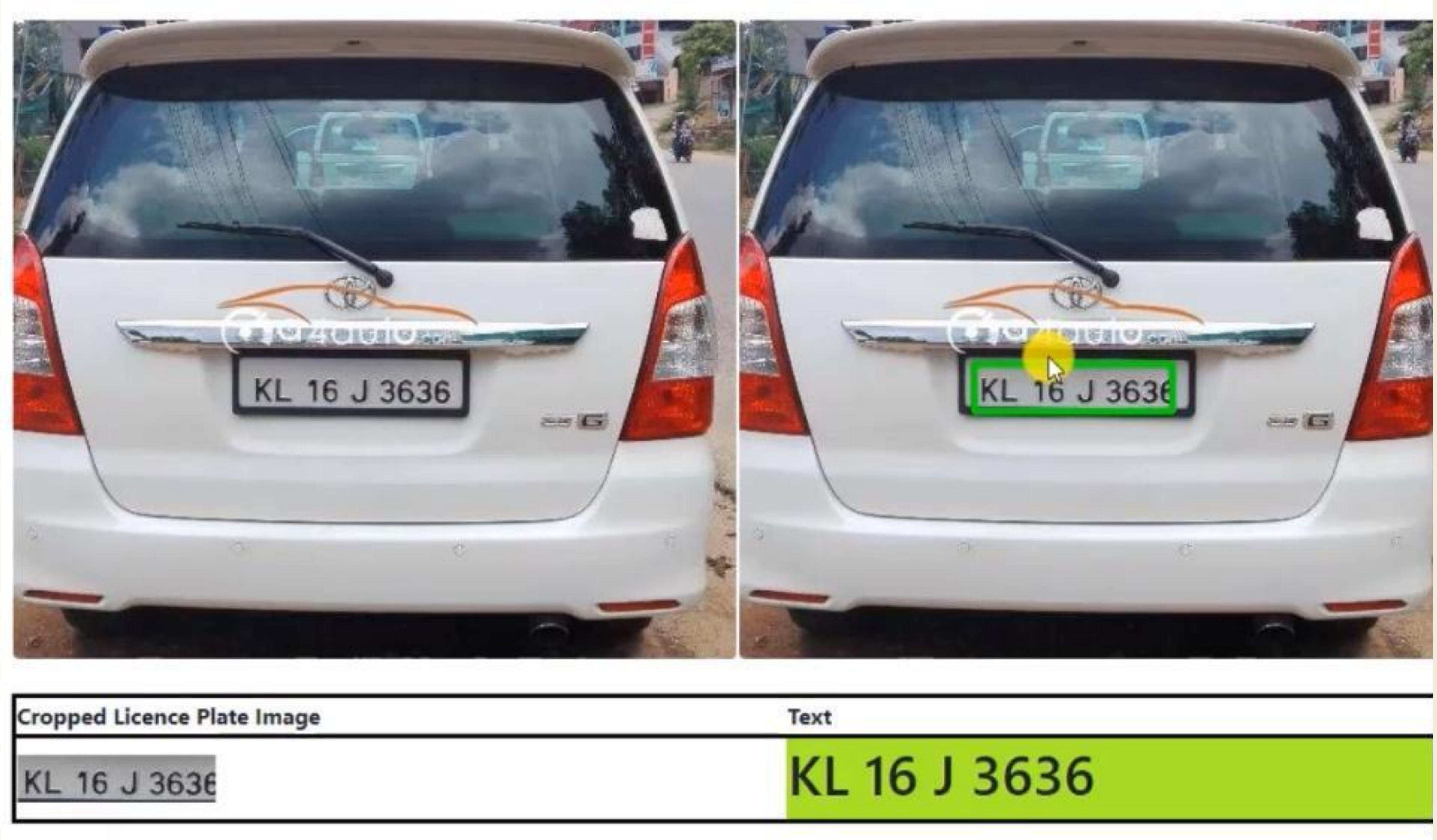
License Plate Detection text

weitsunglin

Outline

- Purpose
- FrameWork
- Download car number plate image
- Labeling to get car number plate object in image
- convert xml file to csv
- normalization data
- deep learning model
- Using model to predict
- Optical Character Recognition
- limitation of pytesseract
- Problem
- Conclusion

Purpose

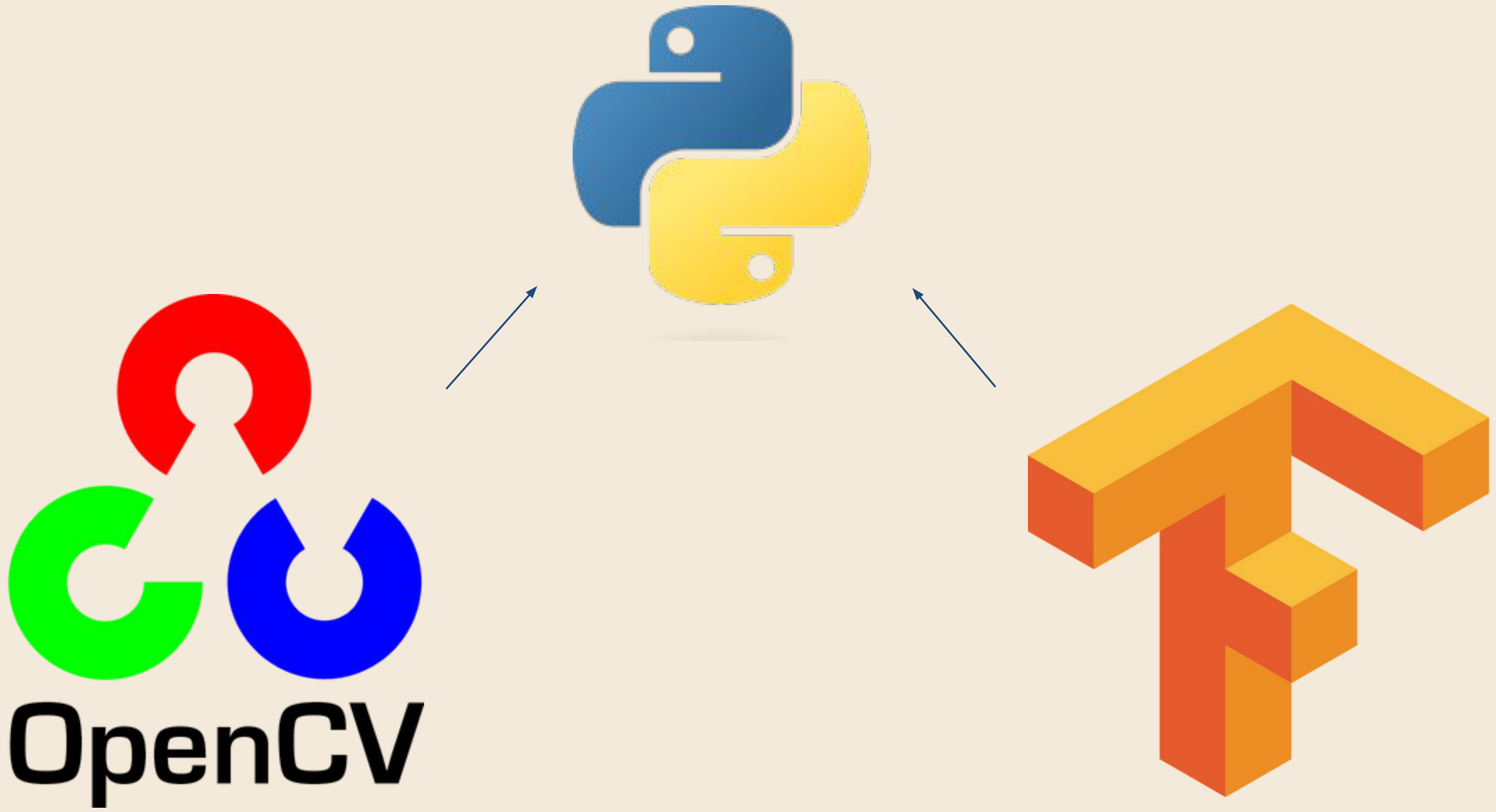


Cropped Licence Plate Image	Text
	KL 16 J 3636

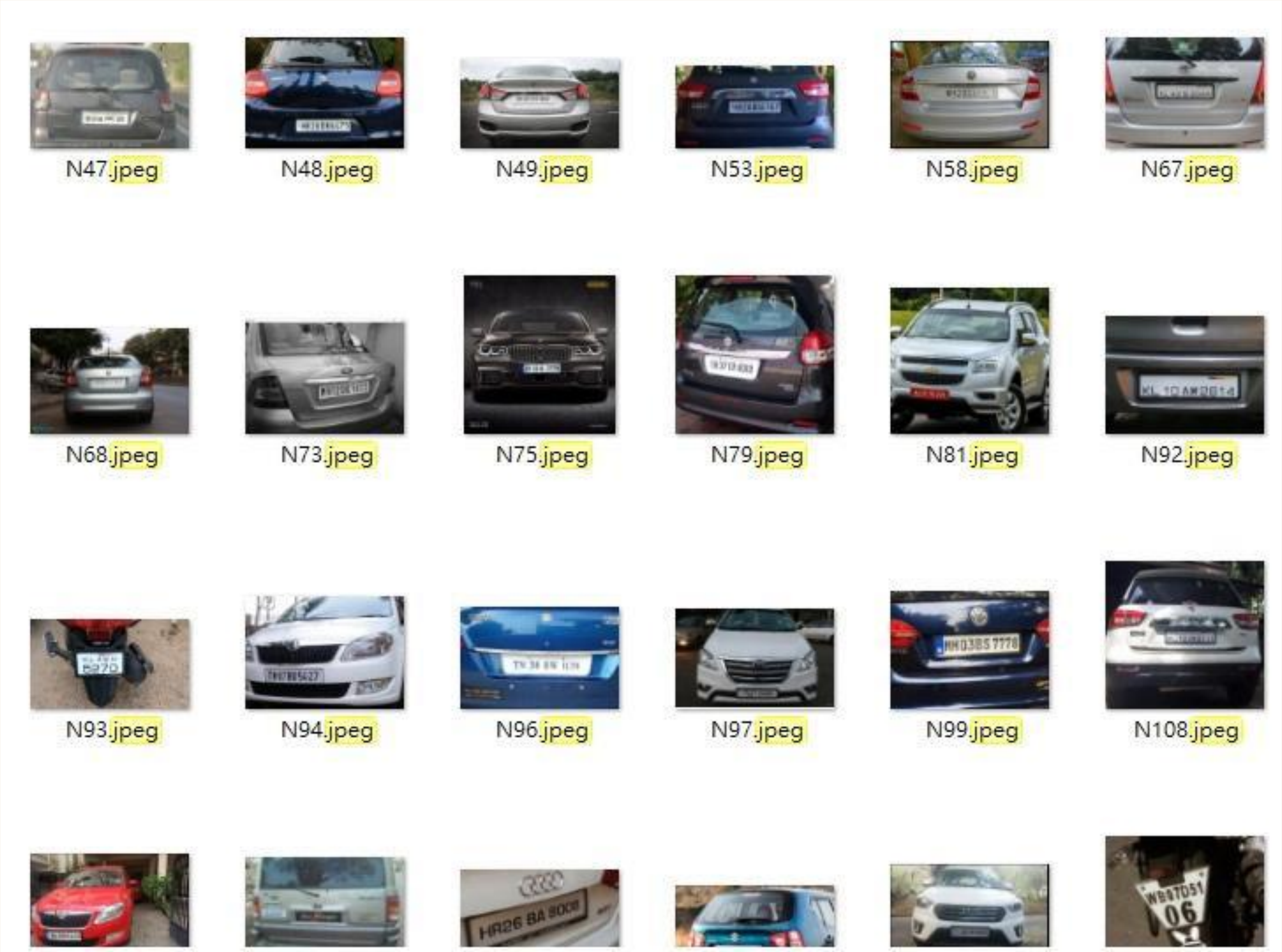
Recognize the license plate in the picture and display the number in it

Framework

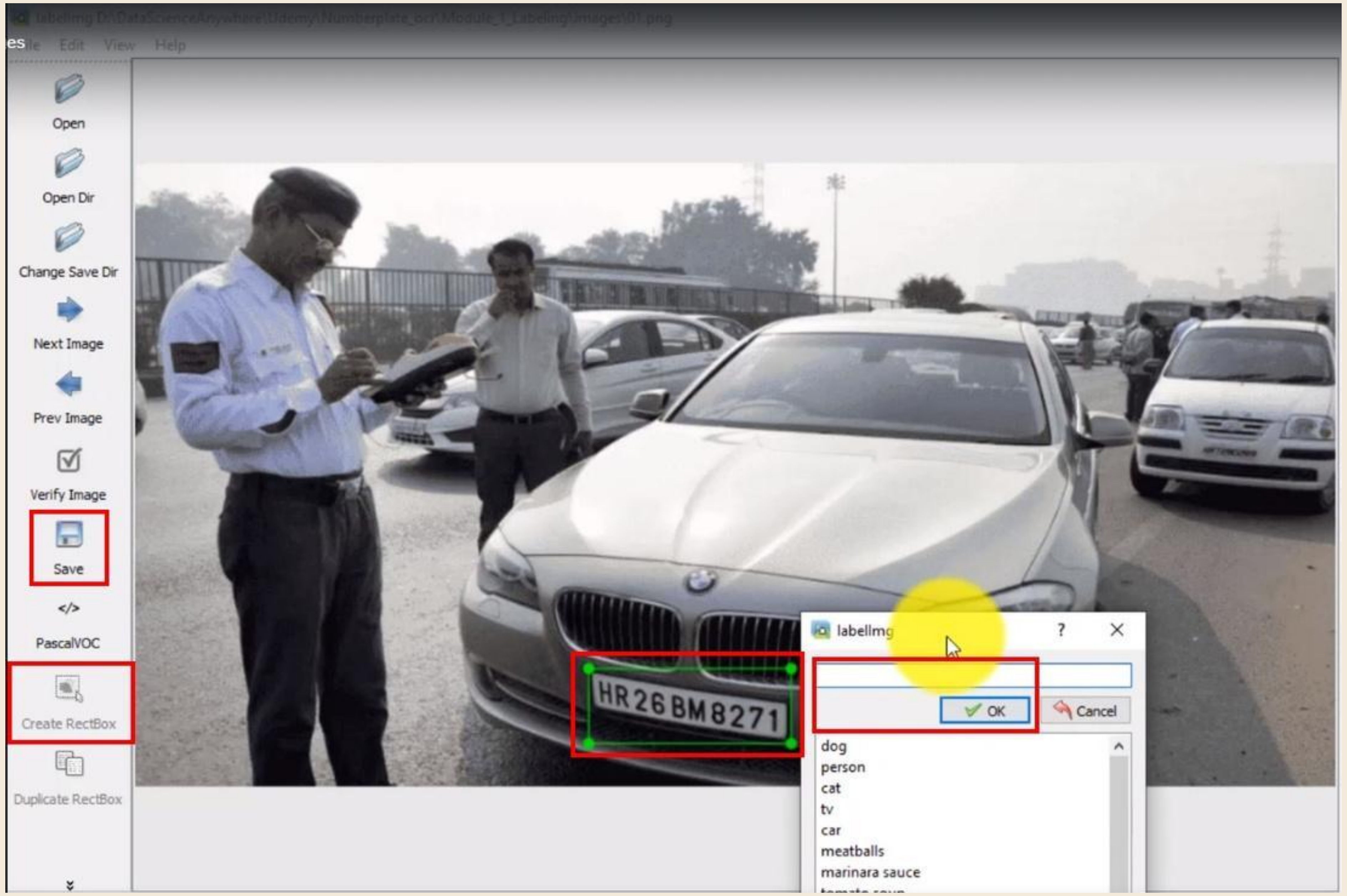
- Python: The main program to process image detection
- Opencv: image and data processing
- Tensorflow: train Deep Learning model and Optical Character Recognition

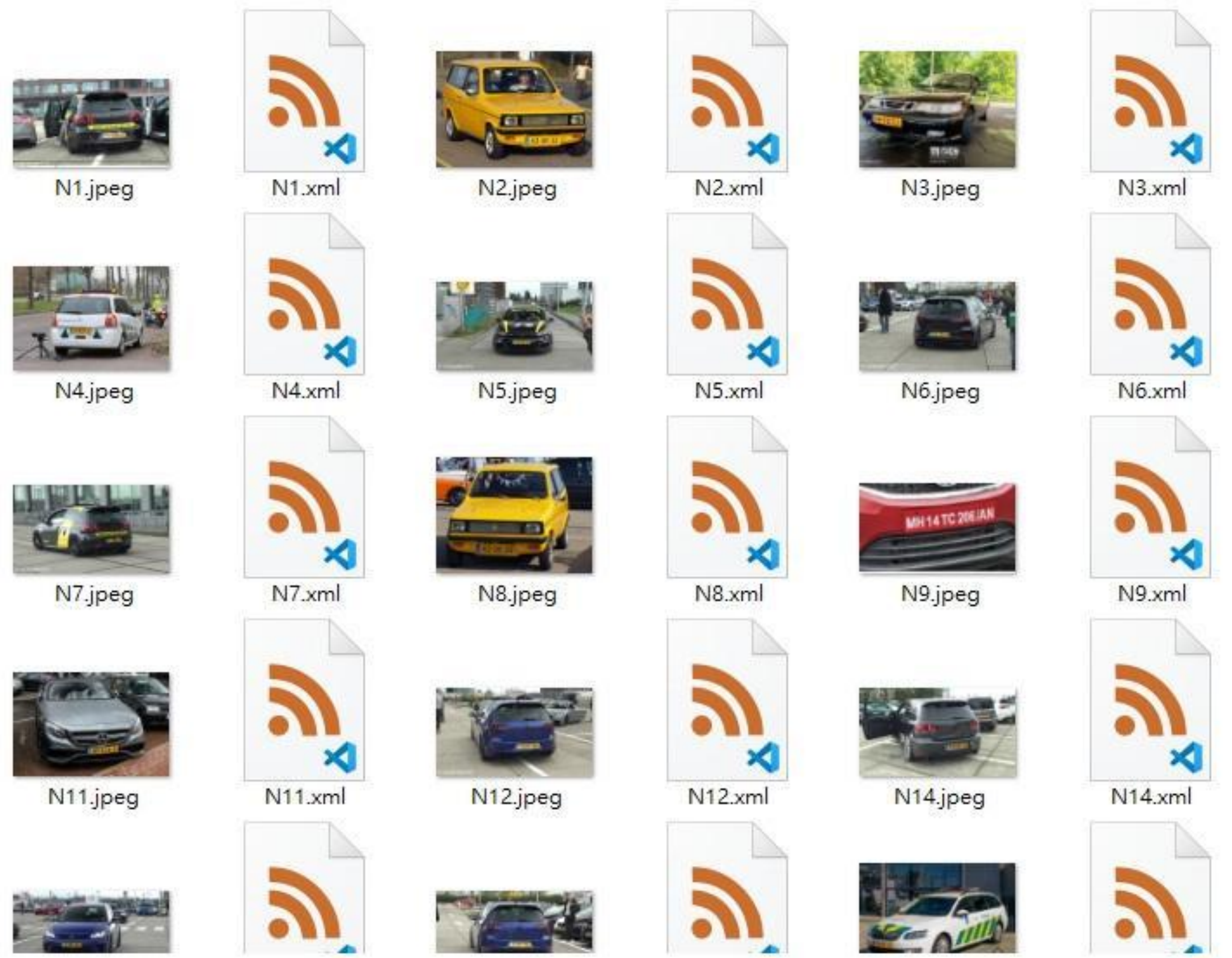


Download car with number plate image

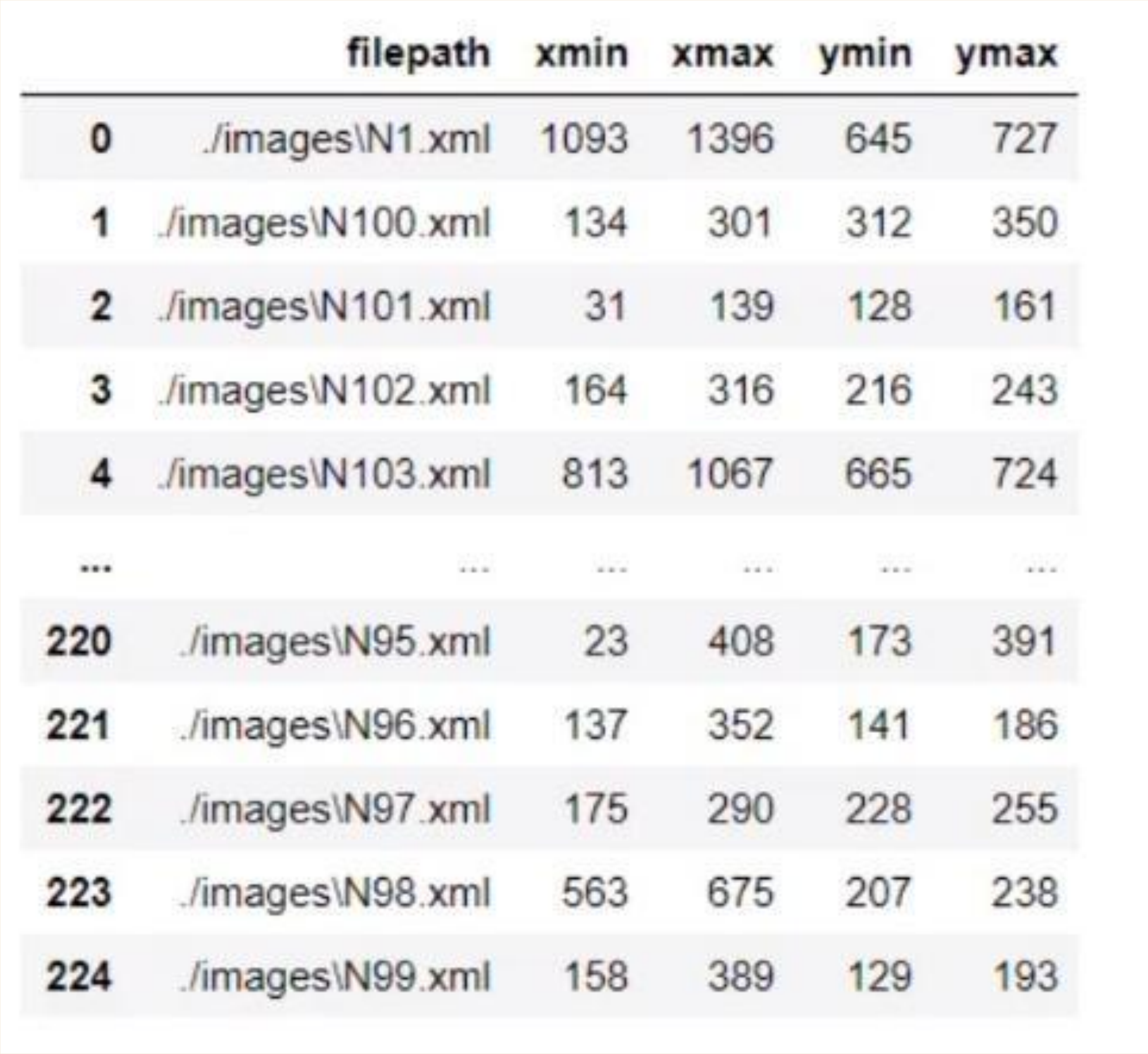


Using labeling to get number plate object in image

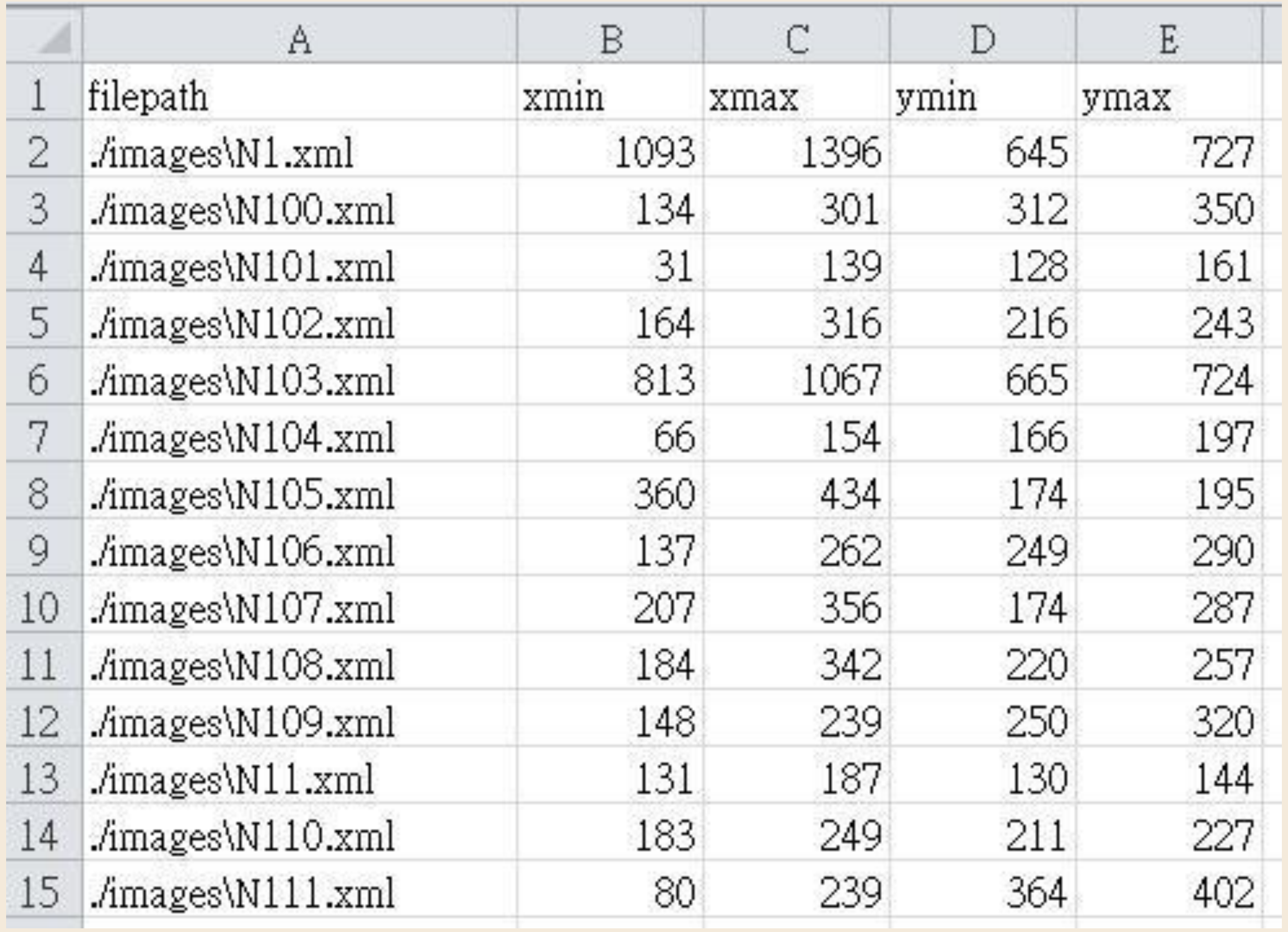




Convert xml file to csv

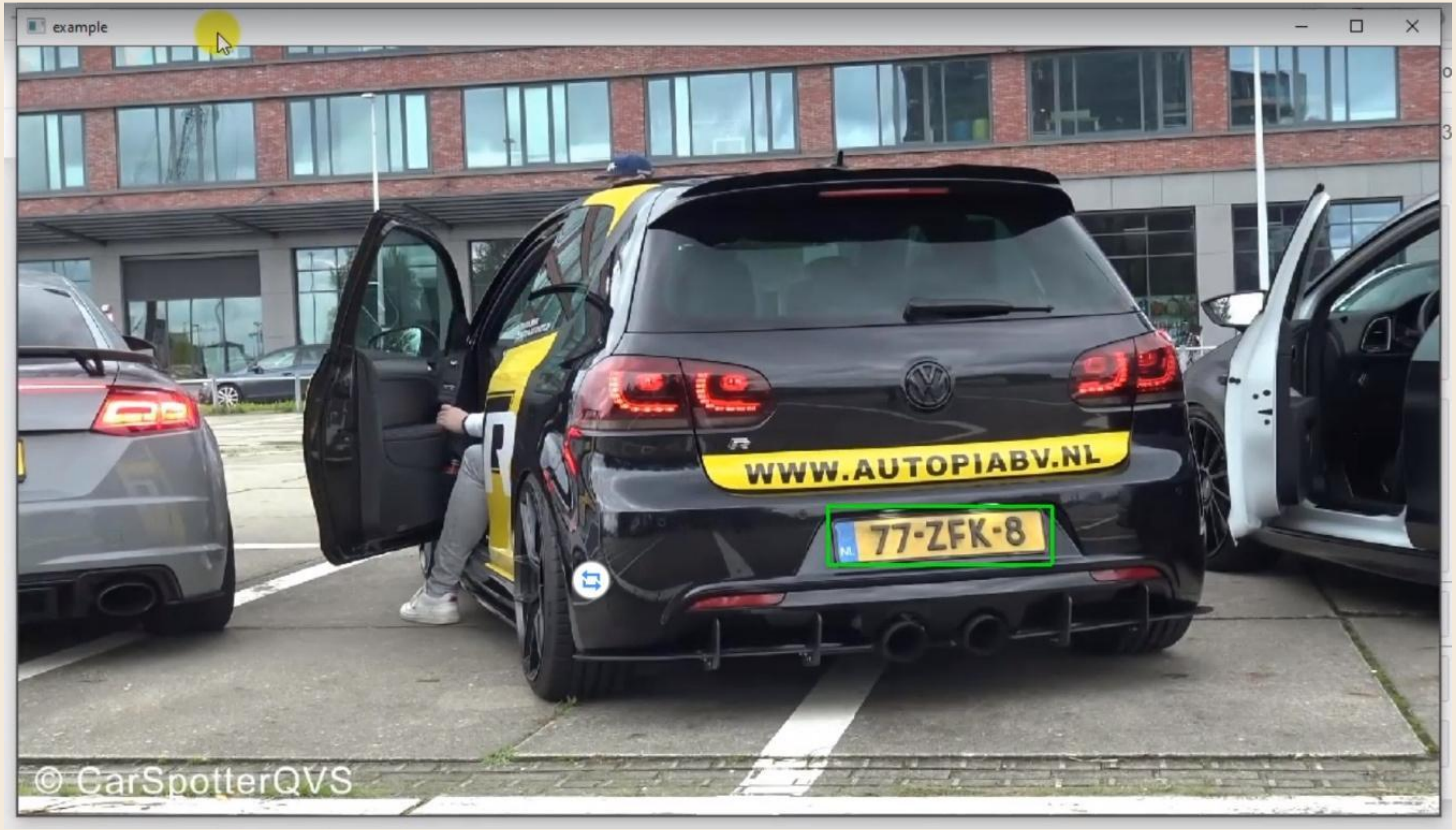


	filepath	xmin	xmax	ymin	ymax
0	./images\N1.xml	1093	1396	645	727
1	./images\N100.xml	134	301	312	350
2	./images\N101.xml	31	139	128	161
3	./images\N102.xml	164	316	216	243
4	./images\N103.xml	813	1067	665	724
...	...	...	...	...	...
220	./images\N95.xml	23	408	173	391
221	./images\N96.xml	137	352	141	186
222	./images\N97.xml	175	290	228	255
223	./images\N98.xml	563	675	207	238
224	./images\N99.xml	158	389	129	193

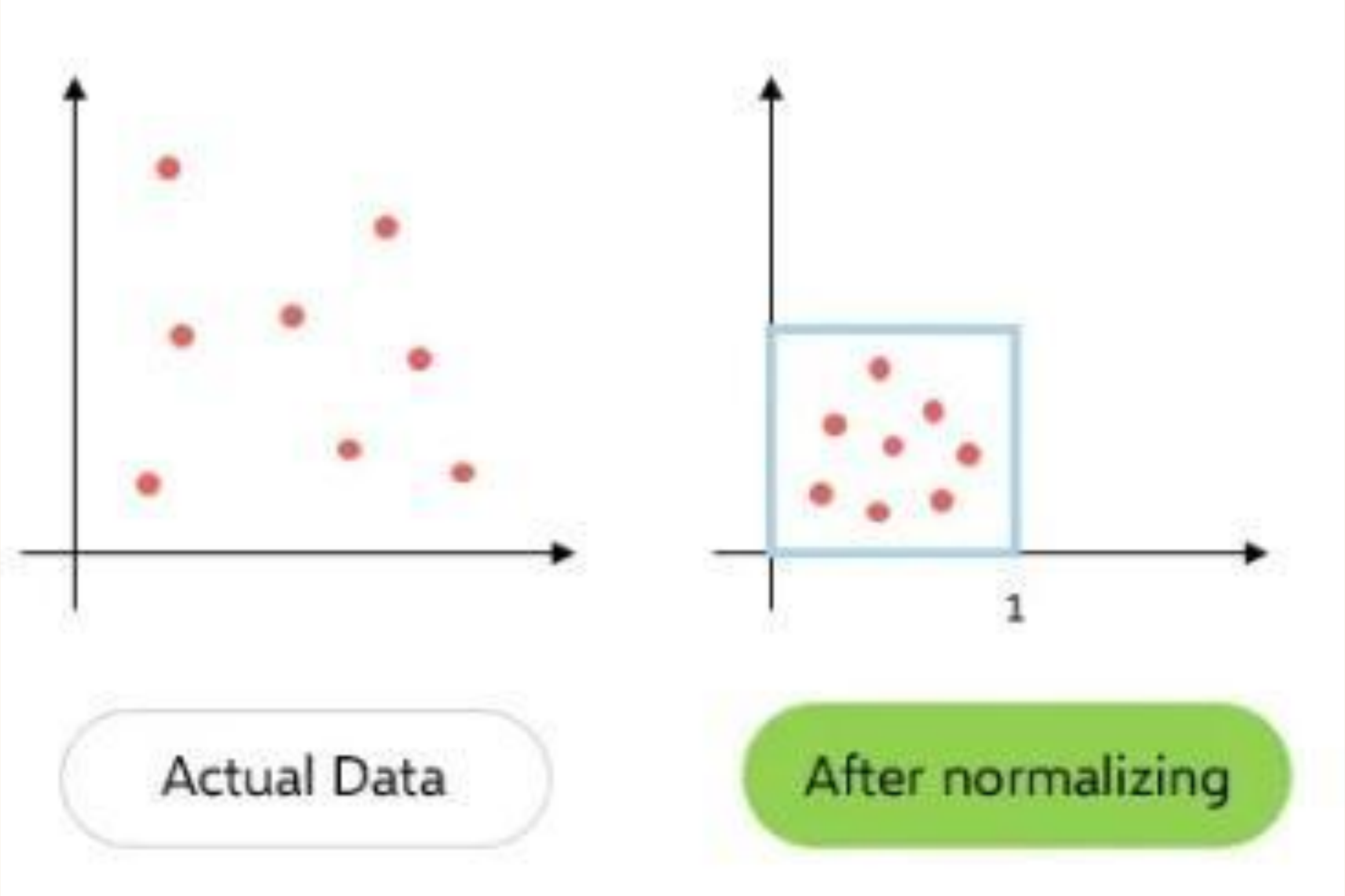


	A	B	C	D	E
1	filepath	xmin	xmax	ymin	ymax
2	./images\N1.xml	1093	1396	645	727
3	./images\N100.xml	134	301	312	350
4	./images\N101.xml	31	139	128	161
5	./images\N102.xml	164	316	216	243
6	./images\N103.xml	813	1067	665	724
7	./images\N104.xml	66	154	166	197
8	./images\N105.xml	360	434	174	195
9	./images\N106.xml	137	262	249	290
10	./images\N107.xml	207	356	174	287
11	./images\N108.xml	184	342	220	257
12	./images\N109.xml	148	239	250	320
13	./images\N11.xml	131	187	130	144
14	./images\N110.xml	183	249	211	227
15	./images\N111.xml	80	239	364	402

Base on xml object's position data to draw rectangle to verify labeled data



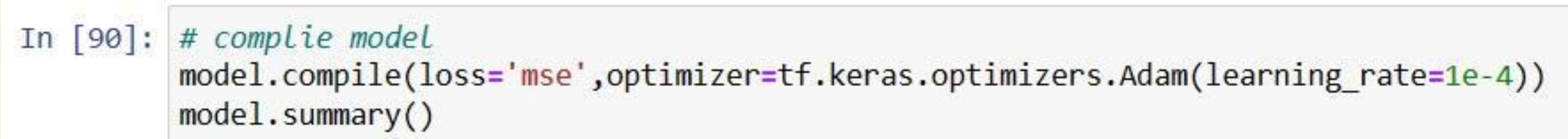
Labeled image's pixel data (0 ~ 255) normalization to (0~1), increase the performance of model training



Deep learning model

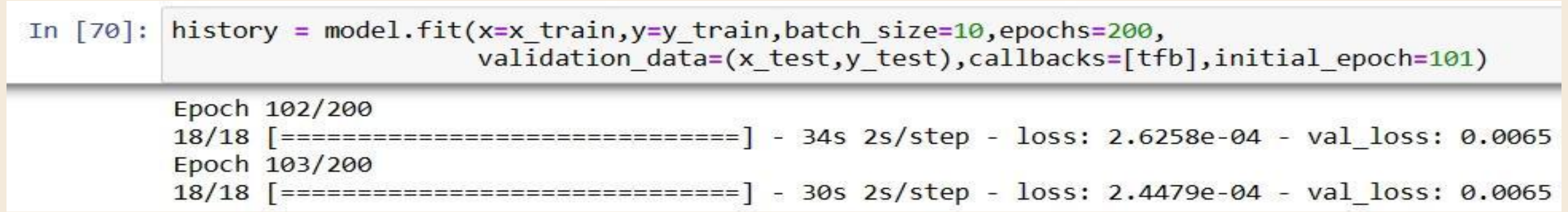
- model setting :
  - ○ Inception ResNet V2 (深度捲積神經網路, 圖像分類、目標檢測、圖像分割)

- model compile :
  - ○ Mean Squared Error (1/N) * Σ (y_true - y_pred) ^ 2)
  - ○ Adaptive Moment Estimation (learning rate越小, 學越久, 但穩定)



```
In [90]: # complie model
         model.compile(loss='mse',optimizer=tf.keras.optimizers.Adam(learning_rate=1e-4))
         model.summary()
```

- model training
  - ○ x_train, y_train is training data (80% original data)
  - ○ batch_size: Number of samples per training
  - ○ epochs: Number of training
  - ○ validation_data: testing model performance's daya (20% original data)
  - ○ callbacks: End of training call tfb board (Visualization training result)
  - ○ init_epoch: start epochs of training

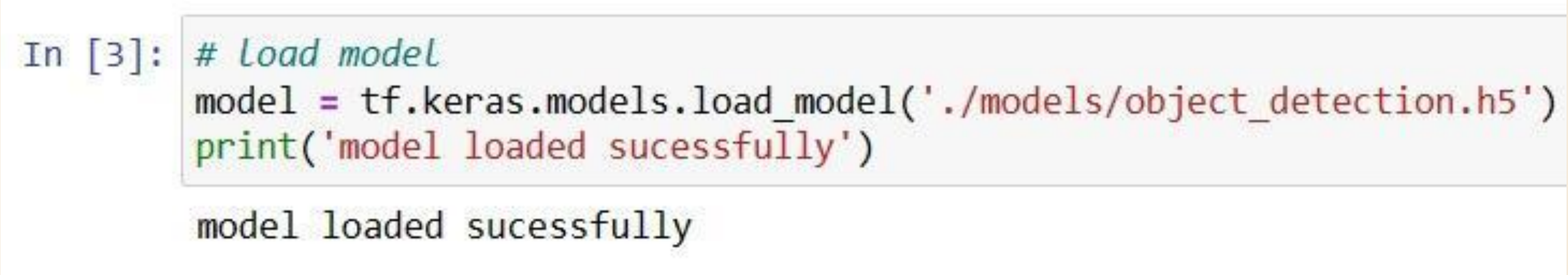


```
In [70]: history = model.fit(x=x_train,y=y_train,batch_size=10,epochs=200,
                             validation_data=(x_test,y_test),callbacks=[tfb],initial_epoch=101)

         Epoch 102/200
         18/18 [==============================] - 34s 2s/step - loss: 2.6258e-04 - val_loss: 0.0065
         Epoch 103/200
         18/18 [==============================] - 30s 2s/step - loss: 2.4479e-04 - val_loss: 0.0065
```

- model save :
  - ○ model.save('./models/object_detection.h5')

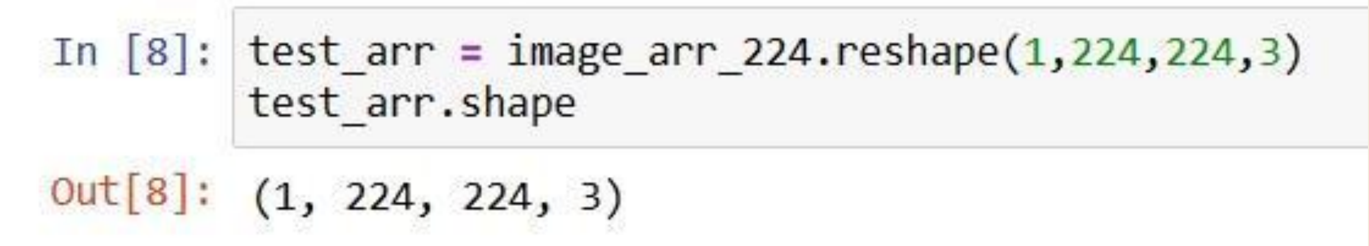
Using model to predict

- load model

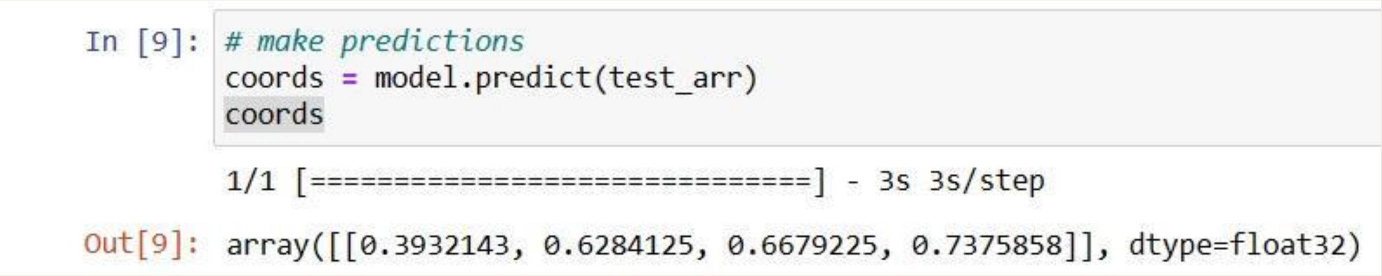


```
In [3]: # load model
        model = tf.keras.models.load_model('./models/object_detection.h5')
        print('model loaded sucessfully')

        model loaded sucessfully
```

- load image
  - convert to pixel datas
  - convert to model-acceptable size
  - normalize size datas
  - convert to model-acceptable format



```
In [8]: test_arr = image_arr_224.reshape(1,224,224,3)
        test_arr.shape

Out[8]: (1, 224, 224, 3)
```

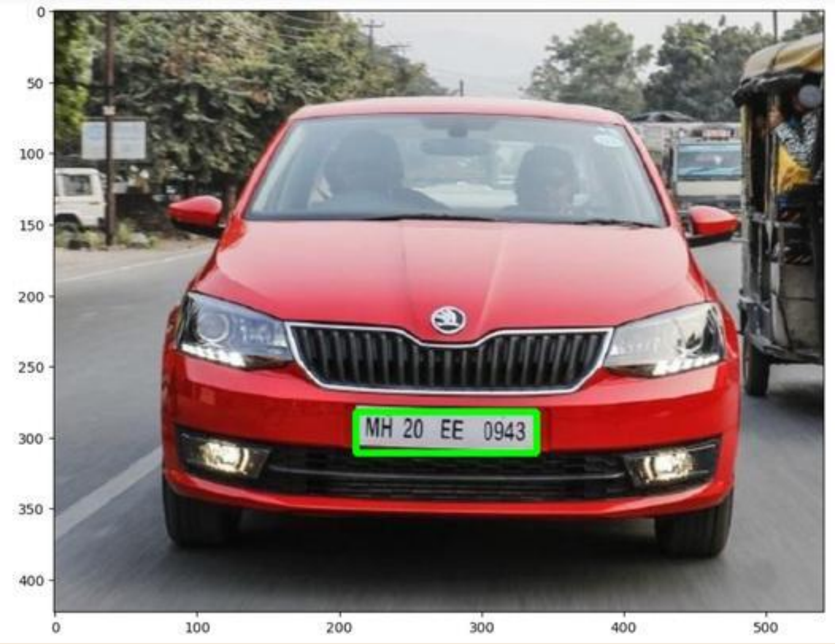


```
In [9]: # make predictions
        coords = model.predict(test_arr)
        coords

        1/1 [==============================] - 3s 3s/step

Out[9]: array([[0.3932143, 0.6284125, 0.6679225, 0.7375858]], dtype=float32)
```

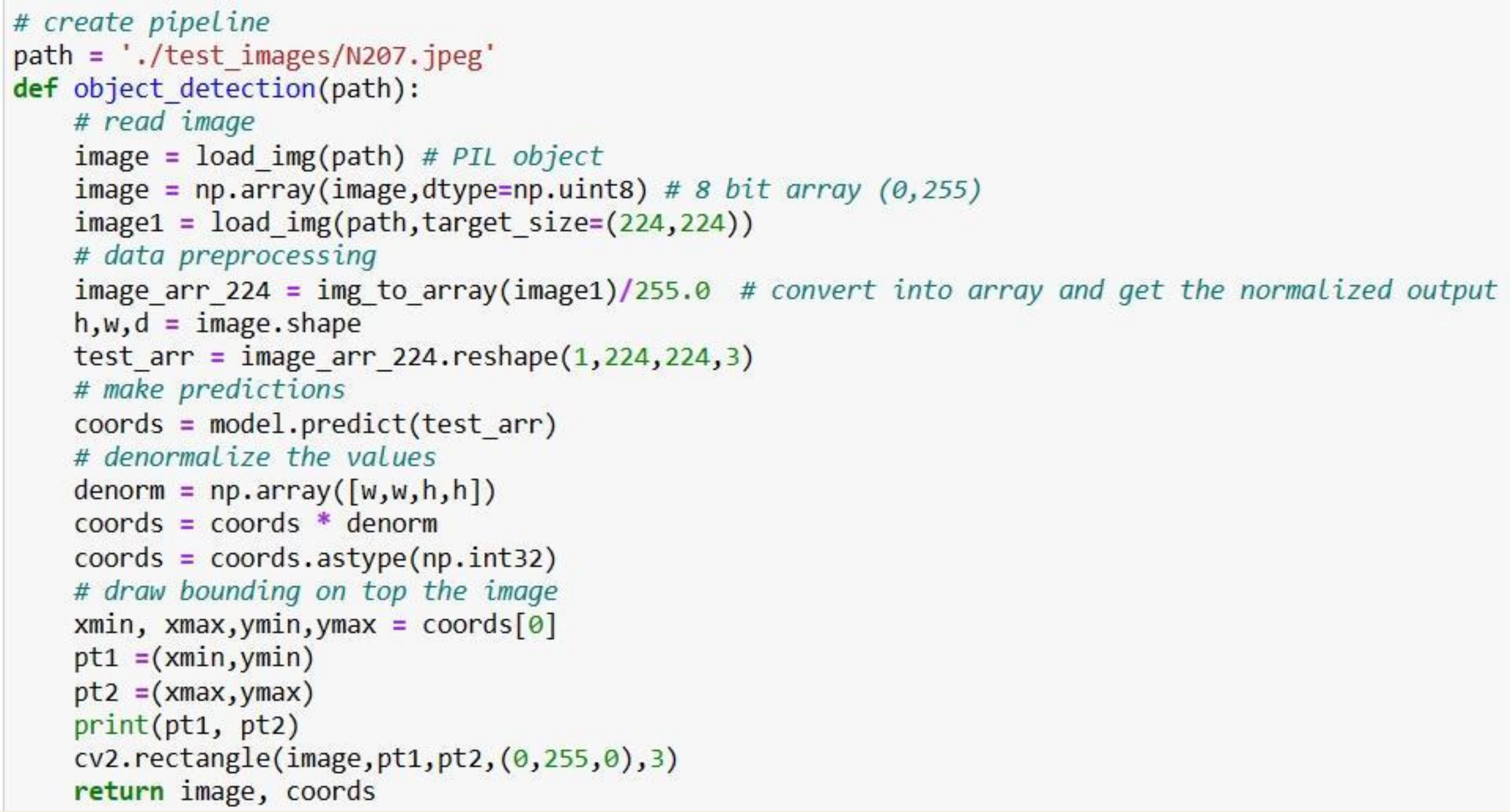
- predict image (return coords)

- denormalize the values to original image size

- draw bounding box on the image

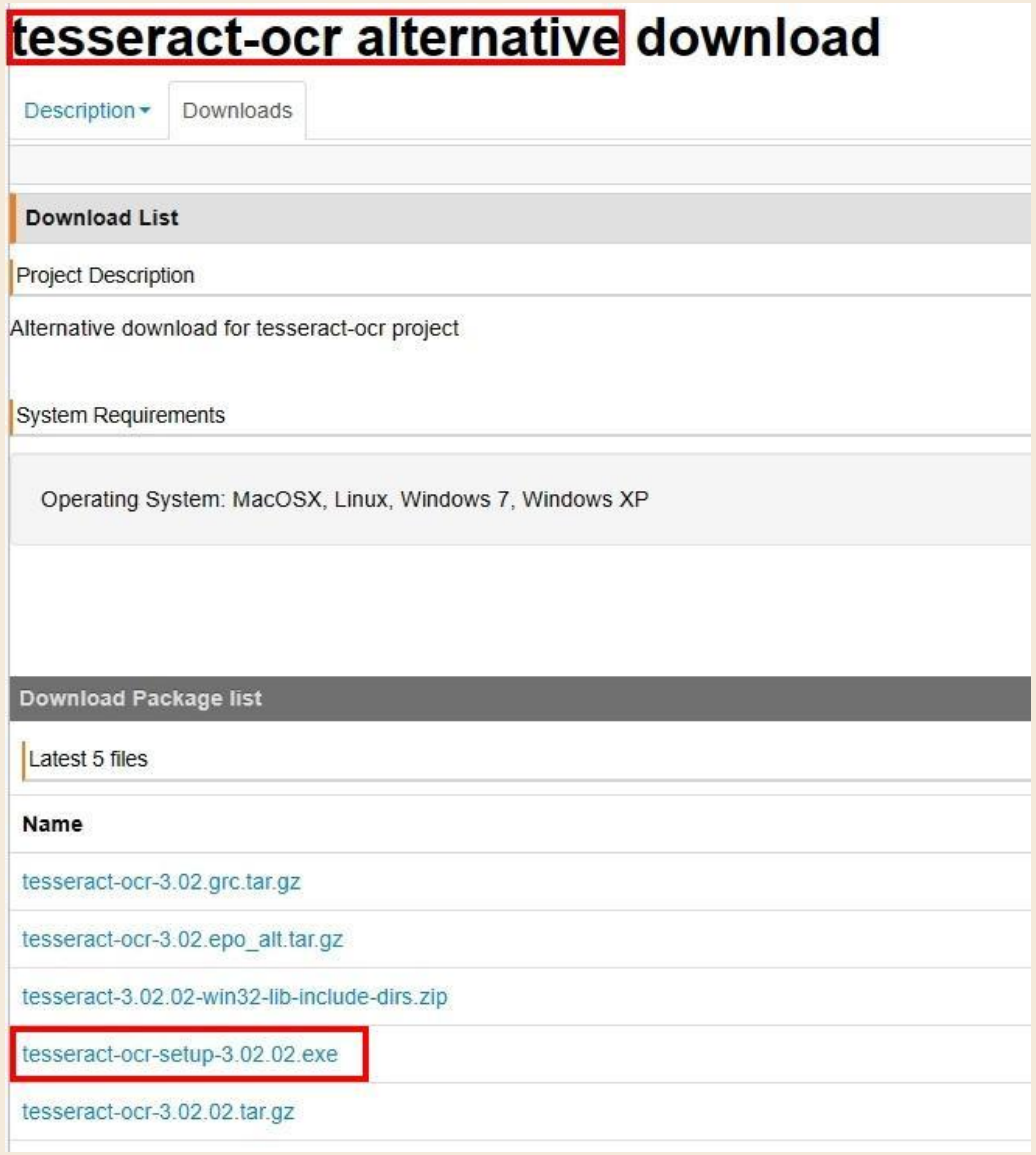


Function about predict number plate



```
# create pipeline
path = './test_images/N207.jpeg'
def object_detection(path):
    # read image
    image = load_img(path) # PIL object
    image = np.array(image,dtype=np.uint8) # 8 bit array (0,255)
    image1 = load_img(path,target_size=(224,224))
    # data preprocessing
    image_arr_224 = img_to_array(image1)/255.0  # convert into array and get the normalized output
    h,w,d = image.shape
    test_arr = image_arr_224.reshape(1,224,224,3)
    # make predictions
    coords = model.predict(test_arr)
    # denormalize the values
    denorm = np.array([w,w,h,h])
    coords = coords * denorm
    coords = coords.astype(np.int32)
    # draw bounding on top the image
    xmin, xmax,ymin,ymax = coords[0]
    pt1 =(xmin,ymin)
    pt2 =(xmax,ymax)
    print(pt1, pt2)
    cv2.rectangle(image,pt1,pt2,(0,255,0),3)
    return image, coords
```

Install tesseract-ocr alternative



tesseract-ocr alternative download

Download List

Project Description

Alternative download for tesseract-ocr project

System Requirements

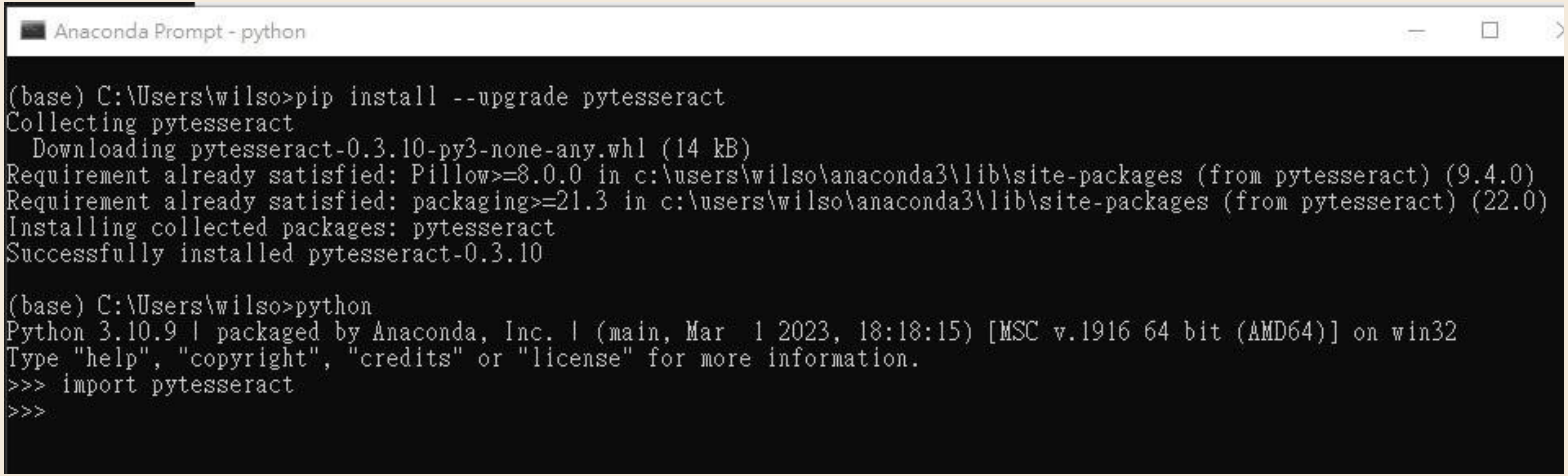
Operating System: MacOSX, Linux, Windows 7, Windows XP

Download Package list

Latest 5 files

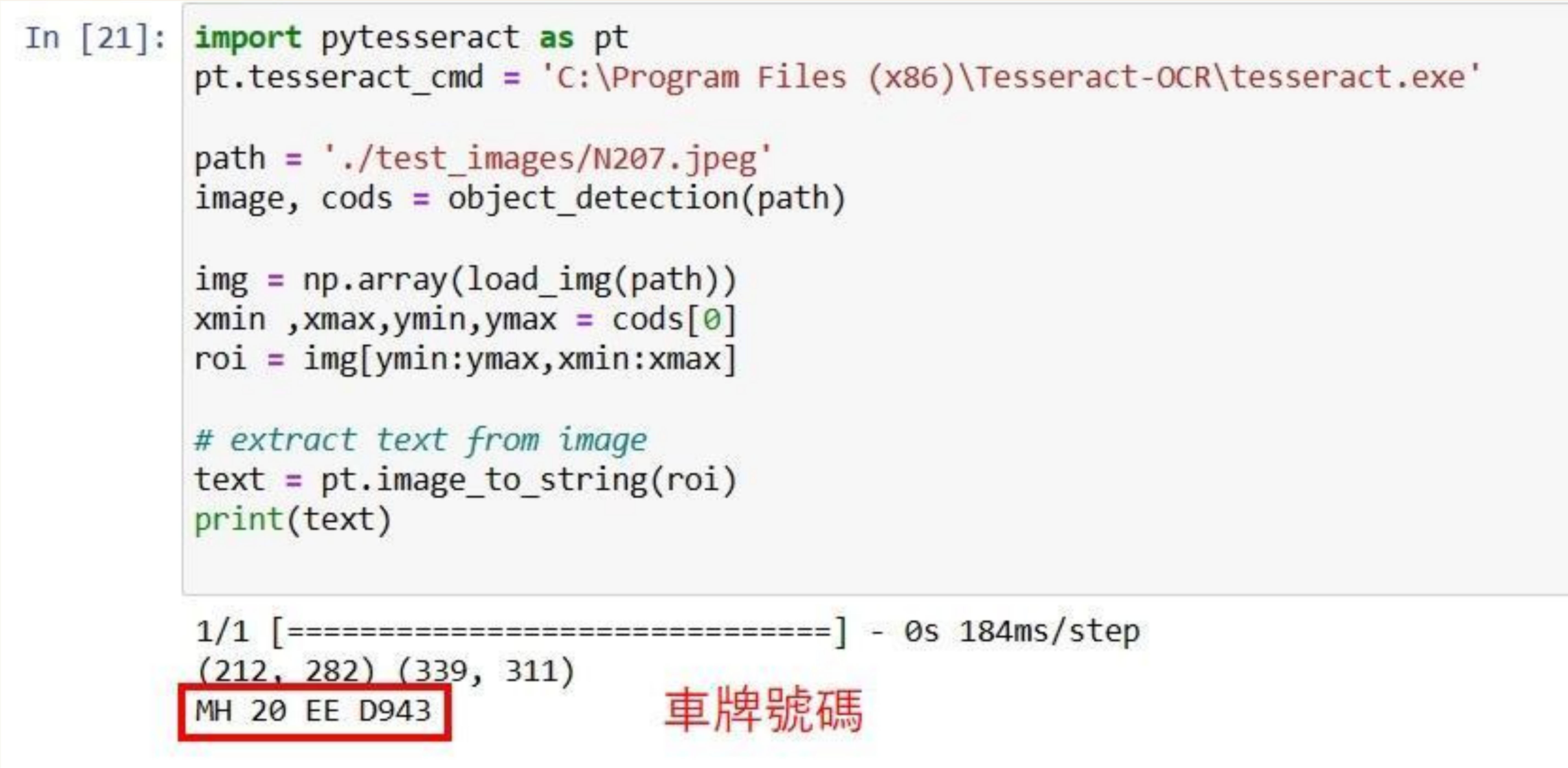
Name
tesseract-ocr-3.02.grc.tar.gz
tesseract-ocr-3.02.epo_alt.tar.gz
tesseract-3.02.02-win32-lib-include-dirs.zip
tesseract-ocr-setup-3.02.02.exe
tesseract-ocr-3.02.02.tar.gz

Install pytesseract



```
(base) C:\Users\wilso>pip install --upgrade pytesseract
Collecting pytesseract
  Downloading pytesseract-0.3.10-py3-none-any.whl (14 kB)
Requirement already satisfied: Pillow>=8.0.0 in c:\users\wilso\anaconda3\lib\site-packages (from pytesseract) (9.4.0)
Requirement already satisfied: packaging>=21.3 in c:\users\wilso\anaconda3\lib\site-packages (from pytesseract) (22.0)
Installing collected packages: pytesseract
Successfully installed pytesseract-0.3.10

(base) C:\Users\wilso>python
Python 3.10.9 | packaged by Anaconda, Inc. | (main, Mar  1 2023, 18:18:15) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import pytesseract
>>>
```

Optical Character Recognition



In [21]:

```
import pytesseract as pt
pt.tesseract_cmd = 'C:\Program Files (x86)\Tesseract-OCR\tesseract.exe'

path = './test_images/N207.jpeg'
image, cods = object_detection(path)

img = np.array(load_img(path))
xmin ,xmax,ymin,ymax = cods[0]
roi = img[ymin:ymax,xmin:xmax]

# extract text from image
text = pt.image_to_string(roi)
print(text)
```

```
1/1 [==============================] - 0s 184ms/step
(212, 282) (339, 311)
MH 20 EE D943
```

車牌號碼

limitation of pytesseract

- 辨識的字體需要在線段上
- 辨識的字體需要是清楚的
- 辨識的字體不要有特效
- 辨識的字體不能是潦草的
- 辨識的字體圖解析度必須大於200dpi，大小至少要300像素

Problem

- pyqt5安裝失敗
  - 先pip uninstall pyqt5，再pip install pyqt5

- Tesseract NotFoundError: tesseract is not installed or it's not in your PATH
  - 安裝tesseract , 加入環境變數, 再重開anaconda

- jupyter檔案想在terminal執行
  - jupyter nbconvert --to script "03-Make Predictions.ipynb"

- model(.h5)檔案太大、images檔案太多, github無法追蹤
  - 只追蹤生成model的python檔案, 分批commit images

- 運行專案時, 需注意python code中的路徑是否match到運行環境

Conclusion

- 這次實作車牌影像辨識, 主要遇到環境問題, 例如缺少了某個套件的安裝, 或是沒設定到環境變數。另外, 訓練出來的成果, 也因樣本的數量, 而有所限制, 因此目前的系統辨識車牌的能力有限, 也就是說, 精準度因樣本少的關係, 而有所下降。因此, 未來可以增加車牌樣本的數量, 並花費更多的時間訓練模型, 進而提升辨識車牌的精準度。