



دانشکده مهندسی برق

تشخیص کاراکترهای پلاک در یک تصویر

گزارش پایانی درس یادگیری عمیق
در رشته مهندسی برق گرایش مخابرات

امیرحسین پورداود

97411279

استاد:

دکتر حدادی

حل تمرین:

مهندس حیدری

تیرماه 1401

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

چکیده

در این پروژه ابتدا به تشخیص پلاک ماشین ها از روی تصاویر پرداخته شده است، سپس کاراکتر های پلاک تشخیص داده شده توسط یک شبکه YOLO تعیین میشود.

واژه های کلیدی: تشخیص پلاک – تشخیص کاراکتر – YOLOv5

فهرست مطالب

| | |
|--|-----------|
| فصل 1: روش تحقیق | 3 |
| 1-1- مقدمه | 4 |
| 1-2- تشخیص پلاک خودرو | 4 |
| 1-2-1- شرح پروژه | 4 |
| 1-2-2- شبکه Yolov5 | 5 |
| 1-2-3- مجموعه داده | 8 |
| 1-3- تشخیص پلاک خودرو(ها) و کاراکترهای آن ها | 10 |
| 1-3-1- شرح پروژه | 11 |
| 1-3-2- شبکه Yolov5 برای کاراکتر | 11 |
| 1-3-3- مجموعه داده | 11 |
| 1-3-4- تشخیص نهایی کاراکتر از روی عکس | 14 |
| فصل 2: نتایج و تفسیر آنها | 16 |
| 2-1- مقدمه | 17 |
| 2-2- تشخیص پلاک خودروها | 17 |
| 2-2-1- دقت شبکه | 17 |
| 2-2-2- خروجی نهایی | 19 |
| 2-3- تشخیص کاراکتر پلاک | 21 |
| 2-3-1- دقت شبکه | 21 |
| 2-3-2- خروجی نهایی | 21 |
| 2-4- تشخیص کاراکتر از روی عکس | 23 |
| 2-4-1- خروجی نهایی | 23 |
| مراجع | 26 |

فصل 1:

روش تحقیق

1-1- مقدمه

در این فصل به نحوه انجام تشخیص ها و ابزار ها و کدها پرداخته شده است.

- در این پروژه میبایست پلاک خودرو و همچنین کاراکترهای آن را به تفکیک در تصاویر و ویدئوها با استفاده از دیتاستی که در اختیار شما قرار خواهد گرفت، تشخیص دهید. انتخاب الگوریتم پیاده سازی کاملاً اختیاری و انتخاب شما خواهد بود اما توصیه میشود از الگوریتم یولو مطابق آموزش های کلاس استفاده کنید.

1-2- تشخیص پلاک خودرو

در این بخش شما باید شبکه ای را طراحی کنید که یک تصویر در ورودی گرفته و پلاک خودرو های احتمالی موجود در تصویر را تشخیص دهد. خروجی مورد انتظار مطابق تصویر زیر است.



1-2-1- شرح پروژه

در این قسمت با توجه به صورت پروژه میبایست از روی تصویر، پلاک خودروی موجود تشخیص داده شود.

برای اینکار از یک شبکه YOLO استفاده شده است و با یک مجموعه تصاویر ماشین دارای پلاک نیز تمرین داده شده است که در ادامه بصورت جزئی تر این شبکه و دیتاست مربوطه شرح داده میشود.

1-2-2- شبکه Yolov5

انسان با نگاهی کوتاه به تصویر بلافاصله می‌فهمد چه اشیا یی در تصویر وجود دارند، موقعیت‌شان در تصویر کجاست و حتی چه ارتباطی با هم دارند. این عمل‌ها برای انسان بسیار ساده هست و سیستم بینایی دقیق و سریع انسان کارهای به مراتب پیچیده‌تری مانند رانندگی را می‌تواند به آسانی انجام دهد. البته، بخش مهمی از رانندگی، شناسایی و موقعیت‌یابی اشیا اطراف خودرو هست که انسان در این زمینه مهارت بالایی دارد. حال، اگر الگوریتم‌های سریع و دقیقی برای شناسایی و موقعیت‌یابی اشیا داشته باشیم، می‌توان امیدوار بود که ماشین‌های خودران بدون نیاز به سنسورهای مخصوص داشته باشیم. شناسایی و موقعیت‌یابی اشیا از جمله زمینه‌های تحقیقاتی قدیمی و مهم در بینایی کامپیوتر است. در بینایی کامپیوتر، به شناسایی و موقعیت‌یابی اشیا در تصویر **Object Detection** گفته می‌شود. معمولاً در فارسی بجای عبارت شناسایی و موقعیت‌یابی اشیا از عبارت “تشخیص اشیا” استفاده می‌شود. در این پروژه، از یکی از سیستم‌های سریع و دقیق تشخیص اشیا به نام YOLO استفاده شده است.

بنابراین، YOLO چیست؟ یک سیستم تشخیص اشیا است. تصویری از خروجی سیستم تشخیص

اشیا را در زیر مشاهده می‌کنید. تشخیص اشیا = شناسایی اشیا + موقعیت‌یابی اشیا



شکل ۱: نمونه تصویر خروجی یک سیستم تشخیص اشیا؛ موقعیت کادرها نشان‌دهنده بخش موقعیت‌یابی و نام اشیا هم نشان‌دهنده بخش شناسایی است.

YOLO مخفف عبارت You Only Look Once ، به معنای “ شما فقط یک بار به تصویر نگاه می کنید ” هست. در واقع، این عبارت به همان قابلیت سیستم بینایی انسان اشاره دارد که با یک نگاه عمل تشخیص اشیا را انجام می دهد. بنابراین، سیستم تشخیص اشیا YOLO با هدف ارائه روشی مشابه کارکرد سیستم بینایی انسان طراحی شده است .

در این پروژه از شبکه برای تمرین دادن شبکه yolov5 بصورت زیر عمل شده است :

- ابتدا مخزن کد ها و مدل های یولو برای تمرین دانلود میشود

Download yolov5 models

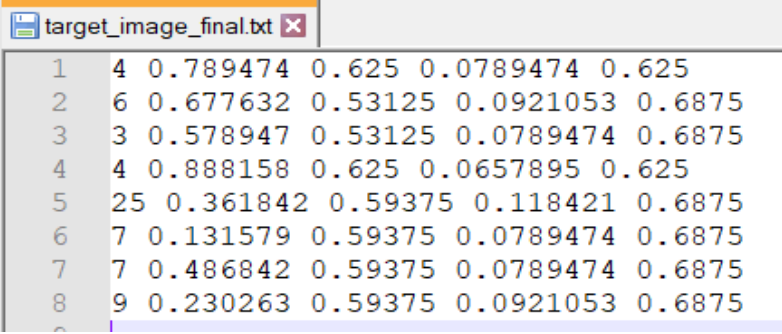
```
!git clone https://github.com/ultralytics/yolov5
%cd yolov5
!pip install -qr requirements.txt

from yolov5 import utils
display = utils.notebook_init()

YOLOv5 v6.1-266-g34df503 Python-3.7.13 torch-1.11.0+cu113 CUDA:0 (Tesla T4, 15110MiB)

Setup complete (2 CPUs, 12.7 GB RAM, 39.1/78.2 GB disk)
```

- بعد از دانلود مدل های از قبل تمرین شده، برای تطبیق دادن این مدل با تشخیص شی های مورد نظر باید یک دیتاست برای شی مورد نظر دانلود یا ایجاد کرد و سپس روی آن پردازش انجام داد و label های آن را متناسب با فرمت یولو تغییر داد و برای تمرین به شبکه داد.
- فرمت لیبل شبکه یولو باید بصورت مختصات محور افقی و عمودی مرکز شی مورد نظر و طول و عرض آن شی باشد.
- لیبل های شی های مختلف در یک عکس باید در یک فایل txt و هر شی در یک خط بصورت زیر باشد که اول لیبل سپس 4 عدد گفته شده در بالا نوشته میشود.



| Index | Class ID | x1 | y1 | x2 | y2 |
|-------|----------|----------|---------|-----------|--------|
| 1 | 4 | 0.789474 | 0.625 | 0.0789474 | 0.625 |
| 2 | 6 | 0.677632 | 0.53125 | 0.0921053 | 0.6875 |
| 3 | 3 | 0.578947 | 0.53125 | 0.0789474 | 0.6875 |
| 4 | 4 | 0.888158 | 0.625 | 0.0657895 | 0.625 |
| 5 | 25 | 0.361842 | 0.59375 | 0.118421 | 0.6875 |
| 6 | 7 | 0.131579 | 0.59375 | 0.0789474 | 0.6875 |
| 7 | 7 | 0.486842 | 0.59375 | 0.0789474 | 0.6875 |
| 8 | 9 | 0.230263 | 0.59375 | 0.0921053 | 0.6875 |

- سپس باید مکان تصاویر train – test – val که دارای دو پوشه images و labels هستند به همراه تعداد لیبل ها و نام کلاس ها در یک فایل yml نوشته شود و به هنگام train به شبکه داده شود تا عملیات تمرین خودکار صورت گیرد.


```

!echo "train: Dataset/train/images" > data/plate-detection.yaml
!echo "val: Dataset/val/images" >> data/plate-detection.yaml

!echo "nc: 1" >> data/plate-detection.yaml
!echo "names: ['license']" >> data/plate-detection.yaml

!cat data/plate-detection.yaml

train: Dataset/train/images
val: Dataset/val/images
nc: 1
names: ['license']

```

- در مرحله آخر پس از پیش پردازش ها بر روی لیبل ها و تصاویر با دستور زیر تعداد epoch و batch را مشخص میکنیم و آدرس فایل yaml. که نشان دهنده مسیر داده های تمرینی و کلاس ها است را در دستور مینویسیم و پس از اجرا صبر میکنیم که یادگیری شبکه مورد نظر به پایان برسد.

```

In [12]: !python train.py --img 416 --batch 16 --epochs 50 --data data/plate-detection.yaml --cfg models/yolov5m.yaml

train: weights=yolov5s.pt, cfg=models/yolov5m.yaml, data=data/plate-detection.yaml, hyp=data/hyps/hyp.scratch-low.yaml, epochs=
50, batch_size=16, imgs=416, rect=False, resume=False, nosave=False, noval=False, noautoanchor=False, noplots=False, evolve=No
ne, bucket=, cache=None, image_weights=False, device=, multi_scale=False, single_cls=False, optimizer=SGD, sync_bn=False, worke
rs=8, project=runs/train, name=exp, exist_ok=False, quad=False, cos_lr=False, label_smoothing=0.0, patience=100, freeze=[0], sa
ve_period=-1, local_rank=-1, entity=None, upload_dataset=False, bbox_interval=-1, artifact_alias=latest
github: up to date with https://github.com/ultralytics/yolov5
YOLOv5 v6.1-266-g34df503 Python-3.7.13 torch-1.11.0+cu113 CUDA:0 (Tesla T4, 15110MiB)

hyperparameters: lr=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_l
r=0.1, box=0.05, cls=0.5, cls_pw=1.0, obj=1.0, obj_pw=1.0, iou_t=0.2, anchor_t=4.0, fl_gamma=0.0, hsv_h=0.015, hsv_s=0.7, hsv_v
=0.4, degrees=0.0, translate=0.1, scale=0.5, shear=0.0, perspective=0.0, flipud=0.0, fliplr=0.5, mosaic=1.0, mixup=0.0, copy_pa
ste=0.0
Weights & Biases: run 'pip install wandb' to automatically track and visualize YOLOv5 runs (RECOMMENDED)
TensorBoard: Start with 'tensorboard --logdir runs/train', view at http://localhost:6006/
Downloading https://ultralytics.com/assets/Arial.ttf to /root/.config/Ultralytics/Arial.ttf...
100% 755k/755k [00:00<00:00, 155MB/s]
Downloading https://github.com/ultralytics/yolov5/releases/download/v6.1/yolov5s.pt to yolov5s.pt...
100% 14.1M/14.1M [00:04<00:00, 3.47MB/s]

Overriding model.yaml nc=80 with nc=1

      from  n  params module                        arguments
      -1  1    5280  models.common.Conv [3, 48, 6, 2, 2]
      1  -1  1    41664 models.common.Conv [48, 96, 3, 2]
      2  -1  2    65280 models.common.C3 [96, 96, 2]
      3  -1  1    166272 models.common.Conv [96, 192, 3, 2]
      4  -1  4    444672 models.common.C3 [192, 192, 4]
      5  -1  1    664320 models.common.Conv [192, 384, 3, 2]
      6  -1  6    2512896 models.common.C3 [384, 384, 6]
      7  -1  1    2655744 models.common.Conv [384, 768, 3, 2]
      8  -1  2    4134912 models.common.C3 [768, 768, 2]
      9  -1  1    1476864 models.common.SPPF [768, 768, 5]
     10  -1  1    295680 models.common.Conv [768, 384, 1, 1]
     11  -1  1         0 torch.nn.modules.upsampling.Upsample [None, 2, 'nearest']
     12  [-1, 6] 1         0 models.common.Concat [1]
     13  -1  2    1182720 models.common.C3 [768, 384, 2, False]
     14  -1  1    74112 models.common.Conv [384, 192, 1, 1]
     15  -1  1         0 torch.nn.modules.upsampling.Upsample [None, 2, 'nearest']
     16  [-1, 4] 1         0 models.common.Concat [1]
     17  -1  2    296448 models.common.C3 [384, 192, 2, False]
     18  -1  1    332160 models.common.Conv [192, 192, 3, 2]
     19  [-1, 14] 1         0 models.common.Concat [1]
     20  -1  2    1035264 models.common.C3 [384, 384, 2, False]
     21  -1  1    1327872 models.common.Conv [384, 384, 3, 2]
     22  [-1, 10] 1         0 models.common.Concat [1]
     23  -1  2    4134912 models.common.C3 [768, 768, 2, False]
     24  [17, 20, 23] 1    24246 models.yolo.Detect [1, [[10, 13, 16, 30, 33, 23], [30, 61, 62, 45, 59,
119], [116, 90, 156, 198, 373, 326]], [192, 384, 768]]
YOLOv5m summary: 369 layers, 20871318 parameters, 20871318 gradients

Transferred 57/481 items from yolov5s.pt
AMP: checks passed
Scaled weight_decay = 0.0005
optimizer: SGD with parameter groups 79 weight (no decay), 82 weight, 82 bias
albumentations: version 1.0.3 required by YOLOv5, but version 0.1.12 is currently installed
train: Scanning '/content/yolov5/Dataset/train/labels' images and labels...433 found, 0 missing, 0 empty, 0 corrupt: 100% 433/4
33 [00:00<00:00, 756.30it/s]

```

- پس از آن بهترین مدل یادگیری شده (best.pt) ذخیره شده و میتوان از آن برای تست و تشخیص تصاویر دیگر استفاده کرد.

3-2-1- مجموعه داده

برای انجام تسک تشخیص پلاک از دیتاست زیر که برای سایت Kaggle است استفاده کردیم که شامل 343 عکس ماشین با پلاک میباشد.

Car License Plate Detection

433 images of license plates



Data Code (39) Discussion (1) Metadata

About Dataset



Usability ⓘ

8.75

License

CC0: Public Domain

Expected update frequency

Never

About this dataset

This dataset contains 433 images with bounding box annotations of the car license plates within the image. Annotations are provided in the PASCAL VOC format.

در این دیتاست لیبل ها که در یک فایل xml هستند شامل کادر پلاک، که مختصات گوشه بالا سمت چپ و گوشه پایین سمت راست و همچنین عرض و طول کادر در آن مشخص شده است.

Cars1.xml (560 B)

```
<annotation>
  <folder>images</folder>
  <filename>Cars1.png</filename>
  <size>
    <width>400</width>
    <height>248</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>licence</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <occluded>0</occluded>
    <difficult>0</difficult>
    <bndbox>
      <xmin>134</xmin>
      <ymin>128</ymin>
      <xmax>262</xmax>
      <ymax>160</ymax>
    </bndbox>
  </object>
</annotation>
```

برای یادگیری شبکه یولو نیاز است که پیش پردازش بر روی این لیبل ها انجام گیرد و فرمت آن با توجه به چیزی که در بخش قبل گفته شد انجام شود که برای اینکار تابعی بصورت زیر نوشته شده که مختصات گوشه ها را به چهار نقطه: مختصات مرکزی و طول و عرض تبدیل کند و به همراه لیبل در هر خط یک فایل txt بنویسد.

```
def cord_converter(size, box):
    """
    convert xml annotation to darknet format coordinates
    :param size: [w,h]
    :param box: anchor box coordinates [upper-left x,uppler-left y,lower-right x, lower-right y]
    :return: converted [x,y,w,h]
    """
    x1 = int(box[0])
    y1 = int(box[1])
    x2 = int(box[2])
    y2 = int(box[3])

    dw = np.float32(1. / int(size[0]))
    dh = np.float32(1. / int(size[1]))

    w = x2 - x1
    h = y2 - y1
    x = x1 + (w / 2)
    y = y1 + (h / 2)

    x = x * dw
    w = w * dw
    y = y * dh
    h = h * dh
    return [x, y, w, h]
```

تابع زیر پارامتر های xml را میخواند و فایل txt را مینویسد:

```
def get_xml_data(file_path, img_xml_file):
    img_path = file_path + '/' + img_xml_file + '.xml'
    print(img_path)

    dom = parse(img_path)
    root = dom.documentElement
    img_name = root.getElementsByTagName("filename")[0].childNodes[0].data
    img_size = root.getElementsByTagName("size")[0]
    objects = root.getElementsByTagName("object")
    img_w = img_size.getElementsByTagName("width")[0].childNodes[0].data
    img_h = img_size.getElementsByTagName("height")[0].childNodes[0].data
    img_c = img_size.getElementsByTagName("depth")[0].childNodes[0].data
    # print("img_name:", img_name)
    # print("image_info:(w,h,c)", img_w, img_h, img_c)
    img_box = []
    for box in objects:
        cls_name = box.getElementsByTagName("name")[0].childNodes[0].data
        x1 = int(box.getElementsByTagName("xmin")[0].childNodes[0].data)
        y1 = int(box.getElementsByTagName("ymin")[0].childNodes[0].data)
        x2 = int(box.getElementsByTagName("xmax")[0].childNodes[0].data)
        y2 = int(box.getElementsByTagName("ymax")[0].childNodes[0].data)
        print("box:(c,xmin,ymin,xmax,ymax)", cls_name, x1, y1, x2, y2)
        img_jpg_file_name = img_xml_file + '.jpg'
        img_box.append([cls_name, x1, y1, x2, y2])
    # print(img_box)
    # test_dataset_box_feature(img_jpg_file_name, img_box)
    save_file(img_xml_file, [img_w, img_h], img_box)

def save_file(img_jpg_file_name, size, img_box):
    classes = ['license']
    save_file_name = DATA_ROOT + DEST_LABELS_PATH + '/' + img_jpg_file_name + '.txt'
    print(save_file_name)
    file_path = open(save_file_name, "a+")
    for box in img_box:
        #cls_num = classes.index(box[0]) # find class_id
        cls_num = 0
        new_box = cord_converter(size, box[1:]) # convert box coord into YOLO x,y,w,h

        file_path.write(f"{cls_num} {new_box[0]} {new_box[1]} {new_box[2]} {new_box[3]}\n")

    file_path.flush()
    file_path.close()
```

1-3- تشخیص پلاک خودرو(ها) و کاراکترهای آن ها

در این بخش شما بایستی الگوریتم طراحی شده در بخش قبل را توسعه دهید، به این صورت که ورودی شبکه شما تصویری شامل خودروهایی خواهد بود و خروجی مورد انتظار پلاک های تشخیص داده شده به همراه کاراکترهای هر پلاک است. (برای این بخش میتوانید از دیتاست پیوست شده استفاده کنید. دقت کنید که فرمت لیبل های دیتاست پیوست شده با فرمت مورد نیاز یولو متفاوت است. در این بخش میبایست مطابق مطالب گفته شده در کلاس لیبل هارا به فرمت یولو تبدیل کنید)

خروجی مورد انتظار مطابق تصویر زیر است.



1-3-1- شرح پروژه

در این قسمت با توجه به صورت پروژه میبایست از روی تصویر، پلاک خودروی موجود را تشخیص داده و سپس کاراکترهای درون پلاک مشخص شده از قسمت قبل نیز مشخص مشخص شود. برای اینکار از دو شبکه YOLO یکی برای تشخیص پلاک که در بخش قبل آماده شد و دیگری برای تشخیص حروف استفاده میکنیم و با یک مجموعه تصاویر پلاک فارسی آن را تمرین داده میشود که در ادامه بصورت جزئی تر این شبکه و دیتاست مربوطه شرح داده میشود.

1-3-2- شبکه Yolov5 برای کاراکتر

در بخش 1-2-2 بطور جزئی شبکه yolo و نحوه کارکرد آن مشخص شده است بنابراین در این قسمت فقط به شبکه نهایی ساخته شده، پرداخته خواهد شد. در این قسمت یک شبکه یولو دیگر با دیتاست پلاک ها با لیبل کاراکترهای آن train میکنیم که با دریافت عکس پلاک، کاراکترهای آن که شامل حروف و عدد میباشد را تشخیص دهد. در این شبکه 35 کلاس وجود دارد که شامل اعداد 0 تا 9 و حروف های فارسی و S , D و معلول میباشد. بنابراین بدلیل وجود تعداد زیاد کلاس ها به دیتای بیشتری نسبت به شبکه قبل داریم.

1-3-3- مجموعه داده

برای انجام تسک تشخیص کاراکتر پلاک از دیتاست زیر که شامل 5000 عکس پلاک میباشد استفاده کردیم.



در این دیتاست لیبل ها که در یک فایل json هستند شامل کادر پلاک، که مختصات x , y و همچنین عرض و طول کادر در آن مشخص شده است و همچنین کاراکتر انگلیسی و فارسی و آیدی آن ها بصورت زیر می باشد.

```
[
  {
    "char_en": "b",
    "char_fa": "ب",
    "char_id": 11,
    "height": 76,
    "width": 83,
    "x": 202,
    "y": 86
  },
  {
    "char_en": "8",
    "char_fa": "8",
    "char_id": 8,
    "height": 76,
    "width": 39,
    "x": 337,
    "y": 77
  },
  {
    "char_en": "6",
    "char_fa": "6",
    "char_id": 6,
    "height": 72,
    "width": 46,
    "x": 383,
    "y": 74
  },
  {
    "char_en": "6",
    "char_fa": "6",
    "char_id": 6,
    "height": 72,
    "width": 38,
    "x": 448,
    "y": 77
  },
]
```

برای یادگیری شبکه یولو مورد نظر نیاز است که پیش پردازش بر روی این لیبل ها انجام گیرد و فرمت آن با توجه به چیزی که در بخش قبل گفته شد انجام شود که برای اینکار تابعی بصورت زیر نوشته شده که

مختصات x, y را به چهار نقطه: مختصات مرکزی و طول و عرض تبدیل کند و به همراه لیبل در هر خط یک فایل txt بنویسد.

```
def save_yolo_format(image_file, json_file, txt_file):
    with open(json_file) as f:
        chars = json.loads(f.read())

    bgr = cv2.imread(image_file)

    result = ""
    for char in chars:
        x_center = (char["x"] + char["width"] // 2) / bgr.shape[1]
        y_center = (char["y"] + char["height"] // 2) / bgr.shape[0]

        width = char["width"] / bgr.shape[1]
        height = char["height"] / bgr.shape[0]

        label = char["char_id"]

        result += str(label) + " " + str(x_center) + " " + str(y_center) + " " + str(width) + " " + str(height) + "\n"

    with open(txt_file, "w") as f:
        f.write(result)

for image in glob.glob('Dataset/train/images/*.jpg'):
    num = image.split('/')[1].split('.')[0]
    save_yolo_format(image,
                     "/content/plate-dataset-char/labels/"+ num + ".json",
                     "Dataset/train/labels/"+ num + ".txt")
```

تعداد لیبل ها :

```
id = []
for json_file in glob.glob('/content/plate-dataset-char/labels/*.json'):
    with open(json_file) as f:
        chars = json.loads(f.read())
        for char in chars:
            label = char["char_id"]
            id.append(label)
set(id)

{0,
1,
2,
3,
4,
5,
6,
7,
8,
9,
10,
11,
12,
13,
14,
15,
16,
17,
18,
19,
20,
21,
22,
23,
24,
25,
26,
27,
28,
29,
31,
32,
33,
34,
35,
36}
```

4-3-1- تشخیص نهایی کاراکتر از روی عکس

تا اینجای کار دو شبکه برای تشخیص پلاک از روی تصویر ماشین و تشخیص کاراکتر از روی عکس پلاک جداگانه یادگیری شده است، حال باید این دو شبکه را به یکدیگر بصورت سری متصل کنیم تا خروجی تشخیص پلاک به شبکه دوم وارد شود و کاراکتر های پلاک با شبکه دوم تشخیص داده شود. برای متصل کردن این دو شبکه ابتدا تصویر مورد نظر را به شبکه اول می‌دهیم تا پلاک خودرو مشخص شود و کادر آن ذخیره شود سپس با توجه به لیبل خروجی که مختصات کادر پلاک را نشان می‌دهد، تصویر پلاک را بدست آورده و جدا می‌کنیم. حال با توجه به اینکه یک تصویر پلاک داریم، این پلاک را به شبکه دوم می‌دهیم که کاراکتر های آن مشخص شود و در نتیجه مقادیر مورد نظر که همان کاراکتر های پلاک ماشین است بدست می‌آید.

```
plate_char = ['0',
              '1',
              '2',
              '3',
              '4',
              '5',
              '6',
              '7',
              '8',
              '9',
              'الف',
              'ب',
              'ج',
              'ل',
              'م',
              'ن',
              'ق',
              'و',
              'ه',
              'ی',
              'د',
              'س',
              'ص',
              'مطلوب',
              'ت',
              'ط',
              'ع',
              'D',
              'S',
              'پ',
              '30',
              'ت',
              'ز',
              'ش',
              'ف',
              'ک',
              'گ']
```


تشخیص پلاک با شبکه اول :

```
!python detect.py --source runs/target_image.jpg --conf 0.4 --weights runs/plate-detection-weights.pt --save-txt --name target_image
```

```
detect: weights=['runs/plate-detection-weights.pt'], source=runs/target_image.jpg, data=data/coco128.yaml, imgsz=[640, 640], conf_thres=0.4, iou_thres=0.45, max_det=1000, device=, view_img=False, save_txt=True, save_conf=False, save_crop=False, nosave=False, classes=None, agnostic_nms=False, augment=False, visualize=False, update=False, project=runs/detect, name=target_image, exist_ok=False, line_thickness=3, hide_labels=False, hide_conf=False, half=False, dnn=False
fatal: cannot change to 'D:\learning\EE': No such file or directory
YOLOv5 2022-6-28 Python-3.9.6 torch-1.9.0+cu111 CUDA:0 (NVIDIA GeForce GTX 1660 Ti, 6144MiB)

Fusing layers...
YOLOv5m summary: 290 layers, 20852934 parameters, 0 gradients
image 1/1 D:\learning\EE Courses\Deep learning (Dr.Hadadi)\TA\Exercise\Project\Final\yolov5\runs\target_image.jpg: 448x640 1 license, Done. (0.026s)
Speed: 0.0ms pre-process, 26.0ms inference, 4.0ms NMS per image at shape (1, 3, 640, 640)
Results saved to runs\detect\target_image
1 labels saved to runs\detect\target_image\labels
```

تشخیص کاراکترهای پلاک با شبکه دوم پس از بدست آمدن تصویر پلاک :

```
!python detect.py --source runs/target_image_final.jpg --conf 0.4 --weights runs/plate-char-detection-weights.pt --save-txt --name target_image_final
```

```
detect: weights=['runs/plate-char-detection-weights.pt'], source=runs/target_image_final.jpg, data=data/coco128.yaml, imgsz=[640, 640], conf_thres=0.4, iou_thres=0.45, max_det=1000, device=, view_img=False, save_txt=True, save_conf=False, save_crop=False, nosave=False, classes=None, agnostic_nms=False, augment=False, visualize=False, update=False, project=runs/detect, name=target_image_final, exist_ok=False, line_thickness=3, hide_labels=False, hide_conf=True, half=False, dnn=False
fatal: cannot change to 'D:\learning\EE': No such file or directory
YOLOv5 2022-6-28 Python-3.9.6 torch-1.9.0+cu111 CUDA:0 (NVIDIA GeForce GTX 1660 Ti, 6144MiB)

Fusing layers...
YOLOv5m summary: 290 layers, 20998410 parameters, 0 gradients
image 1/1 D:\learning\EE Courses\Deep learning (Dr.Hadadi)\TA\Exercise\Project\Final\yolov5\runs\target_image_final.jpg: 160x64 0 1 3, 2 4s, 1 6, 2 7s, 1 9, 1 25, Done. (0.016s)
Speed: 0.0ms pre-process, 16.0ms inference, 4.0ms NMS per image at shape (1, 3, 640, 640)
Results saved to runs\detect\target_image_final
1 labels saved to runs\detect\target_image_final\labels
```

I. باید توجه کرد که چون خروجی شبکه یولو مختصات مرکز پلاک و طول و عرض آن است، برای جدا کردن پلاک از کد زیر برای بدست آوردن مختصات گوشه بالا و پایین برای کراپ کردن و سیو کردن تصویر پلاک استفاده شده است :

```
from PIL import Image

with open(first_label) as f:
    char = f.read()

char = [list(map(float, i.strip().split(' '))) for i in char.strip().split('\n')]
for l in char:
    label, x_center, y_center, width, height = l
    img = Image.open(image_path)
    w_pic, h_pic = img.size

    w = width * w_pic
    h = height * h_pic

    w_1 = int((x_center * w_pic) - w/2)
    h_1 = int((y_center * h_pic) - h/2)

    w_2 = int(w_1 + w)
    h_2 = int(h_1 + h)

    # print(w_1, h_1, w_2, h_2)
    img_f = img.crop((w_1, h_1, w_2, h_2))
    # img_f.show()
    img_f.save("runs/target_image_final.jpg")
```

فصل 2:

نتایج و تفسیر آنها

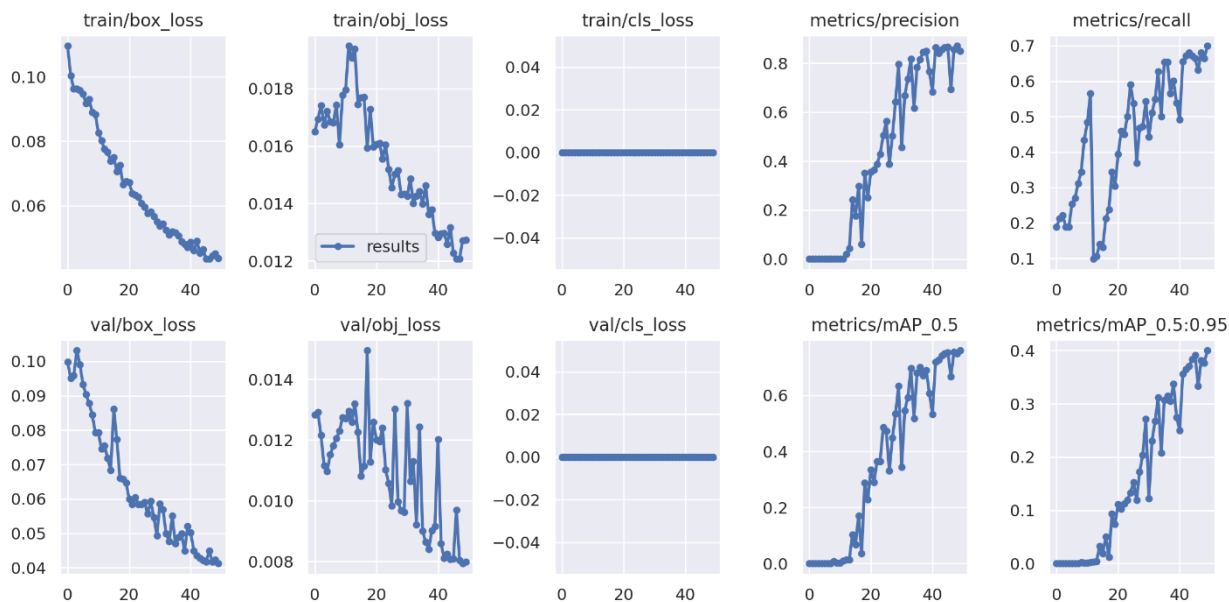
2-1- مقدمه

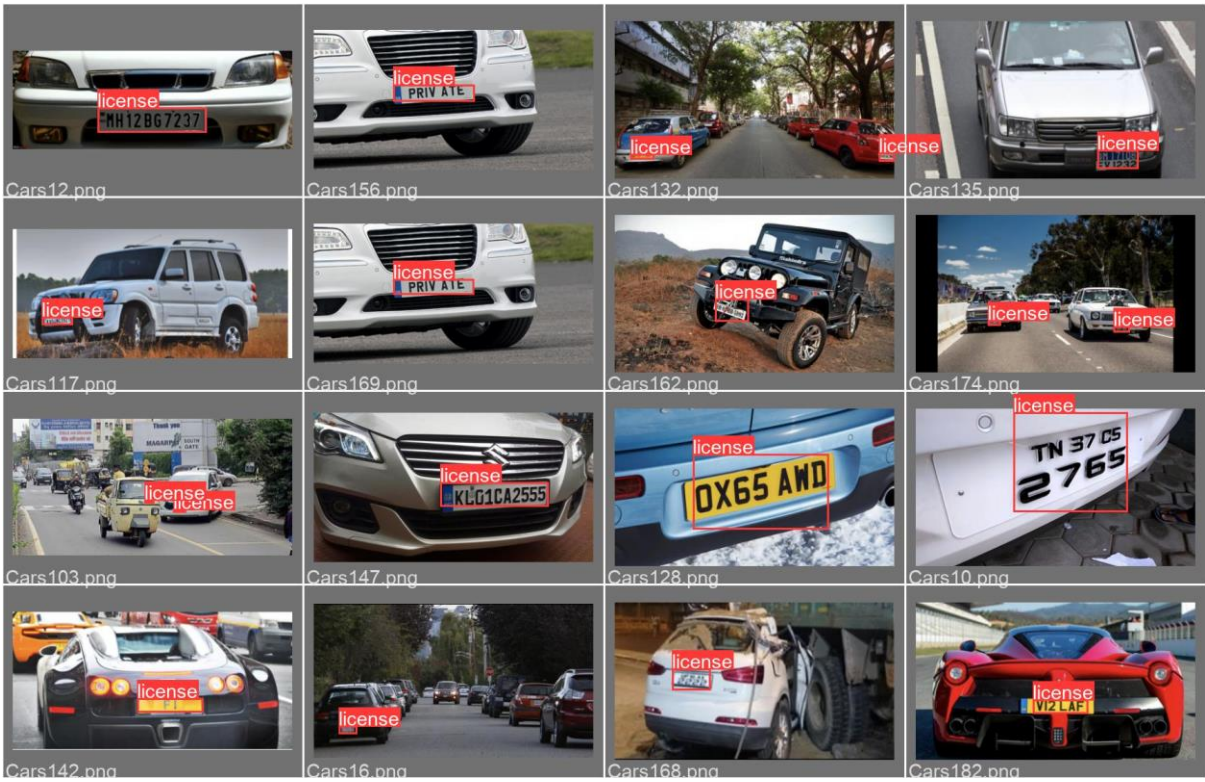
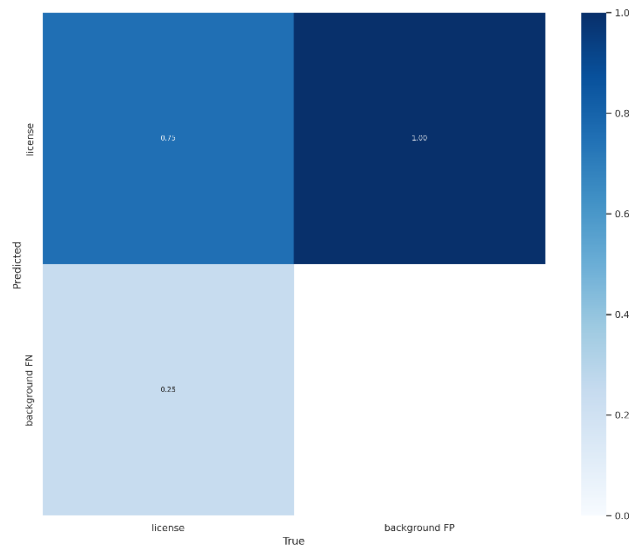
در این قسمت، نتایج و تحلیل و تفسیر آنها در هر قسمت از فصل گذشته برای دو شبکه و بصورت ترکیبی ارائه می‌شود.

2-2- تشخیص پلاک خودروها

در فصل قبل نحوه کارکرد شبکه توضیح داده شد، در این قسمت صرفاً به نتایج و خروجی تصویر وارد شده به شبکه در مقایسه با ورودی آن پرداخته خواهد شد.

2-2-1- دقت شبکه





2-2-2- خروجی نهایی

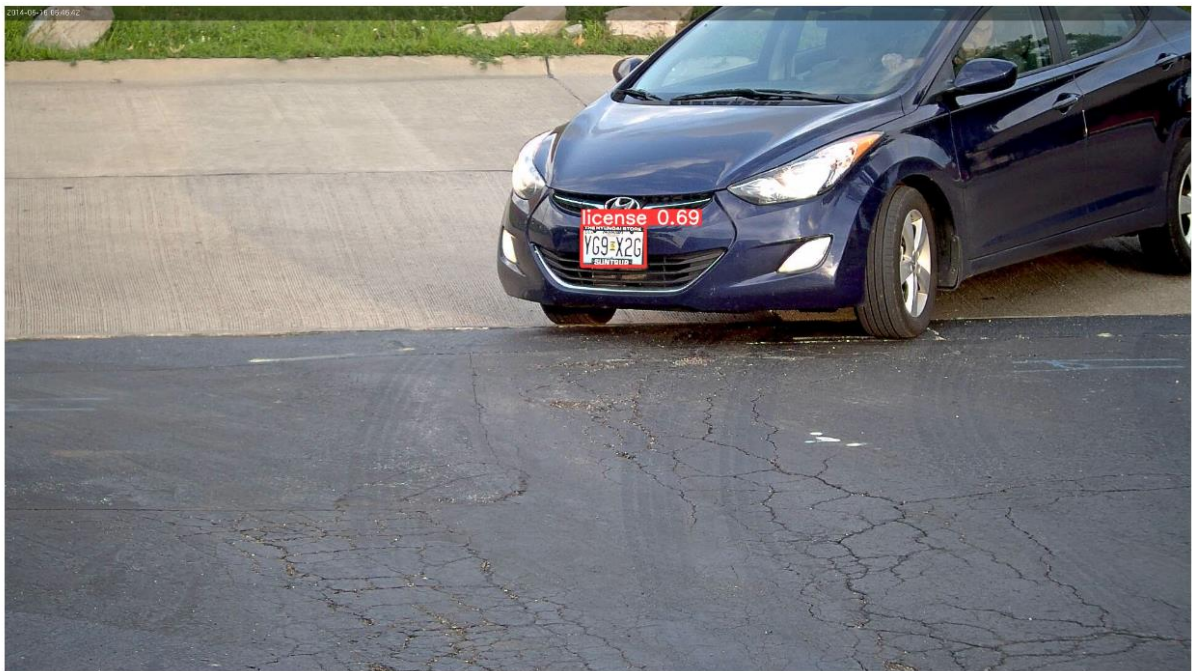
پس از تمرین شبکه، با استفاده از دستور زیر، آدرس وزن ها را داده و تصویر ورودی را نیز مشخص میکنیم، پس از آن تشخیص انجام شده و تصویر نهایی به همراه فایل تکست ذخیره میشود.

```
In [15]: !python detect.py --source ../benchmarks/endoend/us --conf 0.4 --weights runs/train/exp/weights/best.pt --save-txt

detect: weights=['runs/train/exp/weights/best.pt'], source=../benchmarks/endoend/us, data=data/coco128.yaml, imgsz=[640, 640], conf_thres=0.4, iou_thres=0.45, max_det=1000, device=, view_img=False, save_txt=True, save_conf=False, save_crop=False, nosave=False, classes=None, agnostic_nms=False, augment=False, visualize=False, update=False, project=runs/detect, name=exp, exist_ok=False, line_thickness=3, hide_labels=False, hide_conf=False, half=False, dnn=False
YOLOv5 v6.1-266-g34df503 Python-3.7.13 torch-1.11.0+cu113 CUDA:0 (Tesla T4, 15110MiB)

Fusing layers...
YOLOv5m summary: 290 layers, 20852934 parameters, 0 gradients
image 1/222 /content/benchmarks/endoend/us/0b86cecf-67d1-4fc0-87c9-b36b0ee228bb.jpg: 384x640 1 license, Done. (0.023s)
image 2/222 /content/benchmarks/endoend/us/12c6cb72-3ea3-49e7-b381-e0cdfc5e8960.jpg: 384x640 Done. (0.022s)
image 3/222 /content/benchmarks/endoend/us/1e241dc8-8f18-4955-8988-03a0ab49f813.jpg: 384x640 1 license, Done. (0.022s)
image 4/222 /content/benchmarks/endoend/us/21d8c31d-3deb-494b-9c63-c0223306fd82.jpg: 384x640 1 license, Done. (0.022s)
image 5/222 /content/benchmarks/endoend/us/22e54a62-57a8-4a0a-88c1-4b9758f67651.jpg: 384x640 1 license, Done. (0.022s)
image 6/222 /content/benchmarks/endoend/us/316b64c0-55bf-4079-a1c0-d93f461a576f.jpg: 384x640 1 license, Done. (0.022s)
image 7/222 /content/benchmarks/endoend/us/33fa5185-0286-4e8f-b775-46162eba39d4.jpg: 384x640 1 license, Done. (0.022s)
image 8/222 /content/benchmarks/endoend/us/37170dd1-2802-4e38-b982-c5d07c64ff67.jpg: 384x640 Done. (0.022s)
image 9/222 /content/benchmarks/endoend/us/3850ba91-3c64-4c64-acba-0c46b61ec0da.jpg: 384x640 2 licenses, Done. (0.022s)
image 10/222 /content/benchmarks/endoend/us/4be2025c-09f7-4bb0-b1bd-8e8633e6dec1.jpg: 384x640 1 license, Done. (0.021s)
image 11/222 /content/benchmarks/endoend/us/5b562a61-34ad-4f00-9164-d34abb7a38e4.jpg: 384x640 1 license, Done. (0.021s)
```

در اینجا ما بر روی یک سری تصویر تست، تشخیص را انجام دادیم و چند نمونه از تشخیص ها را در زیر مشاهده میکنید :





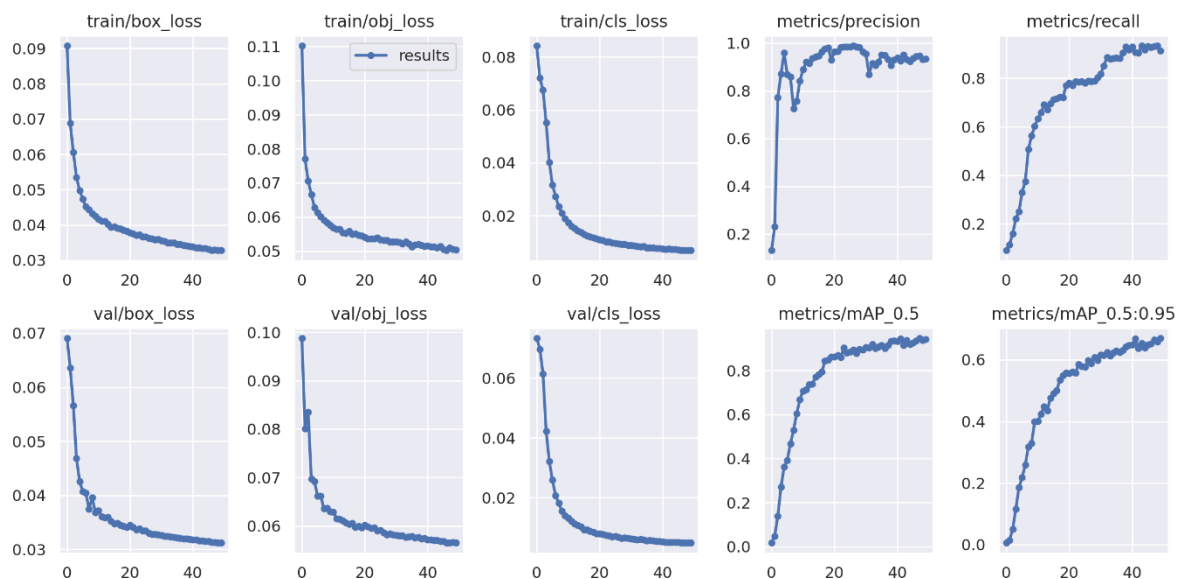
همچنین بصورت جداگانه نیز این شبکه بر روی خودروی ایرانی تست و خروجی بصورت زیر است:



2-3-2- تشخیص کاراکتر پلاک

در فصل قبل نحوه کارکرد شبکه توضیح داده شد، در این قسمت صرفاً به نتایج و خروجی تصویر وارد شده به شبکه در مقایسه با ورودی آن پرداخته خواهد شد.

1-2-3- دقت شبکه



2-3-2- خروجی نهایی

پس از تمرین شبکه، با استفاده از دستور زیر، آدرس وزن ها را داده و تصویر ورودی را نیز مشخص میکنیم، پس از آن تشخیص انجام شده و تصویر نهایی به همراه فایل تکست ذخیره میشود.

```
!python detect.py --source /content/45767y8uhn-1024x580-1.jpg --weights runs/train/exp/weights/best.pt --conf 0.25 --name yolo_p
detect: weights=['runs/train/exp/weights/best.pt'], source=/content/45767y8uhn-1024x580-1.jpg, data=data/coco128.yaml, imgsz=[640, 640], conf_thres=0.25, iou_thres=0.45, max_det=1000, device=, view_img=False, save_txt=False, save_conf=False, save_crop=False, nosave=False, classes=None, agnostic_nms=False, augment=False, visualize=False, update=False, project=runs/detect, name=yolo_plate_det, exist_ok=False, line_thickness=3, hide_labels=False, hide_conf=False, half=False, dnn=False
YOLOv5 v6.1-266-g34df503 Python-3.7.13 torch-1.11.0+cu113 CUDA:0 (Tesla T4, 15110MiB)

Fusing layers...
YOLOv5m summary: 290 layers, 20998410 parameters, 0 gradients
image 1/1 /content/45767y8uhn-1024x580-1.jpg: 384x640 1 0, 2 1s, 2 2s, 1 4, 1 9, 1 22, Done. (0.023s)
Speed: 0.4ms pre-process, 22.7ms inference, 1.3ms NMS per image at shape (1, 3, 640, 640)
Results saved to runs/detect/yolo_plate_det
```

در اینجا ما بر روی یک تصویر تست تشخیص را انجام دادیم و نمونه ای از تشخیص کاراکترها را در زیر مشاهده میکنید :



Plate Characters Detection TEST

```
!python detect.py --source runs/img2.jpg --conf 0.4 --weights runs/plate-char-detection-weights.pt --save-txt --name 2 --hide-co
detect: weights=['runs/plate-char-detection-weights.pt'], source=runs/img2.jpg, data=data/coco128.yaml, imgsz=[640, 640], conf_thres=0.4, iou_thres=0.45, max_det=1000, device=, view_img=False, save_txt=True, save_conf=False, save_crop=False, nosave=False, classes=None, agnostic_nms=False, augment=False, visualize=False, update=False, project=runs/detect, name=2, exist_ok=False, line_thickness=3, hide_labels=False, hide_conf=True, half=False, dnn=False
fatal: cannot change to 'D:\learning\EE': No such file or directory
YOLOv5 2022-6-28 Python-3.9.6 torch-1.9.0+cu111 CUDA:0 (NVIDIA GeForce GTX 1660 Ti, 6144MiB)

Fusing layers...
YOLOv5m summary: 290 layers, 20998410 parameters, 0 gradients
image 1/1 D:\learning\EE Courses\Deep learning (Dr.Hadadi)\TA\Exercise\Project\Final\yolov5\runs\img2.jpg: 384x640 5 1s, 1 2, 1 5, 1 23, Done. (0.022s)
Speed: 1.0ms pre-process, 22.0ms inference, 4.0ms NMS per image at shape (1, 3, 640, 640)
Results saved to runs\detect\2
1 labels saved to runs\detect\2\labels
```

Image(path + "2/img2.jpg")



2-4- تشخیص کاراکتر از روی عکس

در قسمت های قبل دو شبکه بصورت مجزا تشکیل و یادگیری شد و نتایج آن نیز بررسی شد، در این قسمت این دو شبکه را به یکدیگر متصل کرده و خروجی شبکه اول که تشخیص پلاک است را به شبکه دوم که تشخیص کاراکتر است داده میشود و در نهایت نتایج و خروجی تصویر وارد شده به شبکه در مقایسه با ورودی بررسی خواهد شد.

1-2-4- خروجی نهایی

تصویر ورودی ما بصورت زیر خواهد بود و توقع خواهیم داشت که در خروجی کاراکتر های پلاک ماشین سفید رنگ را داشته باشیم:

```
image_path = "runs/target_image.jpg"  
first_label = "runs/detect/target_image/labels/target_image.txt"
```

```
Image(image_path)
```



I. برای اینکار ابتدا تصویر مورد نظر را توسط شبکه یولو اول تشخیص پلاک می‌دهیم :

```
!python detect.py --source runs/target_image.jpg --conf 0.4 --weights runs/plate-detection-weights.pt --save-txt --name target_in

detect: weights=['runs/plate-detection-weights.pt'], source=runs/target_image.jpg, data=data\coco128.yaml, imgsz=[640, 640], co
nf_thres=0.4, iou_thres=0.45, max_det=1000, device=, view_img=False, save_txt=True, save_conf=False, save_crop=False, nosave=Fa
lse, classes=None, agnostic_nms=False, augment=False, visualize=False, update=False, project=runs\detect, name=target_image, ex
ist_ok=False, line_thickness=3, hide_labels=False, hide_conf=False, half=False, dnn=False
fatal: cannot change to 'D:\learning\EE': No such file or directory
YOLOv5 2022-6-28 Python-3.9.6 torch-1.9.0+cu111 CUDA:0 (NVIDIA GeForce GTX 1660 Ti, 6144MiB)

Fusing layers...
YOLOv5m summary: 290 layers, 20852934 parameters, 0 gradients
image 1/1 D:\learning\EE Courses\Deep learning (Dr.Hadadi)\TA\Exercise\Project\Final\yolov5\runs\target_image.jpg: 448x640 1 li
cense, Done. (0.026s)
Speed: 0.0ms pre-process, 26.0ms inference, 4.0ms NMS per image at shape (1, 3, 640, 640)
Results saved to runs\detect\target_image
1 labels saved to runs\detect\target_image\labels
```

II. پس از تشخیص پلاک و ذخیره شدن کادر پلاک در تصویر، پلاک را از تصویر جدا کرده و ذخیره می‌کنیم :

```
from PIL import Image

with open(first_label) as f:
    char = f.read()

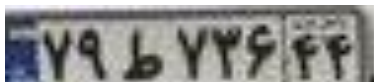
char = [list(map(float, i.strip().split(' '))) for i in char.strip().split('\n')]
for l in char:
    label, x_center, y_center, width, height = l
    img = Image.open(image_path)
    w_pic, h_pic = img.size

    w = width * w_pic
    h = height * h_pic

    w_1 = int((x_center * w_pic) - w/2)
    h_1 = int((y_center * h_pic) - h/2)

    w_2 = int(w_1 + w)
    h_2 = int(h_1 + h)

#     print(w_1, h_1, w_2, h_2)
img_f = img.crop((w_1, h_1, w_2, h_2))
#     img_f.show()
img_f.save("runs/target_image_final.jpg")
```



III. سپس تصویر پلاک ذخیره شده را به شبکه دوم داده تا کاراکترهای آن مشخص شود :

```
!python detect.py --source runs/target_image_final.jpg --conf 0.4 --weights runs/plate-char-detection-weights.pt --save-txt --na

detect: weights=['runs/plate-char-detection-weights.pt'], source=runs/target_image_final.jpg, data=data\coco128.yaml, imgsz=[64
0, 640], conf_thres=0.4, iou_thres=0.45, max_det=1000, device=, view_img=False, save_txt=True, save_conf=False, save_crop=Fals
e, nosave=False, classes=None, agnostic_nms=False, augment=False, visualize=False, update=False, project=runs\detect, name=targ
et_image_final, exist_ok=False, line_thickness=3, hide_labels=False, hide_conf=True, half=False, dnn=False
fatal: cannot change to 'D:\learning\EE': No such file or directory
YOLOv5 2022-6-28 Python-3.9.6 torch-1.9.0+cu111 CUDA:0 (NVIDIA GeForce GTX 1660 Ti, 6144MiB)

Fusing layers...
YOLOv5m summary: 290 layers, 20998410 parameters, 0 gradients
image 1/1 D:\learning\EE Courses\Deep learning (Dr.Hadadi)\TA\Exercise\Project\Final\yolov5\runs\target_image_final.jpg: 160x64
0 1 3, 2 4s, 1 6, 2 7s, 1 9, 1 25, Done. (0.016s)
Speed: 0.0ms pre-process, 16.0ms inference, 4.0ms NMS per image at shape (1, 3, 640, 640)
Results saved to runs\detect\target_image_final
1 labels saved to runs\detect\target_image_final\labels
```

IV. در مرحله آخر کلاس های تشخیص داده شده از فایل txt را میخوانیم و پلاک مورد نظر تشخیص

داده میشود :

```
second_label = first_label = "runs/detect/target_image_final/labels/target_image_final.txt"
with open(first_label) as f:
    char = f.read()

char = [list(map(float, i.strip().split(' '))) for i in char.strip().split('\n')]
pelak = []

for l in char:
    label, x_center, y_center, width, height = l
    pelak.append((x_center, plate_char[int(label)]))

pelak = sorted(pelak, key = lambda i:i[0])
pelak = [i[1] for i in pelak]

for ch in pelak:
    print(ch)
```

7
9
ط
7
3
6
4
4

مراجع

- [1] <https://github.com/ultralytics/yolov5>
- [2] <https://docs.ultralytics.com/tutorials/train-custom-datasets/>
- [3] <https://blog.paperspace.com/train-yolov5-custom-data/>
- [4]