参考）YOLOv2 训练自己的数据集（voc 格式）进行实验，基本上是正确的，但其初始给出的代码并非是在 linux 下可以运行的，因此参考部分博客写了下面的程序，可以实现对文件夹内图片的批量读取以及更改名称符合 VOC 数据集习惯。另原文有部分小错误，本文已经修改，但后文属于转载，版权属原作者所有，本文仅为记录和交流用。如下文所示。

1 准备数据

首先准备好自己的数据集，最好固定格式，此处以 VOC 为例，采用 jpg 格式的图像，在名字上最好使用像 VOC 一样类似 000001.jpg、000002.jpg 这样。可使用下面示例代码

```
#include <dirent.h>
#include <sys/stat.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>
#include <opencv2/opencv.hpp>

#define img_num 2000
char img_file[img_num][1000];

int list_dir_name(char* dirname, int tabs)
{
    DIR* dp;
    struct dirent* dirp;
    struct stat st;
    char tab[tabs + 1];
                    char img_count=0;

    /* open dirent directory */
    if((dp = opendir(dirname)) == NULL)
    {
        perror("opendir");
        return -1;
```

```
    }



    /* fill tab array with tabs */
    memset(tab, '\t', tabs);
    tab[tabs] = 0;

    /**
     * read all files in this dir
     **/
    while((dirp = readdir(dp)) != NULL)

    {
        char fullname[255];
        memset(fullname, 0, sizeof(fullname));

        /* ignore hidden files */
        if(dirp->d_name[0] == '.')
            continue;

        /* display file name */
        //printf("img_name:%s\n", dirp->d_name);

        strncpy(fullname, dirname, sizeof(fullname));
        strncat(fullname, dirp->d_name, sizeof(fullname));
              strcat(img_file[img_count++], fullname);
              printf("Image %3d
path:%s\n",img_count-1,img_file[img_count-1]);//fullname=dir+file name,the absolute path of
the image file

                                        /* get dirent status */
        if(stat(fullname, &st) == -1)
        {
            perror("stat");
            fputs(fullname, stderr);
            return -1;
        }
```

```
        /* if dirent is a directory, call itself */
        if(S_ISDIR(st.st_mode) && list_dir_name(fullname, tabs + 1) == -1)
            return -1;
    }
    return img_count;

}

int main(int argc, char* argv[])
{
          char* dir="/home/robot/Downloads/mark_recognition/car_img/simple_3class/";
        printf("%s\n", dir);

        char sum=list_dir_name(dir, 1);
          printf("Img total num:%d\n",sum);

          int i;
          char order[1000];

          char txt_path[1000];
          char* txt_name="train.txt";
          memset(txt_path, 0, sizeof(txt_path));
          strcat(txt_path,dir);
          strcat(txt_path,txt_name);
          FILE *fp = fopen(txt_path, "w");

          for (i = 0; i<sum; ++i)
          {
                    char img_path[1000];
                    memset(img_path, 0, sizeof(img_path));

                    printf("Source %s\n", img_file[i]);
                    IplImage *pSrc = cvLoadImage(img_file[i]);
                    sprintf(order, "%05d.jpg", i);
                    strcat(img_path,dir);
                    strcat(img_path,order);
                    cvSaveImage(img_path, pSrc);
```

```
                    fprintf(fp, "%05d\n", i);


                    printf("Save as%s\n", img_path);
                    cvReleaseImage(&pSrc);
            }
            fclose(fp);




            return 0;
}
```

准备好了自己的图像后，需要按 VOC 数据集的结构放置图像文件。VOC 的结构如下

[plain] view plain copy

```
1.  --VOC
2.      --Annotations
3.      --ImageSets
4.        --Main
5.        --Layout
6.        --Segmentation
7.      --JPEGImages
8.      --SegmentationClass
9.      --SegmentationObject
```

这里面用到的文件夹是 Annotations、ImageSets 和 JPEGImages。其中文件夹 Annotation 中主要存放 xml 文件，每一个 xml 对应一张图像，并且每个 xml 中存放的是标记的各个目标的位置和类别信息，命名通常与对应的原始图像一样；而 ImageSets 我们只需要用到 Main 文件夹，这里面存放的是一些文本文件，通常为 train.txt、test.txt 等，该文本文件里面的内容是需要用来训练或测试的图像的名字（无后缀无路径）；JPEGImages 文件夹中放我们已按统一规则命名好的原始图像。

因此，首先

1.新建文件夹 VOC2007（通常命名为这个，也可以用其他命名，但一定是名字+年份，例如 MYDATA2016，无论叫什么后面都需要改相关代码匹配这里，本例中以 VOC2007 为例）

2.在 VOC2007 文件夹下新建三个文件夹 Annotations、ImageSets 和 JPEGImages，并把准备好的自己的原始图像放在 JPEGImages 文件夹下

3.在 ImageSets 文件夹中，新建三个空文件夹 Layout、Main、Segmentation，然后把写了训练或测试的图像的名字的文本拷到 Main 文件夹下，按目的命名，我这里所有图像用来训练，故而 Main 文件夹下只有 train.txt 文件。上面代码运行后会在图片文件夹内生成该文件，把它拷进去即可。

2 用 YOLOv2 训练

1.生成相关文件

按 darknet 的说明编译好后，接下来在 darknet-master/scripts 文件夹中新建文件夹 VOCdevkit，然后将整个 VOC2007 文件夹都拷到 VOCdevkit 文件夹下。

然后，需要利用 scripts 文件夹中的 voc_label.py 文件生成一系列训练文件和 label，具体操作如下：

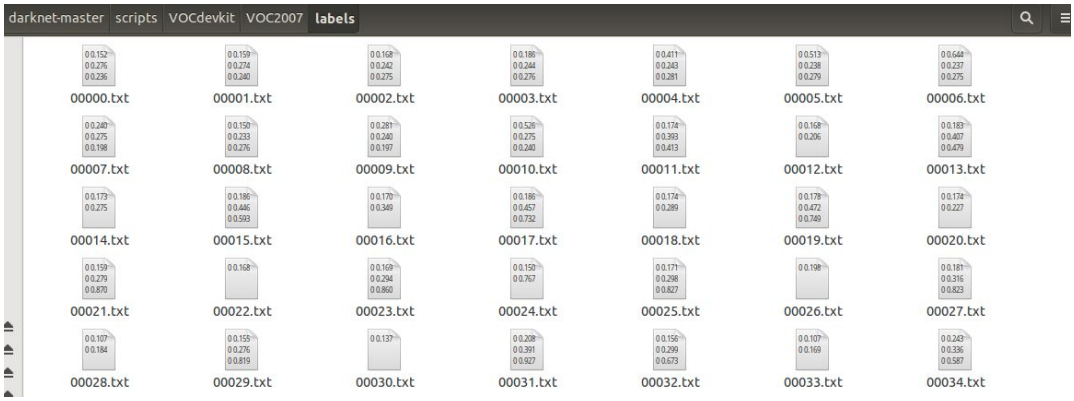
首先需要修改 voc_label.py 中的代码，这里主要修改数据集名，以及类别信息，我的是 VOC2007，并且所有样本用来训练，没有 val 或 test，并且只检测人，故只有一类目标，因此按如下设置

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

#sets=[('2012', 'train'), ('2012', 'val'), ('2007', 'train'), ('2007', 'val'), ('2007', 'test')]

#classes = ["aeroplane", "bicycle", "bird", "boat", "bottle", "bus", "car", "cat", "chair", "cow", "diningtable", "dog", "horse", "motorbike", "person", "pottedplant", "sheep", "sofa", "train", "tvmonitor"]

sets=[('2007', 'train')]
classes = [ "person"]


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(year, image_id):
    in_file = open('VOCdevkit/VOC%s/Annotations/%s.xml'%(year, image_id))  #(如果使用的不是VOC而是自设置数据集名字,则这里需要修改)
    out_file = open('VOCdevkit/VOC%s/labels/%s.txt'%(year, image_id), 'w')  #(同上)
    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')

wd = getcwd()

for year, image_set in sets:
    if not os.path.exists('VOCdevkit/VOC%s/labels/'%(year)):
        os.makedirs('VOCdevkit/VOC%s/labels/'%(year))
    image_ids = open('VOCdevkit/VOC%s/ImageSets/Main/%s.txt'%(year, image_set)).read().strip().split()
    list_file = open('%s_%s.txt'%(year, image_set), 'w')
    for image_id in image_ids:
        list_file.write('%s/VOCdevkit/VOC%s/JPEGImages/%s.jpg\n'%(wd, year, image_id))
```

```
        convert_annotation(year, image_id)
    list_file.close()
```

修改好后在该目录下运行命令：python voc_label.py，之后则在文件夹 scripts\VOCdevkit\VOC2007 下生成了文件夹 lable，该文件夹下的画风是这样的



这里包含了类别和对应归一化后的位置（i guess，如有错请指正）。同时在 scripts\ 下应该也生成了 train_2007.txt 这个文件，里面包含了所有训练样本的绝对路径。

2.配置文件修改

做好了上述准备，就可以根据不同的网络设置（cfg 文件）来训练了。在文件夹 cfg 中有很多 cfg 文件，应该跟 caffe 中的 prototxt 文件是一个意思。这里以 tiny-yolo-voc.cfg 为例，该网络是 yolo-voc 的简版，相对速度会快些。主要修改参数如下

[plain] view plain copy

```
1. .
2. .
3. .
4. [convolutional]
5. size=1
6. stride=1
7. pad=1
8. filters=30  //修改最后一层卷积层核参数个数，计算公式是依旧自己数据的类别数
   filter=num×（classes + coords + 1）=5×（1+4+1）=30
9. activation=linear
10.
11. [region]
12. anchors = 1.08,1.19,  3.42,4.41,  6.63,11.38,  9.42,5.11,  16.62,10.52
13. bias_match=1
14. classes=1  //类别数，本例为 1 类
15. coords=4
16. num=5
17. softmax=1
18. jitter=.2
```

```
rescore=1

object_scale=5
noobject_scale=1
class_scale=1
coord_scale=1

absolute=1
thresh = .6
random=1
```

另外也可根据需要修改 learning_rate、max_batches 等参数。这里歪个楼吐槽一下其他网络配置，一开始是想用 tiny.cfg 来训练的官网作者说它够小也够快，但是它的网络配置最后几层是这样的画风：

[html] view plain copy

```
[convolutional]
filters=1000
size=1
stride=1
pad=1
activation=linear

[avgpool]

[softmax]
groups=1

[cost]
type=sse
```

这里没有类别数，完全不知道怎么修改，强行把最后一层卷积层卷积核个数修改又跑不通会出错，如有大神知道还望赐教。

修改好了 cfg 文件之后，就需要修改两个文件，首先是 data 文件下的 voc.names。打开 voc.names 文件可以看到有 20 类的名称，本例中只有一类，检测人，因此将原来所有内容清空，仅写上 person 并保存，备注：若此处为多个类的训练，请同 voc_label.py 中顺序一致。

接着需要修改 cfg 文件夹中的 voc.data 文件。也是按自己需求修改，我的修改之后是这样的画风：

[plain] view plain copy

```
classes= 1  //类别数
train  = /home/kinglch/darknet-master/scripts/2007_train.txt  //训练样本的绝对路径文件，也就是上文 2.1 中最后生成的
//valid  = /home/pjreddie/data/voc/2007_test.txt  //本例未用到
```

```
4.  names = data/voc.names  //上一步修改的 voc.names 文件
5.  backup = /home/kinglch/darknet-master/results/  //指示训练后生成的权重放在哪
```

修改后按原名保存最好，接下来就可以训练了。
ps：yolo v1 中这些细节是直接在源代码的 yolo.c 中修改的，源代码如下

```
#include "network.h"
#include "detection_layer.h"
#include "cost_layer.h"
#include "utils.h"
#include "parser.h"
#include "box.h"
#include "demo.h"

#ifdef OPENCV
#include "opencv2/highgui/highgui_c.h"
#endif

char *voc_names[] = {"aeroplane", "bicycle", "bird", "boat", "bottle", "bus", "car", "cat", "chair",
"cow", "diningtable", "dog", "horse", "motorbike", "person", "pottedplant", "sheep", "sofa", "train",
"tvmonitor"};

void train_yolo(char *cfgfile, char *weightfile)
{
    char *train_images = "/data/voc/train.txt";
    char *backup_directory = "/home/pjreddie/backup/";
    srand(time(0));
    char *base = basecfg(cfgfile);
    printf("%s\n", base);
    float avg_loss = -1;
    network net = parse_network_cfg(cfgfile);
    if(weightfile){
        load_weights(&net, weightfile);
    }
```

比如这里的类别，训练样本的路径文件和模型保存路径均在此指定，修改后从新编译。而 yolov2 似乎摈弃了这种做法，所以训练的命令也与 v1 版本的不一样。

3.运行训练

上面完成了就可以命令训练了，可以在官网上找到一些预训练的模型作为参数初始值，也可以直接训练，训练命令为

```
./darknet detector train ./cfg/voc.data cfg/tiny-yolo-voc.cfg
```