گزارش پرژوه پایانی یادگیری عمیق

محمد امینی ۴۰۰۶۱۵۵۰۷

تشخیص پلاک خودرو با یادگیری عمیق

قسمت ۱:

در این قسمت که تشخیص پلاک خودرو در تصویر است از یک دیتاست تهیه شده از اینترنت استفاده شده است. این دیتاست در سایت kaggle موجود است و لینک آن https://www.kaggle.com/datasets/andrewmvd/car-plate-detection است.

این دیتاست شامل ۴۳۲ عکس ماشین به همراه پلاک آن ها است. البته در این بخش چون هدف تنها تشخیص پلاک است و نه کاراکتر های آنها پلاک این خودروها فارسی نیست و مشکلی هم ایجاد نمیکند. (فرمت دیتاست به صورت xml و به فرمت یولو است)

برای این پروژه از الگوریتم یولو مطابق نکات تدریس شده در کلاس استفاده شده است که اکنون به توضیح مختلف بخش های کد میپردازیم.

در ابتدا این نکته قابل ذکر است که دیتاست را دانلود کرده و جداگونه در google drive آپلود کردم تا هر بار آن را دانلود نکنم.

برای دسترسی به google drive از کد زیر استفاده کردم :

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

سپس به import کردن کتابخانه های مورد نیاز میپردازیم :

```
import torch
from IPython.display import Image  # for displaying images
import os
import random
import shutil
from sklearn.model_selection import train_test_split
import xml.etree.ElementTree as ET
from xml.dom import minidom
from tqdm import tqdm
from PIL import Image, ImageDraw
import numpy as np
import matplotlib.pyplot as plt

random.seed(108)
```

سپس عکس ها و لیبل ها را براساس اسم آنها مرتب میکنیم و آن ها را به سه بخش train, test, validation تقسیم میکنیم. برای این کار از دستور train_test_split استفاده میکنیم.

```
# Read images and annotations
images = [os.path.join('images', x) for x in os.listdir('images')]
annotations = [os.path.join('annotations', x) for x in os.listdir('annotations') if x[-3:] == "txt"]

images.sort()
annotations.sort()

# Split the dataset into train-valid-test splits
train_images, val_images, train_annotations, val_annotations = train_test_split(images, annotations, test_size = 0.2, random_state = 1)
val_images, test_images, val_annotations, test_annotations = train_test_split(val_images, val_annotations, test_size = 0.5, random_state = 1)
```

و برای هر بخش (عکس ها و لیبل ها) سه فولدر train,test,validation درست میکنیم :

```
!mkdir images/train images/val images/test annotations/train annotations/val annotations/test
```

حال هر یک از عکس ها و لیبل متناظر به آن را به این فولدر ها با تابع زیر منتقل میکنیم :

```python
#Utility function to move images
def move_files_to_folder(list_of_files, destination_folder):
    for f in list_of_files:
        try:
            shutil.move(f, destination_folder)
        except:
            print(f)
            assert False

# Move the splits into their folders
move_files_to_folder(train_images, 'images/train')
move_files_to_folder(val_images, 'images/val/')
move_files_to_folder(test_images, 'images/test/')
move_files_to_folder(train_annotations, 'annotations/train/')
move_files_to_folder(val_annotations, 'annotations/val/')
move_files_to_folder(test_annotations, 'annotations/test/')
```

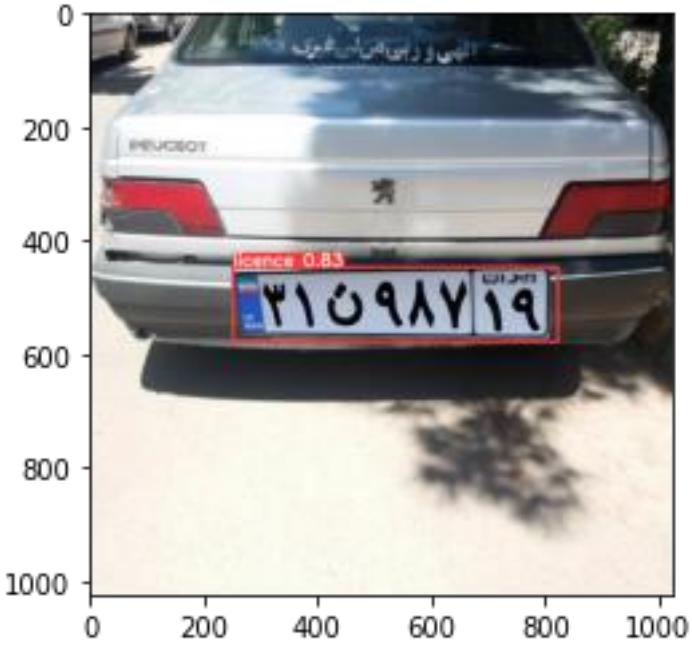
از طرفی از آنجا که در الگوریتم یولو اسم دو فولدر ما باید images و labels باشد، آنها را به این اسامی تغییر میدهیم.

از طرفی فایل yaml را مطابق مساله خود تغییر میدهیم. در اینجا تنها یک کلاس و به اسم lisence داریم.

حال تمامی مراحل پیش پردازش انجام شده است و از مدل آماده یولو استفاده میکنیم.

مدل به تعداد ۵۰ epoch و در هر مرحله با ۱۶ batch آموزش داده شده است.

تویر یک پیش بینی داده شده به مدل به صورت زیر است :



برای تبدیل کردن فایل xml به txt و فرمت یولو از کد استفاده شده در کلاس استفاده شده است.

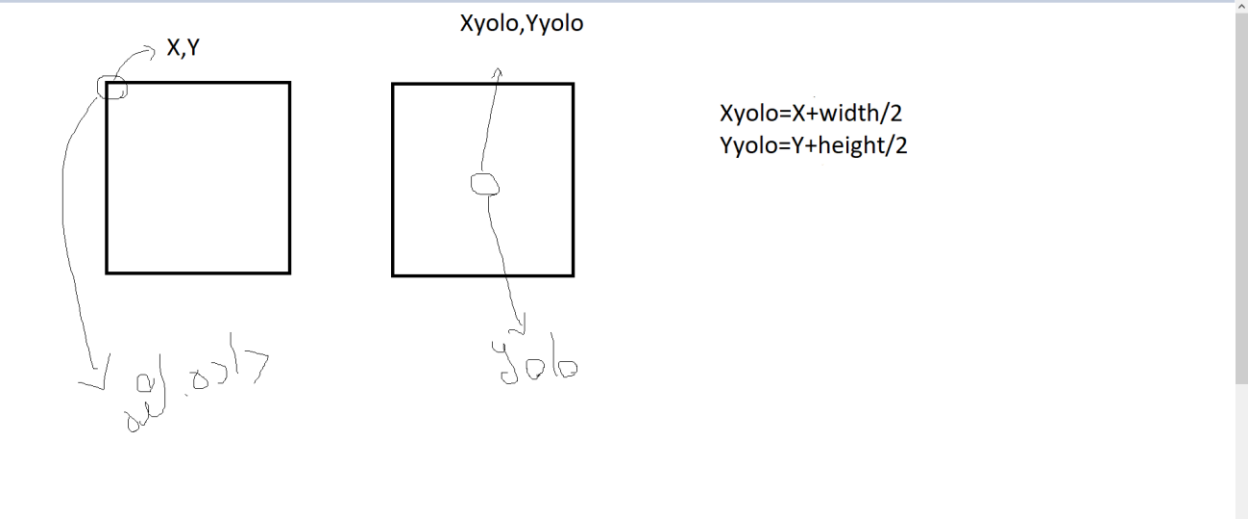
قسمت ۲ :

برای این قسمت دیتاست مورد نیاز به ما داده شده است اما با این تفاوت که فرمت داده شده به شکل یولو نیست.

تفاوت فرمت داده شده با یولو در این است که در فرمت داده شده ، به ما مختصات نقطه بالا سمت چپ مستطیل به همراه طول و عرض آن را داده است، اما در فرمت یولو ما به مختصات نقطه وسط به همراه طول و عرض آن نیاز داریم.

در فایل جیسون داده شده به ما، اولا کلاس هر کاراکتر(که از ۰ تا ۳۶) میتواند باشد به ما داده شده است و ثانیا مختصات نقطه بالا سمت چپ به همراه طول و عرض این مستطیل به ما داده شده است. همانطور که مشخص است این اطلاعات به فرمت یولو نیست و برای تبدیل کردن آن به فرمت یولو مطابق شکل زیر عمل میکنیم. در واقع لیبل ما برای هر عکس باید یک فایل txt باشد که به تربیت شماره کلاس، مختصات نقطه وسط (x,y) و طول و عرض این مستطیل باشد.

یعنی مطابق شکل زیر عمل میکنیم :



برای پیش پردازش این قسمت (تبدیل کردن به فرمت یولو و txt) ابتدا یک کد پایتون جداگانه روی این دیتاست اجرا کردم و سپس مانند قسمت قبل آن ها را در google drive آپلود کردم.

یعنی در نهایت وقتی لیبل ها به فرمت یولو تبدیل شده و به شکل txt است در سایت آپلود شده است.

این کد به شکل زیر است :

```
import numpy as np
import pandas as pd
import os
from PIL import Image

base_path=r'C:\users\USER\Desktop\Dataset1\labels'
for file in os.listdir(base_path):
    location_of_label=r'C:\Users\USER\Desktop\Dataset1\labels\{}'.format(file)
    location_of_image=r'C:\Users\USER\Desktop\Dataset1\images\{}'.format(file)
    location_of_image=location_of_image[:-4]+'jpg'
    json_file=pd.read_json(location_of_label)
    image = Image.open(location_of_image)
    width, height = image.size
    for index,row in json_file.iterrows():
        sd=str(row['char_id'])+' '+str((row['x']+row['width']/2)/width)+' '+str((row['y']+row['height']/2)/height)+' '+str(row['width']/width)+' '+str(row['height']/heigh
        f = open(location_of_label[:-5]+'.txt', "a")
        f.write("{}\n".format(sd))
    os.remove(location_of_label)
    f.close()
```

این کد به این صورت عمل میکند که ابتدا به ازای هر یک از فایل های جیسون، با استفاده از کتابخانه panda آن ها را میخوانیم. سپس کلاس، مختصات طول + طول/۲ ، مختصات عرض + عرض/۲ ، طول و عرض را ذخیره میکنیم. (مطابق توضیحات داده شده)

از طرفی از آنجا که برای فرمت یولو نیاز داریم که این اطلاعات نرمالیزه شده باشد، مقادیر مختصات طول و طول را بر طول تصویر و مختصات عرض و عرض را بر عرض تصویر تقسیم میکنیم. به این ترتیب این اطلاعات نرمالیزه میشوند و در نهایت در یک فایل txt با همان اسم ذخیره کرده و فایل جیسون را حذف میکنیم.

حال دیتاست آماده است و میتوان آن را با استفاده از کد قسمت قبل آموزش داد. فقط چند تفاوت وجود دارد :

```
train: ../images/train/
val:  ../images/val/
test: ../images/test/

# number of classes
nc: 37

# class names
names: ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'alef', 'b', 'j', 'l', 'm', 'n', 'q', 'v', 'h', 'y', 'd', 's', 'sad', 'malol', 't', 'ta
```

۱- تعداد کلاس ها ۳۷ تا است پس باید فایل yaml را تغییر بدیم.

۲- تعداد epoch هارا به ۱۰۰ و تعداد batch هارا به ۳۲ عدد افزایش دادم.

پس از مدت ۲ ساعت آموزش به پایان میرسد.

پس از آن که آموزش یولو دوم به پایان رسید برای تشخیص یک پلاک و کاراکتر های متناسب با آن به این صورت عمل میکنیم که ابتدا عکس را به شبکه اول میدهیم و این شبکه برای ما پلاک را تشخیص میدهد. برای این شبکه آرگومان save-crops– را فعال میکنیم. با این کار خود شبکه اول شی هایی (در اینجا پلاک) را که تشخیص میدهد را crop میکند. حال ما تصویر crop شده را به شبکه دوم میدهیم و این شبکه کاراکترها را برای ما تشخیص میدهد.

عکس داده شده به شبکه دوم به صورت زیر است:

در واقع خود یولو این را برای ما crop کرده است.

در شبکه دوم از دستور save-txt– استفاده کرده ایم. با این کار مختصات باکس تشخیص داده شده در یک فایل تکست ذخیره میکنیم و سپس تابع زیر را روی آن اجرا میکنیم.

```python
def plate_reader(file):
  plate=''
  list_of_dict=[]
  with open(file,'r') as f:
    for line in f:
        array=[]

        for word in line.split():
           array.append(word)
        dict={'class':array[0],'x':array[1]}
        list_of_dict.ap (variable) list_of_dict: list
  list_of_dict= sorted(list_of_dict, key=lambda d: d['x'])
  char_classes = {0: '0', 1: '1', 2: '2', 3: '3', 4: '4', 5: '5', 6: '6', 7: '7', 8: '8', 9: '9',10:'alef',11:'b',12:'j',13:'l',14:'m
  for i in list_of_dict:
      plate+=(char_classes.get(int(i['class'])))
  return plate
```

در واقع کاری که این تابع انجام میدهد این است که با توجه به فایل تکست ذخیره شده و کلاس متناسب به آن ، آن را با کلاس های دیتاست مقایسه میکند و کاراکتر هر کدام را بر میگرداند و با کنار هم قرار دادن این کاراکتر ها میتوانیم به پلاک مورد نظر برسیم.

خروجی این تابع به شکل زیر است:

```
detect: weights=['runs/train/yolo_road_det/weights/best.pt'], source=../Car.jpg,
YOLOv5 🚀 v6.1-273-gd94b470 Python-3.7.13 torch-1.11.0+cu113 CUDA:0 (Tesla T4, 1

Fusing layers...
YOLOv5s summary: 213 layers, 7012822 parameters, 0 gradients, 15.8 GFLOPs
image 1/1 /content/drive/MyDrive/Car.jpg: 640x640 1 lisence, Done. (0.013s)
Speed: 0.6ms pre-process, 12.8ms inference, 1.2ms NMS per image at shape (1, 3, 6
Results saved to ../Part1/final1
detect: weights=['runs/train/yolo_road_det/weights/best.pt'], source=../../Part1/
YOLOv5 🚀 v6.1-272-g8983324 Python-3.7.13 torch-1.11.0+cu113 CUDA:0 (Tesla T4, 1

Fusing layers...
YOLOv5s summary: 213 layers, 7109914 parameters, 0 gradients, 16.1 GFLOPs
image 1/1 /content/drive/MyDrive/Part1/final/crops/lisence/Car.jpg: 192x640 2 1s,
Speed: 0.4ms pre-process, 16.1ms inference, 1.3ms NMS per image at shape (1, 3, 6
Results saved to ../../Part2/final1
1 labels saved to ../../Part2/final1/labels
31n98719
```

که همان پلاک خودرو است.

قسمت ۳:

برای این قسمت ابتدا باید ویدیو را به شبکه اول بدهیم. با این کار شبکه اول قسمت های از ویدیو که در آن ها پلاک تشخیص دهد را crop میکند. حال ما این عکس های crop شده را به شبکه دوم میدهیم. با این کار تمامی پلاک ها و کاراکتر های آن در ویدیو تشخیص داده میشود.