使用Yolov5训练kitti数据

> - 配置Yolov5环境
> - 转换kitti数据集格式
> - 使用Yolov5训练数据集

Windows环境下使用Anaconda配置Yolov5环境

1. 创建虚拟环境

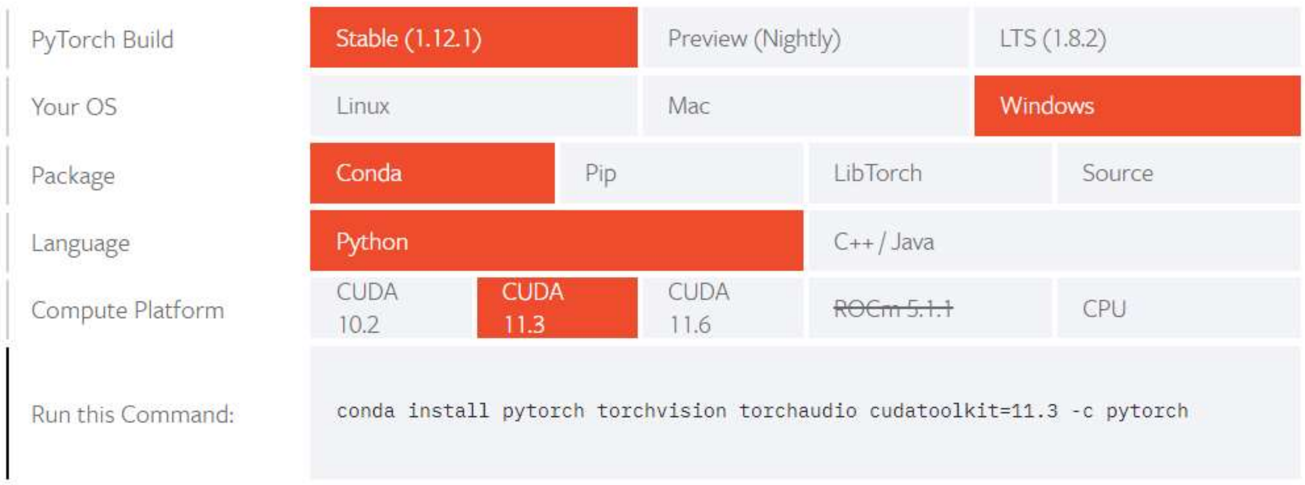
```
conda create -n pytorch python=3.7
```

第一步创建anaconda虚拟环境，命名pytorch

```
conda activate pytorch
```

进入新创建的虚拟环境

2. 安装pytorch



PyTorch Build	[✓ Stable (1.12.1)]	Preview (Nightly)	LTS (1.8.2)		
Your OS	Linux	Mac	[✓ Windows]		
Package	[✓ Conda]	Pip	LibTorch	Source	
Language	[✓ Python]	C++ / Java			
Compute Platform	CUDA 10.2	[✓ CUDA 11.3]	CUDA 11.6	~~ROCm 5.1.1~~	CPU
Run this Command:	`conda install pytorch torchvision torchaudio cudatoolkit=11.3 -c pytorch`				

在虚拟环境中运行这个指令

3. 安装Yolov5

github下载并解压到本地后，按照项目文件里的requirement.txt安装所需包。

```
pip install -r requirement.txt
```

转换kitti数据集格式

1.下载kitti数据集

转到官网

http://www.cvlibs.net/download.php?file=data_object_image_2.zip

http://www.cvlibs.net/download.php?file=data_object_label_2.zip

得到图片和标签

在dataset文件夹按如下框架储存数据，其中labels数据需要转换后在存储

```
|—kitti
├── imgages
|   ├── val
|   |   └── 000000.png
        ├── .......

|   └── train
|   |   └── 000000.png
        ├── .......
|
└── labels
    └── train
```

2.labels数据格式转换

首先将类别简化为三类，其中包括car，Pedestrian，Cyclist。

```
import glob
import string
```

```
txt_list = glob.glob('你下载的标签文件夹的标签路径/*.txt')
def show_category(txt_list):
    category_list= []
    for item in txt_list:
        try:
            with open(item) as tdf:
                for each_line in tdf:
                    labeldata = each_line.strip().split(' ') # 去掉前后多余的字符并把其分开
                    category_list.append(labeldata[0]) # 只要第一个字段，即类别
        except IOError as ioerr:
            print('File error:'+str(ioerr))
    print(set(category_list)) # 输出集合
def merge(line):
    each_line=''
    for i in range(len(line)):
        if i!= (len(line)-1):
            each_line=each_line+line[i]+' '
        else:
            each_line=each_line+line[i] # 最后一条字段后面不加空格
    each_line=each_line+'\n'
    return (each_line)
print('before modify categories are:\n')
show_category(txt_list)
for item in txt_list:
    new_txt=[]
    try:
        with open(item, 'r') as r_tdf:
            for each_line in r_tdf:
                labeldata = each_line.strip().split(' ')
                if labeldata[0] in ['Truck','Van','Tram']: # 合并汽车类
                    labeldata[0] = labeldata[0].replace(labeldata[0],'Car')
                if labeldata[0] == 'Person_sitting': # 合并行人类
                    labeldata[0] = labeldata[0].replace(labeldata[0],'Pedestrian')
                if labeldata[0] == 'DontCare': # 忽略Dontcare类
                    continue
                if labeldata[0] == 'Misc': # 忽略Misc类
                    continue
                new_txt.append(merge(labeldata)) # 重新写入新的txt
```

```
文件
        with open(item,'w+') as w_tdf: # w+是打开原文件将内容删除,
另写新内容进去
            for temp in new_txt:
                w_tdf.write(temp)
    except IOError as ioerr:
        print('File error:'+str(ioerr))
print('\nafter modify categories are:\n')
show_category(txt_list)
```

然后再把它转换为xml文件,创建一个Annotations文件夹用于存放xml

```
from xml.dom.minidom import Document
import cv2
import os
def generate_xml(name,split_lines,img_size,class_ind):
    doc = Document() # 创建DOM文档对象
    annotation = doc.createElement('annotation')
    doc.appendChild(annotation)
    title = doc.createElement('folder')
    title_text = doc.createTextNode('KITTI')
    title.appendChild(title_text)
    annotation.appendChild(title)
    img_name=name+'.png'
    title = doc.createElement('filename')
    title_text = doc.createTextNode(img_name)
    title.appendChild(title_text)
    annotation.appendChild(title)
    source = doc.createElement('source')
    annotation.appendChild(source)
    title = doc.createElement('database')
    title_text = doc.createTextNode('The KITTI Database')
    title.appendChild(title_text)
    source.appendChild(title)
    title = doc.createElement('annotation')
    title_text = doc.createTextNode('KITTI')
    title.appendChild(title_text)
    source.appendChild(title)
    size = doc.createElement('size')
    annotation.appendChild(size)
    title = doc.createElement('width')
    title_text = doc.createTextNode(str(img_size[1]))
    title.appendChild(title_text)
    size.appendChild(title)
    title = doc.createElement('height')
    title_text = doc.createTextNode(str(img_size[0]))
    title.appendChild(title_text)
    size.appendChild(title)
    title = doc.createElement('depth')
    title_text = doc.createTextNode(str(img_size[2]))
    title.appendChild(title_text)
    size.appendChild(title)
    for split_line in split_lines:
        line=split_line.strip().split()
        if line[0] in class_ind:
            object = doc.createElement('object')
            annotation.appendChild(object)
            title = doc.createElement('name')
            title_text = doc.createTextNode(line[0])
            title.appendChild(title_text)
            object.appendChild(title)
            bndbox = doc.createElement('bndbox')
            object.appendChild(bndbox)
            title = doc.createElement('xmin')
            title_text =
doc.createTextNode(str(int(float(line[4]))))
            title.appendChild(title_text)
            bndbox.appendChild(title)
            title = doc.createElement('ymin')
            title_text =
doc.createTextNode(str(int(float(line[5]))))
            title.appendChild(title_text)
            bndbox.appendChild(title)
            title = doc.createElement('xmax')
            title_text =
doc.createTextNode(str(int(float(line[6]))))
            title.appendChild(title_text)
            bndbox.appendChild(title)
            title = doc.createElement('ymax')
            title_text =
doc.createTextNode(str(int(float(line[7]))))
            title.appendChild(title_text)
            bndbox.appendChild(title)
    # 将DOM对象doc写入文件
    f = open('Annotations/trian'+name+'.xml','w')
    f.write(doc.toprettyxml(indent = ''))
    f.close()
if __name__ == '__main__':
    class_ind=('Pedestrian', 'Car', 'Cyclist')
    cur_dir=os.getcwd()
    labels_dir=os.path.join(cur_dir,'Labels')
    for parent, dirnames, filenames in os.walk(labels_dir): # 分别得到根目录，子目录和根目录下文件
        for file_name in filenames:
            full_path=os.path.join(parent, file_name) # 获取文件全路径
            f=open(full_path)
            split_lines = f.readlines()
            name= file_name[:-4] # 后四位是扩展名.txt，只取前面的文件名
            img_name=name+'.png'
            img_path=os.path.join('./JPEGImages/trian',img_name) # 路径需要自行修改
            img_size=cv2.imread(img_path).shape
            generate_xml(name,split_lines,img_size,class_ind)
print('all txts has converted into xmls')
```

最后再把.xml转化为适合于yolo训练的标签模式,也就是darknet的txt格式

```
import glob
import xml.etree.ElementTree as ET
# 这里的类名为我们xml里面的类名，顺序现在不需要考虑
class_names = ['Car', 'Cyclist', 'Pedestrian']
# xml文件路径
path = './Annotations/'
# 转换一个xml文件为txt
def single_xml_to_txt(xml_file):
    tree = ET.parse(xml_file)
    root = tree.getroot()
    # 保存的txt文件路径
    txt_file = xml_file.split('.')[0]+'.'+xml_file.split('.')[1]+'.txt'
    with open(txt_file, 'w') as txt_file:
        for member in root.findall('object'):
            #filename = root.find('filename').text
            picture_width = int(root.find('size')[0].text)
            picture_height = int(root.find('size')[1].text)
            class_name = member[0].text
            # 类名对应的index
            class_num = class_names.index(class_name)

            box_x_min = int(member[1][0].text) # 左上角横坐标
            box_y_min = int(member[1][1].text) # 左上角纵坐标
            box_x_max = int(member[1][2].text) # 右下角横坐标
            box_y_max = int(member[1][3].text) # 右下角纵坐标
            print(box_x_max,box_x_min,box_y_max,box_y_min)
            # 转成相对位置和宽高
            x_center = float(box_x_min + box_x_max) / (2 *
picture_width)
            y_center = float(box_y_min + box_y_max) / (2 *
picture_height)
            width = float(box_x_max - box_x_min) /  picture_width
            height = float(box_y_max - box_y_min) /
picture_height
            print(class_num, x_center, y_center, width, height)
            txt_file.write(str(class_num) + ' ' + str(x_center) +
' ' + str(y_center) + ' ' + str(width) + ' ' + str(height) + '\n')
# 转换文件夹下的所有xml文件为txt
def dir_xml_to_txt(path):
    for xml_file in glob.glob(path + '*.xml'):
        single_xml_to_txt(xml_file)
dir_xml_to_txt(path)
```

最后将得到的Annotations/下的所有txt文件放入之前的dataset/labels中

使用yolov5训练模型

首先在data文件夹中复制一份coco.yaml然后改名kitti.yaml修改内容

```
train: D:\yolov5master\data\kitti\images  # train images (relative
to 'path') 118287 images
val: D:\yolov5master\data\kitti\images  # train images (relative
to 'path') 5000 images

# Classes
nc: 3  # number of classes
names: ['Car','Pedestrian','Cyclist']  # class names
```

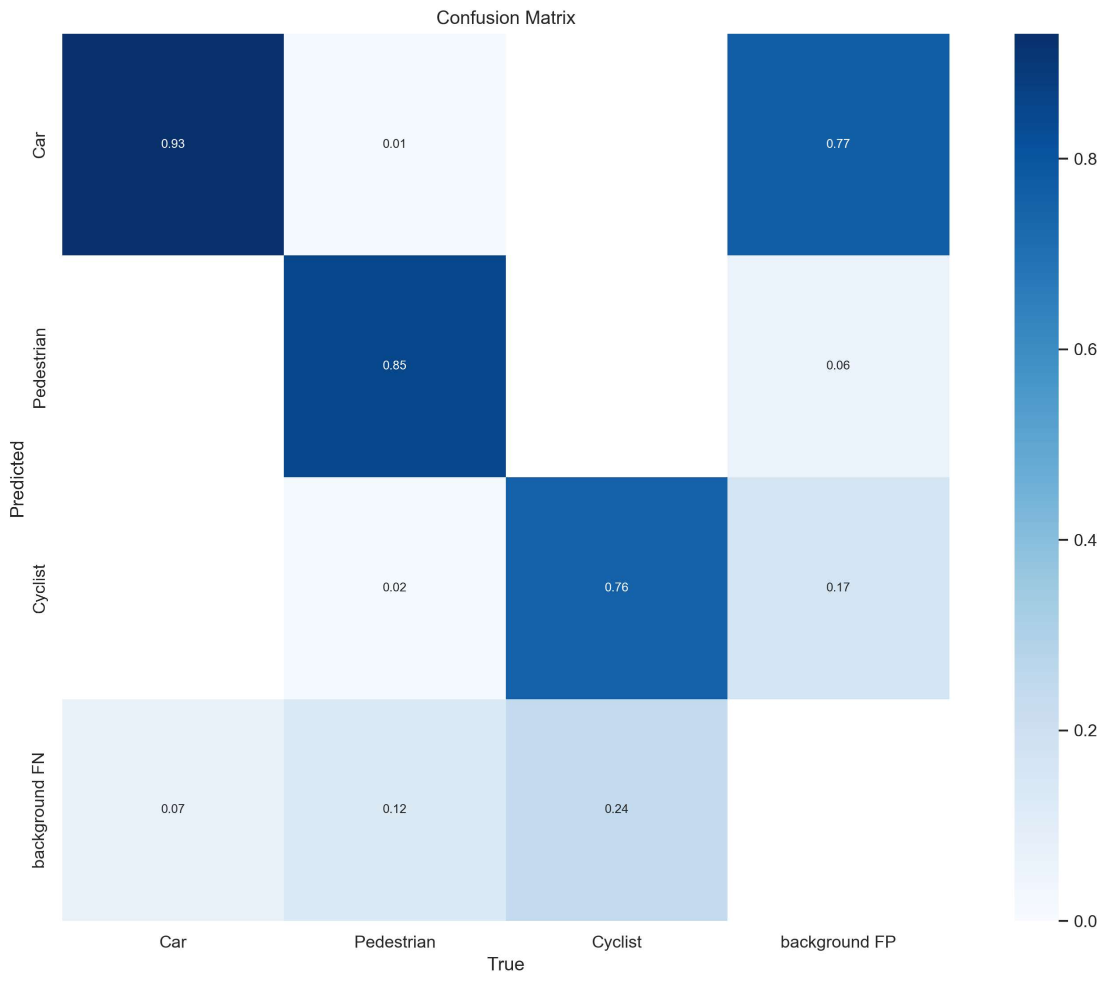
在models文件夹修改yolov5s.yaml内容

```
nc: 3  # number of classes
```

开始训练

```
python train.py --img 640 --batch-size 16 --epochs 10
--data data/kitti.yaml --cfg models/yolov5s.yaml
--weights yolov5s.pt
```

在runs/train/exp下可以看到训练的结果



拿着训练完的最好的权重尝试结果

```
python detect.py --weights runs/train/exp6/weights/best.pt
--source data/kitti/images/val/000000.png --device 0
```

Cyclist 0.74