VOC标签格式转yolo格式并划分训练集和测试集代码及注意事项

上代码

```
import xml.etree.ElementTree as ET
import pickle
import os
from os import listdir, getcwd
from os.path import join
import random
from shutil import copyfile

classes = ["person","dog","cat"]
#一定要和labelimg中定义的类一致

TRAIN_RATIO = 80
 #训练集占80%，验证集20%

def clear_hidden_files(path):
    dir_list = os.listdir(path)
    for i in dir_list:
        abspath = os.path.join(os.path.abspath(path), i)
        if os.path.isfile(abspath):
            if i.startswith("._"):
                os.remove(abspath)
        else:
            clear_hidden_files(abspath)

def convert(size, box):
    dw = 1./size[0]
    dh = 1./size[1]
    x = (box[0] + box[1])/2.0
    y = (box[2] + box[3])/2.0
    w = box[1] - box[0]
    h = box[3] - box[2]
    x = x*dw
    w = w*dw
    y = y*dh
    h = h*dh
    return (x,y,w,h)

def convert_annotation(image_id):
    in_file = open('VOCdevkit/VOC2007/Annotations/%s.xml' %image_id)
    out_file = open('VOCdevkit/VOC2007/YOLOLabels/%s.txt' %image_id, 'w')
    tree=ET.parse(in_file)
```

```
    root = tree.getroot()
    size = root.find('size')
    w = int(size.find('width').text)
    h = int(size.find('height').text)

    for obj in root.iter('object'):
        difficult = obj.find('difficult').text
        cls = obj.find('name').text
        if cls not in classes or int(difficult) == 1:
            continue
        cls_id = classes.index(cls)
        xmlbox = obj.find('bndbox')
        b = (float(xmlbox.find('xmin').text), float(xmlbox.find('xmax').text),
float(xmlbox.find('ymin').text), float(xmlbox.find('ymax').text))
        bb = convert((w,h), b)
        out_file.write(str(cls_id) + " " + " ".join([str(a) for a in bb]) + '\n')
    in_file.close()
    out_file.close()

wd = os.getcwd()
wd = os.getcwd()
data_base_dir = os.path.join(wd, "VOCdevkit/")
if not os.path.isdir(data_base_dir):
    os.mkdir(data_base_dir)
work_sapce_dir = os.path.join(data_base_dir, "VOC2007/")
if not os.path.isdir(work_sapce_dir):
    os.mkdir(work_sapce_dir)
annotation_dir = os.path.join(work_sapce_dir, "Annotations/")
if not os.path.isdir(annotation_dir):
    os.mkdir(annotation_dir)
clear_hidden_files(annotation_dir)
image_dir = os.path.join(work_sapce_dir, "JPEGImages/")
if not os.path.isdir(image_dir):
    os.mkdir(image_dir)
clear_hidden_files(image_dir)
yolo_labels_dir = os.path.join(work_sapce_dir, "YOLOLabels/")
if not os.path.isdir(yolo_labels_dir):
    os.mkdir(yolo_labels_dir)
clear_hidden_files(yolo_labels_dir)
yolov5_images_dir = os.path.join(data_base_dir, "images/")
if not os.path.isdir(yolov5_images_dir):
    os.mkdir(yolov5_images_dir)
clear_hidden_files(yolov5_images_dir)
yolov5_labels_dir = os.path.join(data_base_dir, "labels/")
if not os.path.isdir(yolov5_labels_dir):
    os.mkdir(yolov5_labels_dir)
clear_hidden_files(yolov5_labels_dir)
yolov5_images_train_dir = os.path.join(yolov5_images_dir, "train/")
if not os.path.isdir(yolov5_images_train_dir):
    os.mkdir(yolov5_images_train_dir)
clear_hidden_files(yolov5_images_train_dir)
yolov5_images_test_dir = os.path.join(yolov5_images_dir, "val/")
if not os.path.isdir(yolov5_images_test_dir):
```

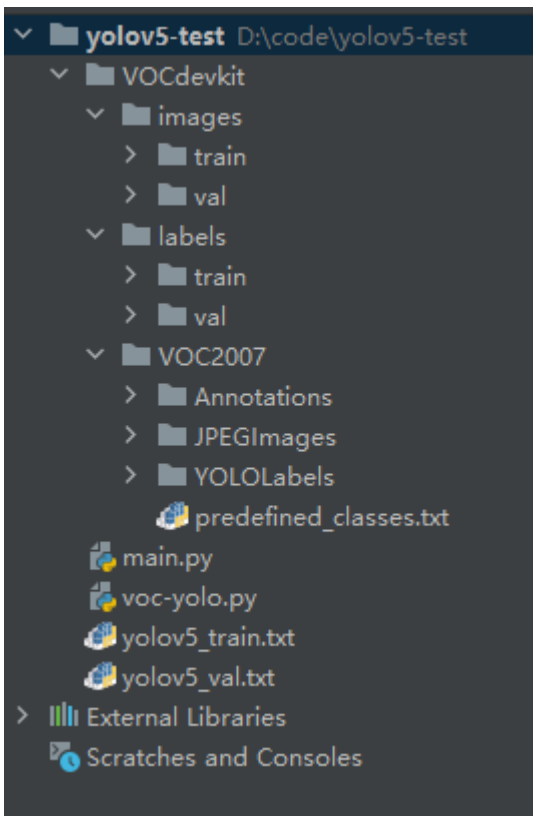
```
        os.mkdir(yolov5_images_test_dir)
clear_hidden_files(yolov5_images_test_dir)
yolov5_labels_train_dir = os.path.join(yolov5_labels_dir, "train/")
if not os.path.isdir(yolov5_labels_train_dir):
        os.mkdir(yolov5_labels_train_dir)
clear_hidden_files(yolov5_labels_train_dir)
yolov5_labels_test_dir = os.path.join(yolov5_labels_dir, "val/")
if not os.path.isdir(yolov5_labels_test_dir):
        os.mkdir(yolov5_labels_test_dir)
clear_hidden_files(yolov5_labels_test_dir)

train_file = open(os.path.join(wd, "yolov5_train.txt"), 'w')
test_file = open(os.path.join(wd, "yolov5_val.txt"), 'w')
train_file.close()
test_file.close()
train_file = open(os.path.join(wd, "yolov5_train.txt"), 'a')
test_file = open(os.path.join(wd, "yolov5_val.txt"), 'a')
list_imgs = os.listdir(image_dir) # list image files
prob = random.randint(1, 100)
print("Probability: %d" % prob)
for i in range(0,len(list_imgs)):
    path = os.path.join(image_dir,list_imgs[i])
    if os.path.isfile(path):
        image_path = image_dir + list_imgs[i]
        voc_path = list_imgs[i]
        (nameWithoutExtention, extention) =
os.path.splitext(os.path.basename(image_path))
        (voc_nameWithoutExtention, voc_extention) =
os.path.splitext(os.path.basename(voc_path))
        annotation_name = nameWithoutExtention + '.xml'
        annotation_path = os.path.join(annotation_dir, annotation_name)
        label_name = nameWithoutExtention + '.txt'
        label_path = os.path.join(yolo_labels_dir, label_name)
    prob = random.randint(1, 100)
    print("Probability: %d" % prob)
    if(prob < TRAIN_RATIO): # train dataset
        if os.path.exists(annotation_path):
            train_file.write(image_path + '\n')
            convert_annotation(nameWithoutExtention) # convert label
            copyfile(image_path, yolov5_images_train_dir + voc_path)
            copyfile(label_path, yolov5_labels_train_dir + label_name)
    else: # test dataset
        if os.path.exists(annotation_path):
            test_file.write(image_path + '\n')
            convert_annotation(nameWithoutExtention) # convert label
            copyfile(image_path, yolov5_images_test_dir + voc_path)
            copyfile(label_path, yolov5_labels_test_dir + label_name)
train_file.close()
test_file.close()
```

注意：

以上代码已经写死了文件名 使用时要注意

养成良好习惯，统一规范

将VOCdevkit文件夹放到代码文件夹下运行可得到：



images：此文件夹中train是训练集的图片 val是验证集图片

labels：此文件夹中train是训练集的标签 val是验证集标签

YOLOLabels：此文件夹下存放着所有txt格式的标签文件

到这里 images和labels文件夹就是训练yolov5模型所要的训练集和验证集