
YOLOv3目标检测实战
训练自己的数据集

目标检测 (Object Detection) = What, and Where

定位 Localization

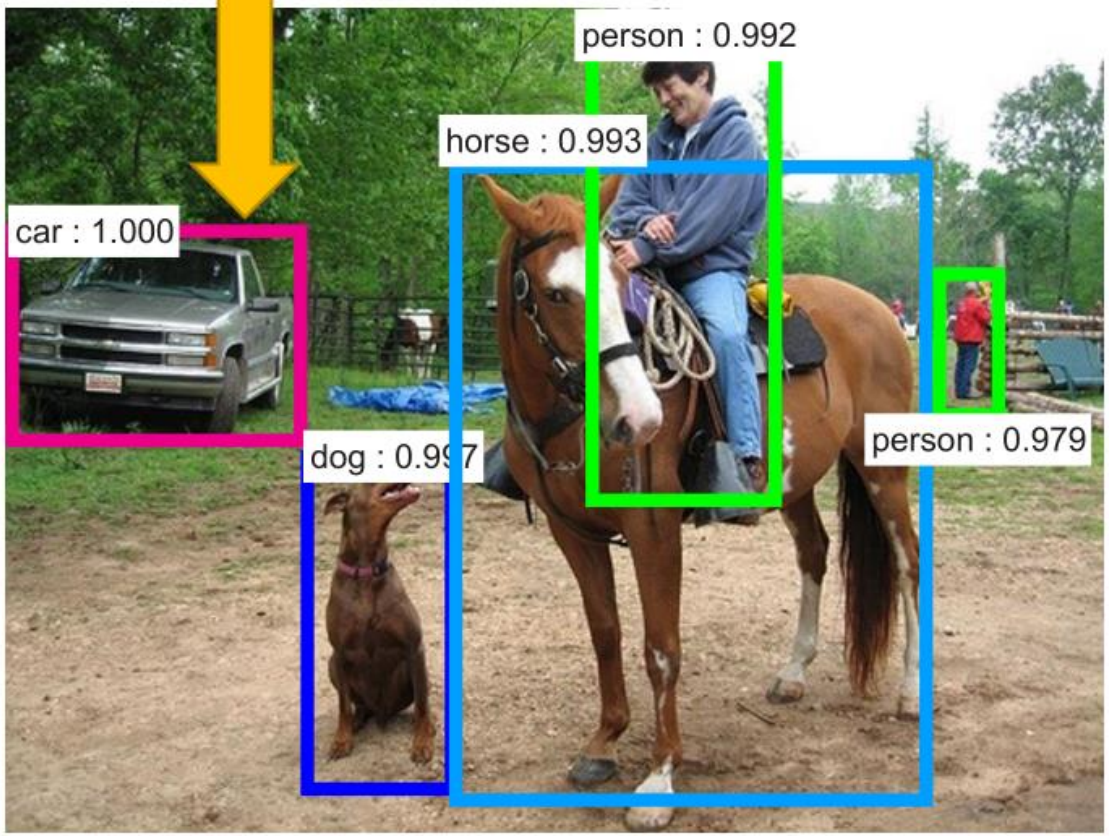
Where?

位置（最小外接矩形，Bounding box）

识别

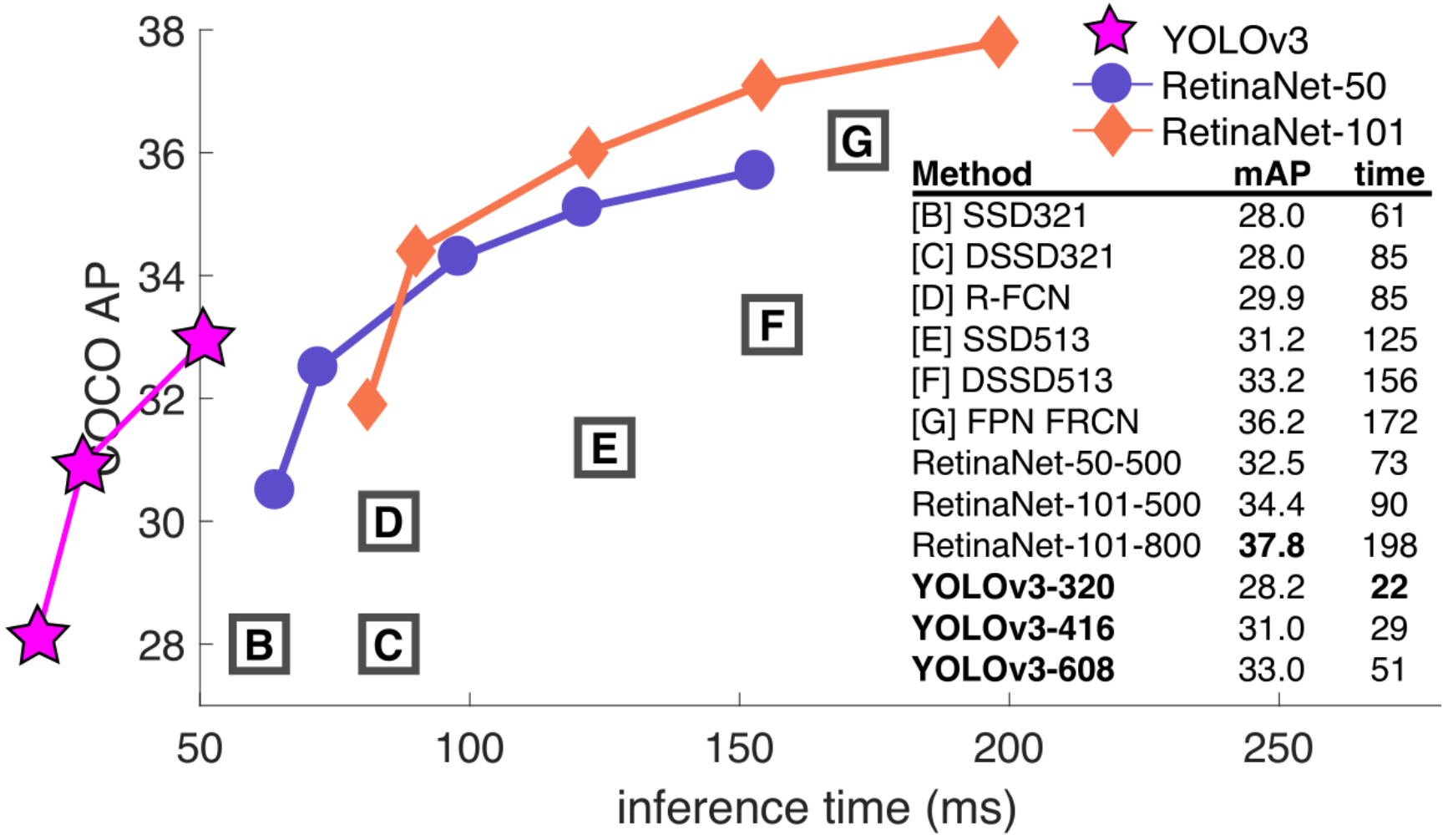
Recognition

What?



类别标签（Category label）

YOLOv3



Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

YOLO算法的基本思想

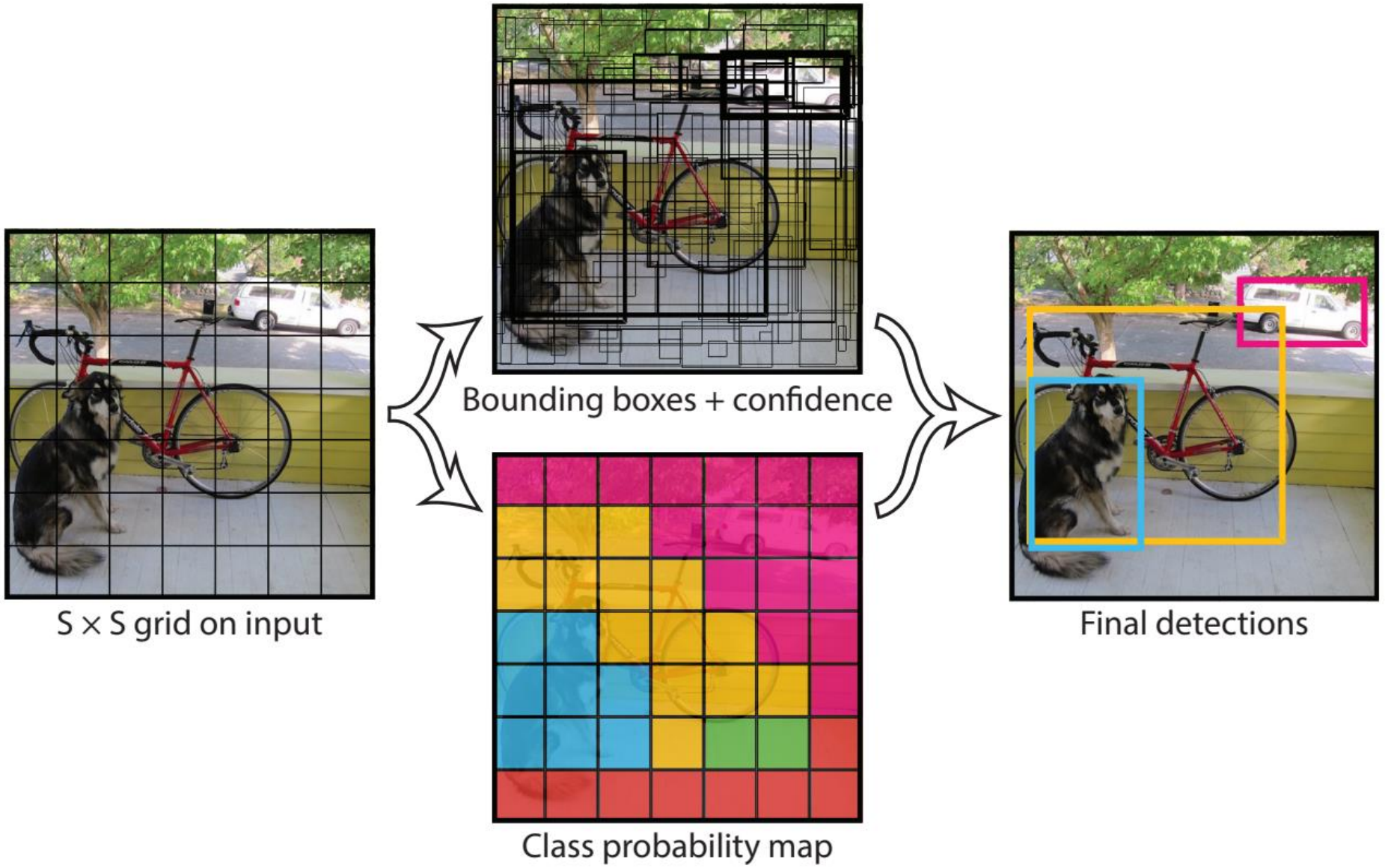
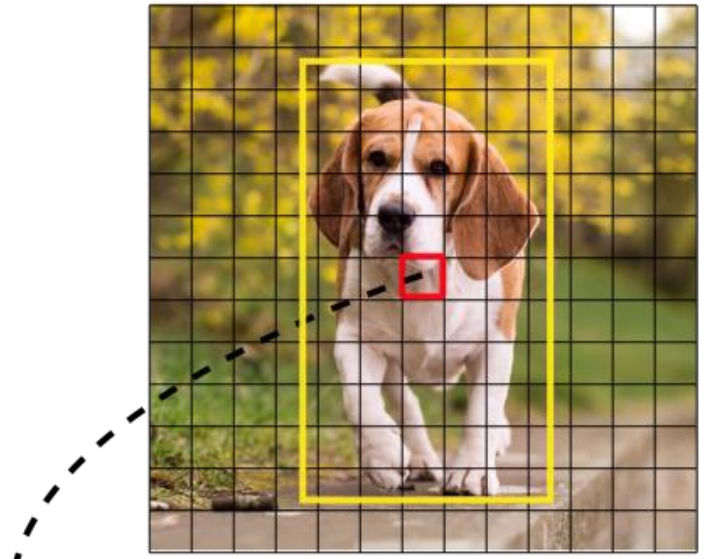
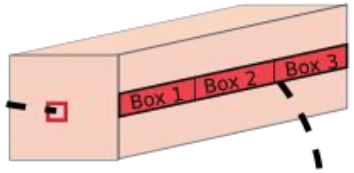


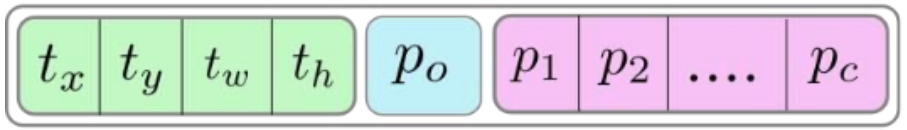
Image Grid. The Red Grid is responsible for detecting the dog



Prediction Feature Map



Attributes of a bounding box



x B

Box Co-ordinates

Objectness Score

Class Scores

Darknet

一个轻型的开源深度学习框架

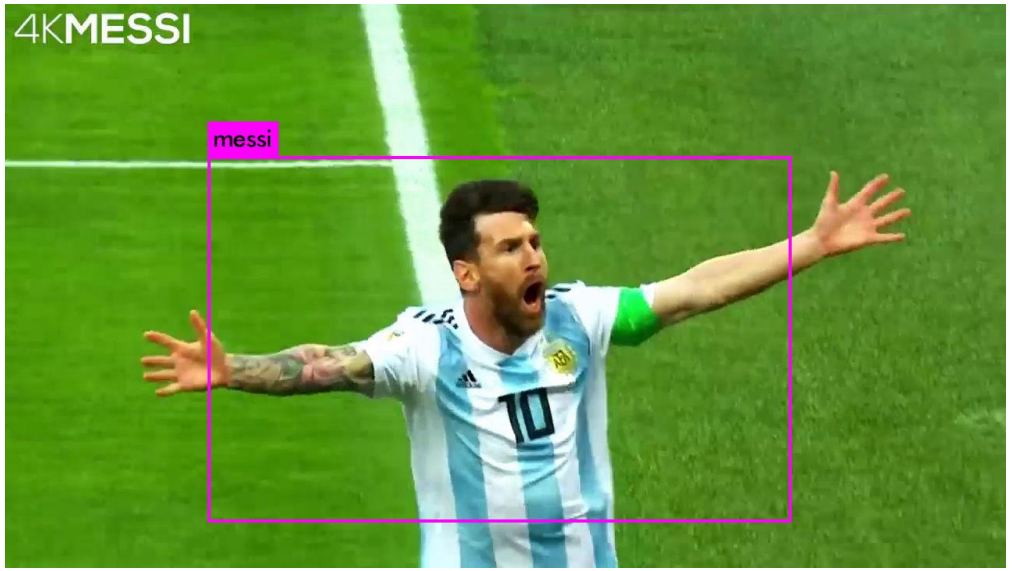
功能：CNN底层实现；YOLO目标检测；RNN；图像实例分割等

特点：

- 由C语言实现
- 没有依赖项 （摄像头和视频处理需要OpenCV）
- 容易安装
- 移植性好
- 支持CPU与GPU(CUDA)两种计算方式

目标检测：足球

目标检测：梅西



目标检测：足球和梅西

项目流程

1. 安装darknet

2. 给自己的数据集打标签

3. 整理自己的数据集

4. 修改配置文件

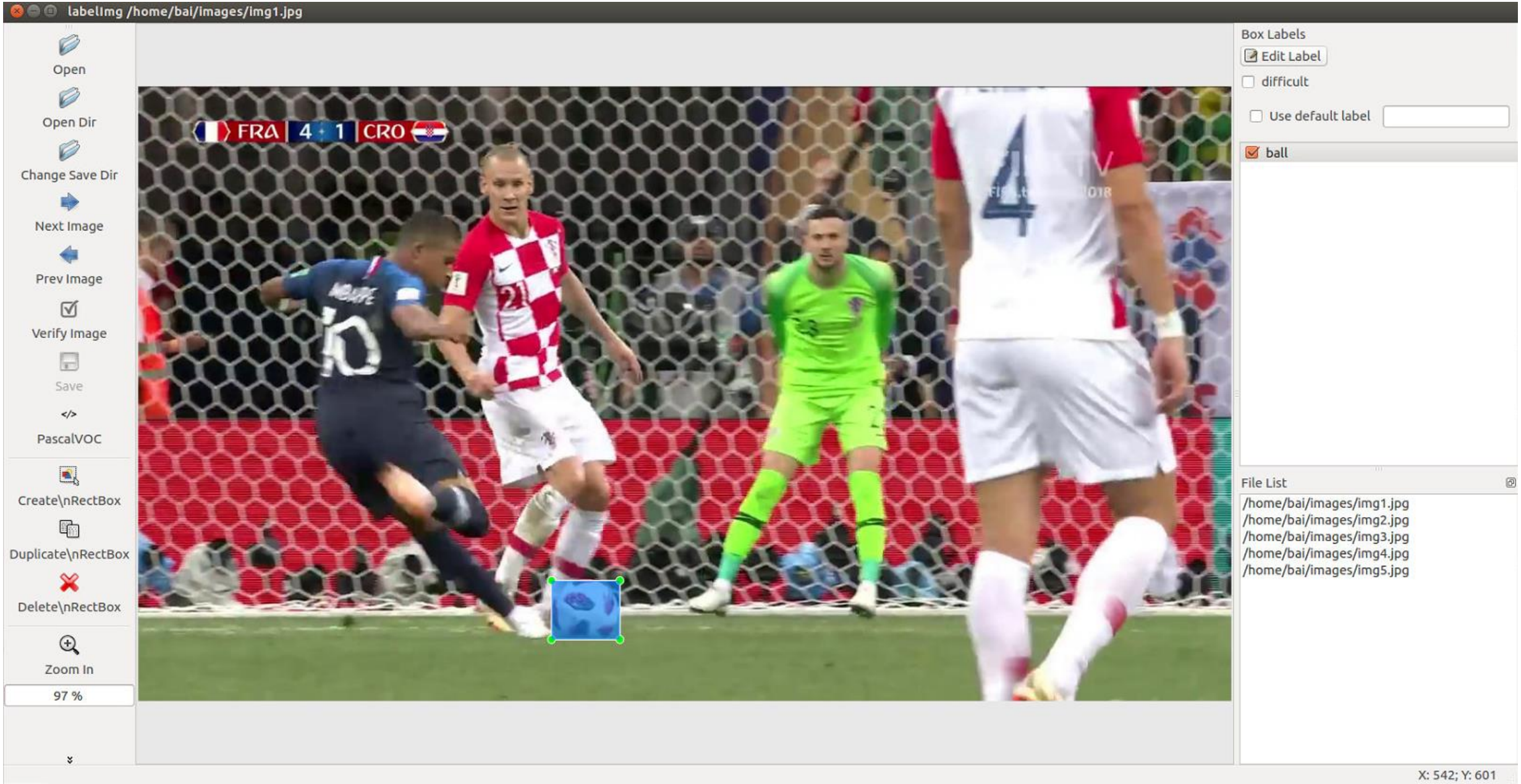
5. 训练自己的数据集

6. 测试训练出的网络模型

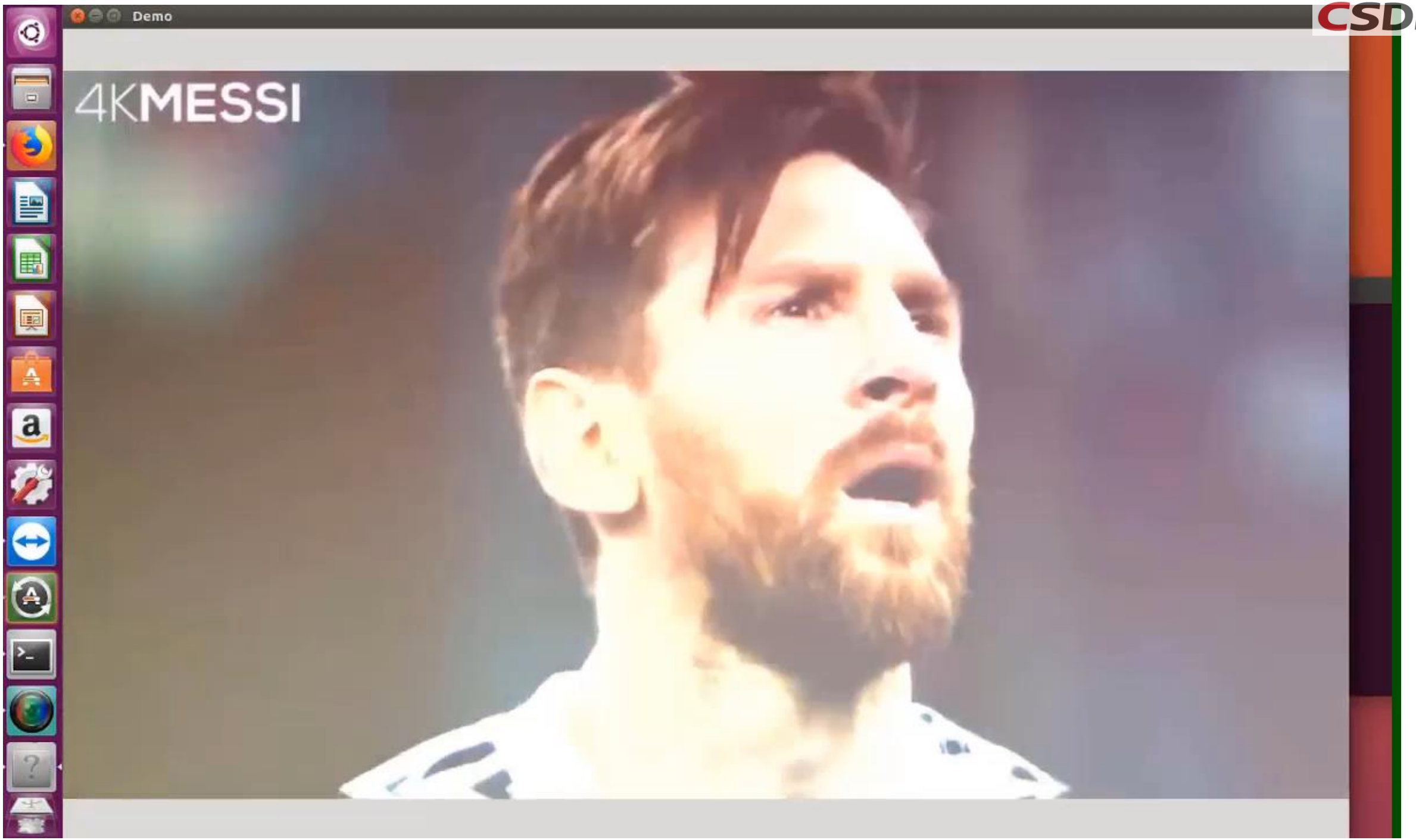
7. 性能统计

8. 先验框聚类与修改

图像标注工具：labelImg




X
Y
$x_{min} = 187$
$y_{min} = 21$
295,324
$x_{max} = 403$
$y_{max} = 627$


Demo
4KMESSI


YOLOv3目标检测实战
训练自己的数据集


YOLOv3目标检测实战
交通标志识别


YOLOv3目标检测
原理与源码解析


YOLOv3目标检测
网络模型改进方法

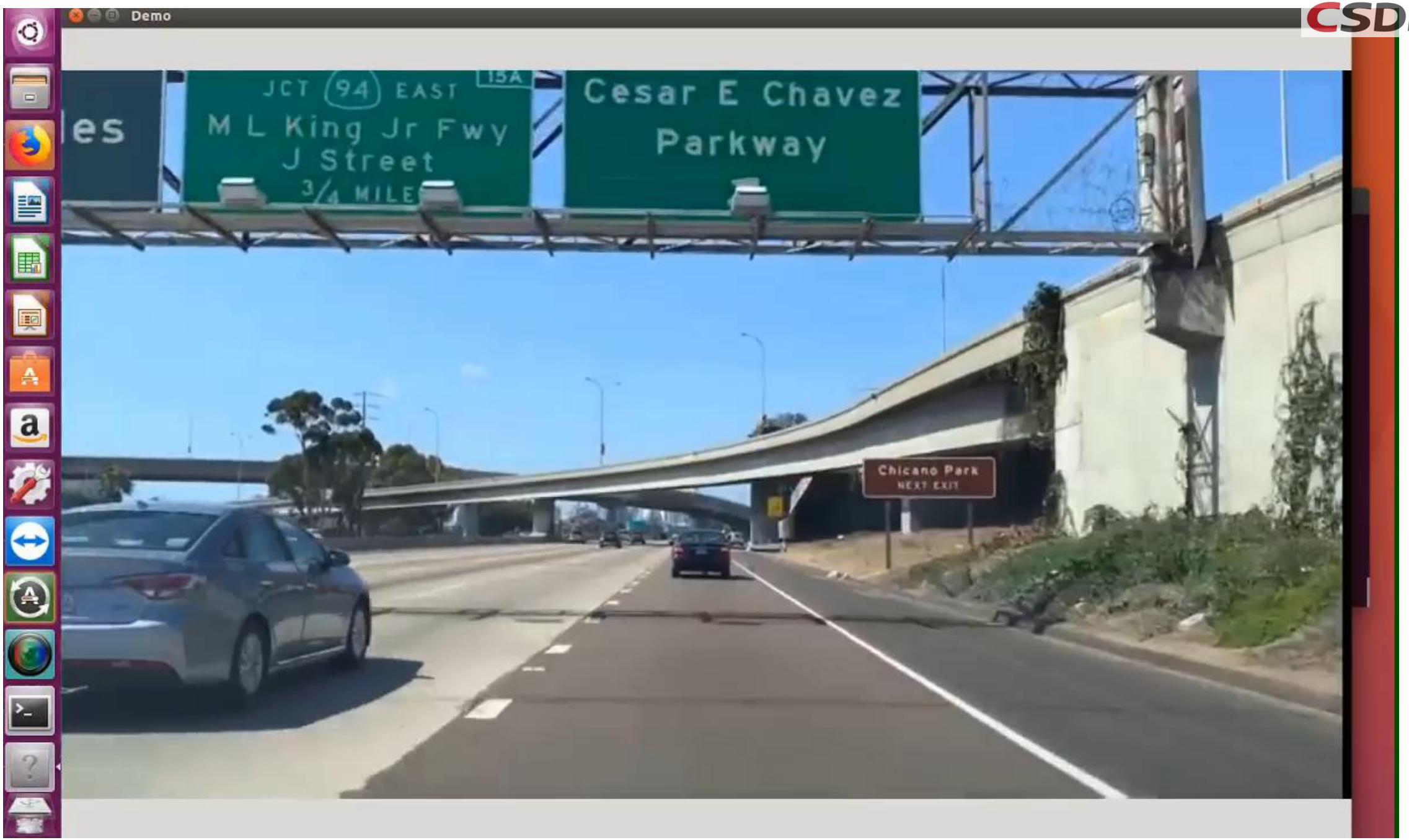

YOLOv3目标检测实战
交通标志识别


speedLimit
35


signalAhead


pedestrianCrossin
HARD ROCK
RIGHT SIDE BRICK BUILDING


stop
STOP


Demo
JCT 94 EAST
15A
M L King Jr Fwy
J Street
3/4 MILE
Cesar E Chavez
Parkway
Chicano Park
NEXT EXIT


YOLOv3目标检测
原理与源码解析

```
/*
** 卷积神经网络反向传播函数
** 主要流程：1）  调用gradient_array()计算当前层l所有输出元素关于加权输入的导数值（也即激活函数关于输入的导数值），
**               并乘上上一次调用backward_convolutional_layer()还没计算完的l.delta，得到当前层最终的误差项；
**           2）  如果网络进行了BN，则；
**           3）  如果网络没有进行BN，则直接调用 backward_bias()计算当前层所有卷积核的偏置更新值；
**           4）  依次调用im2col_cpu()，gemm_nt()函数计算当前层权重系数更新值；
**           5）  如果上一层                                    则依次调用gemm_tn()，col2im_cpu()计算上一层的误差项                                    个步骤）；
** 强调：每次调用本函数会计算                                    同时计算当前层的偏置、权重更新值，除此之外，还会计算上                                    的是，
**       并没有完全计算完，这                                    权输入的导数值。这一步在下一次调用本函数时完成。
*/
void backward_convo                                    volutional_layer l, network net)
{
    int i, j;
    int m = l.n/l.groups;
    // 每一个卷积核元素个数                                    受的输入图片的通道数）个通道上的卷积核元素个数总数，
    // 输入图片有3个通道，因                                    道上，所以实际上这个卷积核是立体的，共有3*3*3=27个元                                    数）
    int n = l.size*l.size
    int k = l.out_w*l.out                                    素个数：out_w，out_h是输出特征图的宽高

    // 计算当前层激活函数对                                    lta相应元素，从而完成当前层误差项的计算，得到当前层的
    // l.output存储了该层网                                    一个batch的输入图片，其中每张图片经卷积处理后得到的特                                    _h，
    // 该层卷积网络共有l.n个                                    卡输出l.n张宽高为l.out_w，l.out_h的特征图（l.outputs为                                    素个数），
    // 所以所有输入图片也即                                    l.n*l.out_                out_h*l.batch；
    // l.delta是一个一维数组                                    ts（其中                                    t_c），在                                    )动态分配内存；
    // 再强调一次：gradient_                                    数对输入h                                    最后一步                                    括函数对输入的导数
    // （注意gradient_arry中                                    成当前层误差项的计算，同时会计算上一层的误差项，但对于                                    全计算完成，还差一步，
    // 每次调用backward_conv                                    ()时来完成，诚如col2im_cpu()中注释一样。
    // 需要等到下一次调用bac
    gradient_array(l.outp                                    l.activation, l.delta);

    if(l.batch_normalize)
        backward_batchnor
    } else {
        // 计算偏置的更新值：  对应公式(Conv-2)
        // 每个卷积核都有一个偏置，偏置的更新值也即误差函数对偏置的导数，这个导数的计算很简单，实际所有的导数已经求完了，都存储在l.delta中，
        // 接下来只需把l.delta中对应同一个卷积核的项加起来就可以（卷积核在图像上逐行逐列跨步移动做卷积，每个位置处都有一个输出，共有l.out_w*l.out_h个，
        // 这些输出都与同一个偏置关联，因此将l.delta中对应同一个卷积核的项加起来即得误差函数对这个偏置的导数）
        backward_bias(l.bias_updates, l.delta, l.batch, l.n, k);
    }
```

读代码

懂原理

若l层为卷积层:

*为卷积

$$\delta^{l-1} = \delta^{l} * \mathrm{rot}180\left(w^{l}\right) \odot \sigma'\left(z^{l-1}\right) \quad \text{(Conv-1)}$$

偏置更新需要的梯度:

偏置项的梯度就是sensitivity map所有误差项之和

$$\frac{\partial C}{\partial b^{l}} = \sum_{u,v} \delta_{w,h} \quad \text{(Conv-2)}$$

其中u,v代表卷积核输出的size的长宽

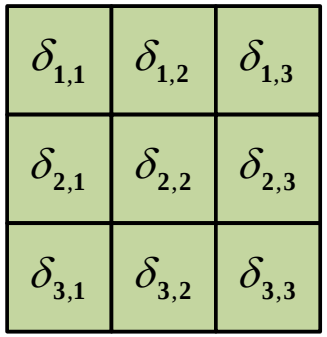
权重更新需要的梯度:

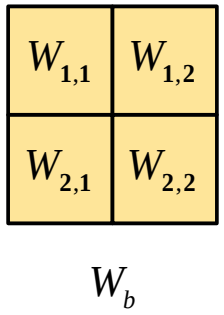
$$\frac{\partial C}{\partial w^{l}} = a^{l-1} * \delta^{l} \quad \text{(Conv-3)}$$

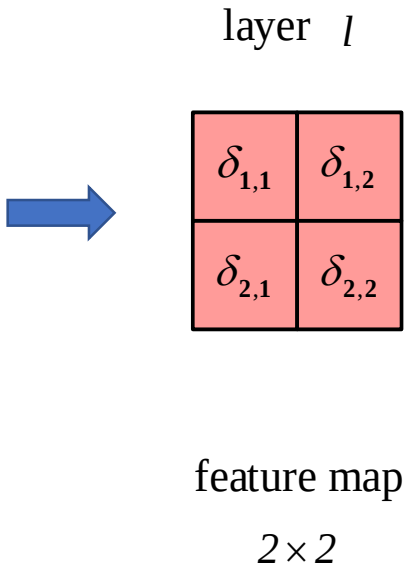
$$b^{l} = b^{l} - \alpha \frac{\partial C}{\partial b^{l}} \quad \text{(FC-4; Conv-4)}$$

$$w^{l} = w^{l} - \alpha \frac{\partial C}{\partial w^{l}} \quad \text{(FC-5; Conv-5)}$$

前向传播：

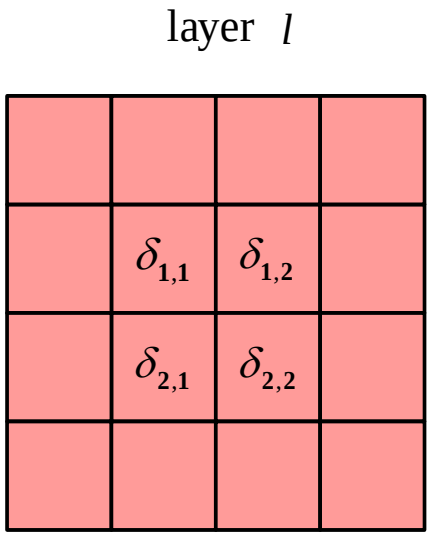


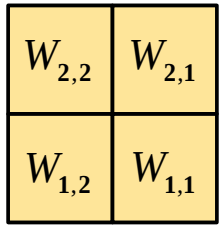


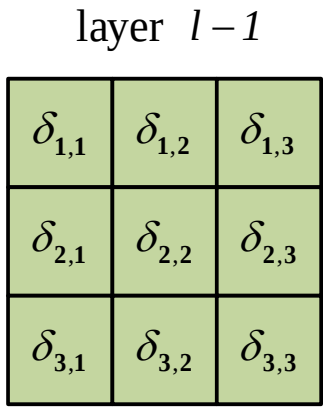


$$O=\frac{W-K+2P}{S}+1$$

反向传播：





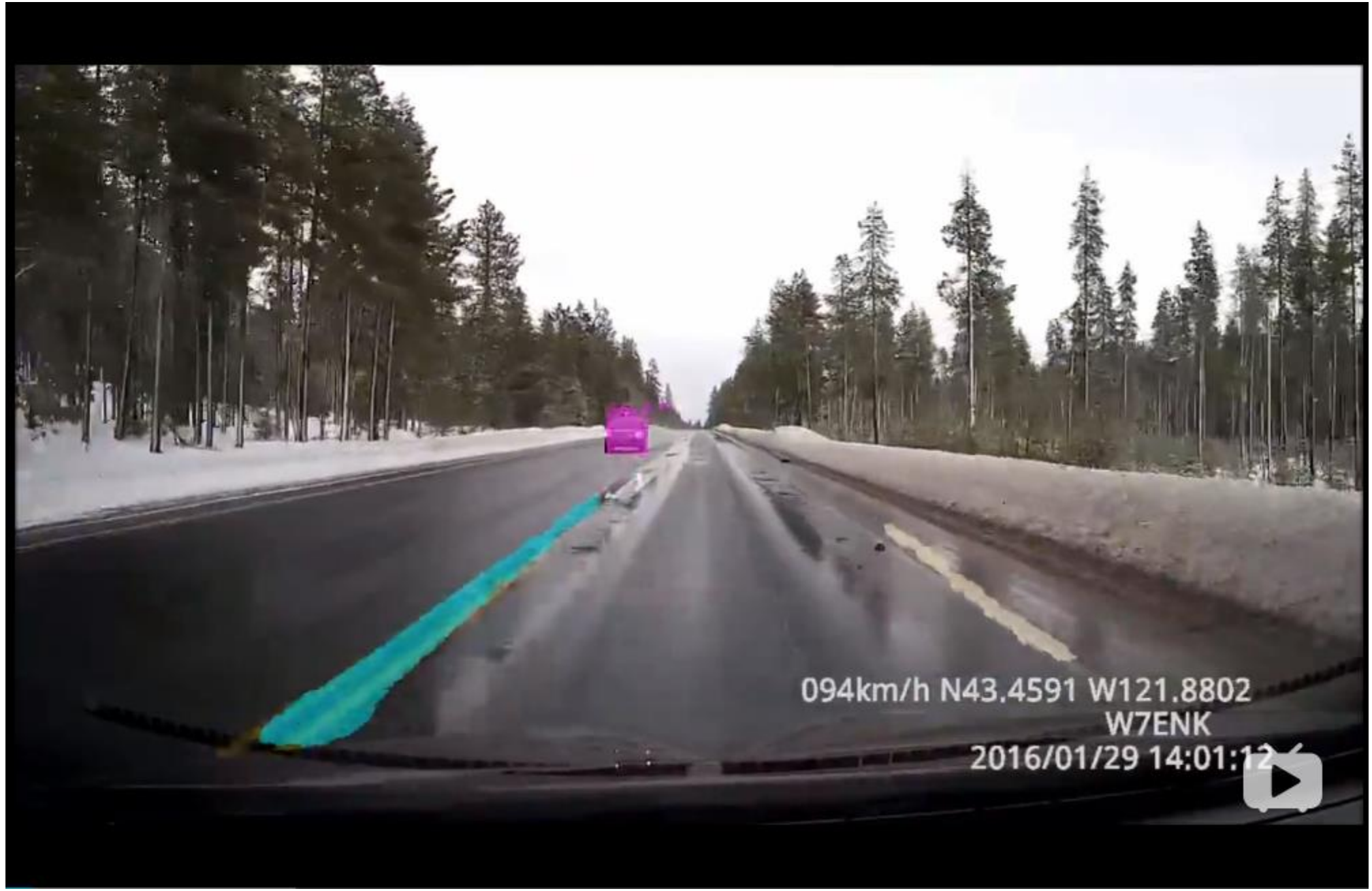



YOLOv3目标检测
网络模型改进方法


Mask R-CNN
图像实例分割实战
训练自己的数据集

Pothole（单类物体）实例分割项目实战

Roadscene（多类物体）实例分割项目实战


094km/h N43.4591 W121.8802
W7ENK
2016/01/29 14:01:12


YES
YOU
CAN

声明

本课程的数据集、程序文件以及课件的演示文稿、视频由讲师白勇拥有知识产权的权利。只限于学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造 。