# 站台停车位置识别与测距

## 1.总体方案

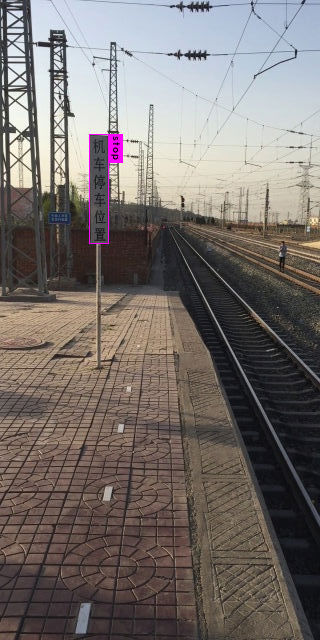
输入为车载摄像头记录的每一帧视频图像数据；

利用深度学习技术检测出停车牌大小（宽度、高度）；

根据相似三角形原理计算停车牌到相机的距离；

输出实时停车牌位置与相机到停车牌距离；

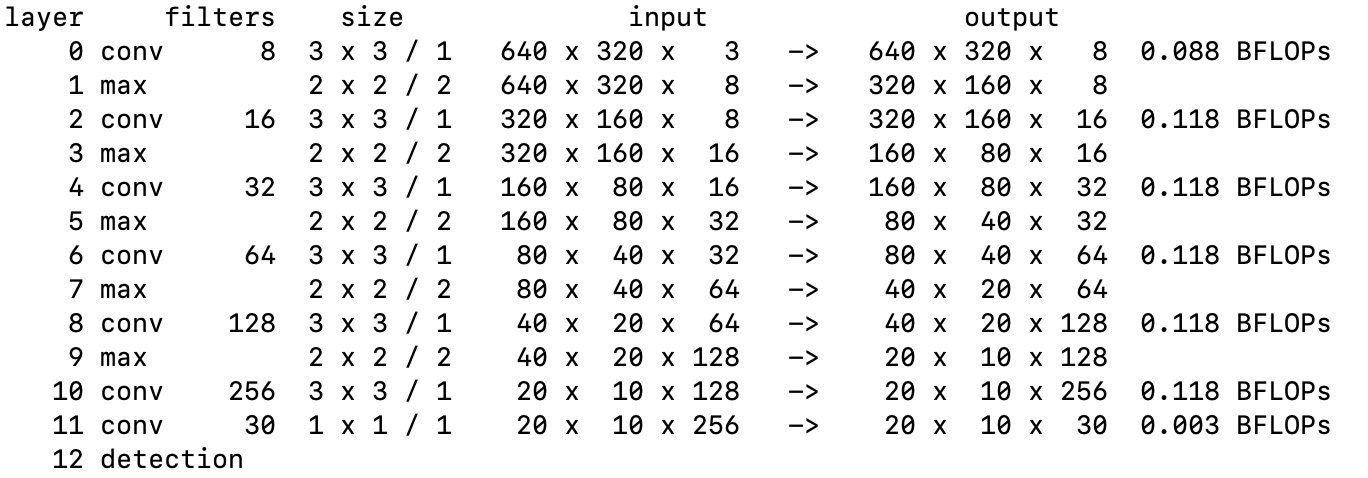
## 2.目标检测算法：



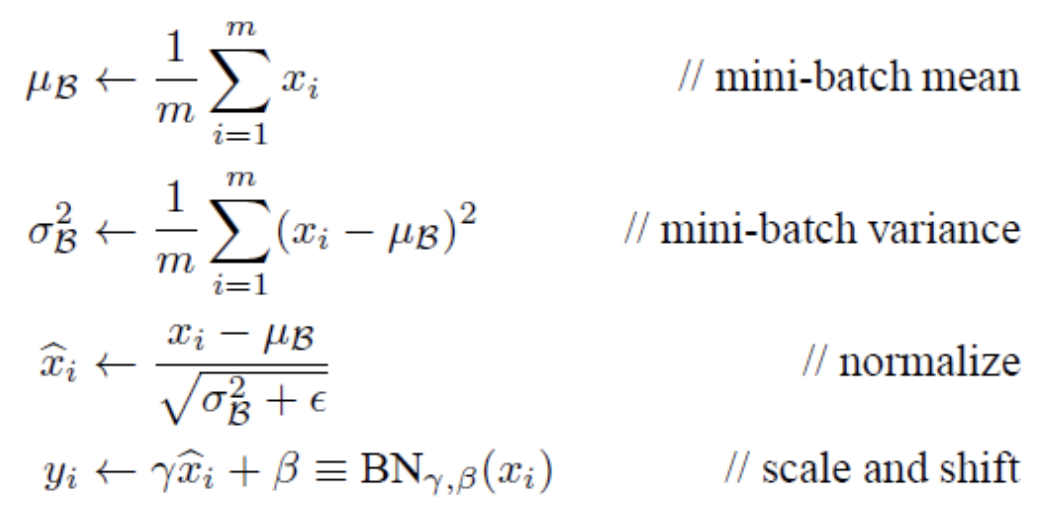
我们采用的深度学习目标检测算法是精简版的YOLO网络(You Only Look Once: Unified, Real-Time Object Detection)。这是一个很常用的目标检测算法，特点是模型参数量小，检测速度快，算法框架（darknet）整体是用C语言写的，很适合在嵌入式平台运行。YOLO将目标检测作为一个回归问题进行求解，输入图像经过一次推理，便能得到图像中所有物体的位置和其所属类别及相应的置信概率。

### 2.1特征提取网络

停车牌检测网络只有特征提取部分和检测部分，结构如下图所示。



其中，其中特征提取部分包括卷积层（conv）用来提取图像特征，卷积操作就是根据卷积窗口，进行像素的加权求和。类似于传统图像处理中边缘检测算子等，只是参数未知，需要通过神经网络训练得到，1x1卷积层用于跨通道信息整合。池化层（max）用于图像下采样，使得卷积神经网络每一层的构建类似图像金字塔结构，增大感受野，并降低特征维度。每个卷积层后面都添加了Batch Normalization层。Batch Normalization可以提升模型收敛速度，而且可以起到一定正则化效果，降低模型的过拟合。Batch Normalization前向传导过程公式：

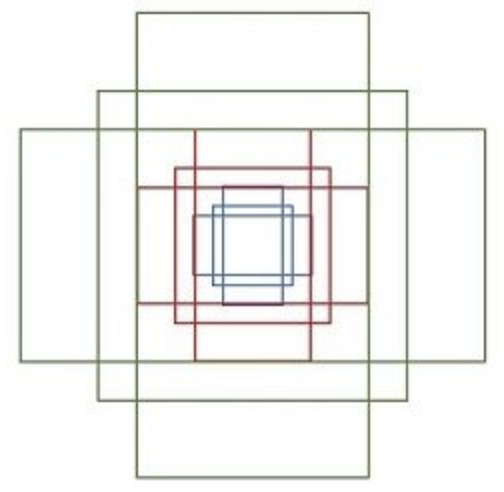


其中为可学习参数，，防止分母为0。

参考材料：<https://arxiv.org/pdf/1409.1556.pdf>

### 2.2 物体分类与坐标回归

检测部分（detection）用来预测物体位置和类别概率值。这里输入为20x10x30的特征图，其中20x10为宽x高，30为通道数。检测层使用候选框机制（anchor boxes），每个格子包含5个候选框，预测时每个候选框都单独预测一套分类概率值和5个数据值，分别是x,y,w,h,和置信度（confidence）。其中x,y是指当前格子预测得到的预测框（bounding box）的中心位置坐标。w,h是预测框的宽度和高度。

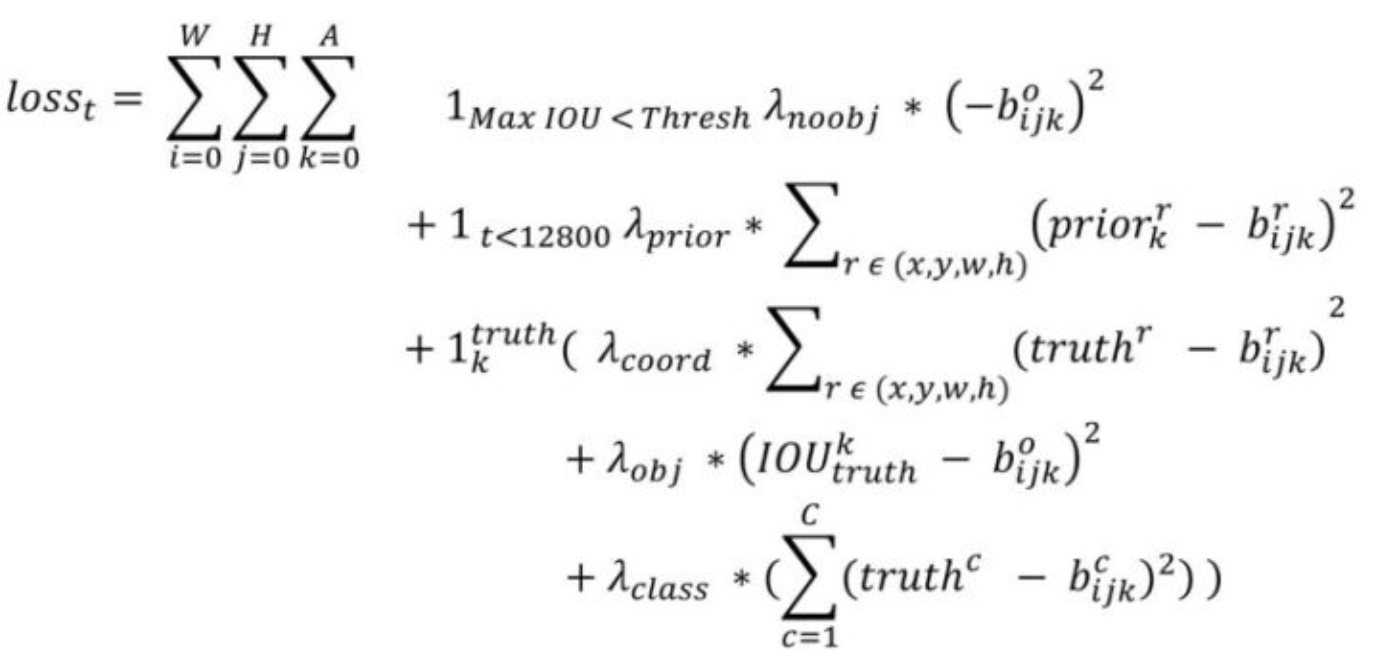


注意：实际训练过程中，w和h的值使用图像的宽度和高度进行归一化到[0,1]区间内；x，y是预测框中心位置相对于当前格子位置的偏移值，并且被归一化到[0,1]。confidence反映当前预测框是否包含物体以及物体位置的准确性，计算方式如下：

其中，若物体框（ground truth）中心落在该格子内，则说明格子对应的预测框包含物体，= 1；否则 = 0。IOU(intersection over union)为预测框与物体框的交集面积（以像素为单位，用真实区域的像素面积归一化到[0,1]区间）。

参考材料：<https://arxiv.org/pdf/1506.01497.pdf>

### 2.3损失函数：



首先 W、H分别指的是特征图（20x10x30）的宽与高，而 A指的是侯选框种类（这里是5），各个 值是各个loss部分的权重系数。第一项loss是计算背景（background）的置信度误差，需要先计算各个预测框和所有物体框的IOU值，并且取最大值Max\_IOU，如果该值小于一定的阈值（这里是0.6），那么这个预测框就标记为背景，需要计算没有包含物体的预测框置信度误差。第二项是计算侯选框与预测框的坐标误差，但是只在前12800个iterations间计算，用于在训练前期使预测框快速学习到侯选框的形状。第三大项计算与某个物体框（ground truth）匹配的预测框各部分loss值，包括坐标误差、置信度误差以及分类误差。先说一下匹配原则，对于某个物体框，首先要确定其中心点要落在哪个格子（cell）上，然后计算这个cell的5种候选框与物体框的IOU值，计算IOU值时不考虑坐标，只考虑形状，所以先将候选框与物体框的中心点都偏移到同一位置（原点），然后计算出对应的IOU值，IOU值最大的那个候选框与物体框匹配，对应的预测框用来预测这个物体。在计算目标置信度时，取该预测框与物体框的IOU值。对于那些没有与物体框匹配的侯选框（与预测框对应），全部忽略，不计算误差。即每一个物体框只会与一个候选框匹配（IOU值最好的）。

参考材料：<https://arxiv.org/pdf/1612.08242.pdf>

### 2.4 参数更新

反向传播算法：<http://jermmy.xyz/2017/12/16/2017-12-16-cnn-back-propagation/>

参数更新算法：<https://blog.csdn.net/weixin_42398658/article/details/84525917>

此过程与目标问题无关，与所用方法无关，无需改动。

## 3.计算目标距离

我们使用相似三角形来计算相机到停车牌的距离。相似三角形是指假设我们有一个宽度为 W 的物体，然后我们将这个目标放在距离我们的相机为 D 的位置。我们用相机对物体进行拍照并且测量物体的像素宽度 P 。这样我们就得出了相机焦距的公式：

当我继续将我的相机移动靠近或者离远物体或者目标时，我可以用相似三角形来计算出物体离相机的距离：

这里我们假设通过预先拍照，已知距离根据照片算出摄像头的焦距，再根据已知的焦距算出接下来的视频图像中停车牌到摄像机的距离。

## 4. 实现流程

1.用labelImg给图片做标注，生成xml格式标注文件。（<https://github.com/tzutalin/labelImg>）

2．整理数据集为VOC格式：

文件夹目录：

darknet:

---VOCdevkit:

------VOC2019:

---------JPEGImages

---------Annotations:

---------ImageSets:

------------Main:

其中，JPEGImages存放图片，Annotations存放xml标注，Main存放数据集划分：train.txt，val.txt，test.txt，里面的txt内容格式为图片名（不带后缀）。

修改voc\_label.py(根据自己情况修改)，运行voc\_label.py，会自动在VOCdevkit同目录下生成5个txt文件：2019\_train.txt，2019\_val.txt，2019\_test.txt，train\_all.txt，train.txt，里面记录了图片位置路径。

3.修改YOLO中配置文件，stop/stop.data：数据集位置，stop / stop.names：物体类别名称，stop / stop.cfg：网络结构配置，其中训练时注释掉testing，打开train，subdivisions=8；测试时注释掉train，打开testing，subdivisions=1；

4. 在darknet目录下载权重文件：wget <https://pjreddie.com/media/files/darknet53.conv.74>

5．训练：

nohup ./darknet detector train stop/stop.data stop/stop.cfg darknet53.conv.74 2>1 | tee visualization/stop.log &

6. 测试：

./darknet detector test stop/stop.data stop/stop.cfg backup/ stop\_final.weights