Licensed Plate Detection and Recognition

With Team. NoAnswer(황지수, 석예찬, 윤형진)

목차

- 1. 모델 선정
- 2. 모델선정 이유와 전략
  - -YOLO V3
  - -Canny Edge
  - -CRNN
- 3. 데이터 준비 및 개발과정
- 4. 성능평가
- 5. 향후 개선방향
- 6. QnA

1. 모델 선정

- Problem : 두개의 데이터셋에 대하여 번호판을 탐지하고 인식

- Problem1 : Parking Detection(YOLO V3)
- Problem2 : CCTV Detection(Canny Edge)
- Problem3 : Recognition(CRNN)

2. 모델선정 이유와 전략

- 전략 :

1. 데이터가 적기 때문에, 학습해서 문제를 해결하려 한다면 Accuracy는 낮을 것이므로, PT가 빠른 모델을 고려하자.

2. IoU가 70%만 넘으면 된다는 사실을 이용하자.

Definition of the Score

$$Score = Score_{park} + Score_{cctv} + 0.1 \times (100 - PT)$$

$$PT = msec./image(average)$$

$$Score_i = Accuracy_{det} + Accuracy_{rec} \quad (i = park\ or\ cctv)$$

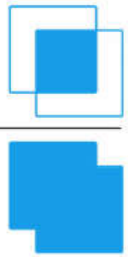
$$Accuracy_{det} = \frac{1}{n}\sum_{i=0}^{n-1}\frac{\#TP_{det} - \#FP_{det}}{\#GT} \times 100\%$$

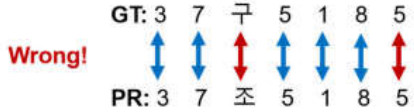
$$Accuracy_{rec} = \frac{1}{n}\sum_{i=0}^{n-1}\frac{\#TP_{rec}}{\#GT} \times 100\%$$

- PT: average processing time of the model (unit: msec.)
- $\#TP_{det}$: number of true positive for detection
- $\#FP_{det}$: number o false positive for detection
- $\#TP_{rec}$: number of true positive for recognition
- #GT: number of ground-truth

- TP_{det}: IoU >= θ, θ=0.7
- FP_{det}: IoU < θ

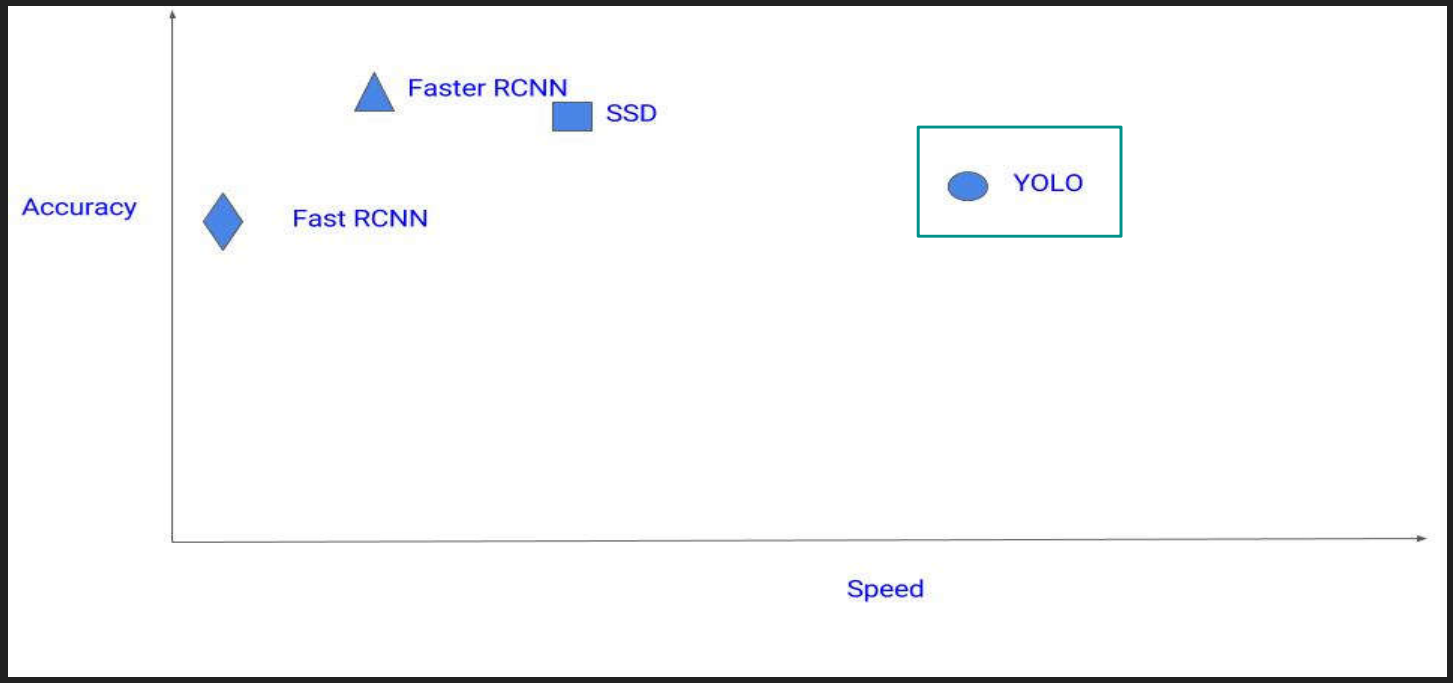
IoU = Area of Overlap / Area of Union





2. 모델선정 이유와 전략(YOLO)

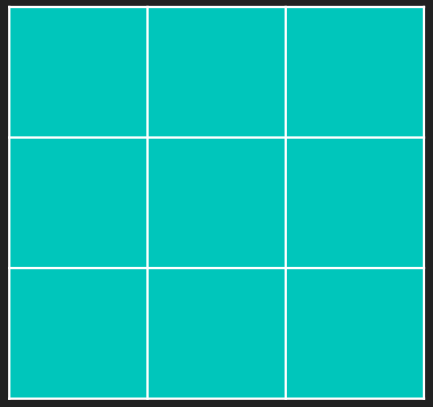
- 모델별 Accuracy 와 speed



2. 모델선정 이유와 전략(YOLO)

- Model Concept

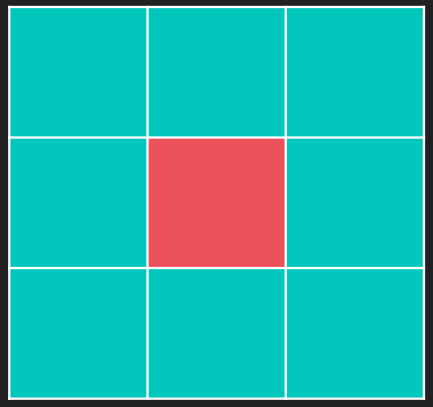
  1. Input Image를 s*s grid로 나누어 각 셀에서 확인한다.



2. 모델선정 이유와 전략(YOLO)

- Model Concept

2. Conv layer를 이용하여 각 셀 별로 B개(1, 2, 3, 4)의 바운딩박스에 대한 정보와, confidence score(물체가 존재할 확률), C개의 class probability(어떤 클래스의 물체일지, 4)를 구하여 합친다.



1. 이 셀의 물체를 포함하는 바운딩 박스의 Center x, y
2. 이 셀의 물체를 포함하는 바운딩 박스의 w, h
3. 이 바운딩 박스에 물체가 존재할 확률
4. 이 셀에 있는 오브젝트의 class probability

2. 모델선정 이유와 전략(YOLO)

- Model Architecture

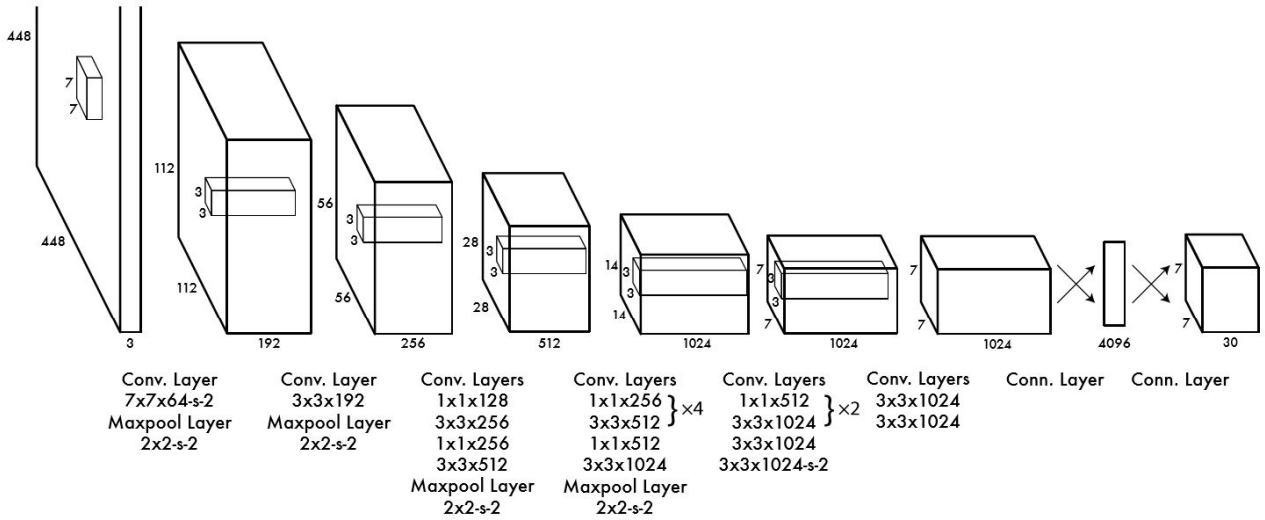
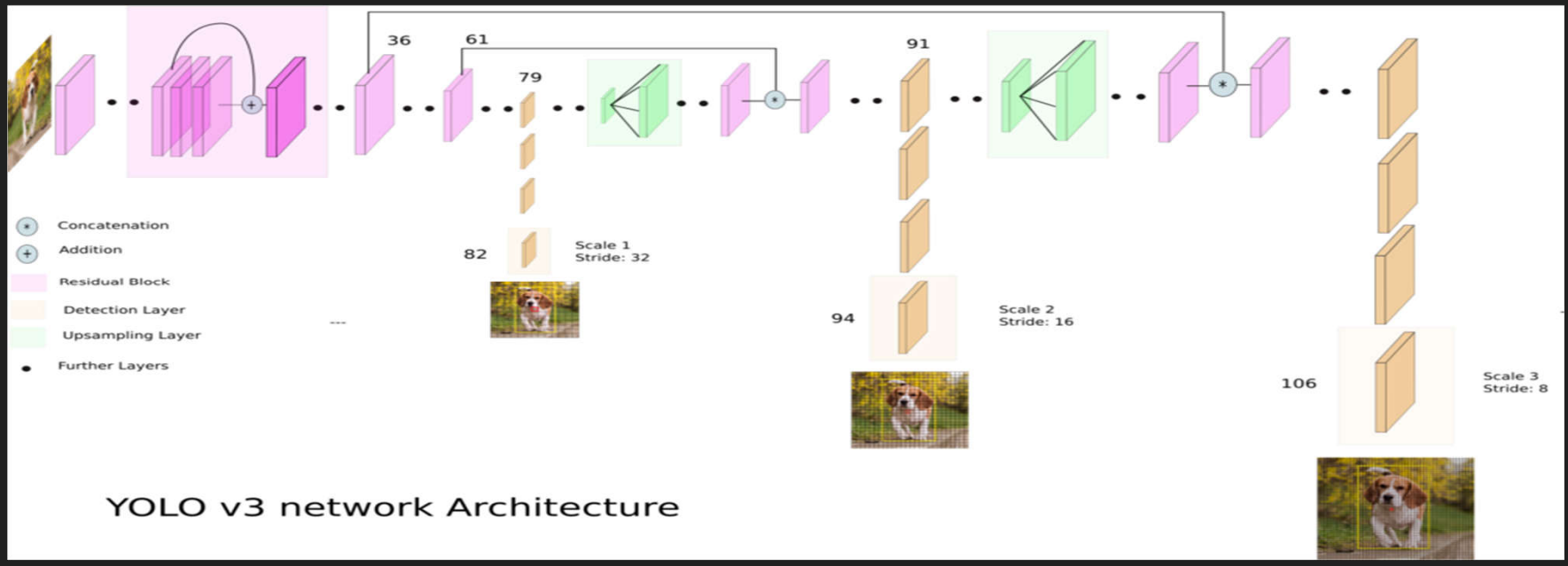


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

2. 모델선정 이유와 전략(YOLO V3)

- What's new in YOLO V3(Detection at three scales)

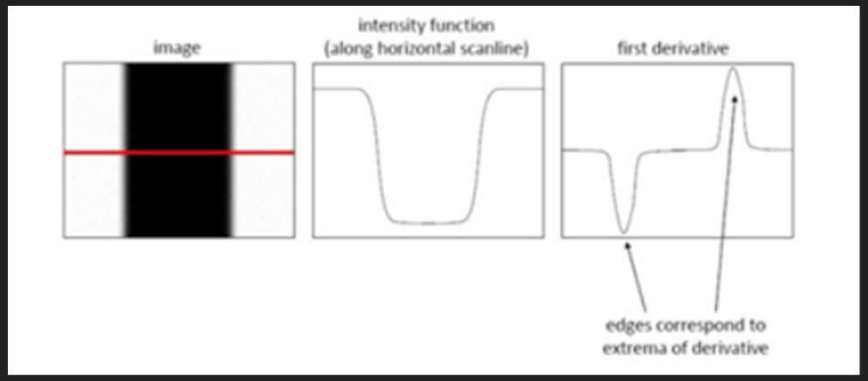


2. 모델선정 이유와 전략(YOLO V3)

단점 :

- 1. s*s그리드로 나누어 각 셀에서 object를 확인하는데, 물체가 작다면 object가 없다고 인식할 수 있어 번호판이 작은 CCTV데이터에선 학습이 안됨.
- 2. 그리드의 사이즈를 작게 해도, 여러 개의 번호판에서 가까운 번호판을 추출하기 어려움.

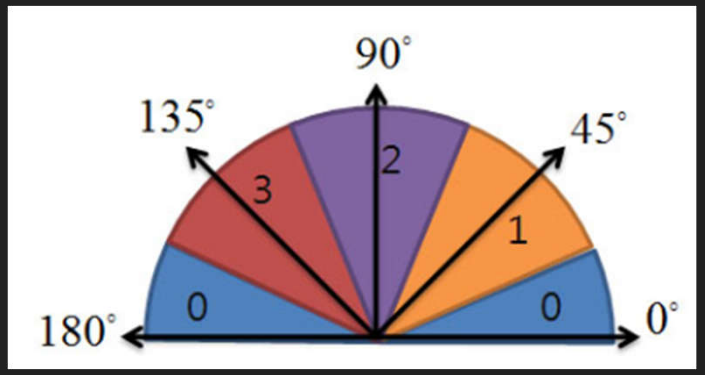
2. 모델선정 이유와 전략(Canny Edge Detection)



- 영상의 pixel변화가 급격한곳이 edge 이다.
- 영상의 noise때문에 edge가 아닌데도 기울기가 급격히 변하여 edge라고 할수있기때문에 가우시안필터를 씌운 뒤에 sobel 필터로 dx ,dy를 구한다.

Canny Edge Detection - non maxima Supr eesion

- 가우시안 필터를 썻음에도 불구하고 dx,dy값이 edge가 아닌데도 꽤 크게나온다.
  따라서 그 어느정도 크기는 제외시켜야 하는데 local maxima를 찾고 그 지점만 제외하고 0으로 만들면 문제를 해결할수있다.



간단 알고리즘 - dx,dy로부터얻은 벡터의 방향의 전후 픽셀과 중심픽셀 3개의 벡터 크기를 비교후에 중심이 maxima가 아니라면 0으로 쓴다.

Canny Edge Detection -double threshold

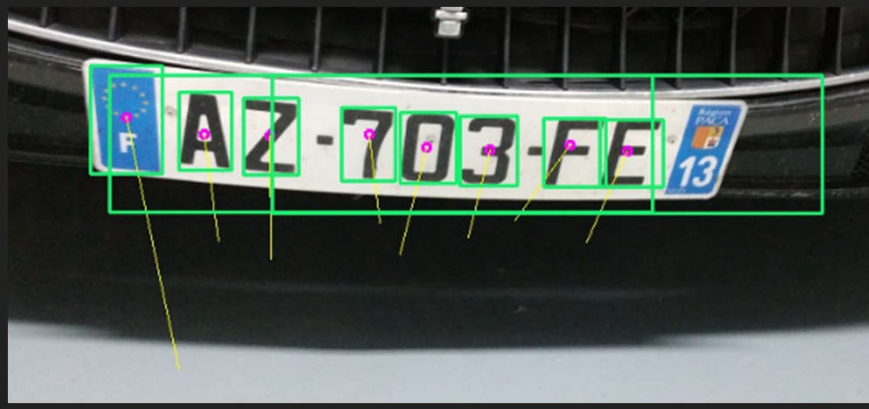
- 이제 threshold를 설정하여 edge인부분 edge가 아닌부분을 구분해야하는데 그냥 하나의 threshold만 그어서 판별하기에는 문제가있다.

- 조금이나마해결.. : high threshold 이상이면 무조건 edge low와 high 사이에 있고 edge로 분류된 픽셀과 인접하면 edge

번호판찾기 : 외곽선따기

```
cnts,contours,hierarchy  = cv2.findContours(canny, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
```

- 외곽선따기 알고리즘은 흰픽셀들을 이어주어 그를 물체로 본다는 것이다.
- APP...SIMPLE : 수평수직 대각성분의 끝점만 저장한다.
- 번호판에서 SIMPLE인자로 외곽선을 따면 문자당 어떤 외곽선이 추출된다.
- 찾은 외곽선에서 일정한 특징를 찾으면 그게 번호판이다.
  특징 : 기울기가 비슷한게 나열되어있다. 간격이좁다



2. 모델선정 이유와 전략(CRNN)

- 번호판은 모든 글자를 전부 제대로 맞춰야 한다.
- 글자 하나하나의 위치를 찾아 잘라내고(Risk 1)
- 해당 부분을 CNN모델을 이용하여 예측해서 찾는 것은 위험성이 크다.(Risk 2)

2. 모델선정 이유와 전략(CRNN)

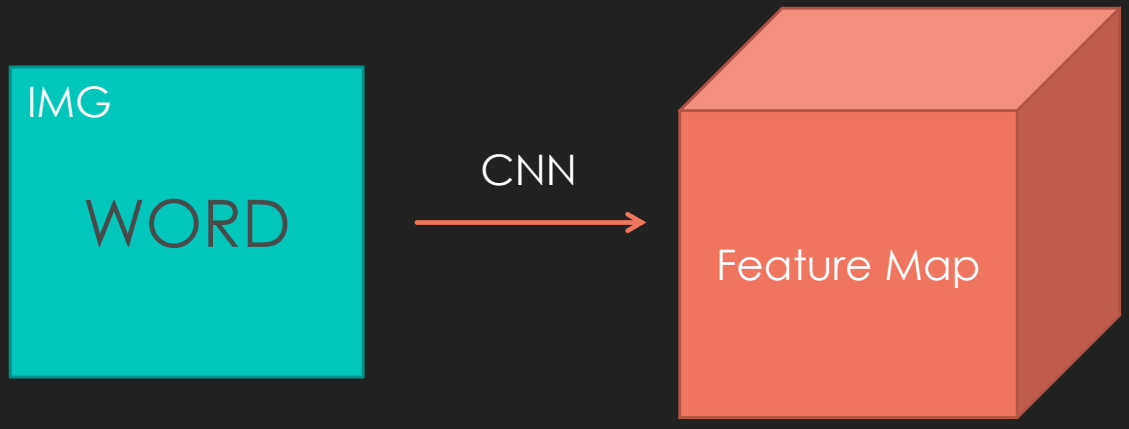
- Model Idea

  -이미지를 Sequential Data로 보아, 각 부분을 RNN을 이용하여 더 정확히 데이터를 처리한다.

2. 모델선정 이유와 전략(CRNN)

- Model Concept

  1. Convolutional Layer을 이용하여, 이미지를 Feature Map으로 바꾼다

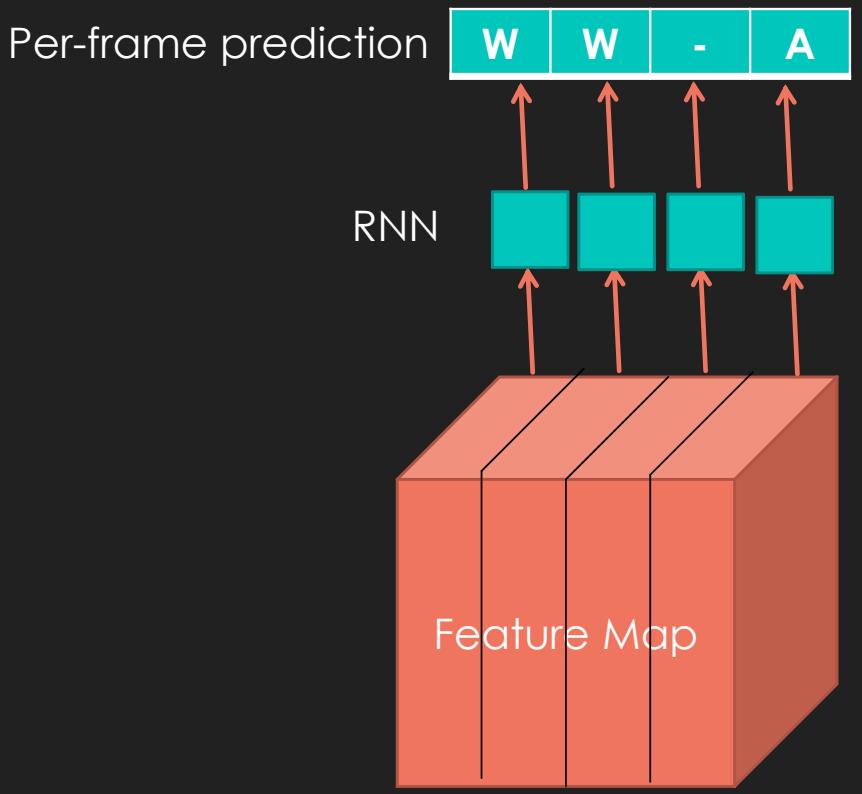


2. 모델선정 이유와 전략(CRNN)

- Model Concept

  2. Recurrent Layer frame별로 예측

이때, 각각의 부분들의 크기를 같게 맞춰주고 RNN에 넣는다.



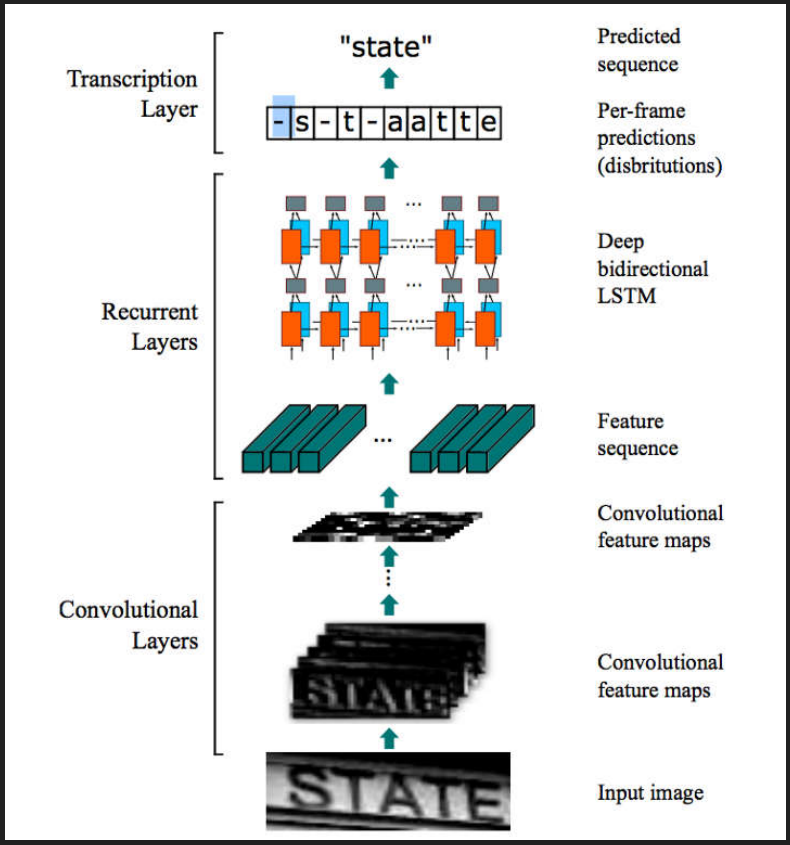
2. 모델선정 이유와 전략(CRNN)

- Model Concept

  3. Transcription Layer

2. 모델선정 이유와 전략(CRNN)

- Model Architecture



3. 데이터 준비 및 개발과정

CRNN을 위한 데이터 : 한줄 번호판

- Sequence로 해석되어야한다.
- 라벨링은 한글자체를 하나의 라벨로 (번호판은 규격에따라 정해진 한글만 있어서 괜찮음)
  초성,중성나누기는 ? : 밑으로 받침이 들어가면 문제가 생김
  바 ([ㅂ][ㅏ]) 버([ㅂ][ㅓ]) 보([?] [?])

CRNN을 위한 데이터 : 두줄 번호판은 제외하자

- Sequence로 해석되어야한다.

- 두줄인 번호판을 트레이닝 하려면... :
  1. 그냥 이미지를 왼쪽에서 오른쪽으로 스캔 : 같은 시퀀스 스캔 상태에서 (서3)이 같은 스캔범위내에서 classification되기때문에 set으로 묶어 서3을 하나의 class로 구분하여야한다.

  2. 빨간색위로 스캔한번 아래로 스캔한번 트레이닝 되어야 한다.

  따라서 한줄인 번호판만 하기로했고 그중에서도 일단 P1만 트레이닝 해보았다.

지도학습의 문제점 : Trainning data set 검증

- 정확한 라벨의 이미지로 Trainning 시켜야한다.
- 깨끗한 이미지만 쓰면 안된다. blurring되거나 변형된 데이터도 라벨을 확인할수 있다면 Trainning 해야한다. - 오버피팅방지
- 이정도는 되야...

- 형체를 알아볼수없는, 라벨을 확인할수없는 너무 작은 바운딩 박스, 글자가 아예가려진 데이터는 안된다. 이런데이터는 라벨이 없는 데이터인데 학습시키면 안된다는 것

지도학습의 문제점 : 잘못된 데이터에 취약

```
</object>
<object>
    <name>P1_23구6044</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
  - <bndbox>
        <xmin>720</xmin>
        <ymin>211</ymin>
        <xmax>728</xmax>
        <ymax>211</ymax>
```

```
error name
P!_10리0284_27.png P!_10리0284
```

- ! 로 라벨링한 분들...
- '투'라는 번호판은 없는데 잘못 읽으신 분들...
- 바운딩 박스를 잘못그리신 분들...

```
error name
P1_28투0145_90.png P1_28투0145
```

```
<path>C:\Users\daegeun\Desktop\08_12121609\24_24917_0290550.png</path>
<source>
    <database>Unknown</database>
</source>
<size>
    <width>1920</width>
    <height>1080</height>
    <depth>3</depth>
</size>
<segmented>0</segmented>
<object>
    <name>P1_51라9883</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
  - <bndbox>
        <xmin>710</xmin>
        <ymin>211</ymin>
        <xmax>725</xmax>
        <ymax>211</ymax>
```

데이터 모으기...

- P1만

```
if plateKind !="P1":
```

- 번호판 규격 데이터만

```
if de_En_codeSystem.isContainHangulKey(hangul)==False:#번호판 mapping 없으면 데이터제외
```

- 번호판 이상하게 쓰신분들...데이터제외

```
#박스의 x,y좌표가 같은사람들....
if boxes[sub_idx][1]==boxes[sub_idx][3] or boxes[sub_idx][0]==boxes[sub_idx][2]:
```

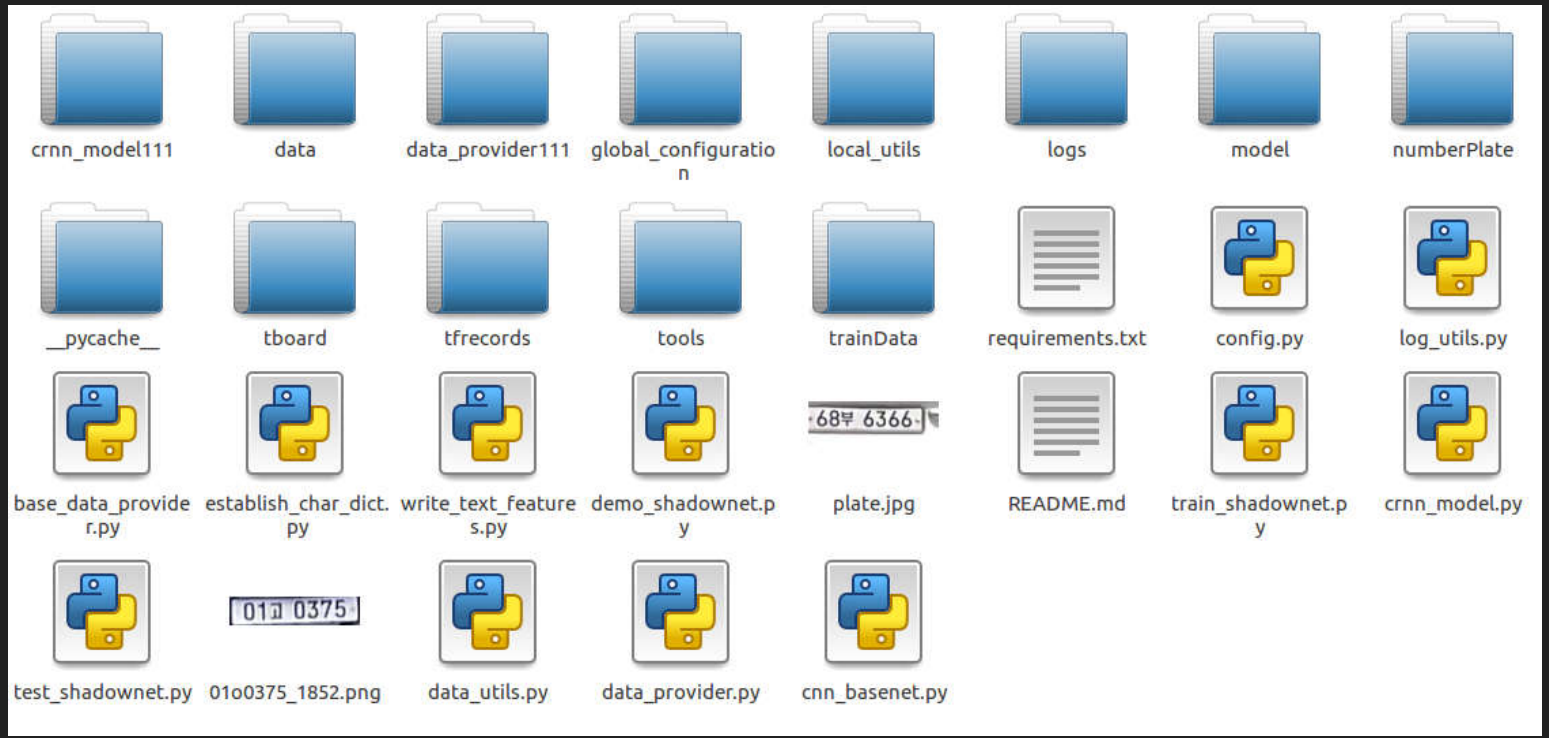
- 좀 큰(식별되는) 번호판만

```
boxSqure=(boxes[sub_idx][3]-boxes[sub_idx][1]) * (boxes[sub_idx][2]-boxes[sub_idx][0])
if boxSqure<2300:
```

- 숫자아닌 이상한거 쓴분들...데이터제외

```
#숫자아닌걸로 써논사람들...
if char4<'0' or '9'<char4:
```

개발과정_CRNN



- CRNN 폴더의 첫 모습

sample.txt

58o8884_494.png

29i7287_336.png

15w9635_433.png

53w7774_487.png

03o0825_340.png

60j4815_516.png

92z0341_460.png

53j2500_606.png

57g5981_521.png

63A6141_470.png

23r2012_495.png

61d5278_476.png

52m2707_395.png

12f0515_510.png

04n0467_431.png

63c2297_473.png

14m2346_505.png

48z2795_466.png

42F8807_411.png

45A0806_670.png

49x9922_246.png

41k8346_467.png

33B6226_279.png

60d6386_558.png

87l7576_330.png

95i1973_546.png

43l6070_444.png

15n5186_320.png

82D7146_235.png

79D3156_504.png

50g4452_38.png

11b1675_439.png

89h3409_217.png

84l7626_251.png

10K8259_436.png

53p4403_410.png

22B7853_459.png

56r6846_316.png

92r4080_920.png

19u9106_455.png

31l6722_536.png

11J9821_842.png

45r6978_449.png

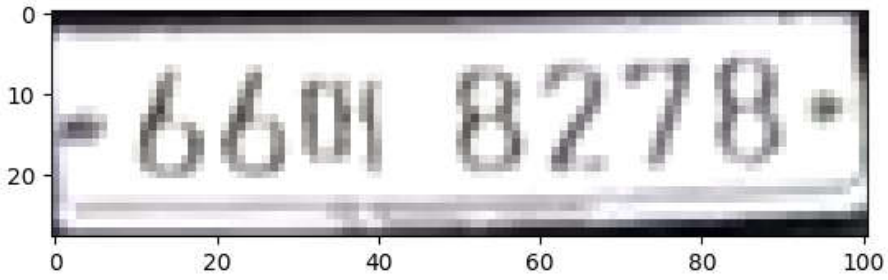
77z9735_858.png

개발과정_CRNN

```
(AI_project) yun@itlab-server98:~/AI_project_speech/CRNN/CRNN_Tensorflow-master$ python ./write_text_features --dataset_dir ./numberPlate --save_dir .
/tfrecords
```

- Tensorflow로 실행시키기 전, data 전처리

CRNN Model Demo

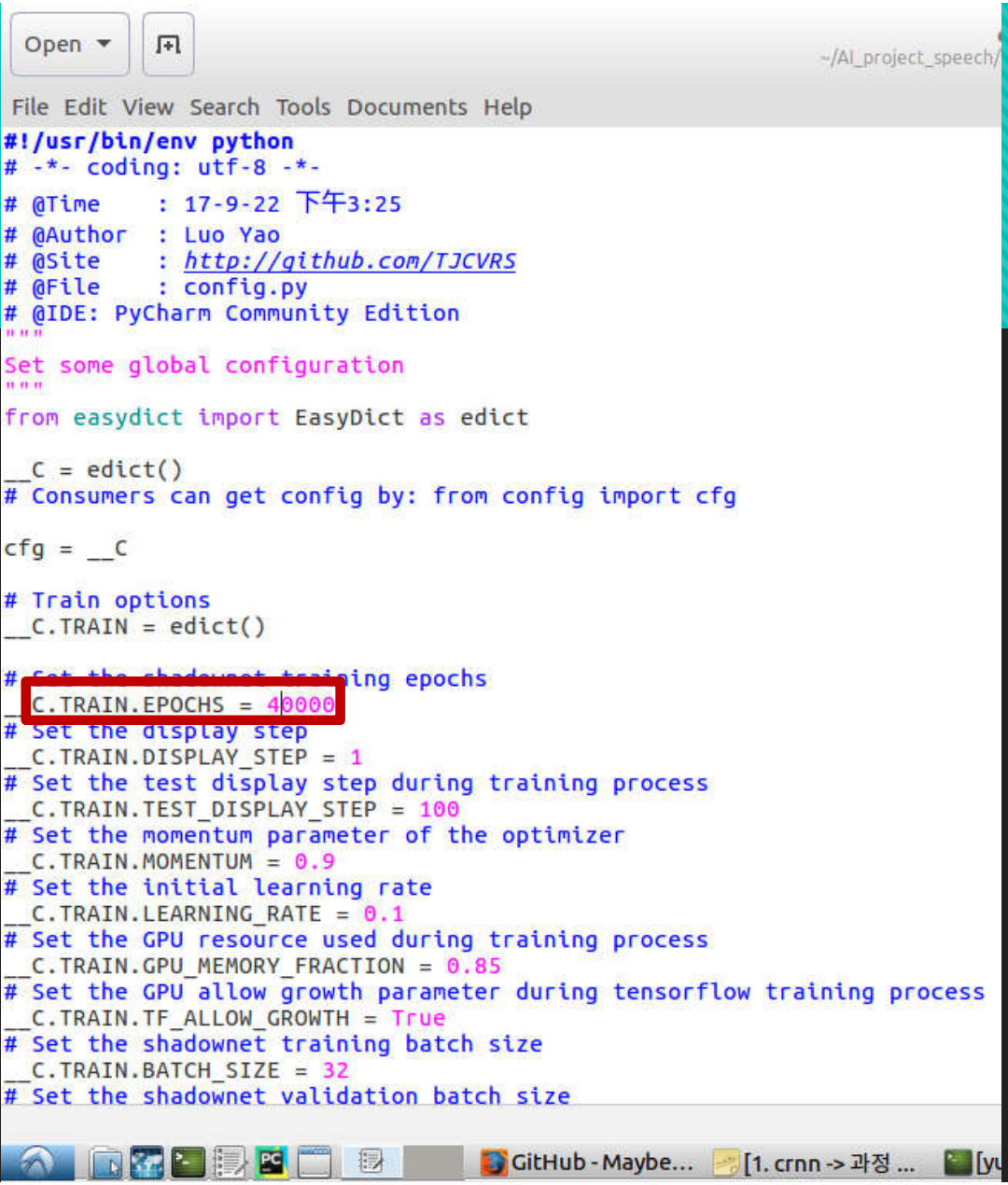


zoom rect

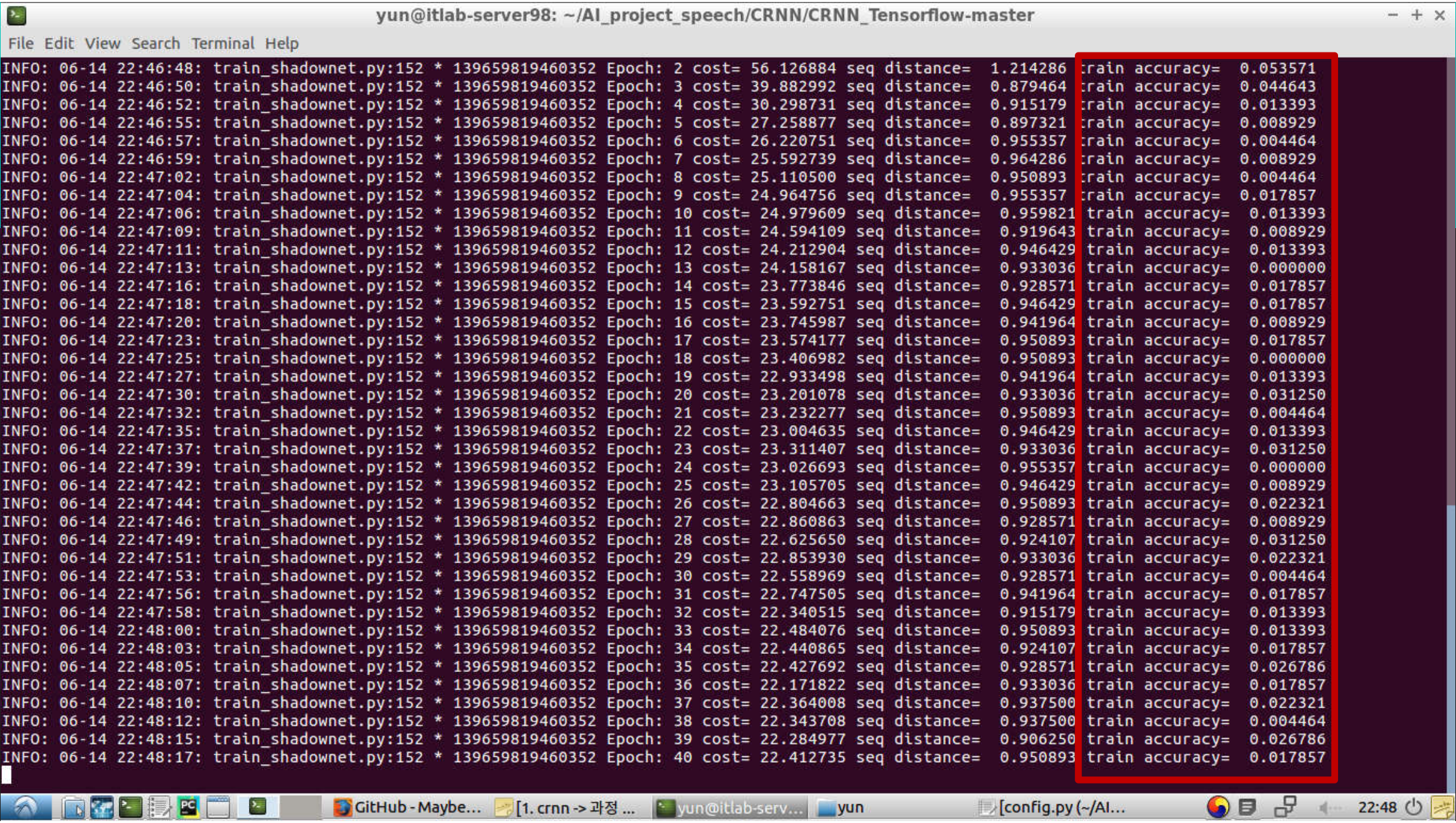
speech/CRNN/CRNN_Tensorflow-master

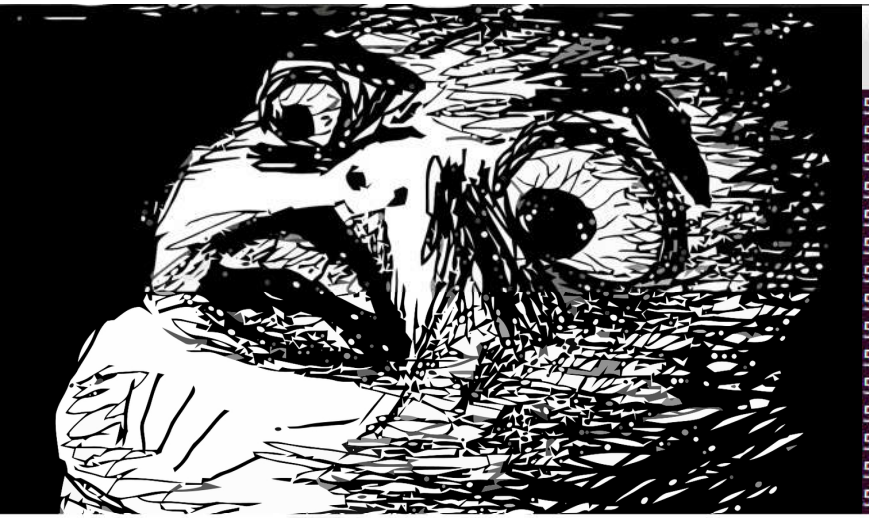
```
8-06-13-05-35-11.ckpt-4659
meters from model/shadownet/shadownet_2018-06-13-05-35-11.ckpt-4659
age 45A0806_670.png label 45a0806

/45A0806_670.png --weights_path model/shadownet/shadownet_2018-06-13-05-35-11.c

w-master$ python ./demo_shadownet.py --image_path ./test_img.png --weights_path

d.cc:45] The TensorFlow library wasn't compiled to use SSE4.1 instructions, but
s.
d.cc:45] The TensorFlow library wasn't compiled to use SSE4.2 instructions, but
s.
d.cc:45] The TensorFlow library wasn't compiled to use AVX instructions, but th

d.cc:45] The TensorFlow library wasn't compiled to use AVX2 instructions, but t

d.cc:45] The TensorFlow library wasn't compiled to use FMA instructions, but th

8-06-13-05-35-11.ckpt-4659
meters from model/shadownet/shadownet_2018-06-13-05-35-11.ckpt-4659
age test_img.png label 100364

/test_img.png --weights_path model/shadownet/shadownet_2018-06-13-05-35-11.ckpt

w-master$ python ./demo_shadownet.py --image_path ./test2_img.png --weights_pat

d.cc:45] The TensorFlow library wasn't compiled to use SSE4.1 instructions, but
s.
d.cc:45] The TensorFlow library wasn't compiled to use SSE4.2 instructions, but
 these are available on your machine and could speed up CPU computations.
2018-06-15 00:38:15.095797: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use AVX instructions, but th
ese are available on your machine and could speed up CPU computations.
2018-06-15 00:38:15.095801: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use AVX2 instructions, but t
hese are available on your machine and could speed up CPU computations.
2018-06-15 00:38:15.095807: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use FMA instructions, but th
ese are available on your machine and could speed up CPU computations.
INFO:tensorflow:Restoring parameters from model/shadownet/shadownet_2018-06-13-05-35-11.ckpt-4659
INFO: 06-15 00:38:15: tf_logging.py:82 * 140473768793856 Restoring parameters from model/shadownet/shadownet_2018-06-13-05-35-11.ckpt-4659
INFO: 06-15 00:38:15: demo_shadownet.py:84 * 140473768793856 Predict image test2_img.png label 002770
```

00:39



- ‘Config’ 파일을 통해 EPOCHS 값 수정


yun@itlab-server98: ~/AI_project_speech/CRNN/CRNN_Tensorflow-master
File Edit View Search Terminal Help
INFO: 06-14 22:46:48: train_shadownet.py:152 * 139659819460352 Epoch: 2 cost= 56.126884 seq distance= 1.214286 train accuracy= 0.053571
INFO: 06-14 22:46:50: train_shadownet.py:152 * 139659819460352 Epoch: 3 cost= 39.882992 seq distance= 0.879464 train accuracy= 0.044643
INFO: 06-14 22:46:52: train_shadownet.py:152 * 139659819460352 Epoch: 4 cost= 30.298731 seq distance= 0.915179 train accuracy= 0.013393
INFO: 06-14 22:46:55: train_shadownet.py:152 * 139659819460352 Epoch: 5 cost= 27.258877 seq distance= 0.897321 train accuracy= 0.008929
INFO: 06-14 22:46:57: train_shadownet.py:152 * 139659819460352 Epoch: 6 cost= 26.220751 seq distance= 0.955357 train accuracy= 0.004464
INFO: 06-14 22:46:59: train_shadownet.py:152 * 139659819460352 Epoch: 7 cost= 25.592739 seq distance= 0.964286 train accuracy= 0.008929
INFO: 06-14 22:47:02: train_shadownet.py:152 * 139659819460352 Epoch: 8 cost= 25.110500 seq distance= 0.950893 train accuracy= 0.004464
INFO: 06-14 22:47:04: train_shadownet.py:152 * 139659819460352 Epoch: 9 cost= 24.964756 seq distance= 0.955357 train accuracy= 0.017857
INFO: 06-14 22:47:06: train_shadownet.py:152 * 139659819460352 Epoch: 10 cost= 24.979609 seq distance= 0.959821 train accuracy= 0.013393
INFO: 06-14 22:47:09: train_shadownet.py:152 * 139659819460352 Epoch: 11 cost= 24.594109 seq distance= 0.919643 train accuracy= 0.008929
INFO: 06-14 22:47:11: train_shadownet.py:152 * 139659819460352 Epoch: 12 cost= 24.212904 seq distance= 0.946429 train accuracy= 0.013393
INFO: 06-14 22:47:13: train_shadownet.py:152 * 139659819460352 Epoch: 13 cost= 24.158167 seq distance= 0.933036 train accuracy= 0.000000
INFO: 06-14 22:47:16: train_shadownet.py:152 * 139659819460352 Epoch: 14 cost= 23.773846 seq distance= 0.928571 train accuracy= 0.017857
INFO: 06-14 22:47:18: train_shadownet.py:152 * 139659819460352 Epoch: 15 cost= 23.592751 seq distance= 0.946429 train accuracy= 0.017857
INFO: 06-14 22:47:20: train_shadownet.py:152 * 139659819460352 Epoch: 16 cost= 23.745987 seq distance= 0.941964 train accuracy= 0.008929
INFO: 06-14 22:47:23: train_shadownet.py:152 * 139659819460352 Epoch: 17 cost= 23.574177 seq distance= 0.950893 train accuracy= 0.017857
INFO: 06-14 22:47:25: train_shadownet.py:152 * 139659819460352 Epoch: 18 cost= 23.406982 seq distance= 0.950893 train accuracy= 0.000000
INFO: 06-14 22:47:27: train_shadownet.py:152 * 139659819460352 Epoch: 19 cost= 22.933498 seq distance= 0.941964 train accuracy= 0.013393
INFO: 06-14 22:47:30: train_shadownet.py:152 * 139659819460352 Epoch: 20 cost= 23.201078 seq distance= 0.933036 train accuracy= 0.031250
INFO: 06-14 22:47:32: train_shadownet.py:152 * 139659819460352 Epoch: 21 cost= 23.232277 seq distance= 0.950893 train accuracy= 0.004464
INFO: 06-14 22:47:35: train_shadownet.py:152 * 139659819460352 Epoch: 22 cost= 23.004635 seq distance= 0.946429 train accuracy= 0.013393
INFO: 06-14 22:47:37: train_shadownet.py:152 * 139659819460352 Epoch: 23 cost= 23.311407 seq distance= 0.933036 train accuracy= 0.031250
INFO: 06-14 22:47:39: train_shadownet.py:152 * 139659819460352 Epoch: 24 cost= 23.026693 seq distance= 0.955357 train accuracy= 0.000000
INFO: 06-14 22:47:42: train_shadownet.py:152 * 139659819460352 Epoch: 25 cost= 23.105705 seq distance= 0.946429 train accuracy= 0.008929
INFO: 06-14 22:47:44: train_shadownet.py:152 * 139659819460352 Epoch: 26 cost= 22.804663 seq distance= 0.950893 train accuracy= 0.022321
INFO: 06-14 22:47:46: train_shadownet.py:152 * 139659819460352 Epoch: 27 cost= 22.860863 seq distance= 0.928571 train accuracy= 0.008929
INFO: 06-14 22:47:49: train_shadownet.py:152 * 139659819460352 Epoch: 28 cost= 22.625650 seq distance= 0.924107 train accuracy= 0.031250
INFO: 06-14 22:47:51: train_shadownet.py:152 * 139659819460352 Epoch: 29 cost= 22.853930 seq distance= 0.933036 train accuracy= 0.022321
INFO: 06-14 22:47:53: train_shadownet.py:152 * 139659819460352 Epoch: 30 cost= 22.558969 seq distance= 0.928571 train accuracy= 0.004464
INFO: 06-14 22:47:56: train_shadownet.py:152 * 139659819460352 Epoch: 31 cost= 22.747505 seq distance= 0.941964 train accuracy= 0.017857
INFO: 06-14 22:47:58: train_shadownet.py:152 * 139659819460352 Epoch: 32 cost= 22.340515 seq distance= 0.915179 train accuracy= 0.013393
INFO: 06-14 22:48:00: train_shadownet.py:152 * 139659819460352 Epoch: 33 cost= 22.484076 seq distance= 0.950893 train accuracy= 0.013393
INFO: 06-14 22:48:03: train_shadownet.py:152 * 139659819460352 Epoch: 34 cost= 22.440865 seq distance= 0.924107 train accuracy= 0.017857
INFO: 06-14 22:48:05: train_shadownet.py:152 * 139659819460352 Epoch: 35 cost= 22.427692 seq distance= 0.928571 train accuracy= 0.026786
INFO: 06-14 22:48:07: train_shadownet.py:152 * 139659819460352 Epoch: 36 cost= 22.171822 seq distance= 0.933036 train accuracy= 0.017857
INFO: 06-14 22:48:10: train_shadownet.py:152 * 139659819460352 Epoch: 37 cost= 22.364008 seq distance= 0.937500 train accuracy= 0.022321
INFO: 06-14 22:48:12: train_shadownet.py:152 * 139659819460352 Epoch: 38 cost= 22.343708 seq distance= 0.937500 train accuracy= 0.004464
INFO: 06-14 22:48:15: train_shadownet.py:152 * 139659819460352 Epoch: 39 cost= 22.284977 seq distance= 0.906250 train accuracy= 0.026786
INFO: 06-14 22:48:17: train_shadownet.py:152 * 139659819460352 Epoch: 40 cost= 22.412735 seq distance= 0.950893 train accuracy= 0.017857
GitHub - Maybe...
[1. crnn -> 과정 ...
yun@itlab-serv...
yun
[config.py (~/AI...
22:48

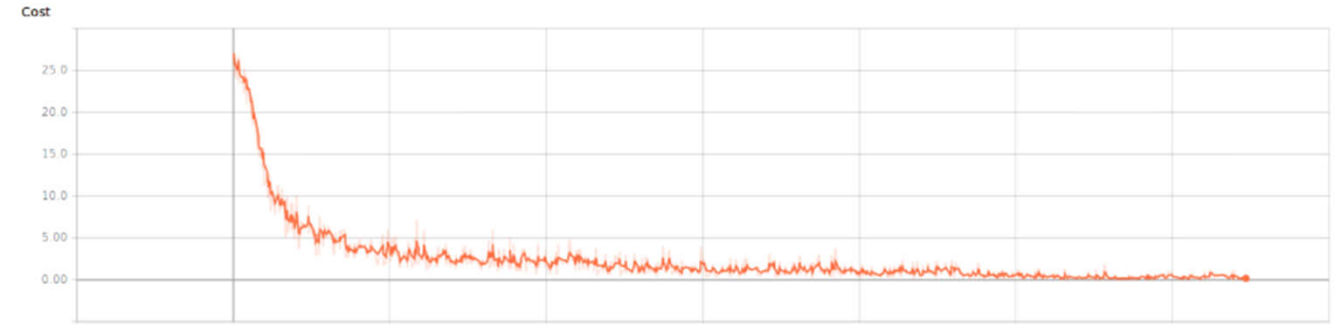


~/AI_project_speech/CRNN/CRNN_Tensorflow-master

```
5616 Epoch: 2 cost=  0.558493 seq distance=  0.026786 train accuracy=  0.897321
5616 Epoch: 3 cost=  0.478799 seq distance=  0.008929 train accuracy=  0.991071
5616 Epoch: 4 cost=  0.341261 seq distance=  0.008929 train accuracy=  0.977679
5616 Epoch: 5 cost=  0.661102 seq distance=  0.022321 train accuracy=  0.937500
5616 Epoch: 6 cost=  0.490162 seq distance=  0.026786 train accuracy=  0.924107
5616 Epoch: 7 cost=  0.526102 seq distance=  0.013393 train accuracy=  0.950893
5616 Epoch: 8 cost=  0.576091 seq distance=  0.013393 train accuracy=  0.973214
5616 Epoch: 9 cost=  0.403551 seq distance=  0.008929 train accuracy=  0.955357
5616 Epoch: 10 cost=  0.403516 seq distance=  0.008929 train accuracy=  0.991071
5616 Epoch: 11 cost=  0.429775 seq distance=  0.013393 train accuracy=  0.986607
5616 Epoch: 12 cost=  0.432648 seq distance=  0.008929 train accuracy=  0.968750
5616 Epoch: 13 cost=  0.446549 seq distance=  0.013393 train accuracy=  0.986607
5616 Epoch: 14 cost=  0.500075 seq distance=  0.008929 train accuracy=  0.991071
5616 Epoch: 15 cost=  0.451168 seq distance=  0.004464 train accuracy=  0.995536
5616 Epoch: 16 cost=  0.477360 seq distance=  0.008929 train accuracy=  0.991071
eq distance=  0.008929 train accuracy=  0.991071
eq distance=  0.004464 train accuracy=  0.977679
eq distance=  0.013393 train accuracy=  0.973214
eq distance=  0.004464 train accuracy=  0.995536
eq distance=  0.022321 train accuracy=  0.928571
eq distance=  0.008929 train accuracy=  0.968750
eq distance=  0.004464 train accuracy=  0.977679
eq distance=  0.008929 train accuracy=  0.973214
eq distance=  0.008929 train accuracy=  0.973214
eq distance=  0.008929 train accuracy=  0.991071
eq distance=  0.004464 train accuracy=  0.977679
eq distance=  0.017857 train accuracy=  0.946429
eq distance=  0.022321 train accuracy=  0.941964
eq distance=  0.017857 train accuracy=  0.982143
eq distance=  0.004464 train accuracy=  0.995536
eq distance=  0.008929 train accuracy=  0.968750
eq distance=  0.008929 train accuracy=  0.991071
eq distance=  0.008929 train accuracy=  0.991071
eq distance=  0.017857 train accuracy=  0.982143
eq distance=  0.004464 train accuracy=  0.995536
eq distance=  0.004464 train accuracy=  0.973214
eq distance=  0.017857 train accuracy=  0.964286
eq distance=  0.008929 train accuracy=  0.991071
eq distance=  0.000000 train accuracy=  1.000000
```

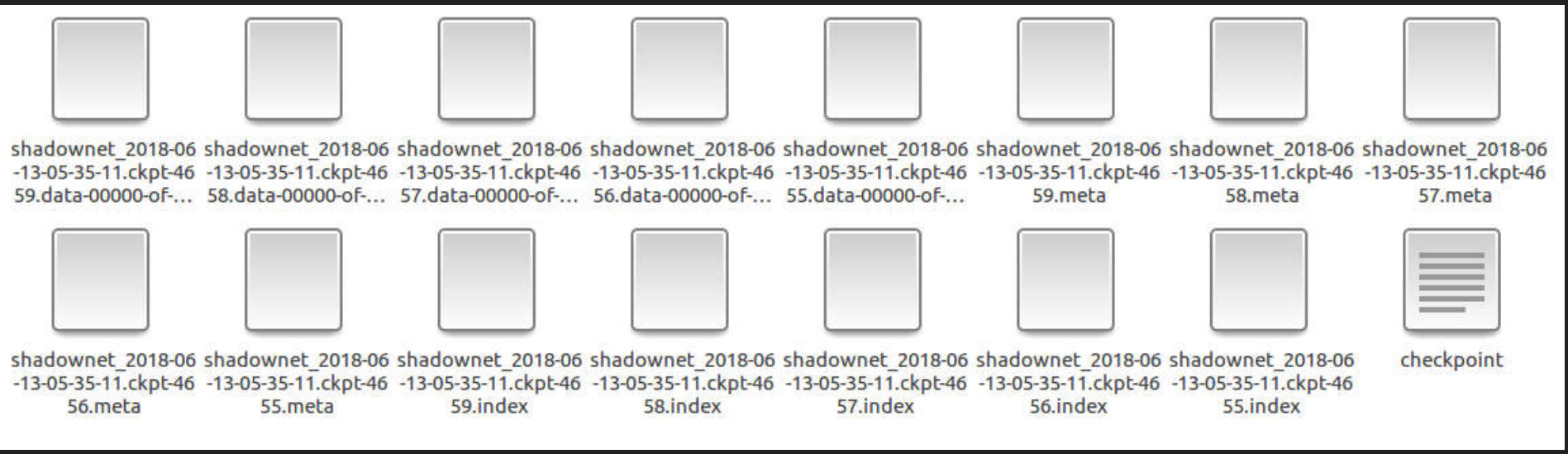
(~/... yun@itlab-se... checkpoint 22:54

During my experiment the `loss` drops as follows

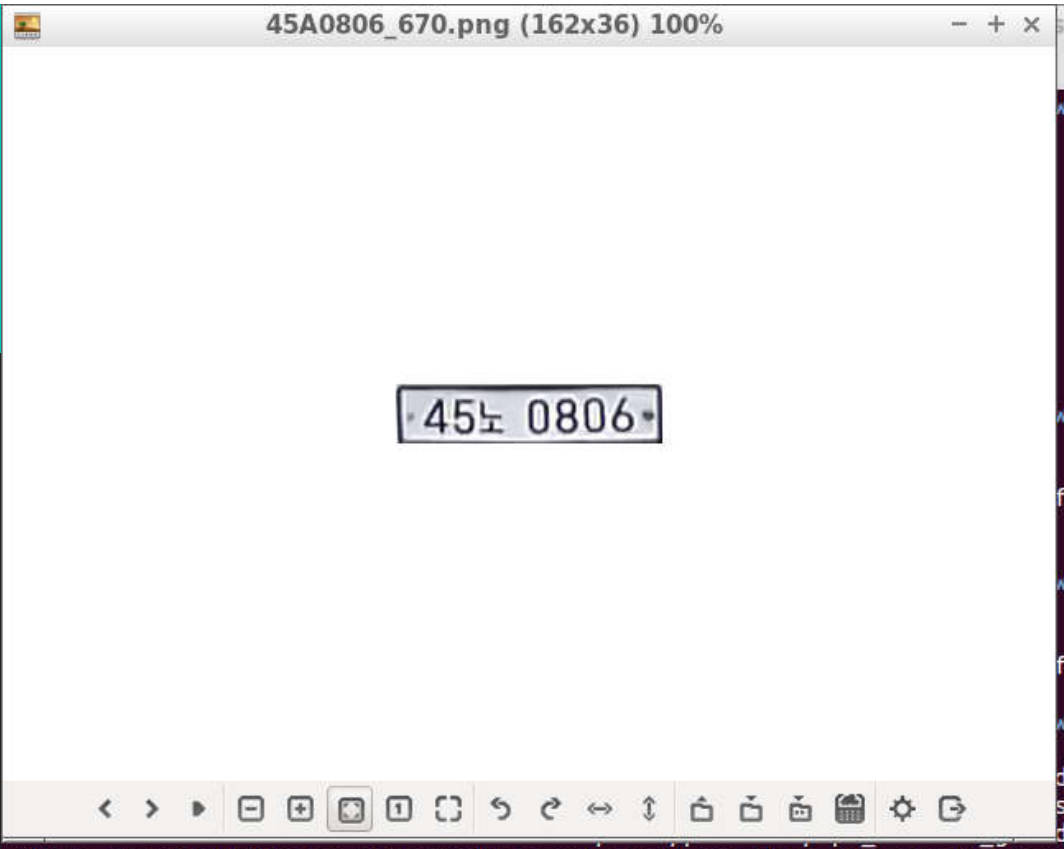


The `distance` between the ground truth and the prediction drops as follows

개발과정_CRNN



- Model 생성 후 모습



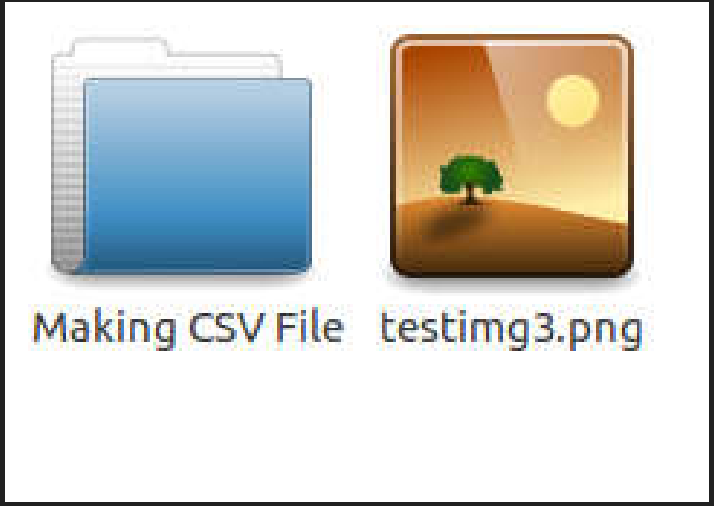

```
speech/CRNN/CRNN_Tensorflow-master
w-master$ ls

w-master$ python ./demo_shadownet.py --image_path ./45A0806_670.png --weights_p

fo(format(ops.split(image_path)[1]), logger.info(preds[0])])
                                                          ^

w-master$ python ./demo_shadownet.py --image_path ./45A0806_670.png --weights_p

fo(format(ops.split(image_path)[1]), logger.info(preds[0])])
                                                          ^
w-master$ python ./demo_shadownet.py --image_path ./45A0806_670.png --weights_p
d.cc:45] The TensorFlow library wasn't compiled to use SSE4.1 instructions, but
s.
d.cc:45] The TensorFlow library wasn't compiled to use SSE4.2 instructions, but
 these are available on your machine and could speed up CPU computations.
2018-06-14 23:31:16.816799: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use AVX instructions, but th
ese are available on your machine and could speed up CPU computations.
2018-06-14 23:31:16.816803: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use AVX2 instructions, but t
hese are available on your machine and could speed up CPU computations.
2018-06-14 23:31:16.816809: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use FMA instructions, but th
ese are available on your machine and could speed up CPU computations.
INFO:tensorflow:Restoring parameters from model/shadownet/shadownet_2018-06-13-05-35-11.ckpt-4659
INFO: 06-14 23:31:16: tf_logging.py:82 * 140114224707328 Restoring parameters from model/shadownet/shadownet_2018-06-13-05-35-11.ckpt-4659
INFO: 06-14 23:31:16: demo_shadownet.py:84 * 140114224707328 Predict image 45A0806_670.png label 45a0806
^Z
[1]+  Stopped                 python ./demo_shadownet.py --image_path ./45A0806_670.png --weights_path model/shadownet/shadownet_2018-06-13-05-35-11.c
kpt-4659
```

개발과정_CRNN

```
{"아": "a" ,"어" : "b","오" : "c","우" : "d","바" : "e","버" : "f","보" : "g","부" : "h",
 "다": "i","더" : "j","도" : "k","두" : "l","가" : "m","거" : "n","고" : "o","구" : "p",
 "자" : "q","저" : "r","조" : "s","주" : "t","마" : "u","머" : "v","모" : "w", "무" : "x",
 "나" : "y","너" : "z","노" : "A","누" : "B","라" : "C","러" : "D","로" : "E","루" : "F",
 "사" : "G","서" : "H","소" : "I","수" : "J","허" : "K"
 #"하" : "_" "호" : " " "후" : "A"
```

- 한글-영어 Matching

개발과정_Canny Edge



- Canny Edge 폴더의 첫 모습

Canny_Edge

File Edit View Bookmarks Go Tools Help

/home/yun/AI_project_speech/Canny_Edge

Directory Tree

- yun
  - AI_project_spe
  - Canny_Edge
    - Making CSV
  - CRNN
    - crnn_genera
    - CRNN_ppt_i
    - CRNN_Tens
      - crnn_mod
      - data
        - char_dic
        - images
        - test_ima
      - data_prov
      - global_cor
      - local_utils
      - logs
      - model
        - shadown
          - <No sub
        - numberPla
        - Test
          - <No sub
        - Train
          - <No sub
      - __pycache_
      - tboard


Making CSV File


testimg3.png

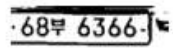

blur.jpg


Canny_Edge_Detection.py


gray.jpg


canny.jpg


snake.jpg



plate_th.jpg

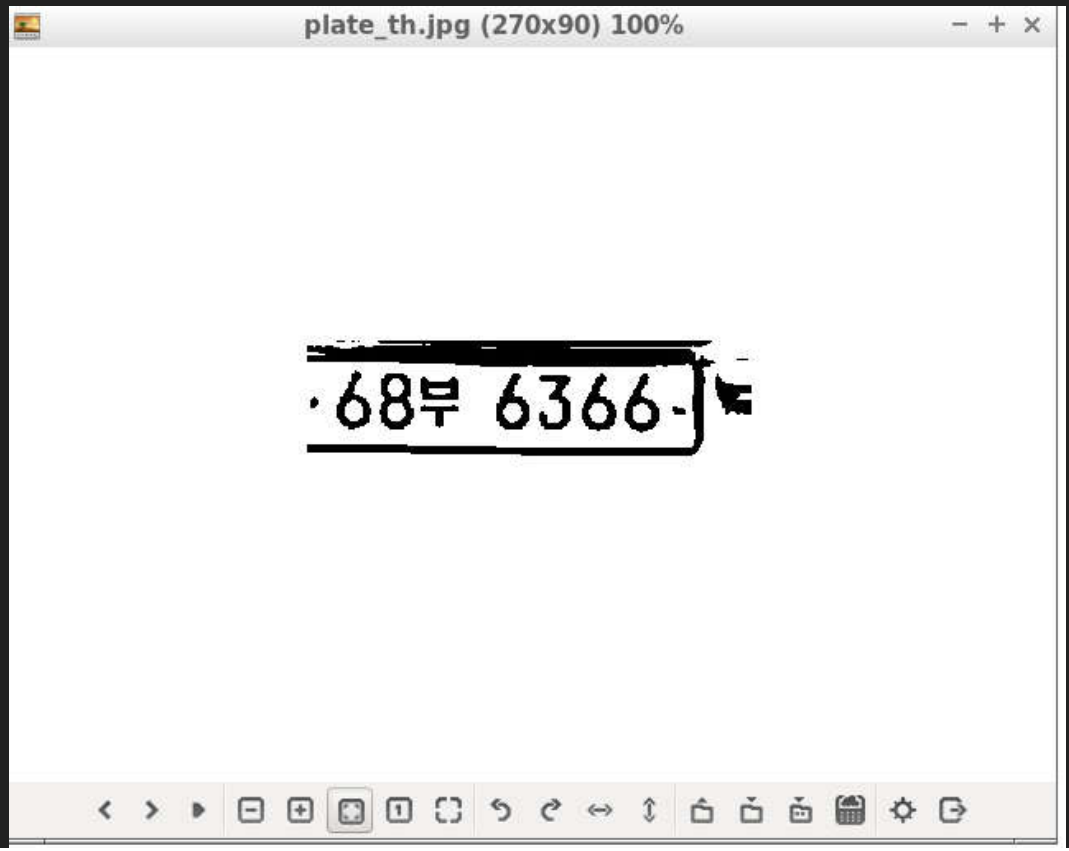
er_plate.jpg

9 items

Free space: 1.4 TiB (Total: 1.8 T

[GitHub - Mayb… [1. crnn -> 과정 … Canny_Edge [checkpoint] [yun@itlab-ser… 23:01 xpad

개발과정_Canny Edge



- Output File의 모습

~/AI_project_speech/Canny Edge

dit View Search Tools Documents Help

config.py × | Canny_Edge_Detection.py ×

```
r/bin/etc python

t cv2
t numpy as np
t pytesseract
 PIL import Image


 Recognition:
f ExtractNumber(self):
   Number = 'testimg3.png'
   img = cv2.imread(Number, cv2.IMREAD_COLOR)
   copy_img = img.copy()
   img2 = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
   cv2.imwrite('gray.jpg', img2)
   blur = cv2.GaussianBlur(img2, (3, 3), 0)
   cv2.imwrite('blur.jpg', blur)
   canny = cv2.Canny(blur, 100, 200)
   cv2.imwrite('canny.jpg', canny)
   cnts, contours, hierarchy = cv2.findContours(canny, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

   box1 = []
   f_count = 0
   select = 0
   plate_width = 0

   for i in range(len(contours)):
       cnt = contours[i]
       area = cv2.contourArea(cnt)
       x, y, w, h = cv2.boundingRect(cnt)
       rect_area = w * h  # area size
       aspect_ratio = float(w) / h  # ratio = width/height

       if (aspect_ratio >= 0.2) and (aspect_ratio <= 1.0) and (rect_area >= 100) and (rect_area <= 700):
           cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 1)
           box1.append(cv2.boundingRect(cnt))
```

○ 하나의 Input File

Python ▾ Tab Width: 8 ▾ Ln 1, Col 1 ▾ INS

[GitHub - Mayb... [1. crnn -> 과정 ... Canny Edge Canny_Edge_D... [checkpoint] 22:57

ew Search Tools Documents Help

config.py × | Canny_Edge_ Detection.py ×

```
        gradient = float(delta_y) / float(delta_x)
        if gradient < 0.25:
            count = count + 1
    # measure number plate size
    if count > f_count:
        select = m
        f_count = count;
        plate_width = delta_x
2.imwrite('snake.jpg', img)

nber_plate = copy_img[box1[select][1] - 10:box1[select][3] + box1[select][1] + 20,
                box1[select][0] - 10:140 + box1[select][0]]

nin = box1[select][1] - 10
nax = box1[select][3] + box1[select][1] + 20
nin = box1[select][0] - 10
nax = 140 + box1[select][0]

int(y_min, y_max, x_min, x_max)

size_plate = cv2.resize(number_plate, None, fx=1.8, fy=1.8, interpolation=cv2.INTER_CUBIC + cv2.INTER_LINEAR)
ate_gray = cv2.cvtColor(resize_plate, cv2.COLOR_BGR2GRAY)
, th_plate = cv2.threshold(plate_gray, 150, 255, cv2.THRESH_BINARY)

2.imwrite('plate_th.jpg', th_plate)
rnel = np.ones((3, 3), np.uint8)
_plate = cv2.erode(th_plate, kernel, iterations=1)
_invplate = er_plate
2.imwrite('er_plate.jpg', er_invplate)
sult = pytesseract.image_to_string(Image.open('er_plate.jpg'), lang='kor')
turn (result.replace(" ", ""))


= Recognition()
ecogtest.ExtractNumber()
t)
```

○ PRINT 추가!

Python ▾ Tab Width: 8 ▾ Ln 1, Col 1 ▾ INS

[GitHub - Mayb... | [1. crnn -> 과정 ... | Canny Edge | Canny_Edge_D... | [checkpoint] | 22:58

~/AI_project_speech/Canny_Edge/Making CSV File

dit View Search Tools Documents Help

```
        x_min = box1[select][0] - 10
        x_max = 140 + box1[select][0]

        #ADDED LINE
        output=[x_min, y_min, x_max, y_max]
        output_total.append(output)

D LINE
    print(output_total)
    f = open('output.csv', 'w', encoding='utf-8', newline='')
    wr = csv.writer(f)
    for i in range(0, len(input_list)):
            wr.writerow([input_list[i], output_total[i][0], output_total[i][1], output_total[i][2], output_total[i][3]])
    f.close()

     resize_plate = cv2.resize(number_plate, None, fx=1.8, fy=1.8, interpolation=cv2.INTER_CUBIC + cv2.INTER_LINEAR)
     plate_gray = cv2.cvtColor(resize_plate, cv2.COLOR_BGR2GRAY)
     ret, th_plate = cv2.threshold(plate_gray, 150, 255, cv2.THRESH_BINARY)

     cv2.imwrite('plate_th.jpg', th_plate)
     kernel = np.ones((3, 3), np.uint8)
     er_plate = cv2.erode(th_plate, kernel, iterations=1)
     er_invplate = er_plate
     cv2.imwrite('er_plate.jpg', er_invplate)
     result = pytesseract.image_to_string(Image.open('er_plate.jpg'), lang='kor')
     return (result.replace(" ", ""))

-------------------------------------------------------------------------------------------------------------------

D LINE
os.path.dirname(os.path.realpath(__file__))
=os.listdir(path)
_png = [i for i in files if i.endswith('.png')]

test = Recognition()
t = recogtest.ExtractNumber(files_png)
t(result)
```

○ 경로 안에 있는 모든 png파일 선택

Python Tab Width: 8 Ln 1, Col 1 INS

[GitH... [1. cr... Maki... [chec... [yun... blur.j... gray.... cann... plate... make... 23:03

~/AI_project_speech/Canny_Edge/Making CSV File

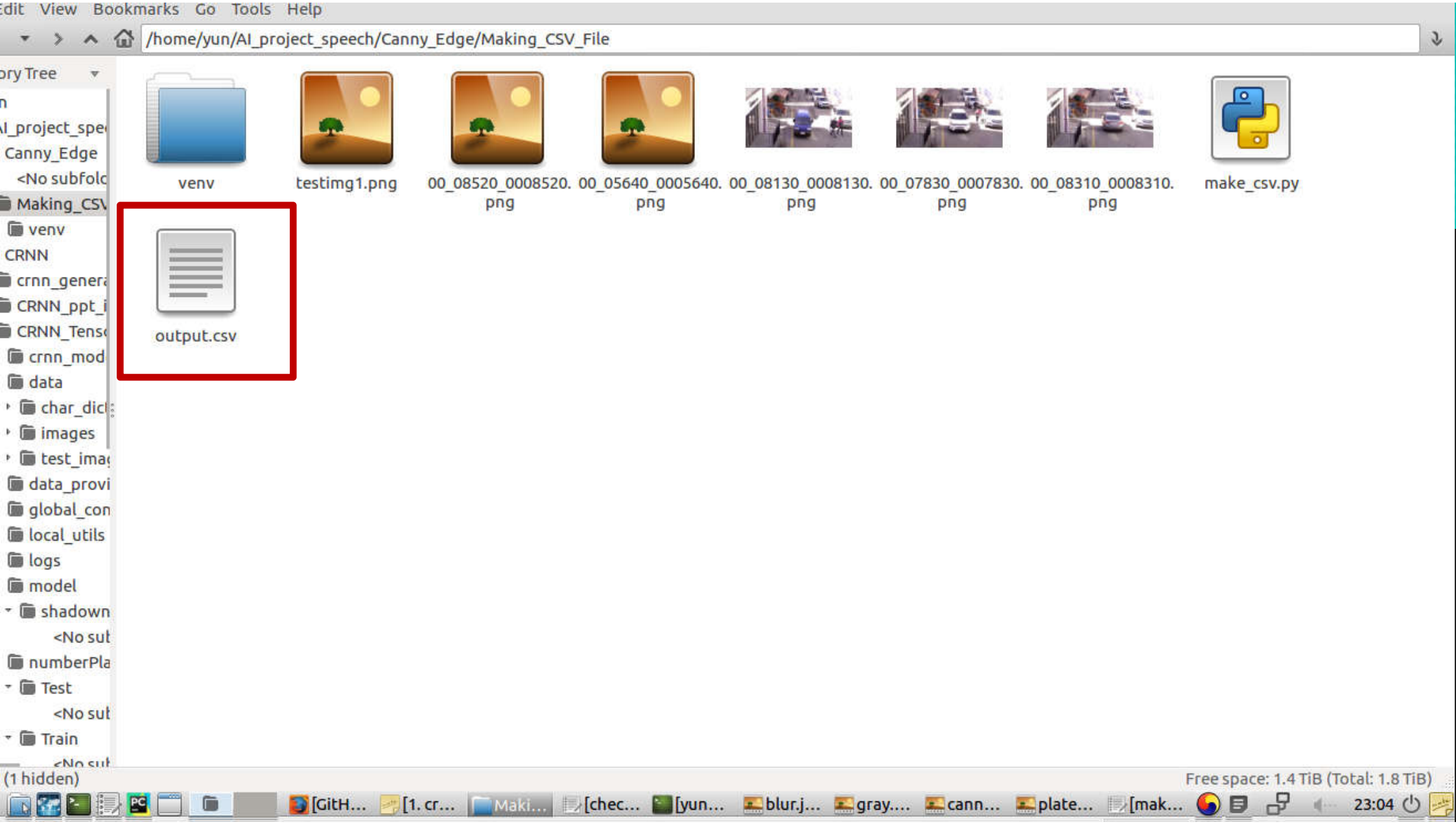
dit View Search Tools Documents Help

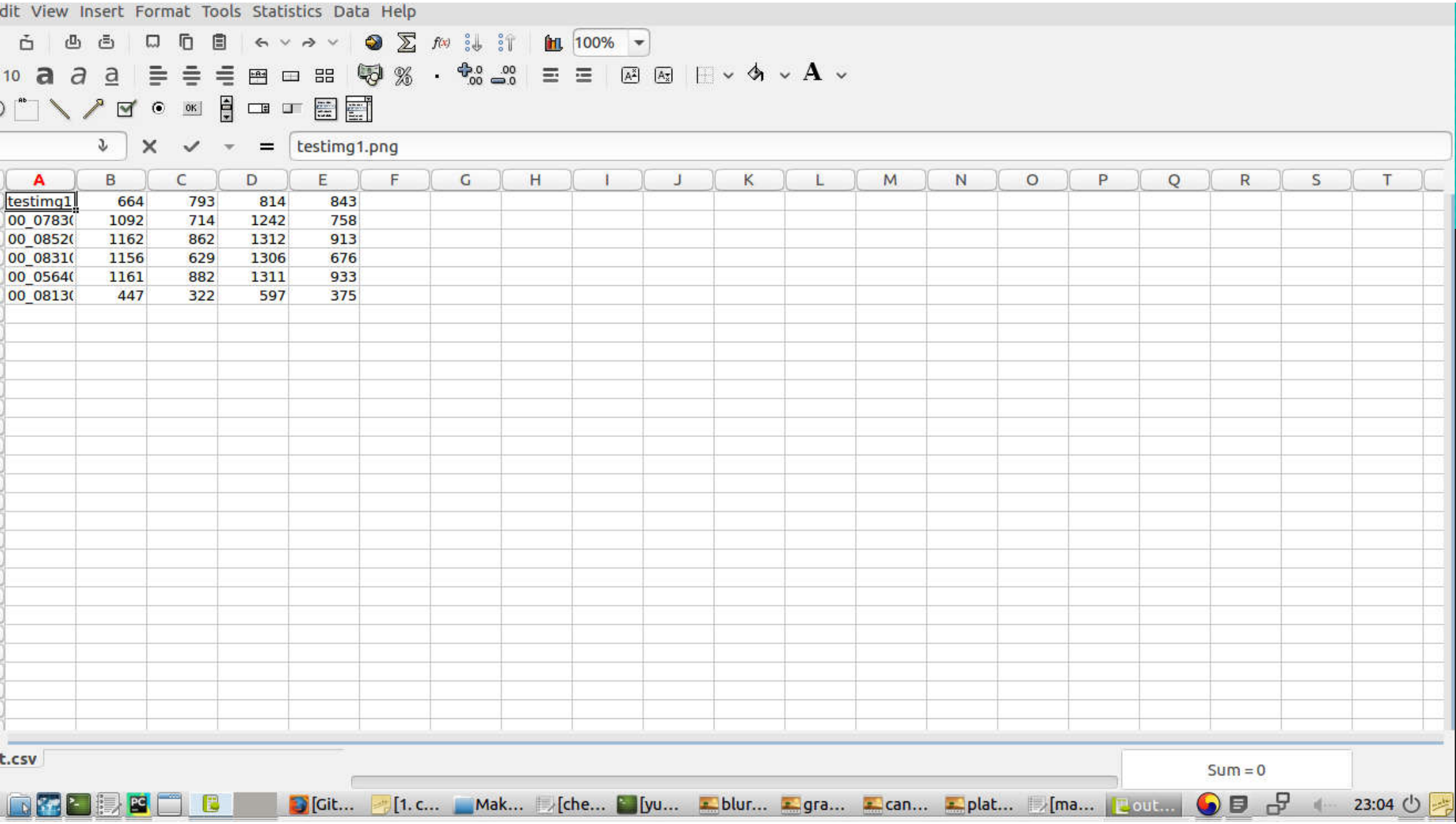
```
            select = m
            f_count = count;
            plate_width = delta_x
        cv2.imwrite('snake.jpg', img)

    number_plate = copy_img[box1[select][1] - 10:box1[select][3] + box1[select][1] + 20,
                            box1[select][0] - 10:140 + box1[select][0]]

    y_min = box1[select][1] - 10
    y_max = box1[select][3] + box1[select][1] + 20
    x_min = box1[select][0] - 10
    x_max = 140 + box1[select][0]

    #ADDED LINE
    output=[x_min, y_min, x_max, y_max]
    output_total.append(output)

print(output_total)
f = open('output.csv', 'w', encoding='utf-8', newline='')
wr = csv.writer(f)
for i in range(0, len(input_list)):
        wr.writerow([input_list[i], output_total[i][0], output_total[i][1], output_total[i][2], output_total[i][3]])
f.close()

    resize_plate = cv2.resize(number_plate, None, fx=1.8, fy=1.8, interpolation=cv2.INTER_CUBIC + cv2.INTER_LINEAR)
    plate_gray = cv2.cvtColor(resize_plate, cv2.COLOR_BGR2GRAY)
    ret, th_plate = cv2.threshold(plate_gray, 150, 255, cv2.THRESH_BINARY)

    cv2.imwrite('plate_th.jpg', th_plate)
    kernel = np.ones((3, 3), np.uint8)
    er_plate = cv2.erode(th_plate, kernel, iterations=1)
    er_invplate = er_plate
    cv2.imwrite('er_plate.jpg', er_invplate)
    result = pytesseract.image_to_string(Image.open('er_plate.jpg'), lang='kor')
    return (result.replace(" ", ""))
```

○ 모든 파일에 대해 CSV 기록 및 출력

Python Tab Width: 8 Ln 1, Col 1 INS

[GitH... [1. cr... Maki... [chec... [yun... blur.j... gray.... cann... plate... make... 23:02


Edit View Bookmarks Go Tools Help
/home/yun/AI_project_speech/Canny_Edge/Making_CSV_File
ory Tree
Canny_Edge
<No subfolc
Making_CSV
venv
CRNN
crnn_genera
CRNN_ppt_i
CRNN_Tens
crnn_mod
data
char_dict
images
test_imag
data_provi
global_con
local_utils
logs
model
shadown
<No sut
numberPla
Test
<No sut
Train
venv
testimg1.png
00_08520_0008520.png
00_05640_0005640.png
00_08130_0008130.png
00_07830_0007830.png
00_08310_0008310.png
make_csv.py
output.csv
(1 hidden)
Free space: 1.4 TiB (Total: 1.8 TiB)
[GitH...
[1. cr...
Maki...
[chec...
[yun...
blur.j...
gray....
cann...
plate...
[mak...
23:04


dit View Insert Format Tools Statistics Data Help
100%
testimg1.png
A B C D E F G H I J K L M N O P Q R S T
testimg1 664 793 814 843
00_0783(1092 714 1242 758
00_0852(1162 862 1312 913
00_0831(1156 629 1306 676
00_0564(1161 882 1311 933
00_0813(447 322 597 375
t.csv
Sum = 0
[Git... [1. c... Mak... [che... [yu... blur... gra... can... plat... [ma... out...
23:04

4. 성능평가

	YOLO v3	canny Edge
fps	11fps	20fps
accuracy	**약 50%**	약 14%
학습	8시간이상	0시간
필요한메모리	weight의량	이미지한장에서 계산 다소적음
필요한메타데이터	weight의량	없음

5. 향후 개선 방향

- 1. Training epoch증가
- 2. 데이터셋의 증가
- 3. 어두운 데이터 밝기 변화
- 4. 유니코드 맵핑

6. QnA