



使用pytorch和yolo模型进行人流分析

姜原



目录

CONTENTS

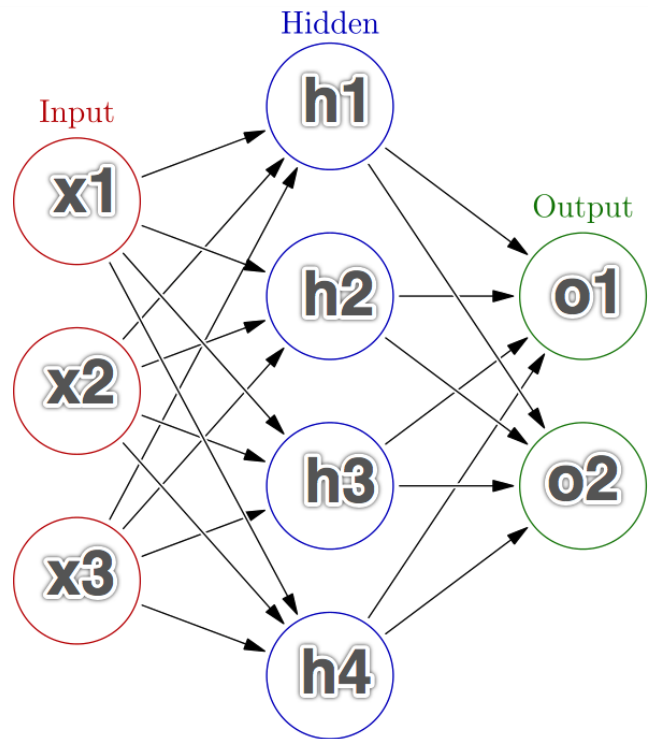
>> 神经网络

>> lenet

>> yolo

>> 行人分析





激活函数 $y = f(x)$

由 x_n 与 h_m 之间的权重为 w_{nm}

$$h_1 = f(x_1 \times w_{11} + x_2 \times w_{21} + x_3 \times w_{31} + b_1)$$

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_n \end{bmatrix}$$

$$W_1 = [w_{11} \quad w_{21} \quad w_{31}]$$

$$h_1 = f(W_1 X + b_1)$$

$$H = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix} = f\left(\begin{bmatrix} W_1 X + b_1 \\ W_2 X + b_2 \\ W_3 X + b_3 \\ W_4 X + b_4 \end{bmatrix}\right) = f\left(\begin{bmatrix} W_1 \\ W_2 \\ W_3 \\ W_4 \end{bmatrix} X + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}\right)$$

$$B = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}$$

$$W = \begin{bmatrix} w_{11} & w_{21} & w_{31} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \\ w_{41} & w_{42} & w_{43} \end{bmatrix}$$

$$H = f(WX + B)$$





$$H = f(WX) + B$$

网络输入 n 维列向量 $X^{n \times 1}$

网络参数 $infeatures, outfeatures$

超参数 W $m \times n$ 矩阵 $W^{m \times n}$

超参数 B m 维列向量 $B^{m \times 1}$

网络输出 m 维列向量 $H^{m \times 1}$

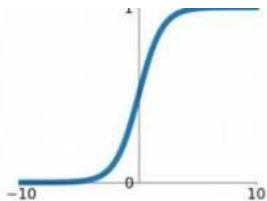




激活函数

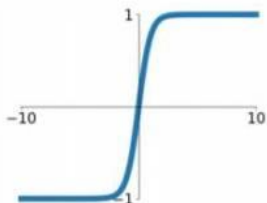
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



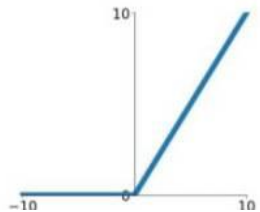
tanh

$$\tanh(x)$$



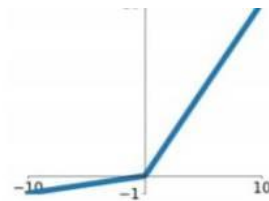
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

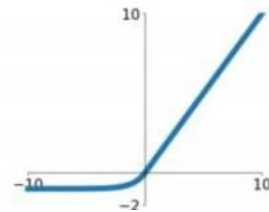


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



要让你的神经网络能够计算出有趣的函数，你必须使用非线性激活函数
如果你使用线性激活函数或者没有使用一个激活函数，
那么无论你的神经网络有多少层一直在做的只是计算线性函数



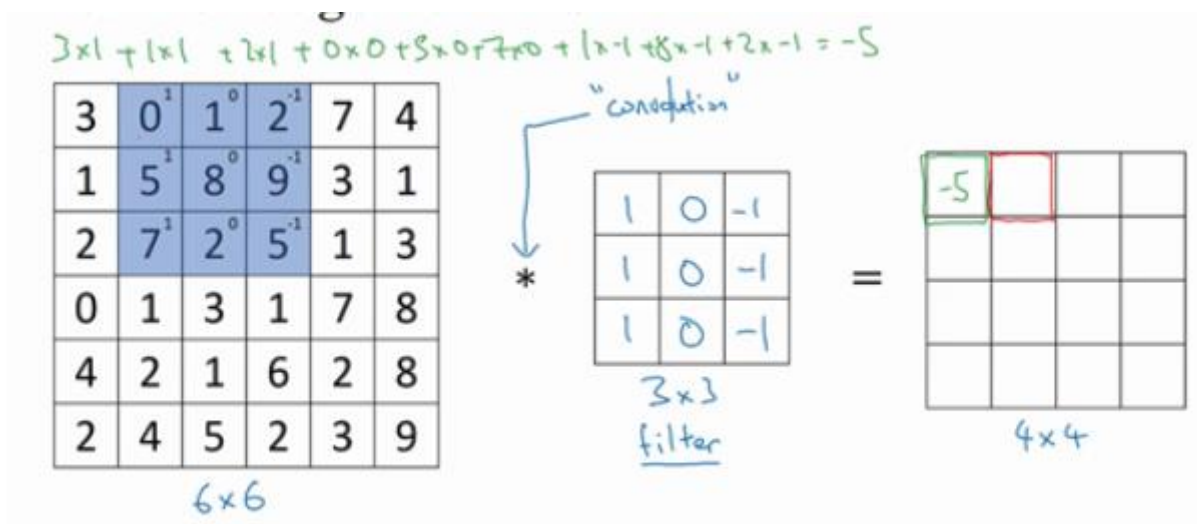


`torch.nn.Linear(in_features, out_features, bias=True)`

- `in_features` – size of each input sample
- `out_features` – size of each output sample
- `bias` – If set to `False`, the layer will not learn an additive bias. Default: `True`

```
liner = nn.Linear(20, 30)
input = torch.randn(128, 20)
output = liner(input)
```





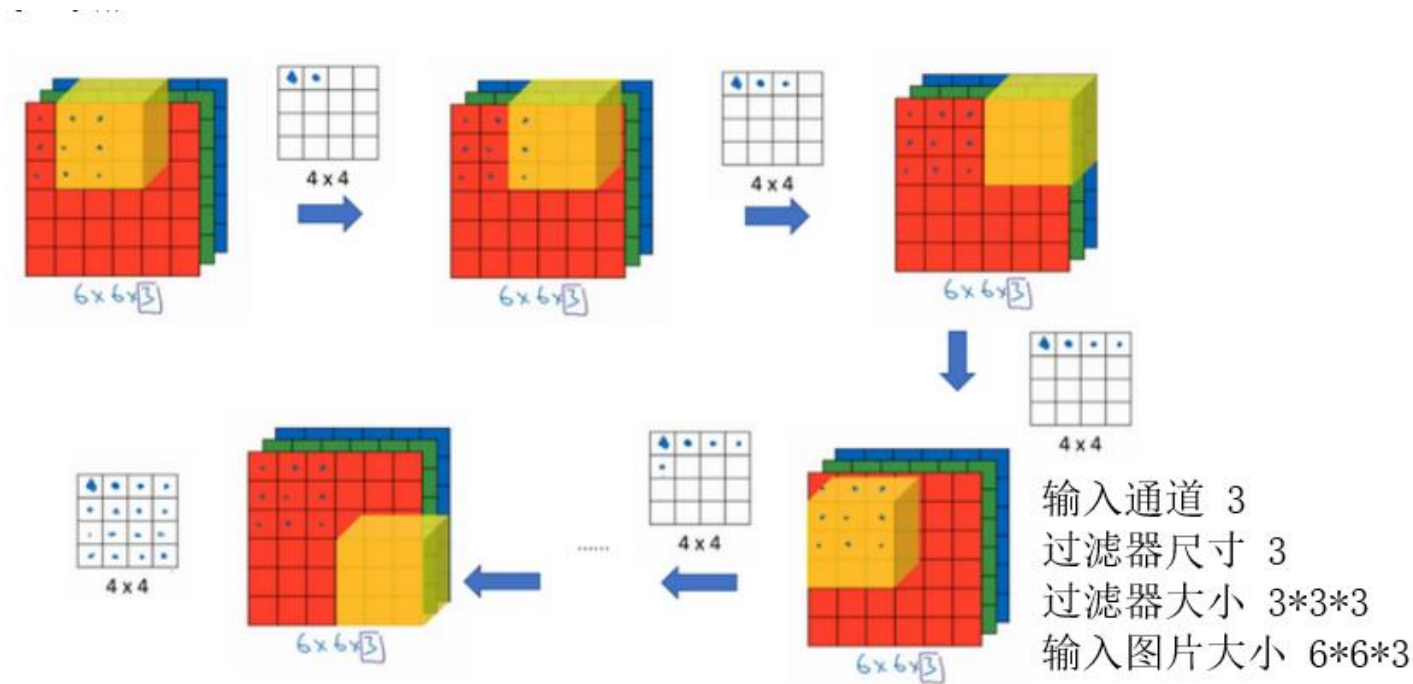
1 _{x1}	1 _{x0}	1 _{x1}	0	0
0 _{x0}	1 _{x1}	1 _{x0}	1	0
0 _{x1}	0 _{x0}	1 _{x1}	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved
Feature





输入： $n \times n \times ic$ 如一张图片 $100 \times 100 \times 3$

网络参数 过滤器尺寸 Filter(kernel) size： f

网络参数 过滤器数量 Numbers of filter / out channels： oc

由输入通道数量和过滤器尺寸可知过滤器大小： $f \times f \times ic$

网络参数 Padding： p

网络参数 步长 Stride： s

网络超参数 Weights： 四维向量 $f \times f \times ic \times oc$

网络超参数 Bias： 四维向量 $1 \times 1 \times 1 \times oc$

输出： $on \times on \times oc$

$$on = \left\lceil \frac{n + 2p - f}{s} + 1 \right\rceil$$



`torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True, padding_mode='zeros')`

Input : $(N, C_{in}, H_{in}, W_{in})$

Output : $(N, C_{out}, H_{out}, W_{out})$

```
conv = nn.Conv2d(3, 50, 3)
input = torch.randn(20, 3, 100, 100)
output = conv(input)
```





Max pooling

Average pooling

3.0	3.0	3.0
3.0	3.0	3.0
3.0	2.0	3.0

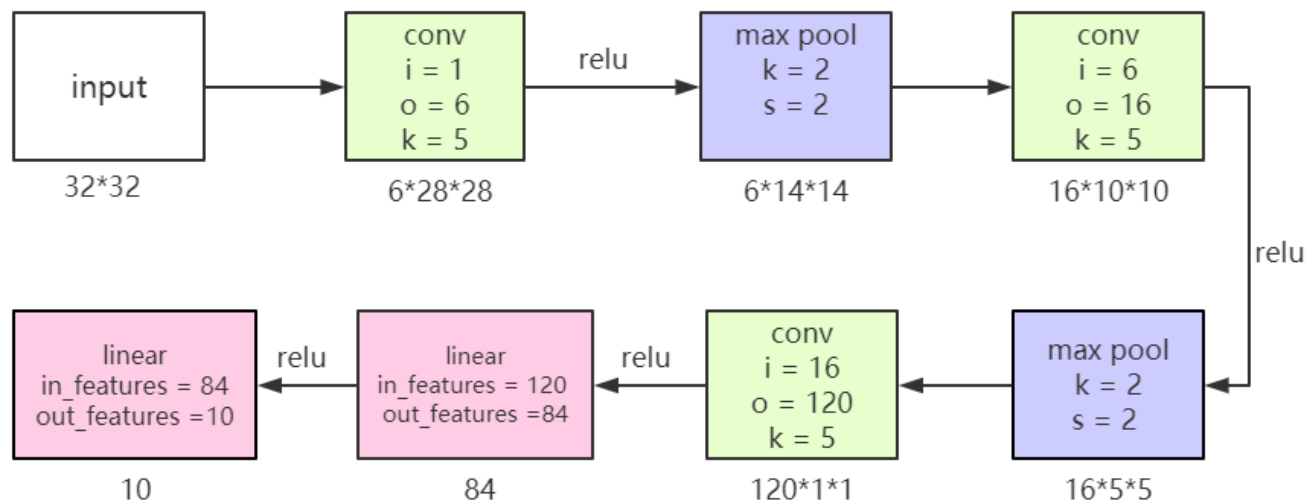
3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1



MNIST 数据集 是 手写数字1 - 9 组成的黑白图片数据集



lenet模型



i : input channels
o : out_channels
k : kernel_size
s : stride
p : padding





这是 Facebook 公司在机器学习和科学计算工具 Torch 的基础上，针对 Python 语言发布的全新的深度学习工具包。PyTorch 类似 NumPy，并且支持 GPU，有着更高级而又易用的功能，可以用来快捷地构建和训练深度神经网络。

PyTorch api设计简洁优雅，方便快速构建模型



在pytorch中定义lenet

```
class LeNet(nn.Module):
    def __init__(self):
        super(LeNet, self).__init__()
        self.pool = nn.MaxPool2d(2, 2)
        self.conv1 = nn.Conv2d(1, 6, 5)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.conv3 = nn.Conv2d(16, 120, 5)
        self.fc1 = nn.Linear(120, 84)
        self.fc2 = nn.Linear(84, 10)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = self.pool(x)
        x = F.relu(self.conv2(x))
        x = self.pool(x)
        x = F.relu(self.conv3(x))
        x = x.view(-1, 120)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```





```
def train(net, n):  
    criterion = nn.CrossEntropyLoss()  
    optimizer = optim.SGD(net.parameters(), lr=1e-2)  
    for epoch in range(1, n+1):  
        for data in train_loader:  
            optimizer.zero_grad()  
            inputs, labels = data  
            prediction = net(inputs)  
            loss = criterion(prediction, labels)  
            loss.backward()  
            optimizer.step()  
            print(loss)
```



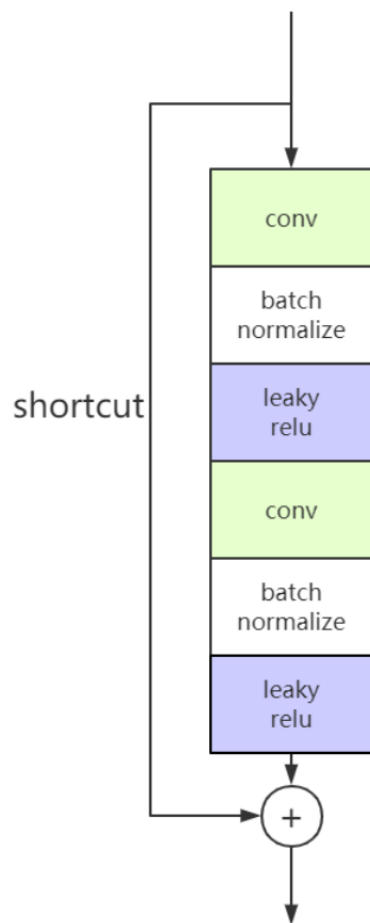


You only look once 你只看一次

YOLO是一个可以一次性预测多个Box位置和类别的卷积神经网络，能够实现端到端的目标检测和识别，其最大的优势就是速度快。



残差块 Residual block



```
conv = nn.Conv2d(in_channel, out_channel, kernel_size)
bn = nn.BatchNorm2d(out_channel)
relu = nn.ReLU()
```

```
x = self.conv(input)
x = self.bn(x)
x = self.relu(x)
x = self.conv(x)
x = self.bn(x)
x = self.relu(x)
```

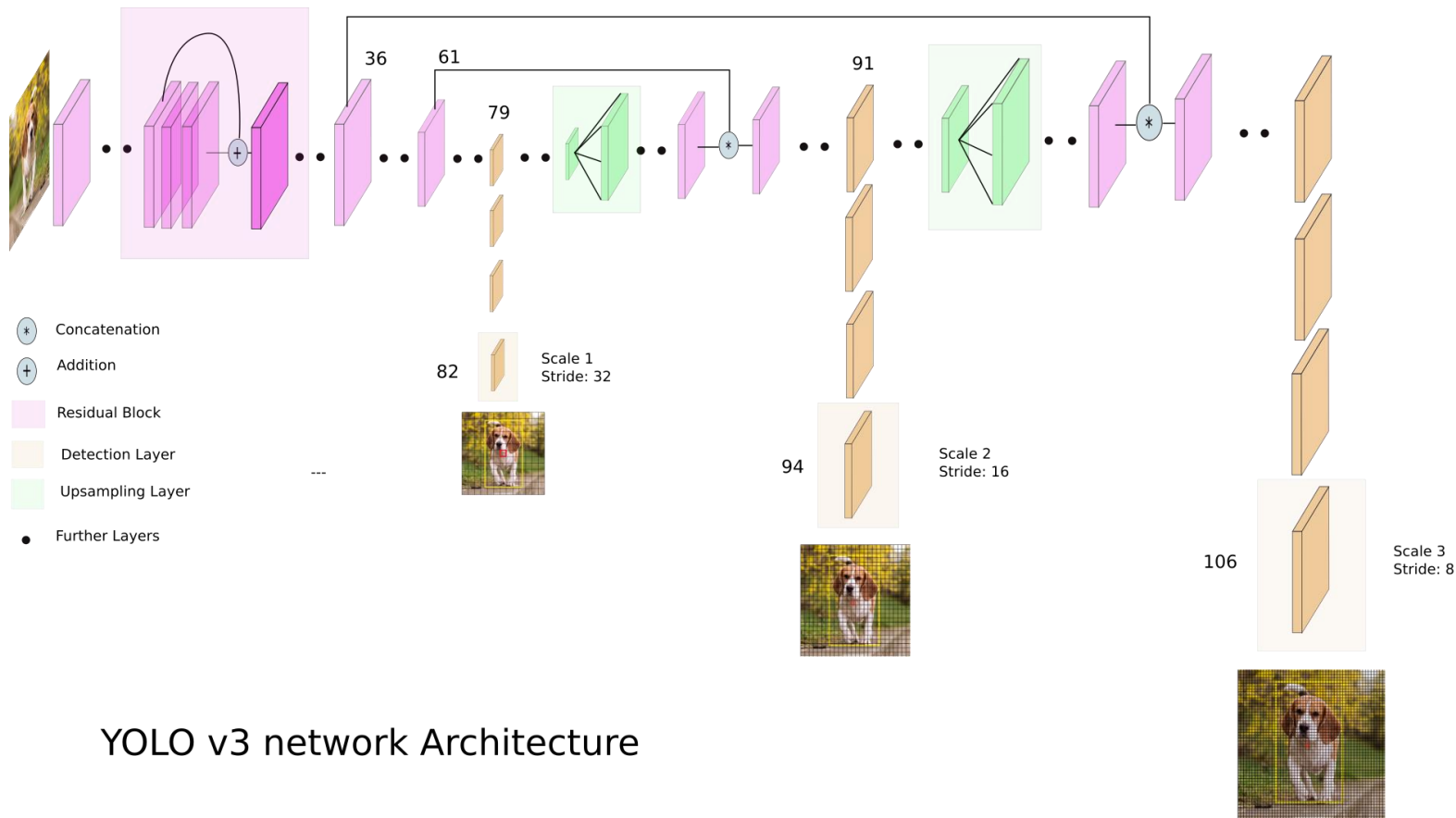
```
output = x + input
```

```
# output = x # 非残差块
```

```
return self.relu(output)
```



darknet 53

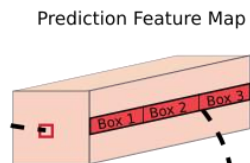
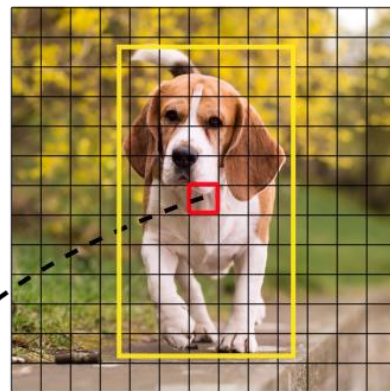
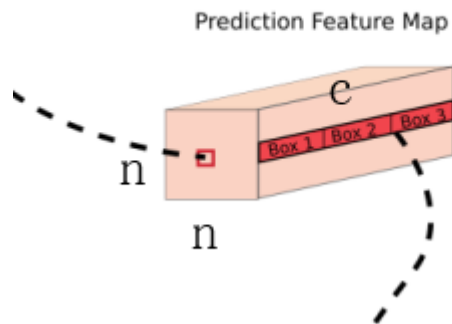


YOLO v3 network Architecture

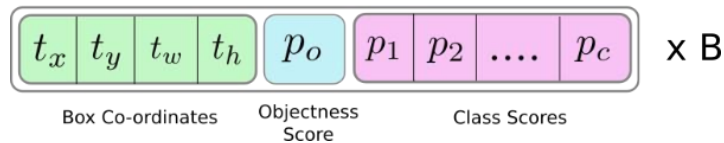


anchor box

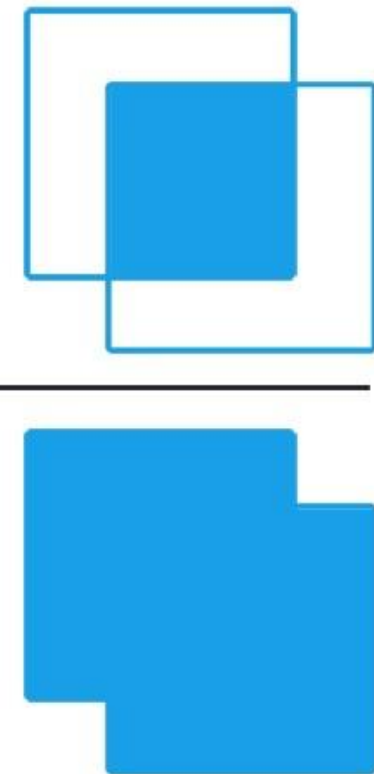
Image Grid. The Red Grid is responsible for detecting the dog



Attributes of a bounding box

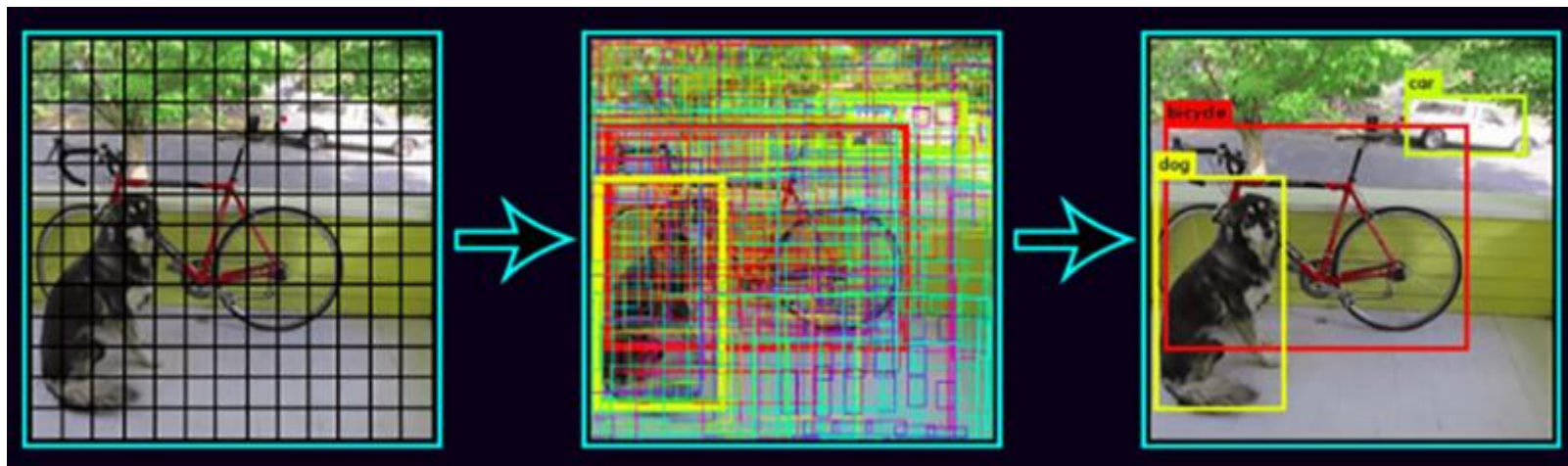


交并比 (Intersection over union)

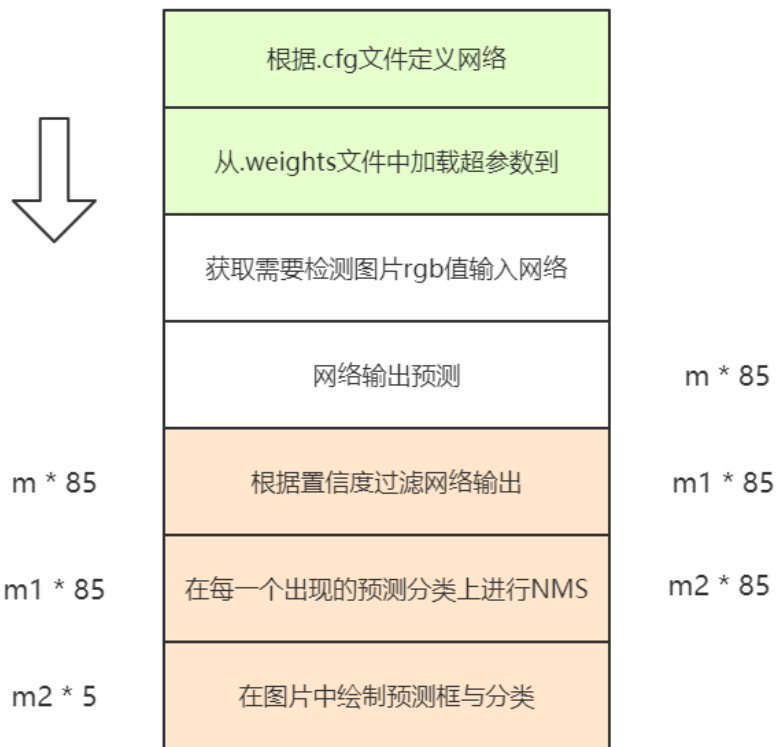
$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


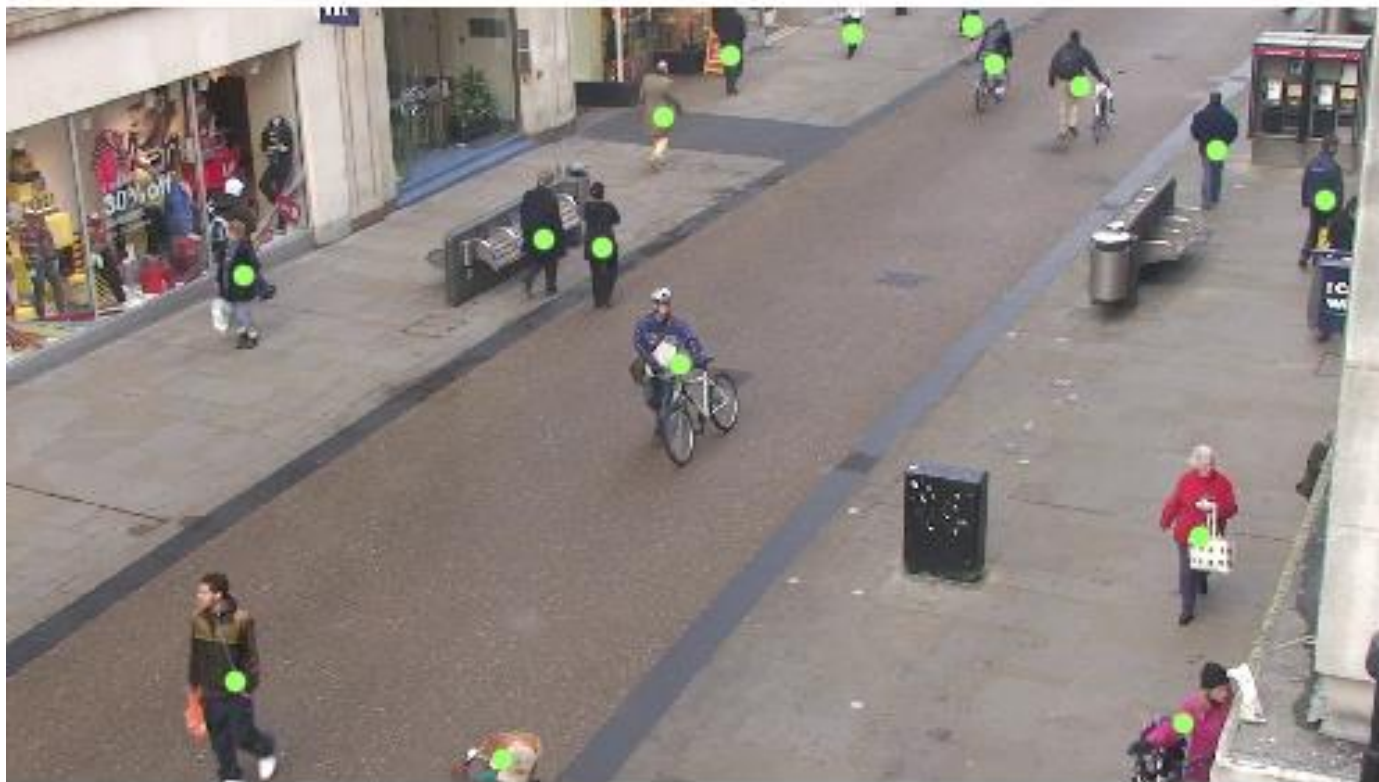


非极大值抑制 (Non-max suppression)



yolo 预测过程







使用opencv-python 获取视频每一帧作为网络输入

获得该帧所有检测出的对象的边框坐标与分类

根据分类过滤对象, 将边框坐标转化为边框中心点





```
class Perdestrian:
    v = (0, 0)
    not_show_frames = 0

    def __init__(self, id, point):
        self.id = id
        self.point = point
        self.track = [point]

    """下一帧有匹配点"""
    def show(self, point):...

    """行人速度"""
    def update_v(self):...

    """下一帧无匹配点"""
    def is_leave(self):...
```





```
current_people = people in first frame
```

```
for frame in frames:
```

```
    # frame 是一个中心点的列表 point list
```

计算 frame 中的各点与 current_people 中各点的距离
根据距离从小到大

将 point 与 people 匹配

更新 people (更新 people 中的速度)

多余的 point 为新加入的行人

对于 current_people 中 未匹配行人, 我们按他当前速度进行更新
统计其累加其未出匹配点的帧数,

if 行人更新后的 point 大于边界值:
 标记为离开

if 未出匹配点的帧数 大于某一阈值:
 标记为离开







refer

- <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>
- https://github.com/fengdu78/deeplearning_ai_books
- <https://pjreddie.com/darknet/yolo/>
- <https://blog.paperspace.com/how-to-implement-a-yolo-object-detector-in-pytorch/>
- <https://pjreddie.com/media/files/papers/YOLOv3.pdf>





THANK YOU

微信: jy5322

