**无人驾驶中的目标检测**

**实验指导书**

**指导者：**

**时 间：**

**地 点：**

[基于摄像头的目标检测实验](#_Toc5723_WPSOffice_Level1)

[一、 实验目的](#_Toc27156_WPSOffice_Level1) [1](#_Toc27156_WPSOffice_Level1)

[二、 实验原理](#_Toc7277_WPSOffice_Level1) [1](#_Toc7277_WPSOffice_Level1)

[三、 实验装置](#_Toc15543_WPSOffice_Level1) [2](#_Toc15543_WPSOffice_Level1)

[四、 实验设备和材料](#_Toc21577_WPSOffice_Level1) [2](#_Toc21577_WPSOffice_Level1)

[五、 实验内容](#_Toc2289_WPSOffice_Level1) [2](#_Toc2289_WPSOffice_Level1)

[六、 实验步骤](#_Toc2774_WPSOffice_Level1) [3](#_Toc2774_WPSOffice_Level1)

[七、 注意事项](#_Toc12224_WPSOffice_Level1) [8](#_Toc12224_WPSOffice_Level1)

[八、 问题讨论](#_Toc25253_WPSOffice_Level1) [8](#_Toc25253_WPSOffice_Level1)

[九、 实验结果](#_Toc16571_WPSOffice_Level1) [10](#_Toc16571_WPSOffice_Level1)

**基于摄像头的目标检测实验**

# 实验目的

1. 了解LabelImage数据标注方法
2. 了解YOlOv3（You Only Look Once）目标检测算法
3. 运用YOLOv3实时目标检测算法解决驾驶视频目标检测问题。

# 实验原理

针对目标检测算法受环境条件影响鲁棒性差、小目标识别能力不高的问题，建立了涵盖多种天气环境、包含疑难目标的驾驶视频样本数据库，提出了疑难样本训练方法，训练出可在多种天气环境中良好识别小型汽车、行人、公交车、货车、摩托车、自行车和停车标志的YOLOv3检测模型。实验结果表明，该训练方法可有效提升目标检测性能；所得检测模型具有较高的召回率和精确度，可初步应用于无人驾驶汽车前向防碰撞系统中的实时驾驶视频的目标检测。

1. 前向防碰撞决策系统技术路线

使用单目视觉方法实现无人驾驶汽车前向碰撞控制的研究较为成熟，常见的方案由五部分主城：摄像机采集图像；目标检测；目标距离估计；形成防碰撞策略；执行控制。目标检测作为该技术的重要环节，其检测精度、检测速度对汽车行驶安全性影响大。运用YOLOv3检测算法对行驶环境中的目标进行识别，有助于实现实时地目标检测。

1. YOLOv3检测系统

YOLOv3检测系统的预测任务主要分为两部分：分类预测和位置预测。

输入图片首先经尺寸调整并被划为N×N的正方形栅格，包含目标中心的栅格单元负责检测该目标。输入图片通过YOLOv3神经网络计算后输出含有分类信息的特征栅格图。YOLOv3的核心网络框架为具有53层卷积层的特征提取网络Darknet-53.

目标位置预测运用K-means聚类方法获得锚点框（Anchor Box）尺寸，通过对锚点框尺寸进行数学变换获得预测框（Bounding Box）尺寸。YOLOv3使用跨尺度预测机制（Predictions Across Scales）——在三种尺度的特征图上执行预测，融合多尺度采样特征。

1. 建立驾驶视频样本数据库

车辆行驶环境的开放性增加了目标检测的难度，亮度变化、天气情况（雷雨天气等）、交通路况（并道河流、转弯、上下坡等）等对检测性能影响大。通过行车记录仪获取驾驶视频样本图片，涵盖上述复杂环境及路况。

检测目标类别为8类：行人（person）、自行车（bicycle）、小型汽车（car）、摩托车（motorbike）、公交车（bus）、货车（truck）、交通灯（traffic light）、停车标志（stop sign）。使用LabelImg工具添加样本标签，建立基础数据库。训练样本选取具有良好特征的目标进行标记，测试数据则需对样本图片中的所有目标进行标记，完成基础样本数据库的建立。

1. 网络模型训练

以Darknet53为网络框架，YOLOv3为检测模型，训练驾驶视频目标检测系统。

为提高训练效率，选择开源的预训练权重darknet53.conv.74.weights初始化卷积层。此预训练权重通过Darknet53网络模型在ImageNet数据集上训练多个循环后获得。设置冲量稀疏为0.9，权值衰减系数为0.0005，最大训练批次为50000步。学习率使用分布调整策略（由10-4至10-6），避免训练出现过拟合。

1. 训练技巧

批样本数量（Batch）与样本分支数量（Subdivision）之比为单个分支中含有的数据量（Subdivision Size），该数据量大小受到硬件计算能力的限制。适当减小批样本数量和样本分支数量可减少硬件计算负担，同时仍可获得良好的模型检测性能。

为提升模型检测精度，更改输入图像分辨率适应大尺寸图像的目标识别。驾驶视频中图像的尺寸普遍较大，将输入图像尺寸由默认的416\*416像素调整为544\*544像素。检测系统对大输入图片尺寸的检测速度低但精度高。执行跨尺度预测时，调整后的模型得到密度更高的特征栅格图。

# 实验装置



# 实验设备和材料

1. 硬件：PC机一台
2. 软件：Ubuntu系统，darknet环境开发
3. 配置GPU驱动等：cuda+cudnn

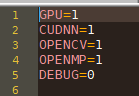
# 实验内容

1. 搜集数据
2. 标注数据
3. 训练数据
4. 调试工程代码

# 实验步骤

1. 下载工程
2. git clone <https://github.com/pjreddie/darknet.git>
3. 编译工程

若使用GPU和OpenCV，在编译工程前需要修改Makefile文件，将GPU=1，CUDNN=1，OPEENCV=1，



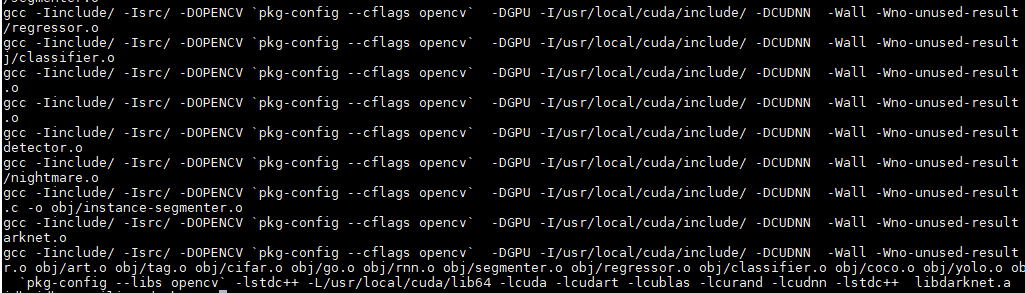
编译：



如果遇到缺失包错误，将缺失的包安装上去以后重新编译，重复以下两个操作，直到所有包都安装好：



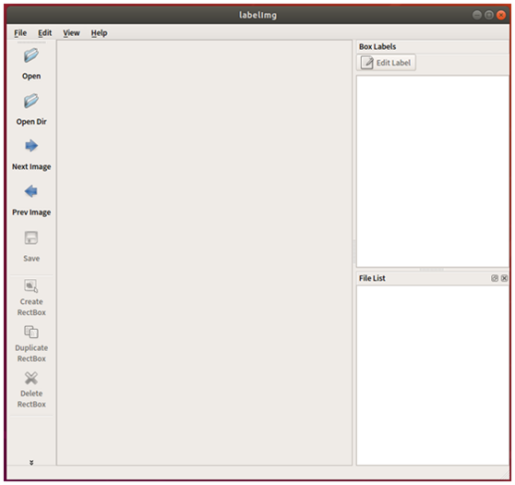
最终编译成功后出现下列信息：



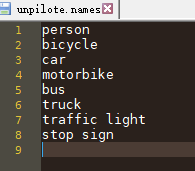
1. 数据集的制作：

工具：LabelImge

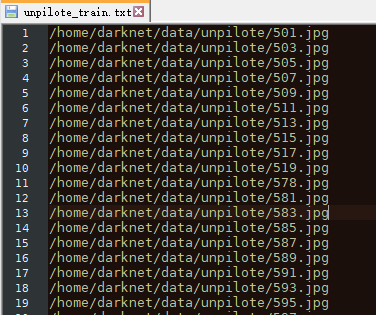
生成YOLO格式的TXT标注文件，一幅图像对应一个.txt文件。



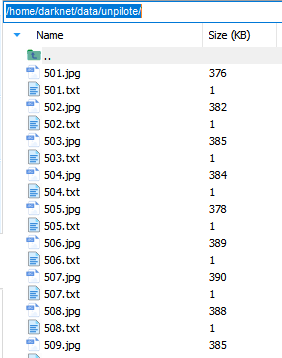
1. 文件的修改：
2. 关于data/unpilote.names文件的修改。



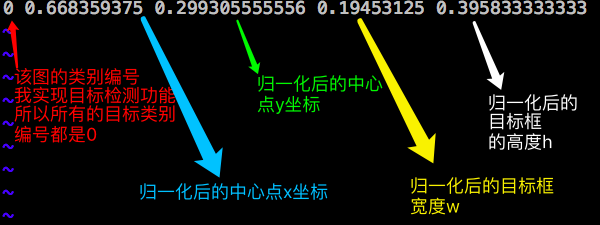
1. 将标注的数据集里的80%作为训练数据集，20%作为测试数据集，训练集（unpilote\_train.txt）和测试集（unpilote\_test.txt）位置文件，存放图片的位置，这两个.txt文件存放的路径没有严格要求，放在任意自己觉得舒服的位置，后面在工程的.data文件中指定这两个文件的路径即可。（经验之谈：将图片直接选中拖动便可在.txt文件中生成路径）



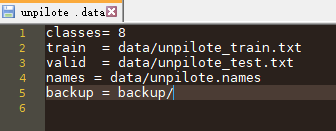
在当前目录，每张图片都会有对应的label文件，在/home/darknet/data/unpilote/文件夹下：图片和对应的label文件



标注的.txt文件的数据格式如下图所示：



1. 关于cfg/unpilote.data文件的修改



classes:类别数

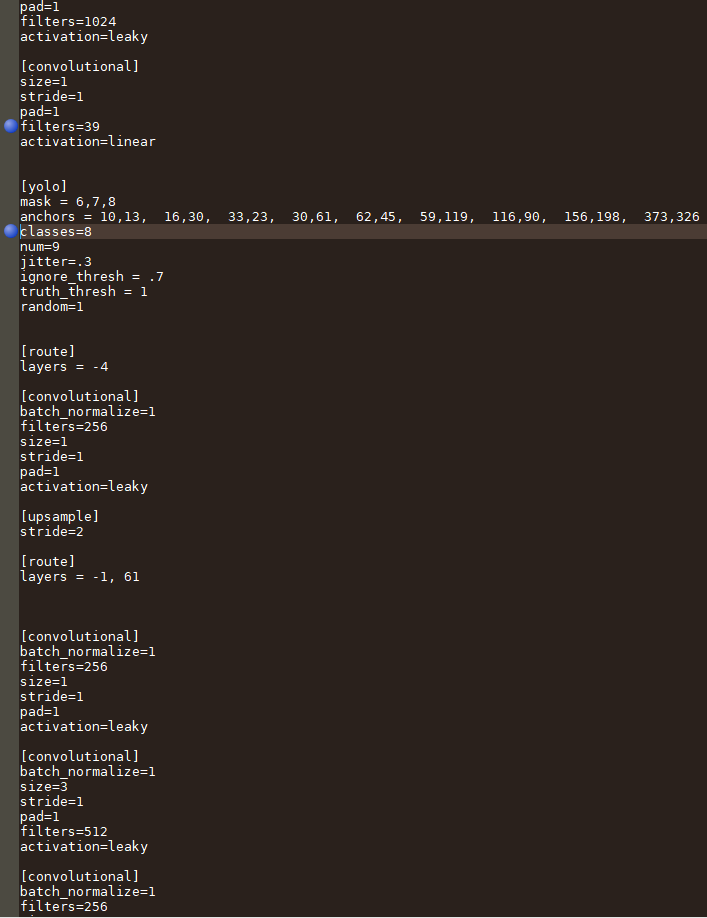
train： 前面说的训练数据的路径

test: 前面说的测试数据的路径

names: 指向存放类别名的文件，自己训练数据的类别名就放到这个指定的文件中，自己新建一个.names文件，一行放一个类名

backup: 模型存放的路径

1. 关于网络配置文件cfg的修改，（在cfg/yolov3.cfg或者cfg/yolov3-tiny.cfg的基础上修改）。以八类目标检测为例，主要有以下3处调整。3处[yolo]前一层filters大小（即最后一层卷积层，filters= 3\*（classes+4+1））和classes。



anchors：预选框，可以手工挑选，也可以通过k-means从训练样本中学出。

num：预选框的个数，即anchors总数

mask:当前属于第几个预选框

classes：网络需要识别的物体种类数（这里训练了8类，所以填8）

jitter：通过抖动增加噪声来抑制过拟合

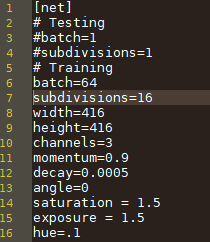
ignore\_thresh: 过滤阈值

random：设置为0，表示关闭多尺度训练（显存小可以设置为0）

yolo层钱满的卷积层和filters数目是怎么计算的：3\*（classes+4+1）3是mask的参数，4表示四个参数（预测boundingbox的：x,y,w,h）,1表示置信度。和聚类数目分布有关。如果想修改默认anchors数值，使用k-means即可。

1. 训练

把cfg/unpolite.cfg中training的参数设置打开，testing的参数设置关闭



直接从头开始训练（开始loss比较大，慢慢降下去）



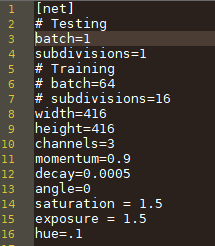
训练的过程中backup目录里面每到一定迭代次数，会有模型参数文件保存下来（比如yolov3\_unpolite\_1000.weights）

如果中断训练想要继续前面的模型训练:



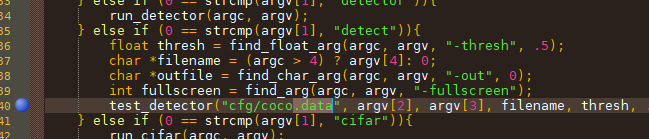
1. 测试

把cfg/unpolite.cfg中testing的参数设置打开，training的参数设置关闭



改源码，需要重新编译（先make clean，然后再make）

原因是作者在代码中设置了默认，所以修改examples/darknet.c文件，将断点指示的文件都替换成自己的。然后重新编译即可。



运行指令：

测试摄像头：

./darknet detector demo cfg/unpolite.data cfg/unpolite.cfg backup/yolov3\_unpolite.weights -c 1



测试视频：

./darknet detector demo cfg/unpolite.data cfg/unpolite.cfg backup/yolov3\_unpolite.weights x.avi

# 注意事项

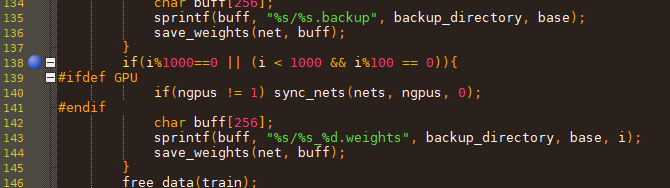
1. 模型保存问题

A. 保存模型出错：一般是.data文件中指定的文件夹无法创建，导致模型保存时出错。自己手动创建即可。

B. 模型什么时候保存？如何更改

迭代次数小于1000时，每100次保存依次，大于1000时，每10000次保存一次。

代码位置：examples/detector.c line138



1. 使用预训练模型直接保存问题

Darknet53.conv.74作为预训练权重文件，因为只包含卷积层，所以可以从头开始训练。

# 问题讨论

1、测试时出现的问题

CUDA Error: out of memory

darknet: ./src/cuda.c:36: check\_error: Assertion `0' failed.

解决办法：

这是因为在yolov3的cfg文件中目前开启的是train模式，batchsize为64，要做的是 vim cfg/unpolite.cfg 按照下面步骤即可

# Testing

batch=1 取消注释

subdivisions=1

# Training

#batch=64 注释掉

#subdivisions=16

2、Loading weights from yolo.weights...Done!

CUDA Error:invalid device function

darknet: ./src/cuda.c:21: check\_error: Assertion `0' failed.

Aborted (core dumped)

这是因为配置文件Makefile中配置的GPU架构和本机GPU型号不一致导致的。

更改前默认配置如下（不同版本可能有变）：

ARCH= -gencode arch=compute\_30,code=sm\_30 \

-gencode arch=compute\_35,code=sm\_35 \

-gencode arch=compute\_50,code=[sm\_50,compute\_50] \

-gencode arch=compute\_52,code=[sm\_52,compute\_52]

# -gencode arch=compute\_20,code=[sm\_20,sm\_21] \ This one is deprecated?

# This is what I use, uncomment if you know your arch and want to specify

# ARCH= -gencode arch=compute\_52,code=compute\_52

compute\_30表示显卡的计算能力是3.0，几款主流GPU的compute capability列表：

|  |  |  |
| --- | --- | --- |
| 显卡 | 参数 | 计算能力 |
| Tesla K10 | compute\_30 | 3.0 |
| Tesla K20 | compute\_35 | 3.5 |
| Tesla K40 | compute\_35 | 3.5 |
| Tesla K80 | compute\_37 | 3.7 |
| GTX Titan X | compute\_52 | 5.2 |
| GTX 980 | compute\_52 | 5.2 |
| Jetson TX2 | compute\_62 | 6.2 |
| Jetson TX1 | compute\_53 | 5.3 |
| Jetson TK1 | compute\_32 | 3.2 |

根据自己的GPU型号的计算能力进行配置，例如配置为：

ARCH= -gencode arch=compute\_35,code=compute\_35 \

然后重新编译即可。

1. 实验结果

# 41

