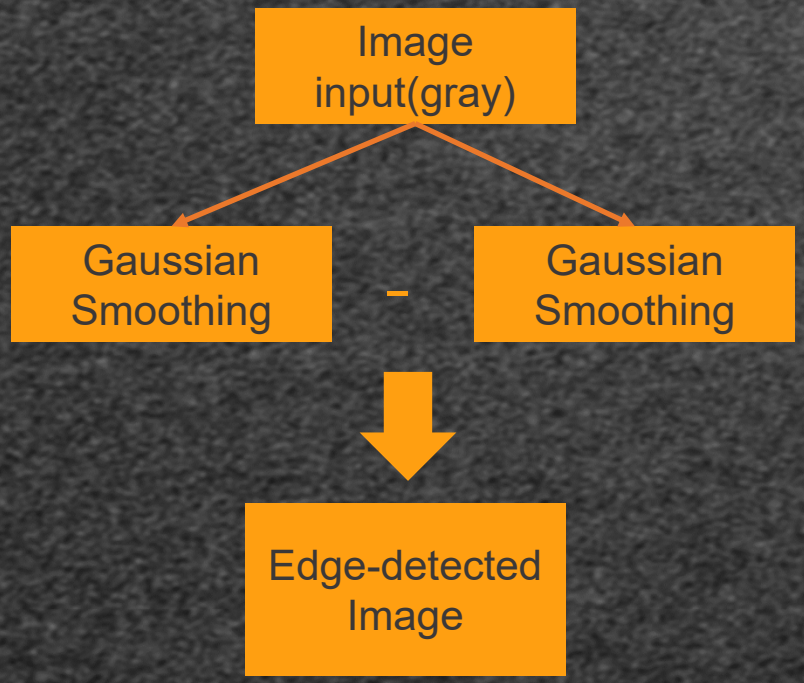
Canny Edge Detection은 왼쪽과 같은 알고리즘으로 이루어져 있습니다.



Canny Edge Detection 알고리즘을 통해 얻은 edge image는 눈으로
보기에도 아주 명확한 글자의 edge를 검출해 주었습니다.

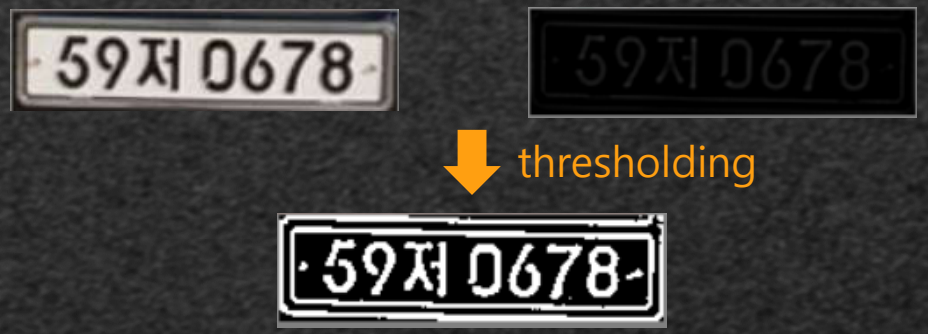
하지만 입력되는 image에 따라 threshold 값의 영향을 크게 받았고,
또한 복잡한 알고리즘 때문에 processing time이 증가한다는 단점이
존재하였습니다.

3.Object Classification



Difference of Gaussian

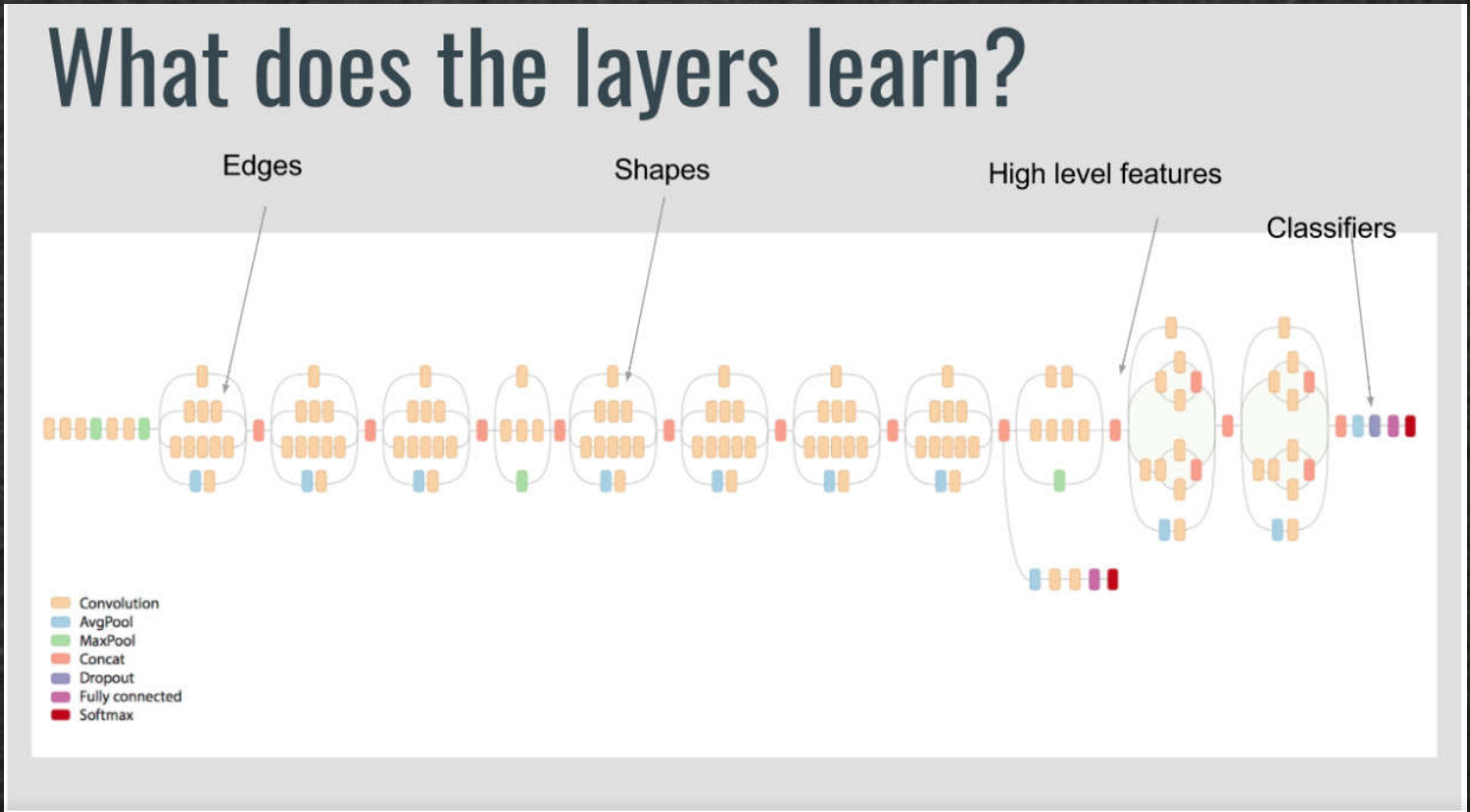
따라서 다음으로 고안한 방법은, DoG였습니다.



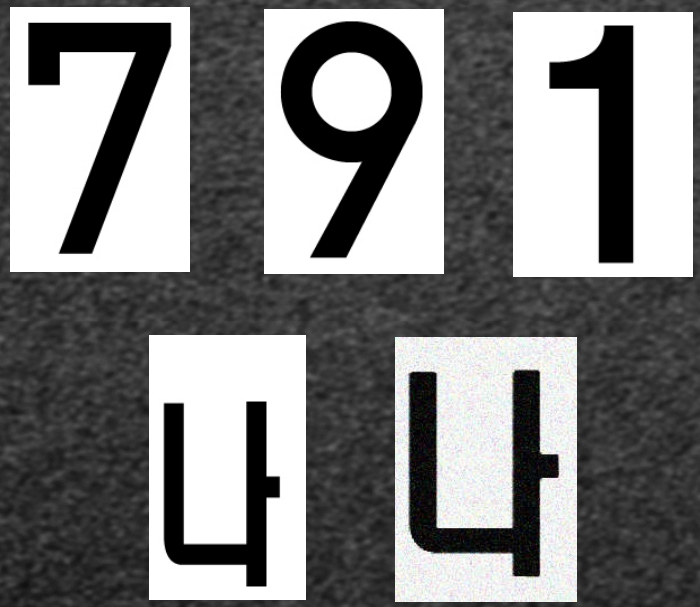
Canny Edge Detection에 비해 간단한 알고리즘 때문에
더 빠른 시간 내에 글자의 edge를 검출할 수 있었고,
입력된 image에 큰 상관없이 edge를 잘 검출해 주었습니다.

3.Object Classification

Inception v3



3.Object Classification



Inception v3

그리고 각각의 문자들을 분류해줄 classifier로써
Google의 inception v3 모델을 사용하여 tensorflow에서
Transfer learning하였습니다.

각각의 번호와 글자에 대해 training시키기 위하여 번호판의 폰트를
찾아보았지만 숫자 7과 9 등 실제 번호판의 번호 및 문자와 완벽하게
동일한 폰트를 찾지 못해 현행 번호판의 규격을 찾아 직접
illustrator로 직접 각각의 번호와 문자를 그려 왼쪽의 사진과 같이 저장하였습니다.

또한 입력 영상의 노이즈에 대비하기 위하여 training data에
Random noise를 추가한 image 또한 추가하였습니다.

4. 결과 분석

4.결과 분석

- Dection에서 Parking 영역에서는 좋은 성능을 보였지만 CCTV 영역에서는 주차된 차량을 인식하여서 인식하고자 하는 차량을 인식하지 못하였다.

  이미지의 절반 아래 부분에서 인식을 하여 개선하였다.

- 어두운 이미지에 대해서 OpenCV를 이용하여 Pre-Processing을 하여 성능을 높혔다.

- CCTV 영역에서 제공해준 GT에 오류가 많아서 정확하게 분석하고 파악하는데 시간이 많이 소모되었다.

- Classifier Accuricy에 관해서는 class 별 training image 수가 적어 Accurcy가 많이 떨어졌다.

감사합니다