Congestion Analyzer

EatCoder

지도 교수 : 장재영

김한솔 – 1292011
이승헌 – 1292034
황희재 – 1292047
용영재 - 1292027

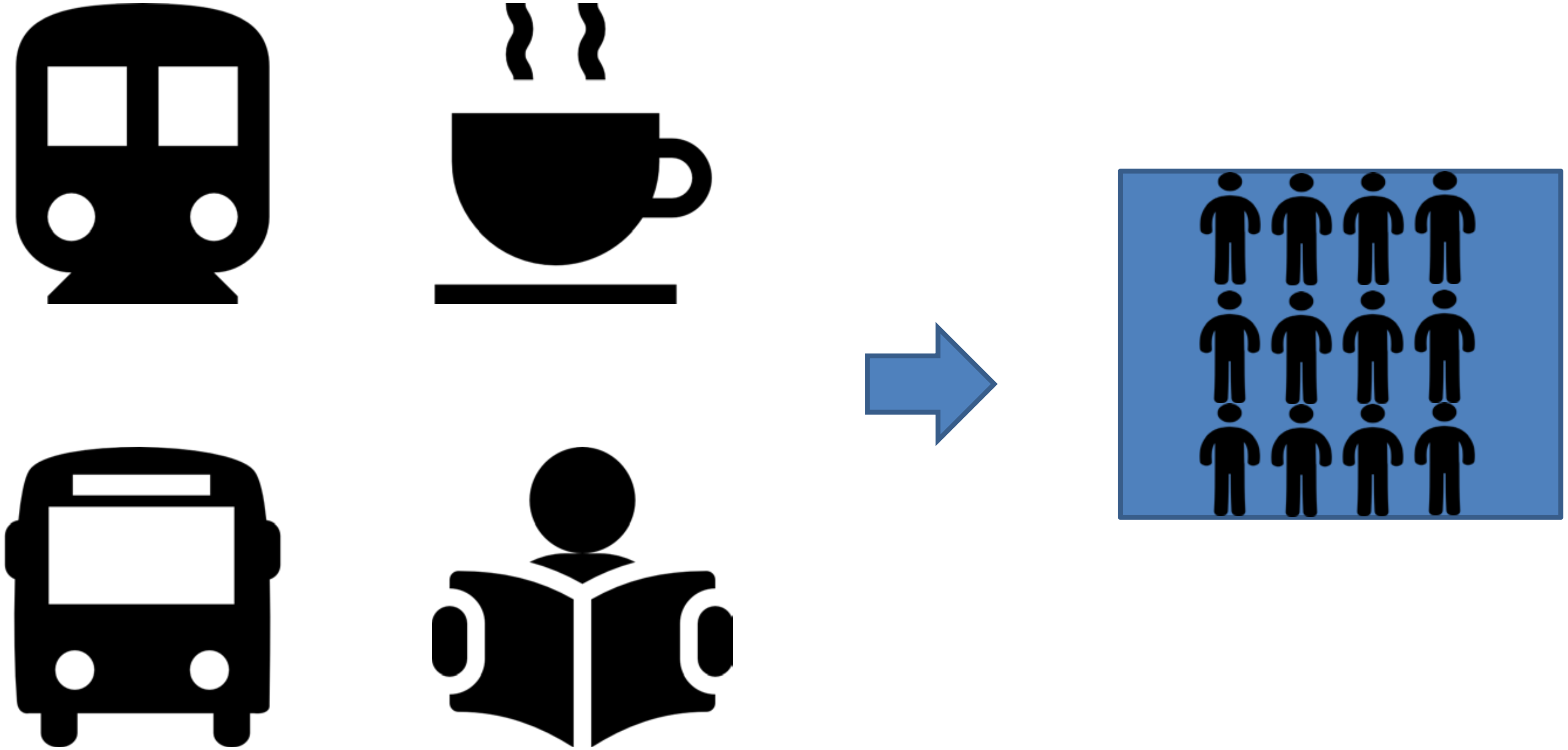
Intro

1. 프로젝트 개요
2. 프로젝트 구조
3. 프로젝트 세부사항
4. Q&A

프로젝트 개요

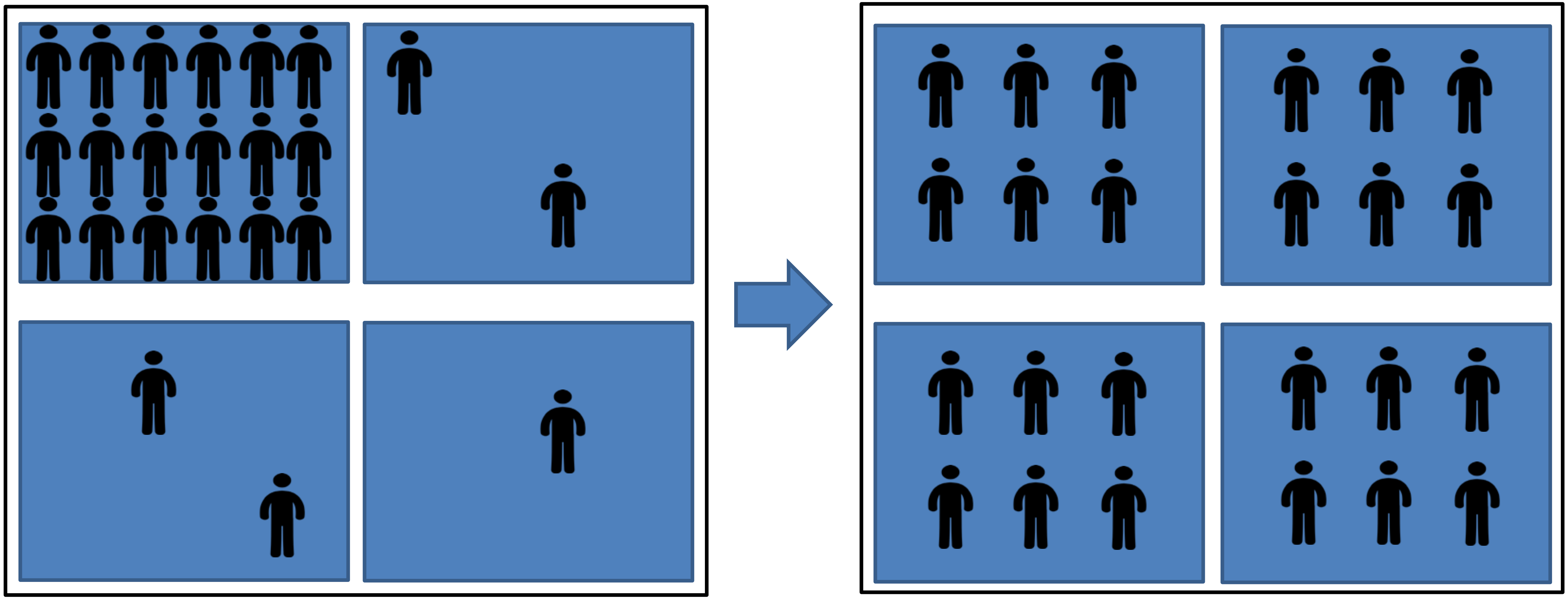
프로젝트 개요

인원이 집중되는 장소



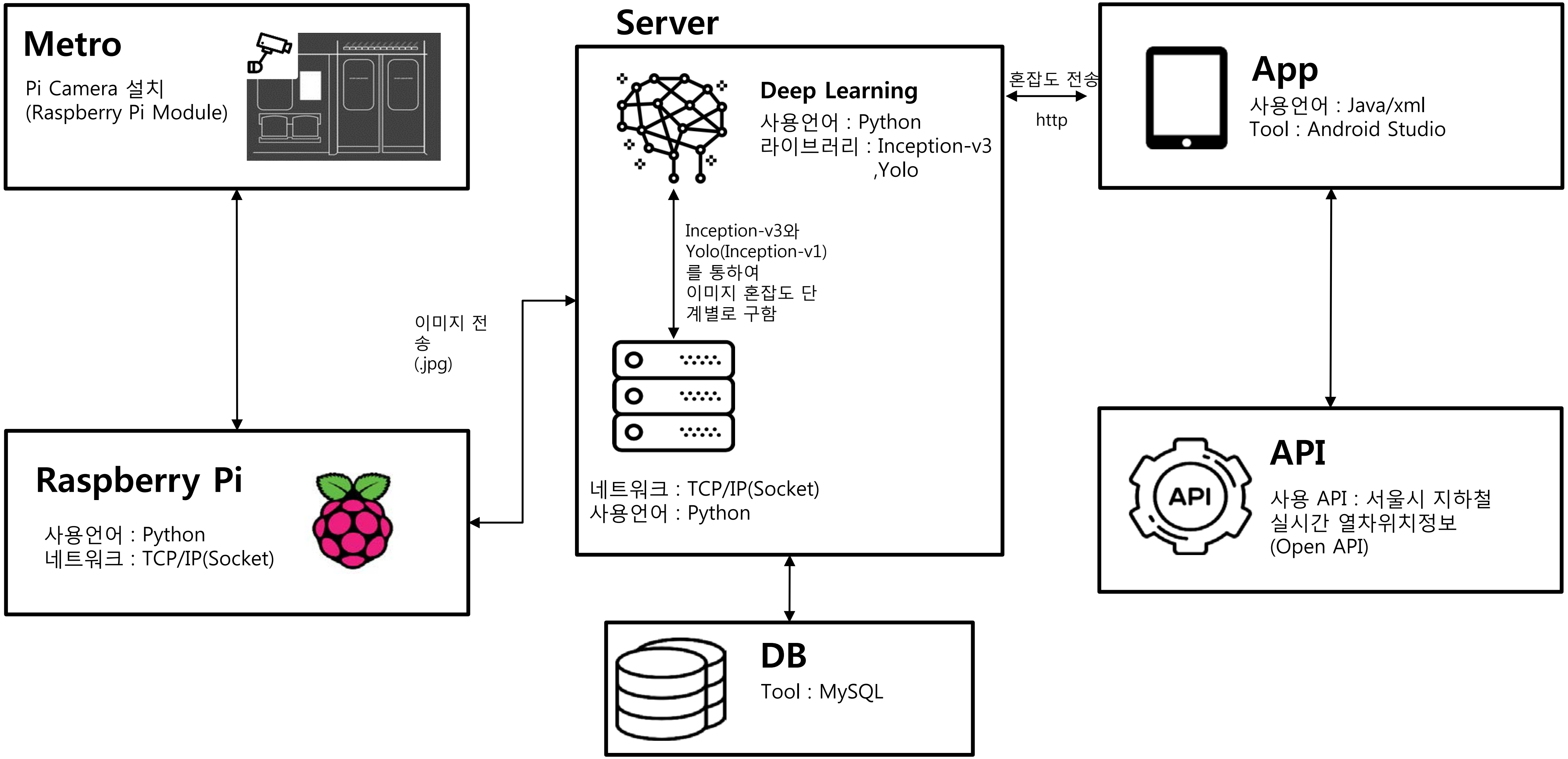
프로젝트 개요

특정 시간대, 특정 칸, 특정 구역



프로젝트 구조

프로젝트 구조



Metro

Pi Camera 설치
(Raspberry Pi Module)

Raspberry Pi

사용언어 : Python
네트워크 : TCP/IP(Socket)

이미지 전송
(.jpg)

Server

Deep Learning

사용언어 : Python
라이브러리 : Inception-v3
,Yolo

Inception-v3와
Yolo(Inception-v1)
를 통하여
이미지 혼잡도 단계별로 구함

네트워크 : TCP/IP(Socket)
사용언어 : Python

DB

Tool : MySQL

혼잡도 전송
http

App

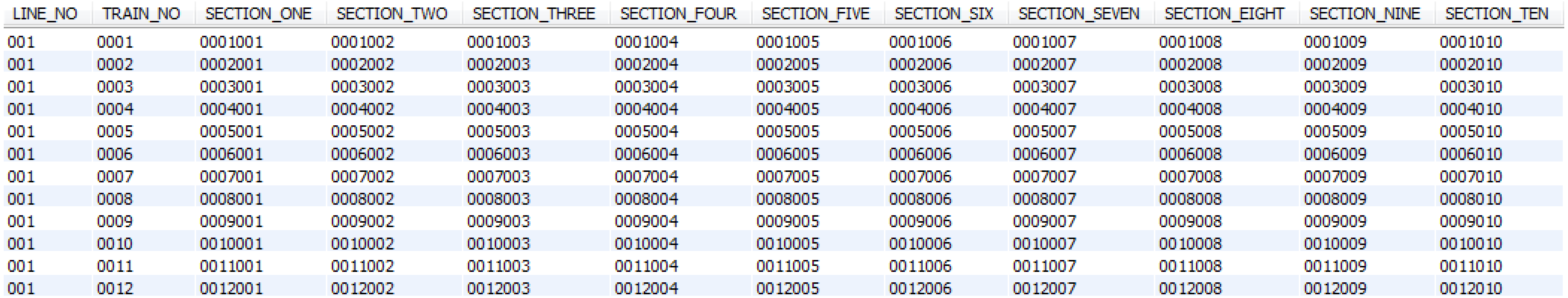
사용언어 : Java/xml
Tool : Android Studio

API

사용 API : 서울시 지하철
실시간 열차위치정보
(Open API)

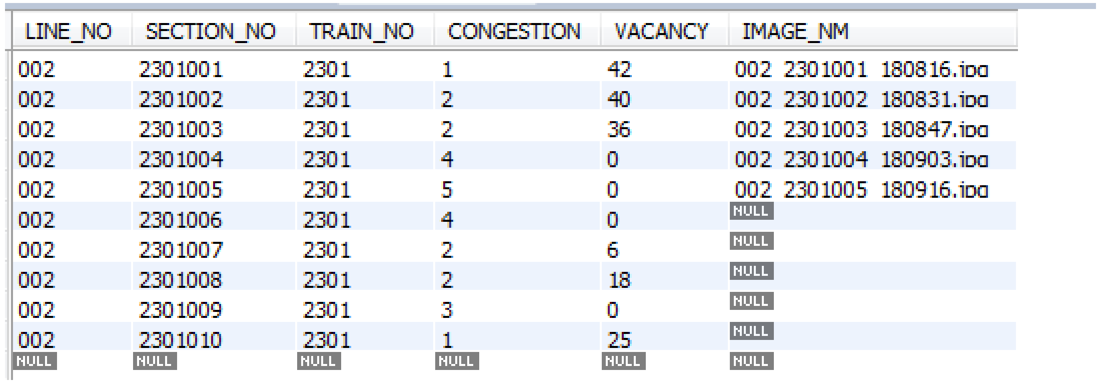
프로젝트 구조

METRO_SECTION



LINE_NO	TRAIN_NO	SECTION_ONE	SECTION_TWO	SECTION_THREE	SECTION_FOUR	SECTION_FIVE	SECTION_SIX	SECTION_SEVEN	SECTION_EIGHT	SECTION_NINE	SECTION_TEN
001	0001	0001001	0001002	0001003	0001004	0001005	0001006	0001007	0001008	0001009	0001010
001	0002	0002001	0002002	0002003	0002004	0002005	0002006	0002007	0002008	0002009	0002010
001	0003	0003001	0003002	0003003	0003004	0003005	0003006	0003007	0003008	0003009	0003010
001	0004	0004001	0004002	0004003	0004004	0004005	0004006	0004007	0004008	0004009	0004010
001	0005	0005001	0005002	0005003	0005004	0005005	0005006	0005007	0005008	0005009	0005010
001	0006	0006001	0006002	0006003	0006004	0006005	0006006	0006007	0006008	0006009	0006010
001	0007	0007001	0007002	0007003	0007004	0007005	0007006	0007007	0007008	0007009	0007010
001	0008	0008001	0008002	0008003	0008004	0008005	0008006	0008007	0008008	0008009	0008010
001	0009	0009001	0009002	0009003	0009004	0009005	0009006	0009007	0009008	0009009	0009010
001	0010	0010001	0010002	0010003	0010004	0010005	0010006	0010007	0010008	0010009	0010010
001	0011	0011001	0011002	0011003	0011004	0011005	0011006	0011007	0011008	0011009	0011010
001	0012	0012001	0012002	0012003	0012004	0012005	0012006	0012007	0012008	0012009	0012010

SECTION_CONGESTION

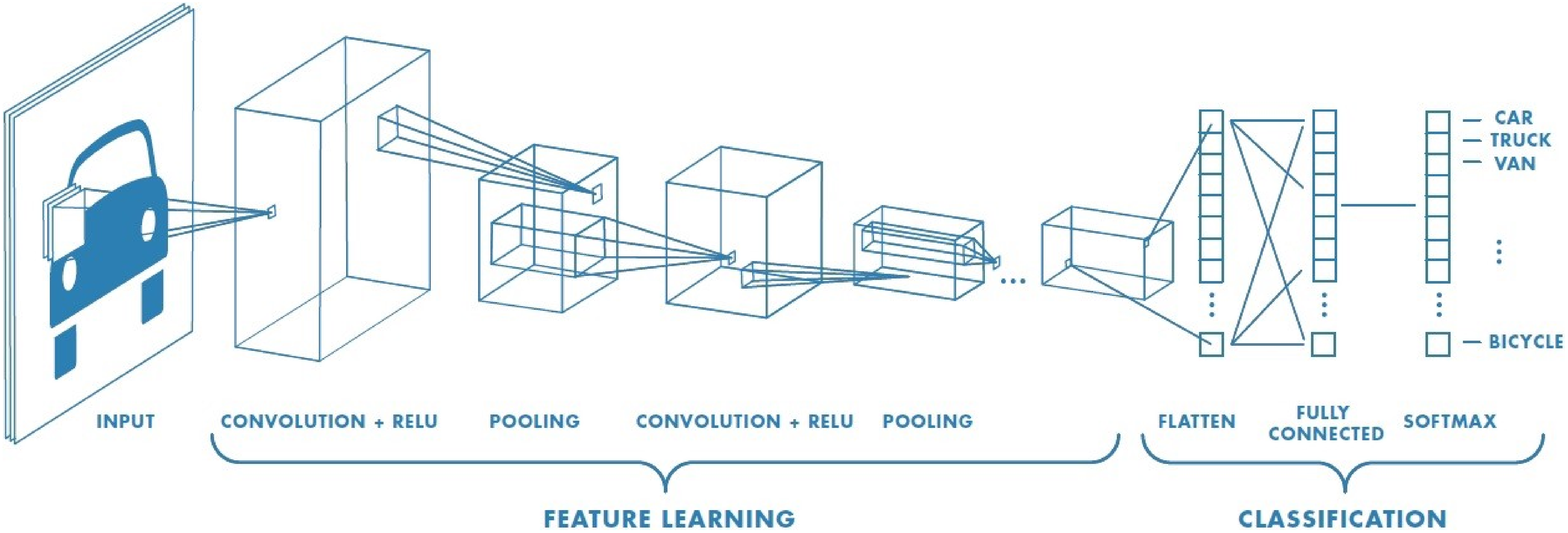


LINE_NO	SECTION_NO	TRAIN_NO	CONGESTION	VACANCY	IMAGE_NM
002	2301001	2301	1	42	002 2301001 180816.jpg
002	2301002	2301	2	40	002 2301002 180831.jpg
002	2301003	2301	2	36	002 2301003 180847.jpg
002	2301004	2301	4	0	002 2301004 180903.jpg
002	2301005	2301	5	0	002 2301005 180916.jpg
002	2301006	2301	4	0	NULL
002	2301007	2301	2	6	NULL
002	2301008	2301	2	18	NULL
002	2301009	2301	3	0	NULL
002	2301010	2301	1	25	NULL
NULL	NULL	NULL	NULL	NULL	NULL

프로젝트 세부사항

프로젝트 세부사항(Deep learning)

CNN(Inception v3)

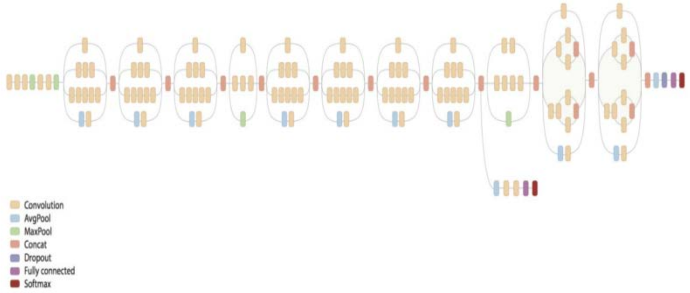


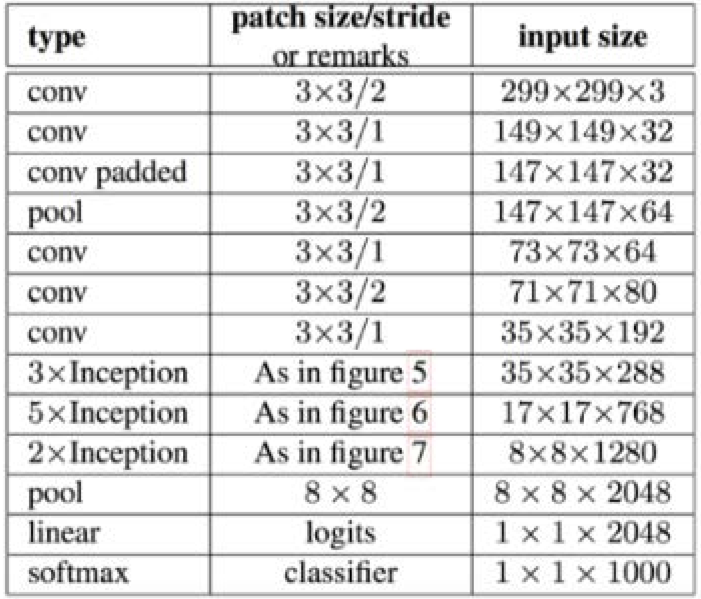
Input – Conv – RELU – Pool – RELU – Conv – RELU – PooL – Fully connected

프로젝트 세부사항

Inception-v3

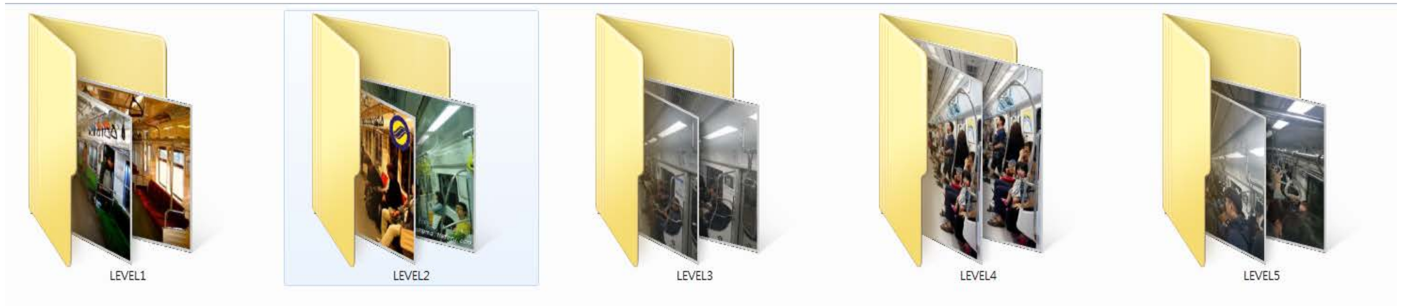
모델 구성





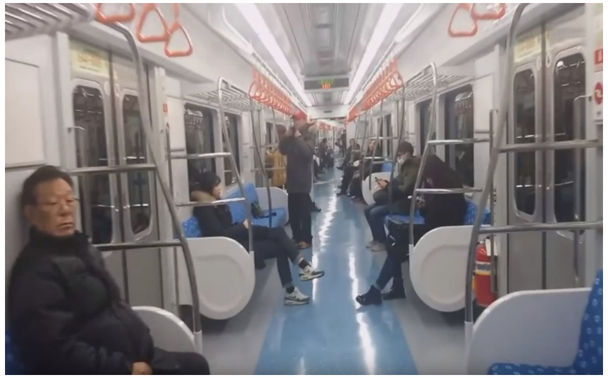
type	patch size/stride or remarks	input size
conv	3×3/2	299×299×3
conv	3×3/1	149×149×32
conv padded	3×3/1	147×147×32
pool	3×3/2	147×147×64
conv	3×3/1	73×73×64
conv	3×3/2	71×71×80
conv	3×3/1	35×35×192
3×Inception	As in figure 5	35×35×288
5×Inception	As in figure 6	17×17×768
2×Inception	As in figure 7	8×8×1280
pool	8 × 8	8 × 8 × 2048
linear	logits	1 × 1 × 2048
softmax	classifier	1 × 1 × 1000

Train data

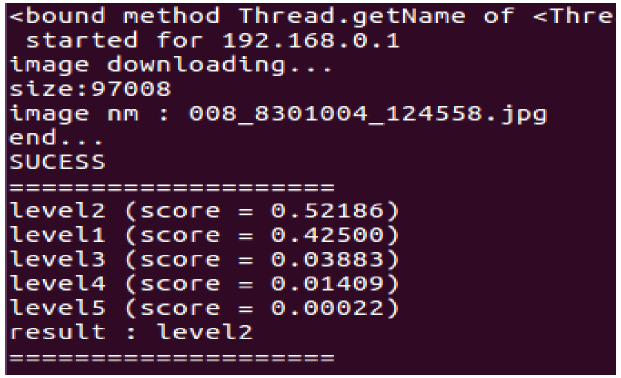


LEVEL1::94개 LEVEL2::155개 LEVEL3:182개 LEVEL4:160개 LEVEL5:131개

input data

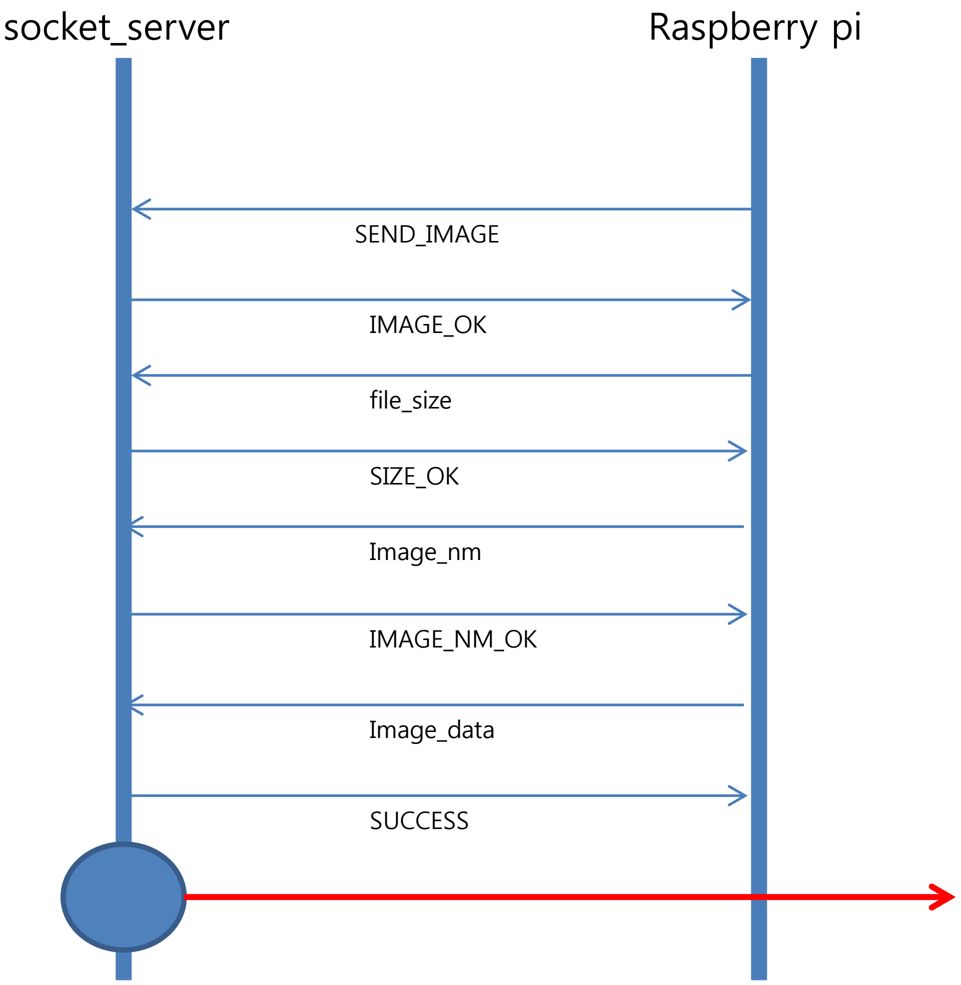


result



```
<bound method Thread.getName of <Thre
 started for 192.168.0.1
image downloading...
size:97008
image nm : 008_8301004_124558.jpg
end...
SUCESS
====================
level2 (score = 0.52186)
level1 (score = 0.42500)
level3 (score = 0.03883)
level4 (score = 0.01409)
level5 (score = 0.00022)
result : level2
====================
```

Protocol



socket_server

Raspberry pi

SEND_IMAGE

IMAGE_OK

file_size

SIZE_OK

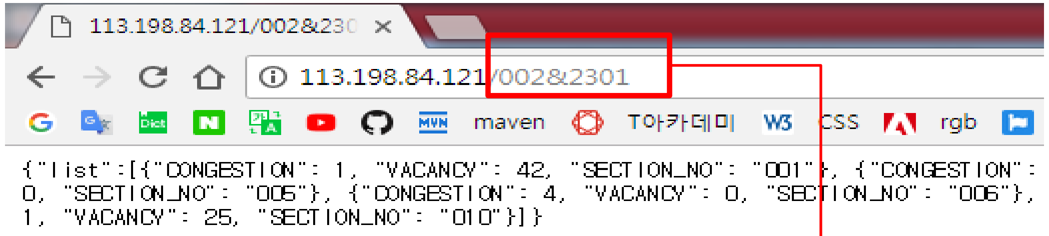
Image_nm

IMAGE_NM_OK

Image_data

SUCCESS

http_server

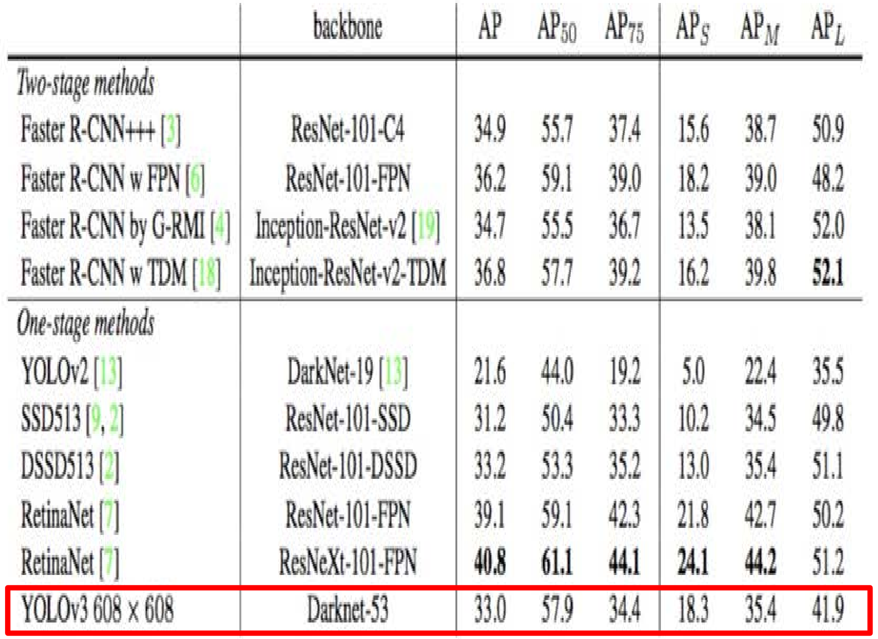


```
{"list":[{"CONGESTION": 1, "VACANCY": 42, "SECTION_NO": "001"}, {"CONGESTION":
0, "SECTION_NO": "005"}, {"CONGESTION": 4, "VACANCY": 0, "SECTION_NO": "006"},
1, "VACANCY": 25, "SECTION_NO": "010"}]}
```

호선&열차번호

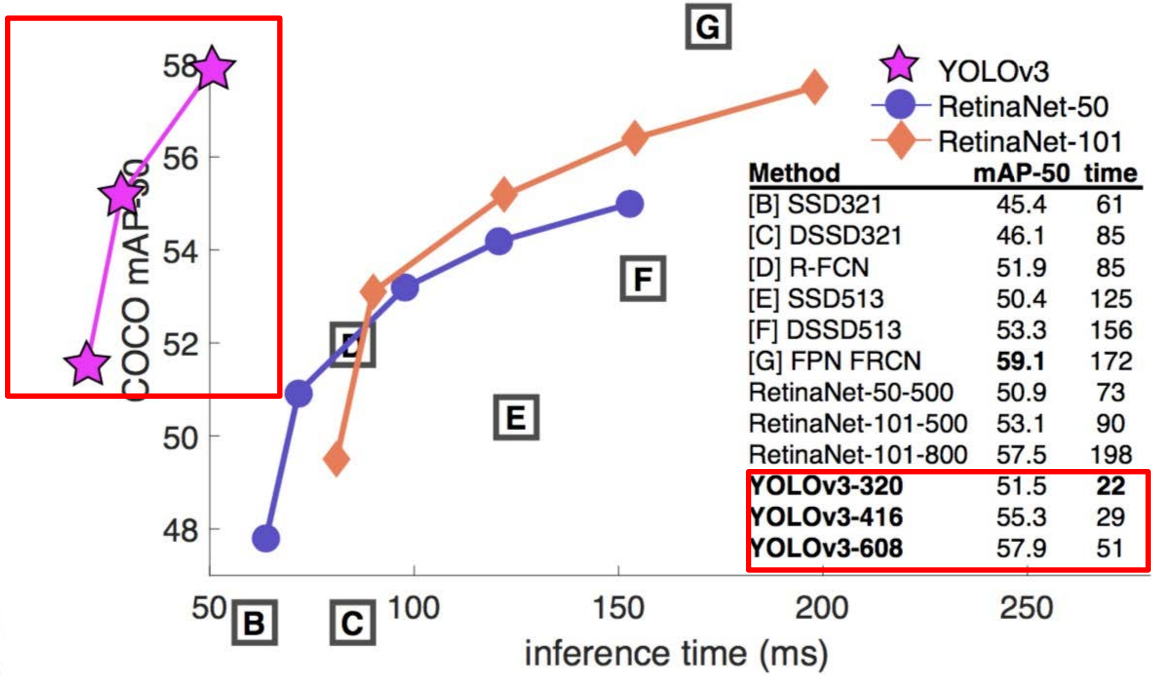
1. Image > deeplearning
2. LEVEL 분류
3. If) LEVEL == 1 or LEVEL == 2
      YOLO를 사용하여 빈자리 구함
   else) 자리 없음
4. DB에 호선/열차번호/열차 칸 번호에 따라서 UPDATE

Yolo



	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [3]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [6]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [4]	Inception-ResNet-v2 [19]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [18]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [13]	DarkNet-19 [13]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [9, 2]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [2]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [7]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet [7]	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2
YOLOv3 608 × 608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9

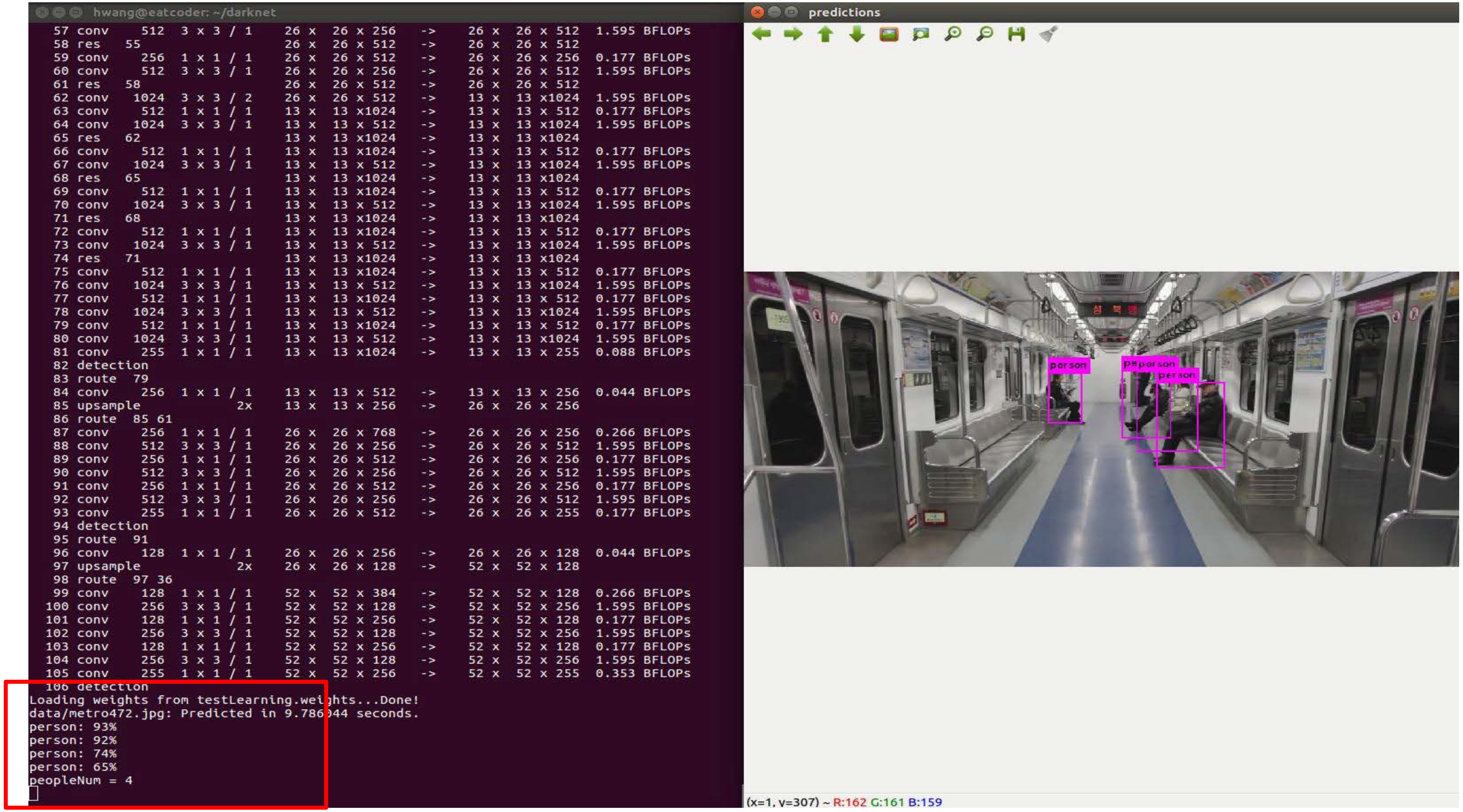
Table 3. I'm seriously just stealing all these tables from [7] they take soooo long to make from scratch. Ok, YOLOv3 is doing alright. Keep in mind that RetinaNet has like 3.8× longer to process an image. YOLOv3 is much better than SSD variants and comparable to state-of-the-art models on the AP_{50} metric.



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

Figure 3. Again adapted from the [7], this time displaying speed/accuracy tradeoff on the mAP at .5 IOU metric. You can tell YOLOv3 is good because it's very high and far to the left. Can you cite your own paper? Guess who's going to try, this guy → [14].

Custom Training 이후 정확도 상향



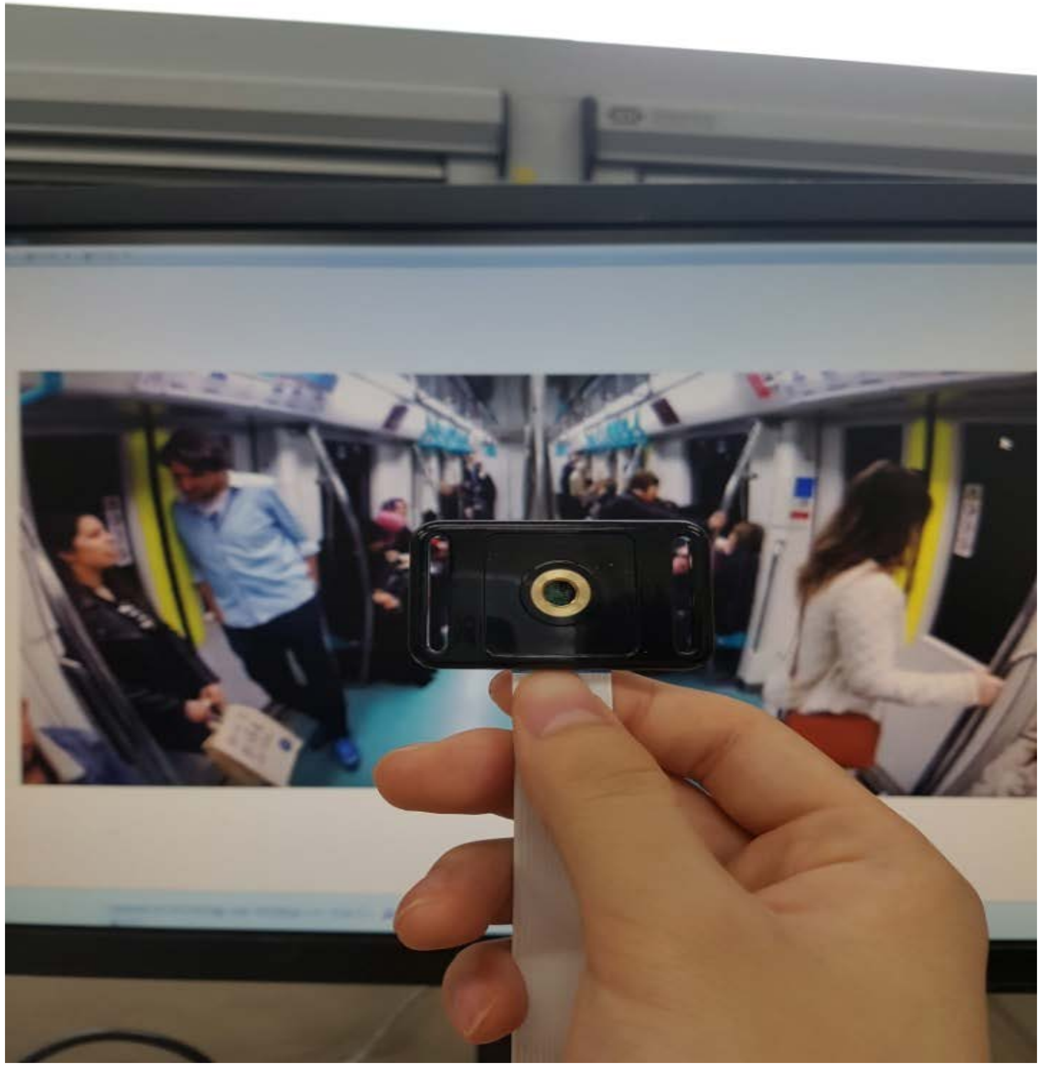
hwang@eatcoder: ~/darknet

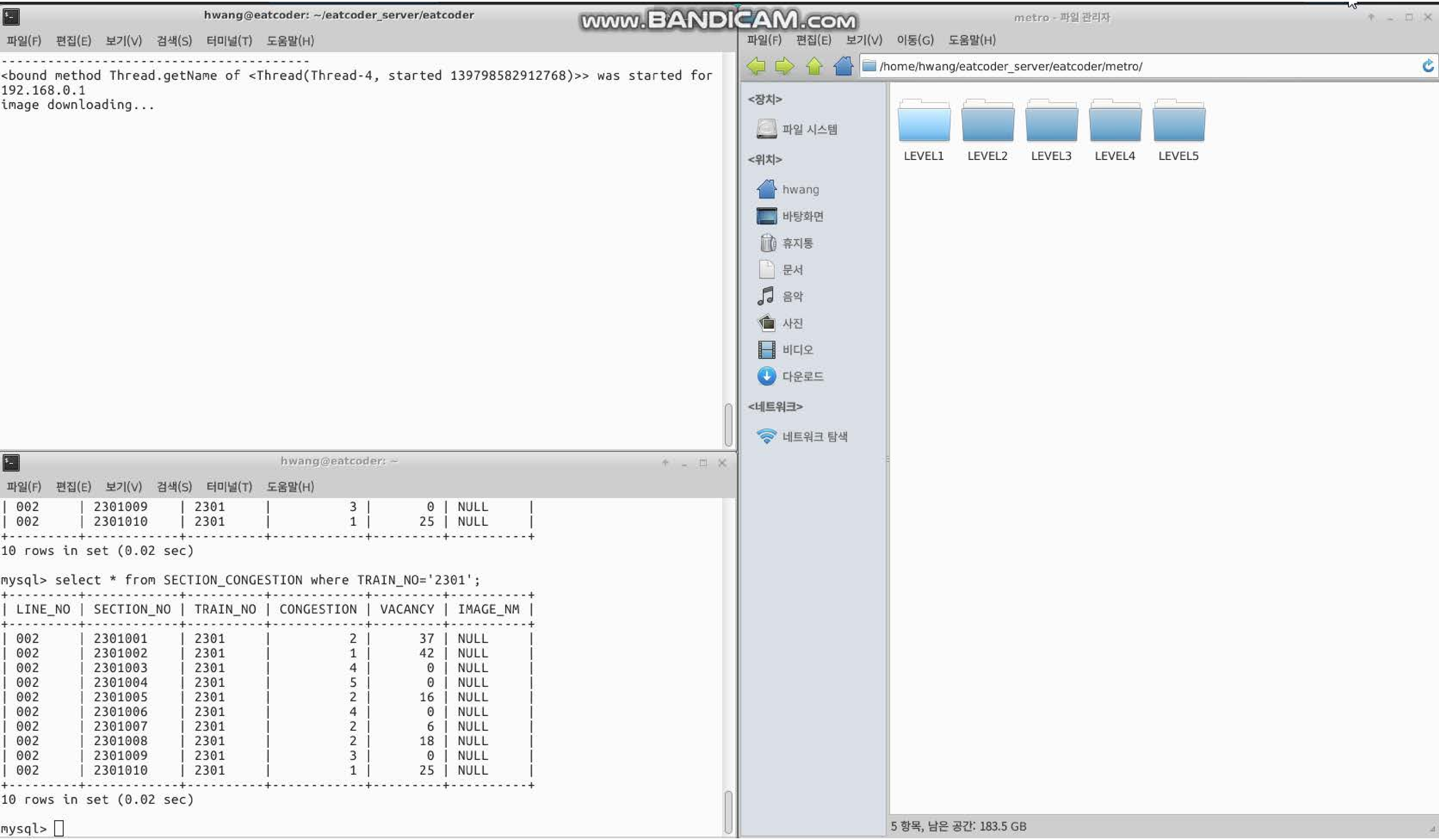
```
   57 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512  1.595 BFLOPs
   58 res    55                 26 x  26 x 512   ->    26 x  26 x 512
   59 conv    256  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x 256  0.177 BFLOPs
   60 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512  1.595 BFLOPs
   61 res    58                 26 x  26 x 512   ->    26 x  26 x 512
   62 conv   1024  3 x 3 / 2    26 x  26 x 512   ->    13 x  13 x1024  1.595 BFLOPs
   63 conv    512  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 512  0.177 BFLOPs
   64 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024  1.595 BFLOPs
   65 res    62                 13 x  13 x1024   ->    13 x  13 x1024
   66 conv    512  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 512  0.177 BFLOPs
   67 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024  1.595 BFLOPs
   68 res    65                 13 x  13 x1024   ->    13 x  13 x1024
   69 conv    512  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 512  0.177 BFLOPs
   70 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024  1.595 BFLOPs
   71 res    68                 13 x  13 x1024   ->    13 x  13 x1024
   72 conv    512  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 512  0.177 BFLOPs
   73 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024  1.595 BFLOPs
   74 res    71                 13 x  13 x1024   ->    13 x  13 x1024
   75 conv    512  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 512  0.177 BFLOPs
   76 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024  1.595 BFLOPs
   77 conv    512  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 512  0.177 BFLOPs
   78 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024  1.595 BFLOPs
   79 conv    512  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 512  0.177 BFLOPs
   80 conv   1024  3 x 3 / 1    13 x  13 x 512   ->    13 x  13 x1024  1.595 BFLOPs
   81 conv    255  1 x 1 / 1    13 x  13 x1024   ->    13 x  13 x 255  0.088 BFLOPs
   82 detection
   83 route  79
   84 conv    256  1 x 1 / 1    13 x  13 x 512   ->    13 x  13 x 256  0.044 BFLOPs
   85 upsample            2x    13 x  13 x 256   ->    26 x  26 x 256
   86 route  85 61
   87 conv    256  1 x 1 / 1    26 x  26 x 768   ->    26 x  26 x 256  0.266 BFLOPs
   88 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512  1.595 BFLOPs
   89 conv    256  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x 256  0.177 BFLOPs
   90 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512  1.595 BFLOPs
   91 conv    256  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x 256  0.177 BFLOPs
   92 conv    512  3 x 3 / 1    26 x  26 x 256   ->    26 x  26 x 512  1.595 BFLOPs
   93 conv    255  1 x 1 / 1    26 x  26 x 512   ->    26 x  26 x 255  0.177 BFLOPs
   94 detection
   95 route  91
   96 conv    128  1 x 1 / 1    26 x  26 x 256   ->    26 x  26 x 128  0.044 BFLOPs
   97 upsample            2x    26 x  26 x 128   ->    52 x  52 x 128
   98 route  97 36
   99 conv    128  1 x 1 / 1    52 x  52 x 384   ->    52 x  52 x 128  0.266 BFLOPs
  100 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256  1.595 BFLOPs
  101 conv    128  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 128  0.177 BFLOPs
  102 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256  1.595 BFLOPs
  103 conv    128  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 128  0.177 BFLOPs
  104 conv    256  3 x 3 / 1    52 x  52 x 128   ->    52 x  52 x 256  1.595 BFLOPs
  105 conv    255  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 255  0.353 BFLOPs
  106 detection
Loading weights from testLearning.weights...Done!
data/metro472.jpg: Predicted in 9.786044 seconds.
person: 93%
person: 92%
person: 74%
person: 65%
peopleNum = 4
```

predictions

(x=1, y=307) ~ R:162 G:161 B:159

Yolo

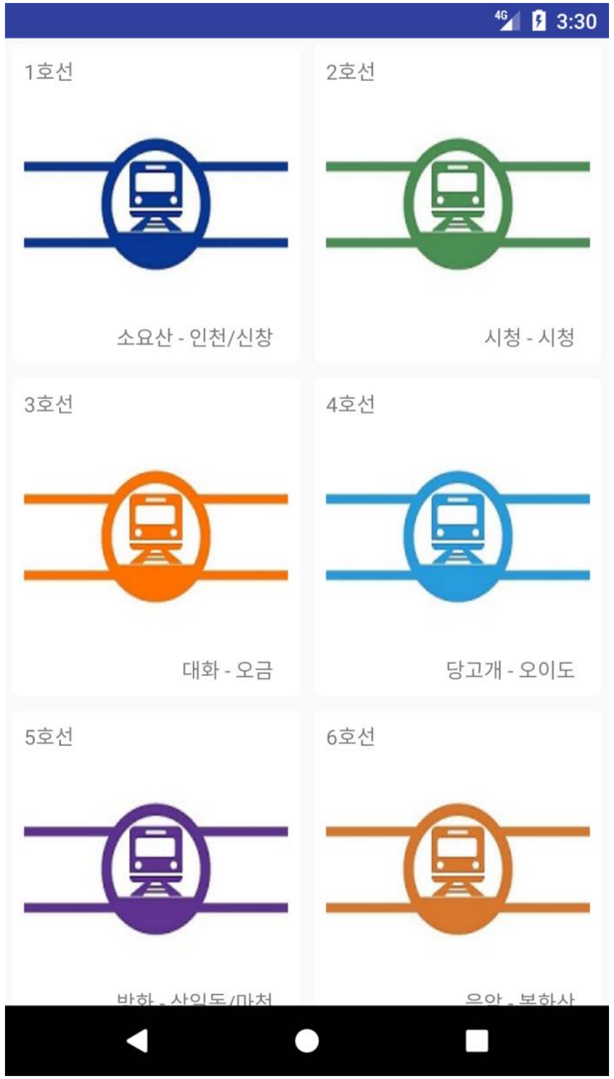


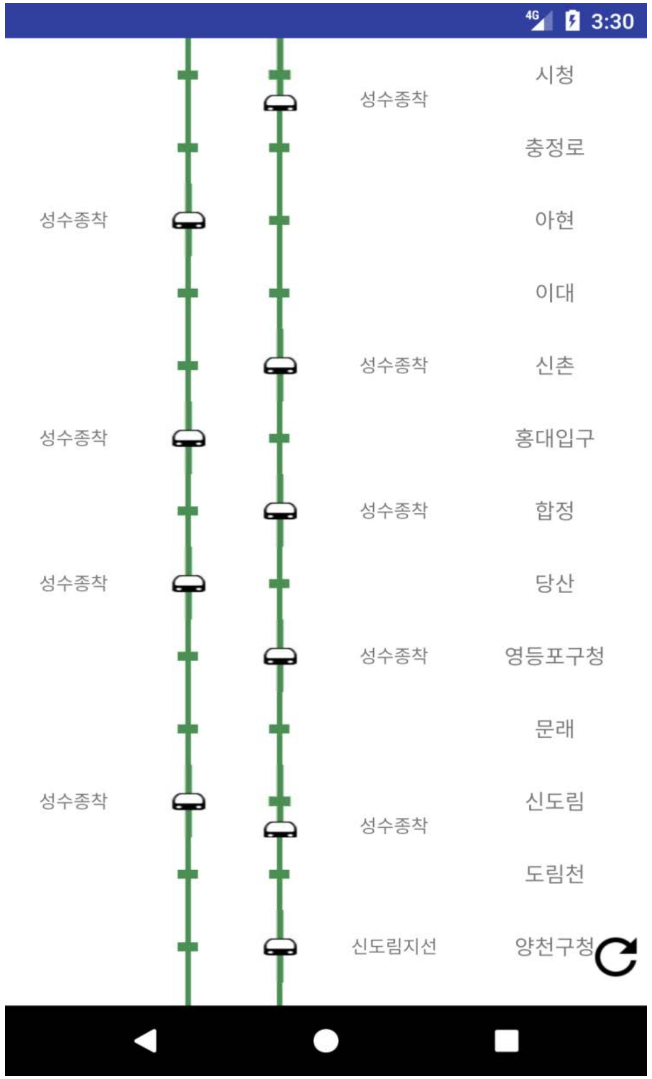

www.BANDICAM.com
hwang@eatcoder: ~/eatcoder_server/eatcoder
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
hwang@eatcoder: ~
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
metro - 파일 관리자
파일(F) 편집(E) 보기(V) 이동(G) 도움말(H)
/home/hwang/eatcoder_server/eatcoder/metro/
<장치>
파일 시스템
<위치>
hwang
바탕화면
휴지통
문서
음악
사진
비디오
다운로드
<네트워크>
네트워크 탐색
LEVEL1
LEVEL2
LEVEL3
LEVEL4
LEVEL5
5 항목, 남은 공간: 183.5 GB

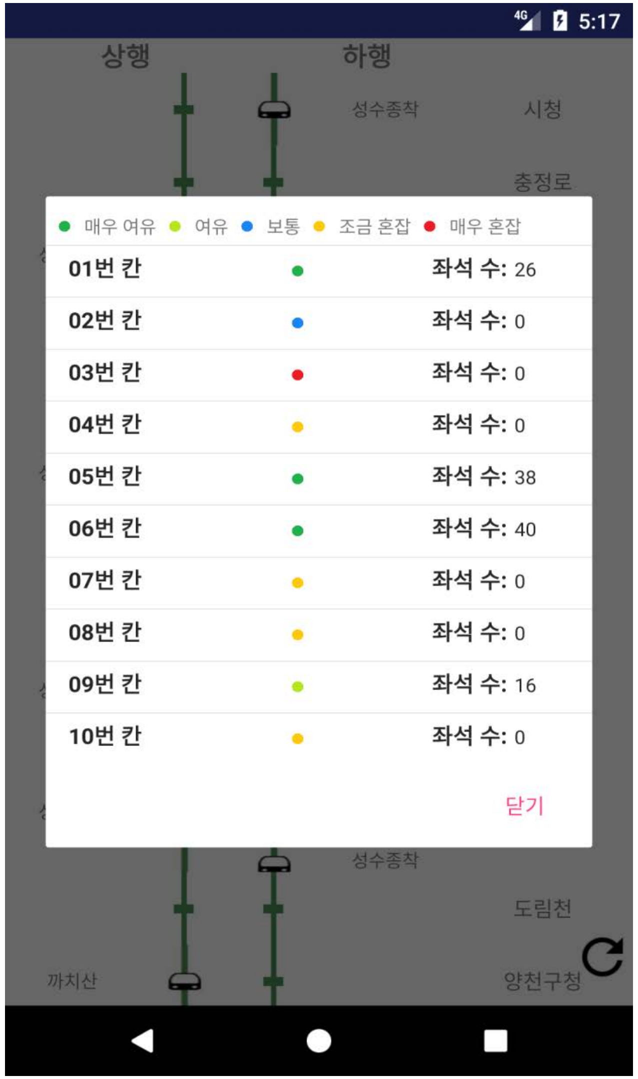
```
----------------------------------------
<bound method Thread.getName of <Thread(Thread-4, started 139798582912768)>> was started for
192.168.0.1
image downloading...
```

```
| 002      | 2301009    | 2301     |          3 |       0 | NULL     |
| 002      | 2301010    | 2301     |          1 |      25 | NULL     |
+----------+------------+----------+------------+---------+----------+
10 rows in set (0.02 sec)

mysql> select * from SECTION_CONGESTION where TRAIN_NO='2301';
+----------+------------+----------+------------+---------+----------+
| LINE_NO  | SECTION_NO | TRAIN_NO | CONGESTION | VACANCY | IMAGE_NM |
+----------+------------+----------+------------+---------+----------+
| 002      | 2301001    | 2301     |          2 |      37 | NULL     |
| 002      | 2301002    | 2301     |          1 |      42 | NULL     |
| 002      | 2301003    | 2301     |          4 |       0 | NULL     |
| 002      | 2301004    | 2301     |          5 |       0 | NULL     |
| 002      | 2301005    | 2301     |          2 |      16 | NULL     |
| 002      | 2301006    | 2301     |          4 |       0 | NULL     |
| 002      | 2301007    | 2301     |          2 |       6 | NULL     |
| 002      | 2301008    | 2301     |          2 |      18 | NULL     |
| 002      | 2301009    | 2301     |          3 |       0 | NULL     |
| 002      | 2301010    | 2301     |          1 |      25 | NULL     |
+----------+------------+----------+------------+---------+----------+
10 rows in set (0.02 sec)

mysql> 
```

프로젝트 세부사항(Android)





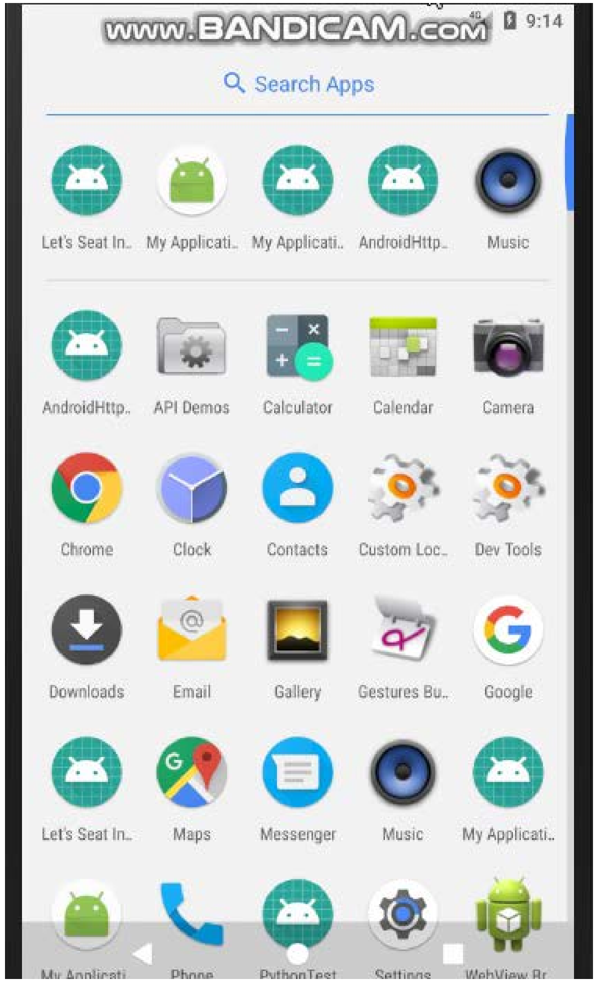


● 매우 여유 ● 여유 ● 보통 ● 조금 혼잡 ● 매우 혼잡

01번 칸	●	좌석 수: 26
02번 칸	●	좌석 수: 0
03번 칸	●	좌석 수: 0
04번 칸	●	좌석 수: 0
05번 칸	●	좌석 수: 38
06번 칸	●	좌석 수: 40
07번 칸	●	좌석 수: 0
08번 칸	●	좌석 수: 0
09번 칸	●	좌석 수: 16
10번 칸	●	좌석 수: 0

닫기

프로젝트 세부사항


www.BANDICAM.com
9:14
Search Apps
Let's Seat In..
My Applicati..
My Applicati..
AndroidHttp..
Music
AndroidHttp..
API Demos
Calculator
Calendar
Camera
Chrome
Clock
Contacts
Custom Loc..
Dev Tools
Downloads
Email
Gallery
Gestures Bu..
Google
Let's Seat In..
Maps
Messenger
Music
My Applicati..
My Applicati..
Phone
PythonTest
Settings
WebView Br..

Q&A