LICENSE PLATE RECOGNITION SYSTEM

Saidrasulkhon {S
Bunyodbek B
Shokhrukh S}

인하대학교

HCI Lab
Inha University
Human-Computer Interaction Lab

EXISTING TECHNOLOGIES

Automatic **N**umber-**P**late **R**ecognition (ANPR)

Algorithms that software requires

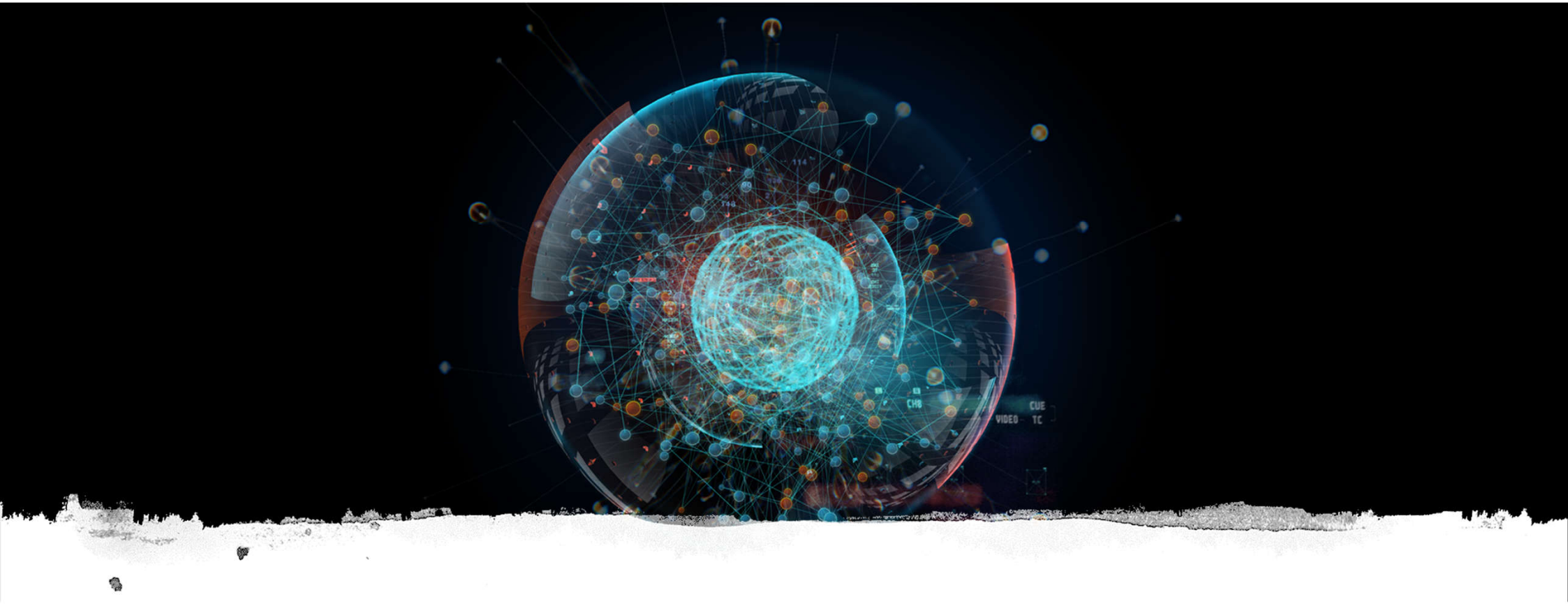
- Plate localization
- Plate orientation and sizing
- Normalization
- Character segmentation
- Optical character recognition.
- Syntactical/Geometrical analysis
- The averaging of the recognized value over multiple fields/images to produce a more reliable or confident result.

EXISTING TECHNOLOGIES

Automatic **N**umber-**P**late **R**ecognition (ANPR)

Difficulties in detection

- Poor file resolution
- Blurry images.
- Poor lighting and low contrast.
- An object obscuring.
- A different font.

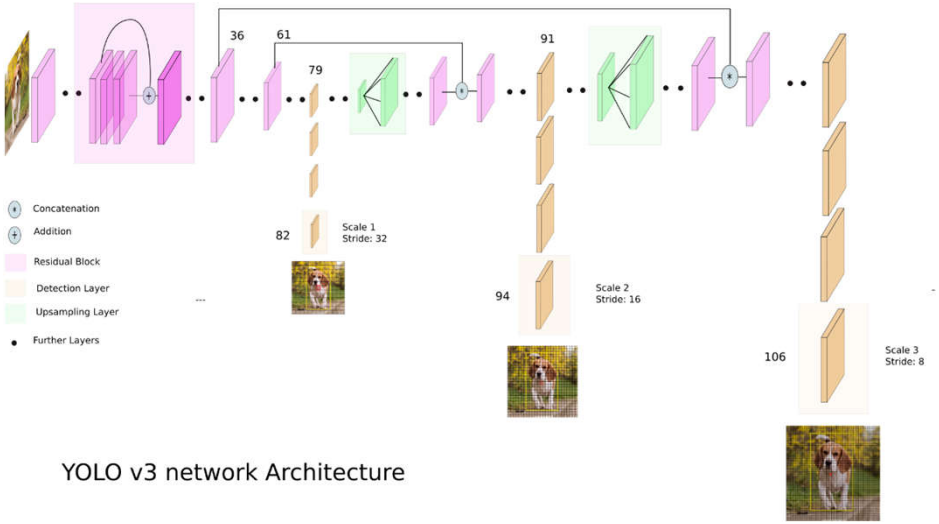


Our approach

- Various deep learning mechanisms were implemented, such as YOLOv3 and CRNN

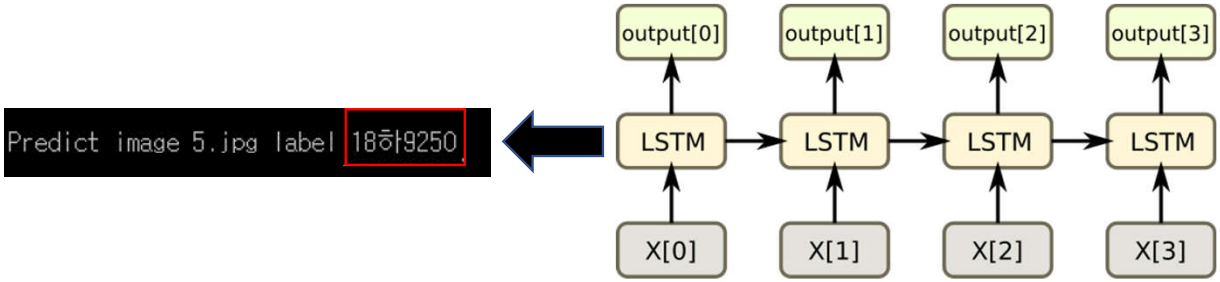
Detection and recognition using Deep Learning

YOLOv3

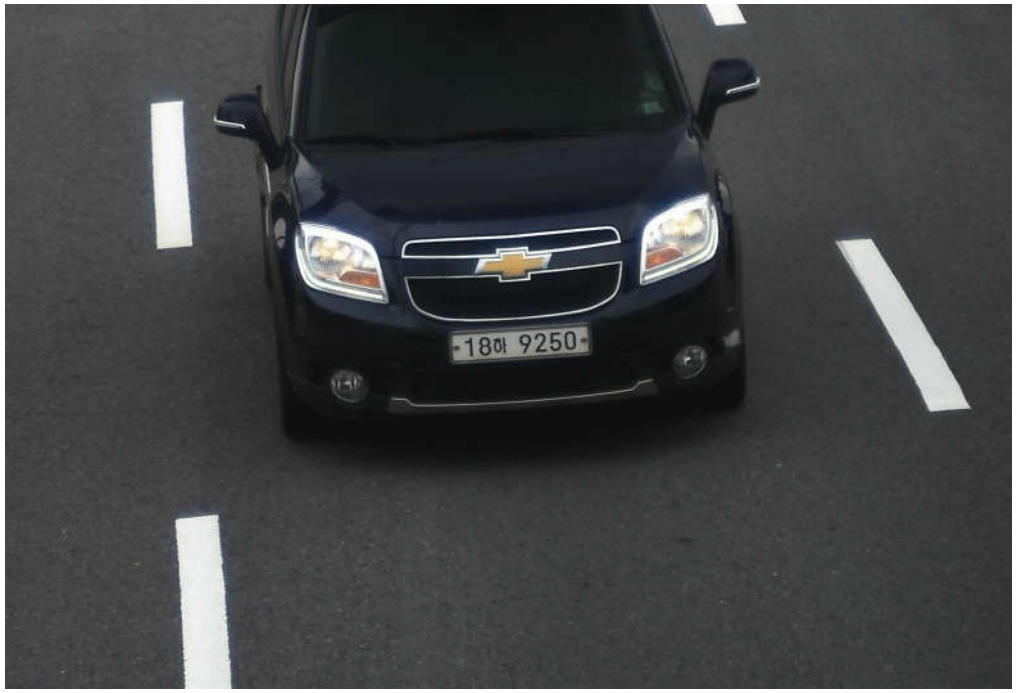


YOLO v3 network Architecture

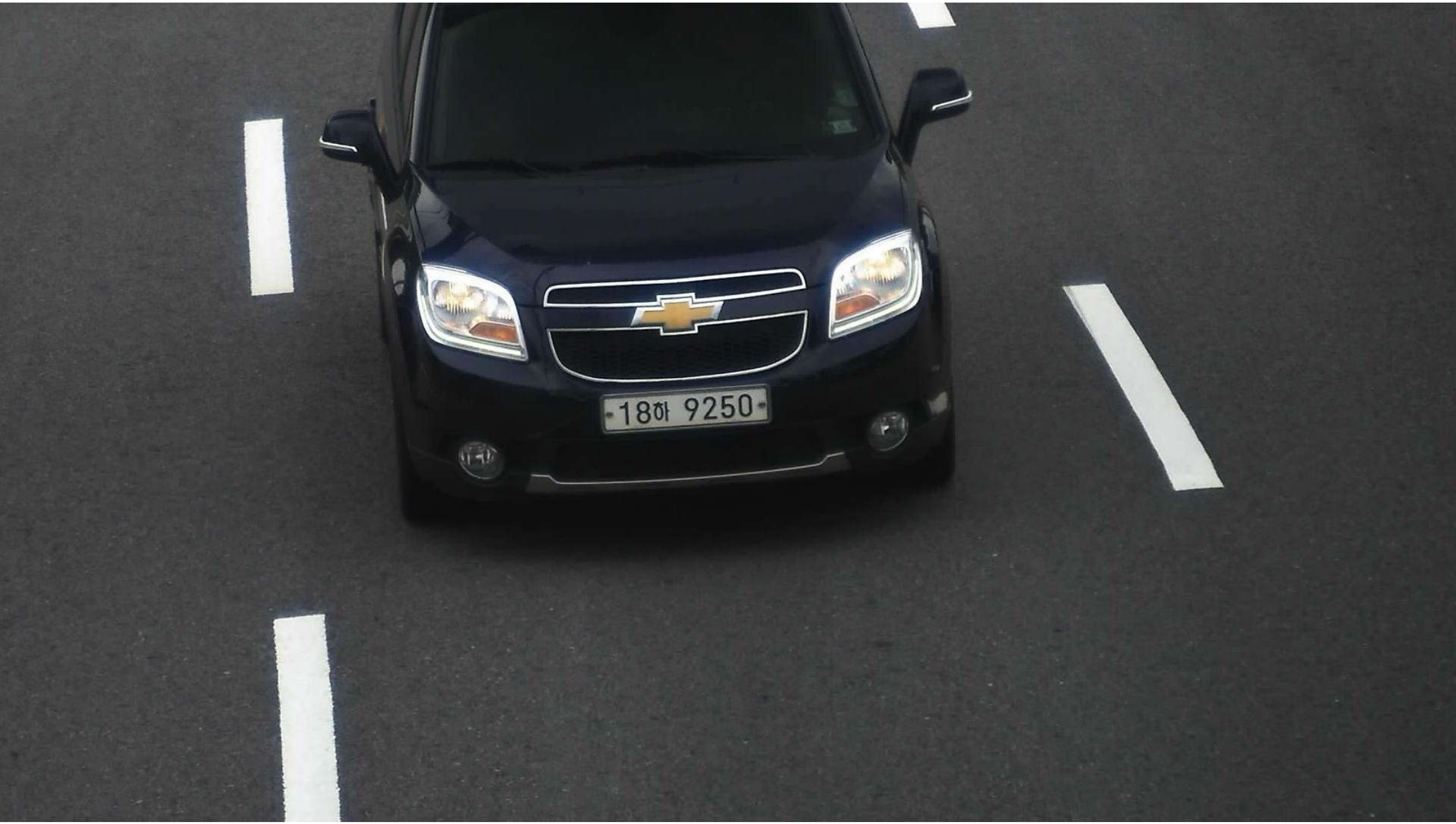
CRNN

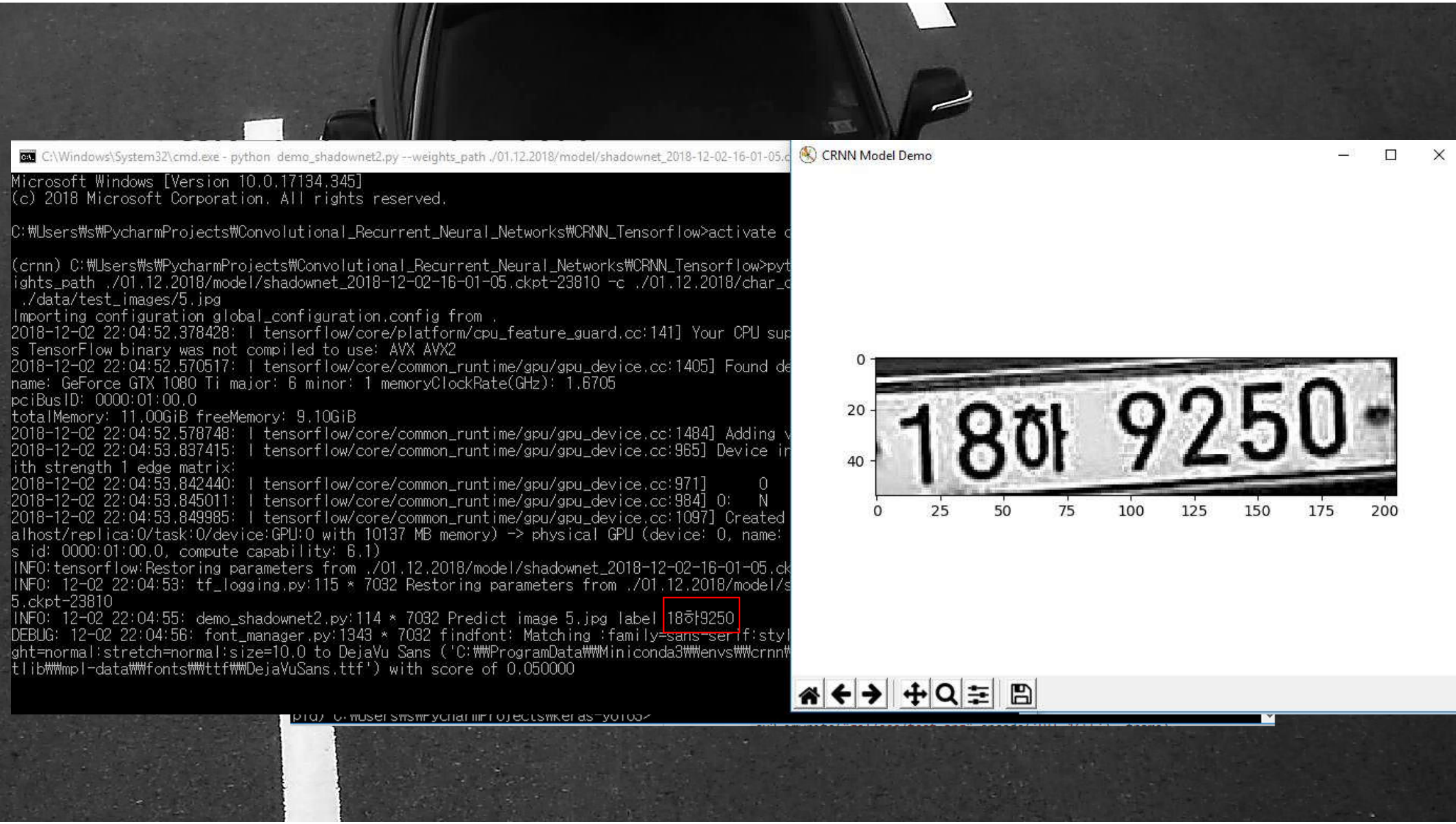


CRNN – Convolutional Recurrent Neural Network

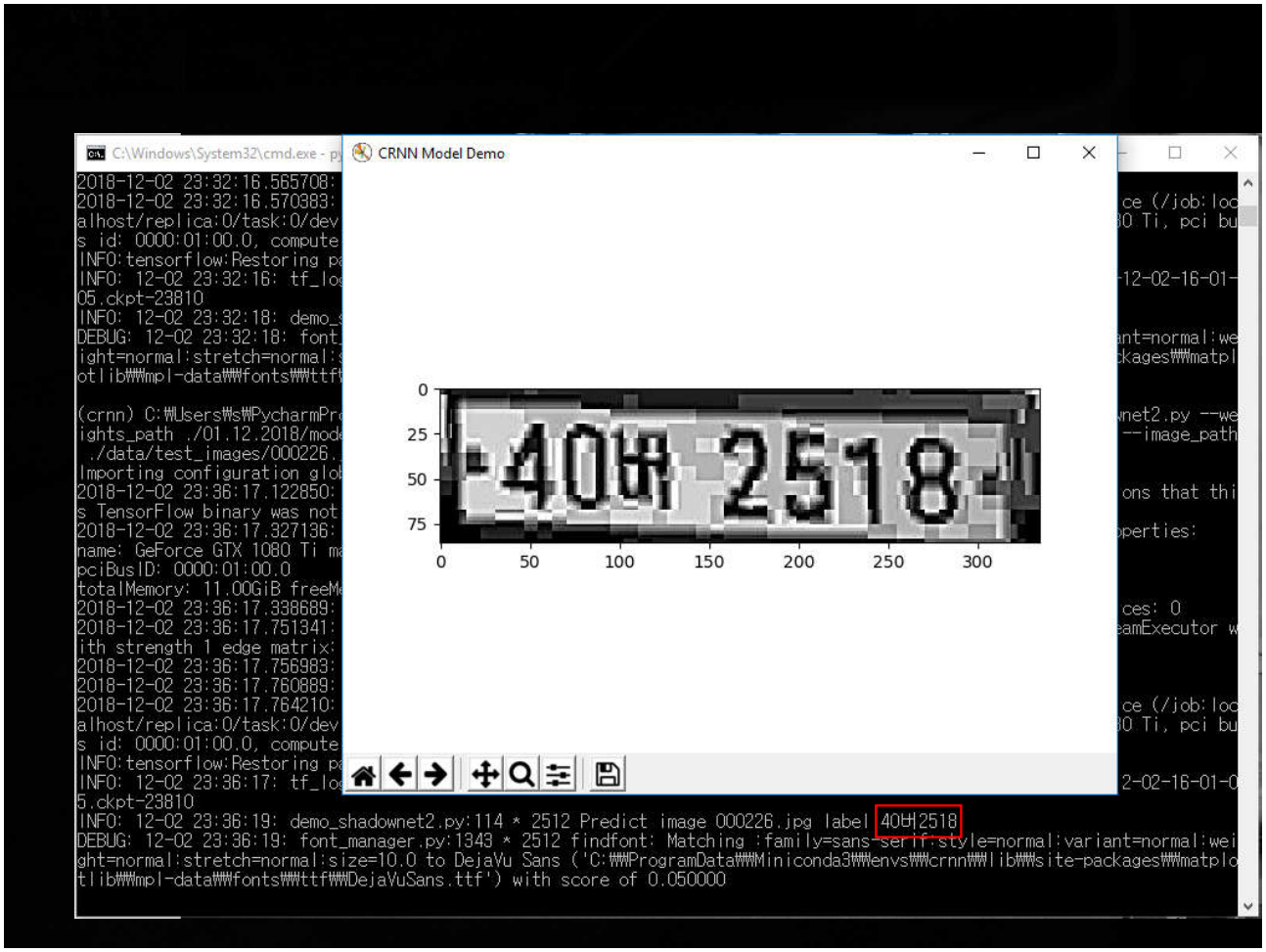


YOLOv3 for plate detection

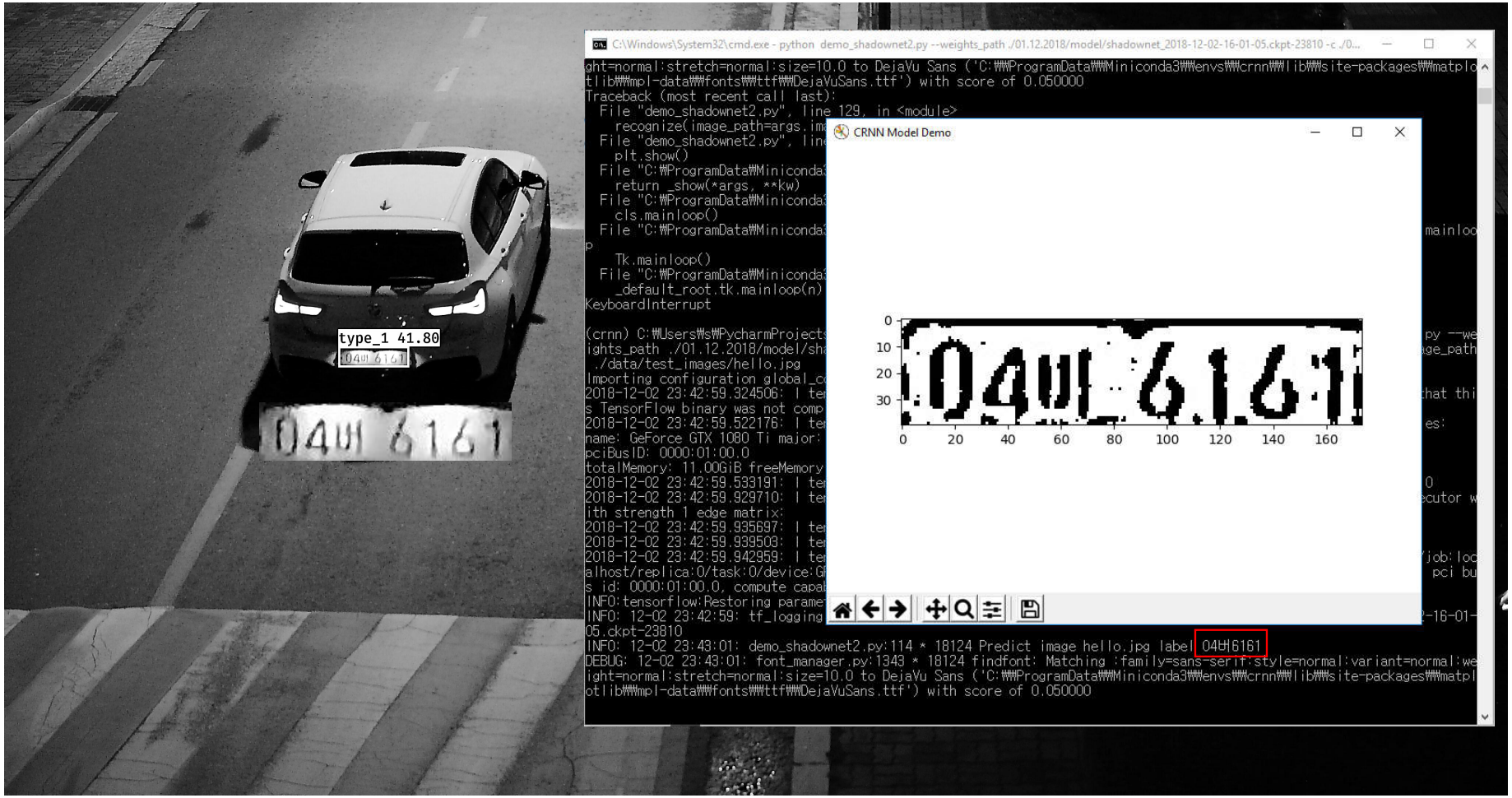

18하 9250


C:\Windows\System32\cmd.exe - python demo_shadownet2.py --weights_path ./01.12.2018/model/shadownet_2018-12-02-16-01-05.c
Microsoft Windows [Version 10.0.17134.345]
(c) 2018 Microsoft Corporation. All rights reserved.
C:₩Users₩s₩PycharmProjects₩Convolutional_Recurrent_Neural_Networks₩CRNN_Tensorflow>activate c
(crnn) C:₩Users₩s₩PycharmProjects₩Convolutional_Recurrent_Neural_Networks₩CRNN_Tensorflow>pyt
ights_path ./01.12.2018/model/shadownet_2018-12-02-16-01-05.ckpt-23810 -c ./01.12.2018/char_d
./data/test_images/5.jpg
Importing configuration global_configuration.config from .
2018-12-02 22:04:52.378428: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU sup
s TensorFlow binary was not compiled to use: AVX AVX2
2018-12-02 22:04:52.570517: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1405] Found de
name: GeForce GTX 1080 Ti major: 6 minor: 1 memoryClockRate(GHz): 1.6705
pciBusID: 0000:01:00.0
totalMemory: 11.00GiB freeMemory: 9.10GiB
2018-12-02 22:04:52.578748: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1484] Adding v
2018-12-02 22:04:53.837415: I tensorflow/core/common_runtime/gpu/gpu_device.cc:965] Device in
ith strength 1 edge matrix:
2018-12-02 22:04:53.842440: I tensorflow/core/common_runtime/gpu/gpu_device.cc:971] 0
2018-12-02 22:04:53.845011: I tensorflow/core/common_runtime/gpu/gpu_device.cc:984] 0: N
2018-12-02 22:04:53.849985: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1097] Created
alhost/replica:0/task:0/device:GPU:0 with 10137 MB memory) -> physical GPU (device: 0, name:
s id: 0000:01:00.0, compute capability: 6.1)
INFO:tensorflow:Restoring parameters from ./01.12.2018/model/shadownet_2018-12-02-16-01-05.ck
INFO: 12-02 22:04:53: tf_logging.py:115 * 7032 Restoring parameters from ./01.12.2018/model/s
5.ckpt-23810
INFO: 12-02 22:04:55: demo_shadownet2.py:114 * 7032 Predict image 5.jpg label 18하9250
DEBUG: 12-02 22:04:56: font_manager.py:1343 * 7032 findfont: Matching :family=sans-serif:styl
ght=normal:stretch=normal:size=10.0 to DejaVu Sans ('C:₩₩ProgramData₩₩Miniconda3₩₩envs₩₩crnn₩
tlib₩₩mpl-data₩₩fonts₩₩ttf₩₩DejaVuSans.ttf') with score of 0.050000
CRNN Model Demo
0
20
40
0
25
50
75
100
125
150
175
200

Low lighting and poor visible images



Low lighting and poor visible images

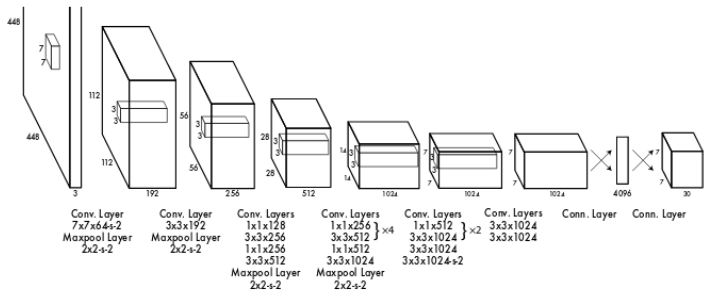


Used tools

YOLOv3 – 2 models

```
x1 = compose(
        DarknetConv2D_BN_Leaky(16, (3,3)),
        MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='same'),
        DarknetConv2D_BN_Leaky(32, (3,3)),
        MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='same'),
        DarknetConv2D_BN_Leaky(64, (3,3)),
        MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='same'),
        DarknetConv2D_BN_Leaky(128, (3,3)),
        MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='same'),
        DarknetConv2D_BN_Leaky(256, (3,3)))(inputs)
x2 = compose(
        MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='same'),
        DarknetConv2D_BN_Leaky(512, (3,3)),
        MaxPooling2D(pool_size=(2,2), strides=(1,1), padding='same'),
        DarknetConv2D_BN_Leaky(1024, (3,3)),
        DarknetConv2D_BN_Leaky(256, (1,1)))(x1)
y1 = compose(
        DarknetConv2D_BN_Leaky(512, (3,3)),
        DarknetConv2D(num_anchors*(num_classes+5), (1,1)))(x2)

x2 = compose(
        DarknetConv2D_BN_Leaky(128, (1,1)),
        UpSampling2D(2))(x2)
y2 = compose(
        Concatenate(),
        DarknetConv2D_BN_Leaky(256, (3,3)),
        DarknetConv2D(num_anchors*(num_classes+5), (1,1)))([x2,x1])
```

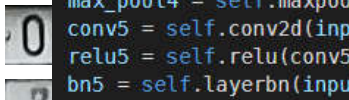


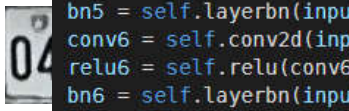
MODEL: 9500(5000 real + 4500 augmented)

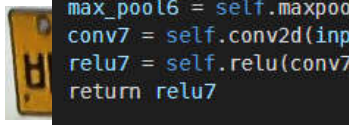
CRNN model

- Only one model was used
- >
- >

```
conv1 = self.__conv_stage(inputdata=inputdata, out_dims=64, name='conv1')  # batch*16*50*64
conv2 = self.__conv_stage(inputdata=conv1, out_dims=128, name='conv2')  # batch*8*25*128
conv3 = self.conv2d(inputdata=conv2, out_channel=256, kernel_size=3, stride=1, use_bias=False, name='conv3')  # batch*8*25*256
relu3 = self.relu(conv3) # batch*8*25*256
conv4 = self.conv2d(inputdata=relu3, out_channel=256, kernel_size=3, stride=1, use_bias=False, name='conv4')  # batch*8*25*256
relu4 = self.relu(conv4)  # batch*8*25*256
max_pool4 = self.maxpooling(inputdata=relu4, kernel_size=[2, 1], stride=[2, 1], padding='VALID')  # batch*4*25*256
conv5 = self.conv2d(inputdata=max_pool4, out_channel=512, kernel_size=3, stride=1, use_bias=False, name='conv5')  # batch*4*25*512
relu5 = self.relu(conv5)  # batch*4*25*512
bn5 = self.layerbn(inputdata=relu5, is_training=self.phase == 'train')  # batch*4*25*512
conv6 = self.conv2d(inputdata=bn5, out_channel=512, kernel_size=3, stride=1, use_bias=False, name='conv6')  # batch*4*25*512
relu6 = self.relu(conv6)  # batch*4*25*512
bn6 = self.layerbn(inputdata=relu6, is_training=self.phase == 'train')  # batch*4*25*512
max_pool6 = self.maxpooling(inputdata=bn6, kernel_size=[2, 1], stride=[2, 1])  # batch*2*25*512
conv7 = self.conv2d(inputdata=max_pool6, out_channel=512, kernel_size=2, stride=[2, 1], use_bias=False, name='conv7')  # batch*1*25*512
relu7 = self.relu(conv7)  # batch*1*25*512
return relu7
```







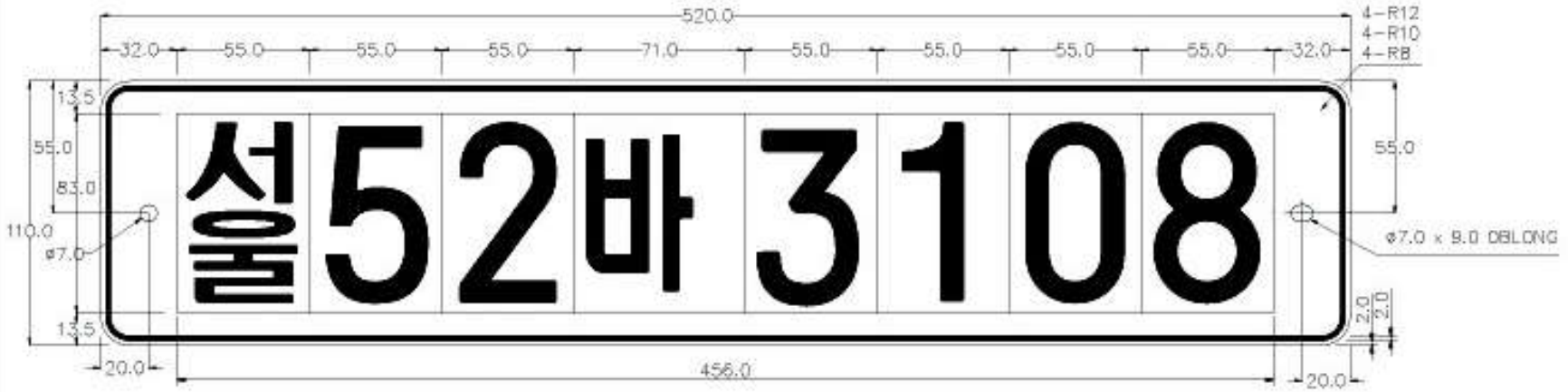
[별표 4]

자동차 운수 사업용 보통 등록 번호판 (520mm x 110mm)

1. 번호판의 규격

Width:
32mm - 103px
55mm - 178px
71mm - 229px
520mm - 1679px

Height:
13.5mm - 44px
83mm - 268px
110mm - 355px



가로 520mm x 세로 110mm(허용 범위: ± 0.5mm)

Results:

Our test results

num_bbox_examples	451
num_bbox_corrects	396
bbox_accuracy	87.80
num_rec_examples	451
num_rec_corrects	204
rec_accuracy	45.23
avg_pt	68.26
score	136.20

num_bbox_examples	285
num_bbox_corrects	285
bbox_accuracy	100
num_rec_examples	285
num_rec_corrects	187
rec_accuracy	65.61
avg_pt	72.68
Score	168.34

Total: 304.54

Real test results

num_bbox_examples	451
num_bbox_corrects	362
bbox_accuracy	80.27
num_rec_examples	436
num_rec_corrects	197
rec_accuracy	45.18
avg_pt	109.35
score	124.51

num_bbox_examples	285
num_bbox_corrects	279
bbox_accuracy	97.89
num_rec_examples	285
num_rec_corrects	148
rec_accuracy	51.93
avg_pt	112.28
score	148.6

Total: 273.11

Q&A