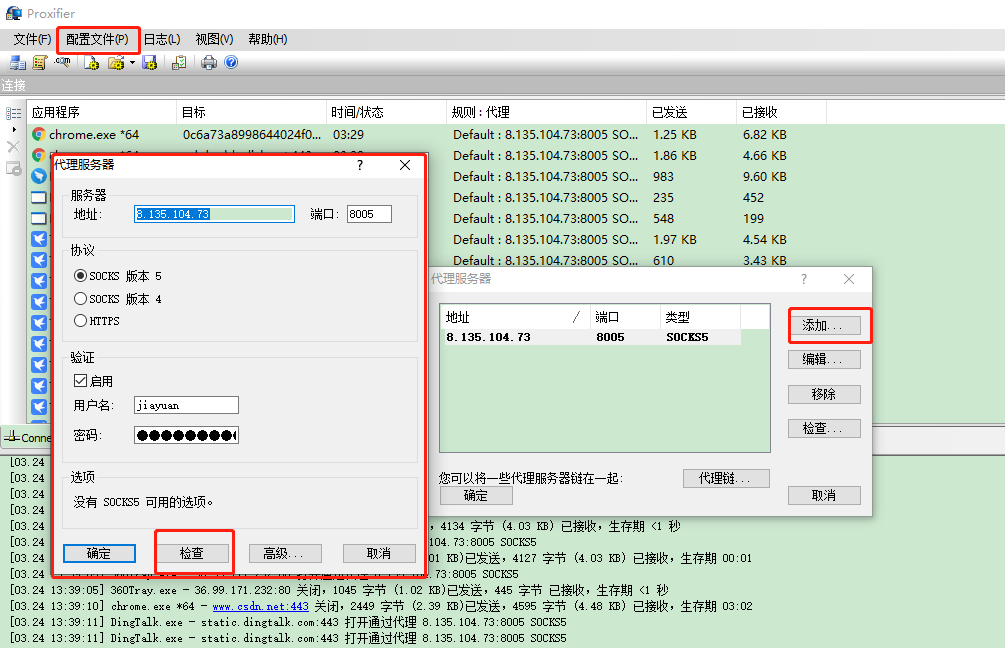
# 基本开发环境

## 广州SVN信息

VPN地址: 8.135.104.73:8005

用户名/密码:jiayuan/jiayuan123

参考: <https://blog.csdn.net/Amdy_amdy/article/details/84585063>



地址：svn://116.56.140.138

用户名:minglu

密码:L2m9n3g4zll

## GitLab账户

<http://git.jy1024.cn/>

用户名:minglu

密码:L2m9n3g4zll

## 为知笔记

[850268547@qq.com](mailto:850268547@qq.com)

## 外网连接到ssh工作站（大服务器）

Ip : 376g6u6711.qicp.vip:34286

用户名:zhks

密码: zhks2020@\*#&

## 代理服务：

参考：<https://blog.csdn.net/Amdy_amdy/article/details/84585063>

广州内网vpn 8.135.104.73:8005 jiayuan/jiayuan123

## 麻地梁远程链接

使用向日葵客户端远程链接到：



对方系统的密码：123

## 麻地梁统一安全平台

用户名:mdl

密码：@12345678

要远程连接此安全平台的话，需要使用代理：

47.119.169.152:8003

用户名:jiayuanjiegou

密码:jiayuan~+!\_543

## 界沟视频监控平台

### 代理信息

IP:47.119.169.152:8009

用户名:jiayuan

密码:jiayuan#$%86389

### 海康监控平台信息

Ip地址：192.168.168.15

用户名:xufei

密码:zhks2021

### 远程web地址

<http://218.22.237.158:82/>

用户名: admin

密码:jg123456

## 界沟服务器

界沟服务器ssh 47.119.169.152:8008  
用户名:zhks  
密码:zhks2020@\*#&

## 邮箱

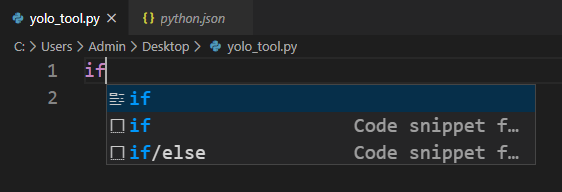
账号：[luming@ahzhks.com](mailto:luming@ahzhks.com)

密码：L2m9n3g4zll

# VS code使用方法

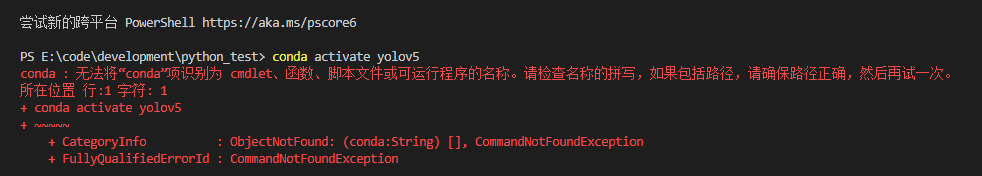
## 请使用1.5以下版本

对于高版本，可能提示不够全面。如在1.42.1版本中提示如下：



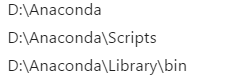
提示很全面。

## 报错：conda : 无法将“conda”项识别为 cmdlet



<https://www.cnblogs.com/nmydt/p/14493789.html>

在系统变量path中新增：



# Linux中安装svnserver

## 参考：

<https://blog.csdn.net/zengsange/article/details/80618301>

<https://blog.csdn.net/ArvinFei/article/details/45460373>

## 步骤

服务器是阿里云的 系统ubuntu 14.04 64位(查看系统版本 uname -a)

### 查看是否安装

安装SVN服务器之前 查看一下是否已经安装了 查看命令 svn

如下图 是已经安装了SVN

https://img-blog.csdn.net/20180608093348857?watermark/2/text/aHR0cHM6Ly9ibG9nLmNzZG4ubmV0L3plbmdzYW5nZQ==/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70

如下图是没有安装SVN

https://img-blog.csdn.net/20180608093444549?watermark/2/text/aHR0cHM6Ly9ibG9nLmNzZG4ubmV0L3plbmdzYW5nZQ==/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70

安装了的话 可以用如下命令删除

apt-get remove --purge subversion

### 安装SVN

安装之前更新一下

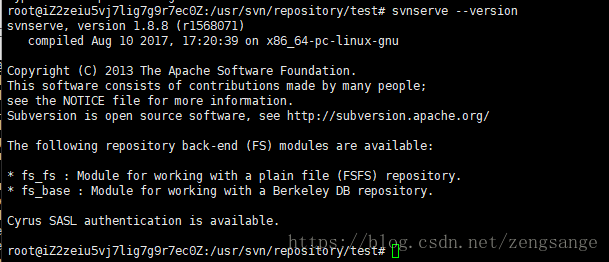
apt-get update （可以跳过）

apt-get install subversion

安装好之后 查看是否安装成功

svnserve --version

如下图 显示版本信息 表示安装成功



### 创建SVN版本库

 进入 cd /usr 文件

mkdir svn

进入 cd /svn

mkdir repository

给 repository 权限 chmod -R 777 /repository

创建svn仓库

svnadmin create /usr/svn/repository

执行命令后，会在repository下生成以下文件：

https://img-blog.csdn.net/20180608094731390?watermark/2/text/aHR0cHM6Ly9ibG9nLmNzZG4ubmV0L3plbmdzYW5nZQ==/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70

对db进入权限设置

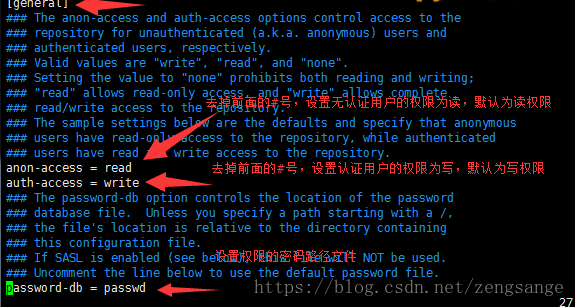
chmod -R 777 db

### 设置访问权限

#### 修改配置文件conf/svnserve.conf

cd /conf

vi svnserve.conf

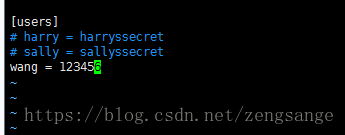


注意：1）authz-db = authz可以不打开。2）将”#” 删除并将其顶格后保存并退出

#### 添加访问用户

vi passwd

用户名 wang 密码 123456



#### 设置用户权限

vi authz



对以上进行解释：

admin = wang //用户王属于admin权限组

@admin = rw //admin权限组的权限是读和写

\* = r 所有的组都具有读权限

### 测试服务器

启动SVN服务器

svnserve -d -r /usr/svn/

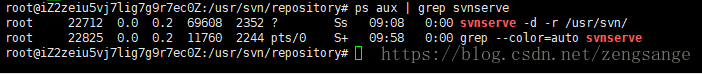
-d：表示在后台运行

-r：指定服务器的根目录

查看是否启动成功

ps aux | grep svnserve

如下图 成功启动

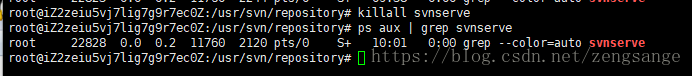


注意：这里必须出现” svnserve -d -r /usr/svn/”才算成功的，只有--color的不算。

停止服务器

killall svnserve

如下图 说明已经关闭了



查看svnserve所监听的端口:

Netstat –anp | grep svnserve

这样访问服务器时就可以直接用”svn://服务器ip:port/repository”来访问了。

### 开机自启动svn

在/etc/init.d目录建立一个脚本文件start\_svnserver.sh。在其中输入如下内容：

#!/bin/bash

sudo svnserve -d -r /usr/svn/

exit 0

保存退出。更新，修改权限：

chmod 777 start\_svnserver.sh

添加到自启动项目中

# update-rc.d start\_svnserver.sh defaults

或者在/etc/rc.local中的exit 0之前写入如下信息：

#!/bin/sh

sudo /bin/bash /etc/init.d/start\_svnserver.sh

# Nvidia 显卡信息查询

## 驱动查询

cat /proc/driver/nvidia/version

## 显卡设备信息

## 删除旧的显卡驱动和cuda

sudo apt-get --purge remove nvidia\*

sudo apt autoremove

To remove CUDA Toolkit:

**$** 

To remove NVIDIA Drivers:

**$** sudo apt-get --purge remove "\*nvidia\*"

参考：<https://blog.csdn.net/qq_33200967/article/details/80689543>

# 安装Teamviewer

<https://www.cnblogs.com/wmr95/p/7574615.html>

注意：可能会用到：sudo apt-get --fix-broken install

# Ubuntu 18.04上安装cuda

<https://blog.csdn.net/zhiman_zhong/article/details/91491780>

## 禁用 nouveau驱动

**禁用 nouveau驱动**



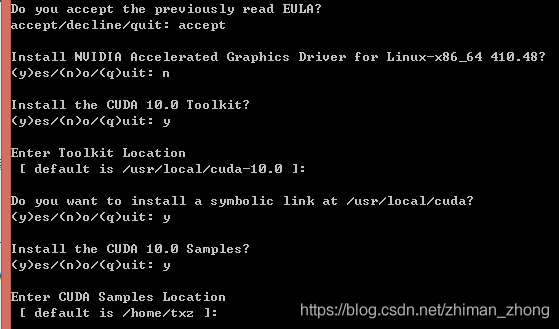
lsmod | grep nouveau 若无内容输出，则禁用成功

**我执行以上操作后，还是有输出，reboot重启后，无输出了。**

## **执行安装脚本**

sudo sh cuda\_\*\*\*\_linux.run (你下载的runfile安装包名字）

注意：先accept，安装项里不要选择安装驱动，避免有坑，其他都yes



## 设置环境变量

在主目录下的~/.bashrc文件添加如下路径

1. export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:/usr/local/cuda-10.0/lib64
2. export PATH=$PATH:/usr/local/cuda-10.0/bin

# Ubantu18.04.4：依赖: g++-7 (>= 7.3.0-12~) 但是它将不会被安装

<https://blog.csdn.net/qq_41608919/article/details/104748719>

# Python学习笔记

## 路径操作

<https://jingyan.baidu.com/article/6d704a13cc28cd68db51caa1.html>

可以通过os.path和pathlib来操作

## glob



## 函数值传递、地址传递(又称为引用传递)

通过上面介绍可以得出如下两个结论：

1、不管什么类型的参数，在 Python 函数中对参数直接使用“=”符号赋值是没用的，直接使用“=”符号赋值并不能改变参数。

2、如果需要让函数修改某些数据，则可以通过把这些数据包装成列表、字典等可变对象，然后把列表、字典等可变对象作为参数传入函数，在函数中通过列表、字典的方法修改它们，这样才能改变这些数据。

# 安全帽检测

<https://www.cnblogs.com/supersayajin/p/11445401.html>

<https://haoji007.blog.csdn.net/article/details/112504931>

<https://github.com/HCIILAB/SCUT-HEAD-Dataset-Release>

# YOLOV5教程

## 使用详解

<https://shliang.blog.csdn.net/article/details/106785253>

<https://www.cnblogs.com/shuimuqingyang/p/14462363.html>

## YOLOV5核心内容

<https://zhuanlan.zhihu.com/p/172121380/>

### Mosaic数据增强

Yolov5的输入端采用了和Yolov4一样的Mosaic数据增强的方式。

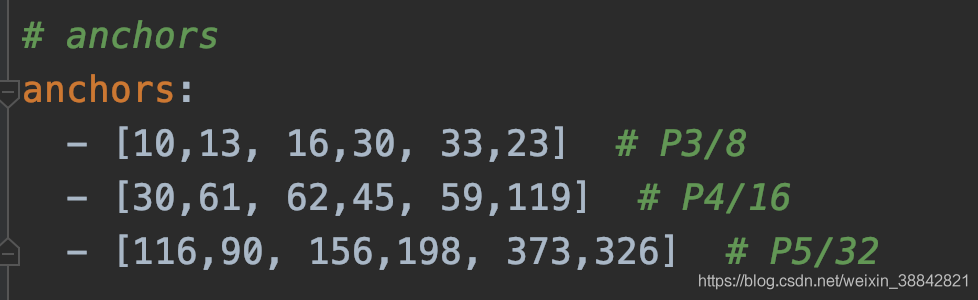
随机缩放、随机裁剪、随机排布的方式进行拼接，对于小目标的检测效果很不错。

### 自适应锚框计算

在Yolo算法中，针对不同的数据集，都会有初始设定长宽的锚框。

在网络训练中，网络在初始锚框的基础上输出预测框，进而和真实框groundtruth进行比对，计算两者差距，再反向更新，迭代网络参数。

yolov5s初始设定的anchor



Yolov5中将此功能嵌入到代码中，每次训练时，自适应的计算不同训练集中的最佳锚框值。

当然，如果觉得计算的锚框效果不是很好，也可以在代码中将自动计算锚框功能关闭。

parser.add\_argument('--noautoanchor', action='store\_true', help='disable autoanchor check')

1

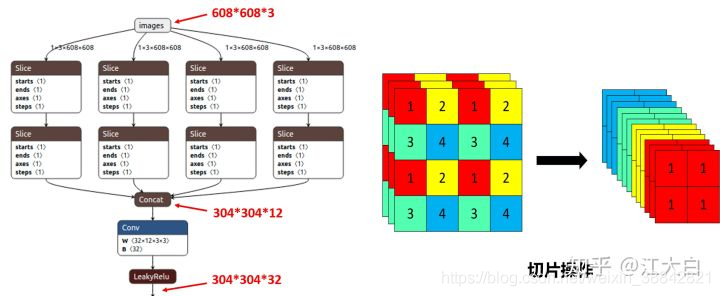
train.py中上面一行代码，设置成False，每次训练时，不会自动计算。

### 自适应图片缩放

参见utils/datasets.py中的letterbox()函数



### Focus结构



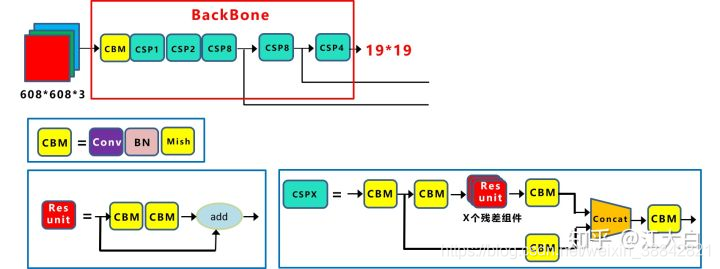
Focus是Yolov5新增的操作，右图就是将443的图像切片后变成2212的特征图。

以Yolov5s的结构为例，原始6086083的图像输入Focus结构，采用切片操作，先变成30430412的特征图，再经过一次32个卷积核的卷积操作，最终变成30430432的特征图。

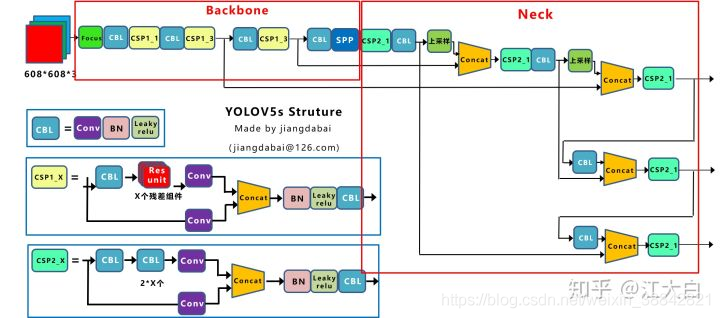
**需要注意的是：Yolov5s的Focus结构最后使用了32个卷积核，而其他三种结构，使用的数量有所增加。**

### CSP结构

Yolov4借鉴了CSPNet的设计思路，在**主干网络**中设计了CSP结构，但**只有主干网络**使用了CSP结构。



Yolov5中设计了两种CSP结构，以Yolov5s网络为例，CSP1\_X结构应用于Backbone主干网络，另一种CSP2\_X结构则应用于Neck中。



**CSPNet**(Cross Stage Partial Network):跨阶段局部网络，以缓解以前需要大量推理计算的问题。

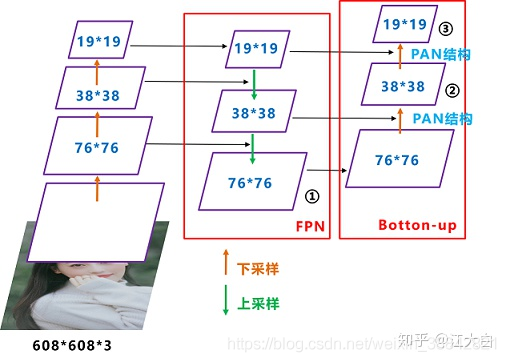
* 增强了CNN的学习能力，能够在轻量化的同时保持准确性。
* 降低计算瓶颈。
* 降低内存成本。

CSPNet通过将梯度的变化从头到尾地集成到特征图中，在减少了计算量的同时可以保证准确率。

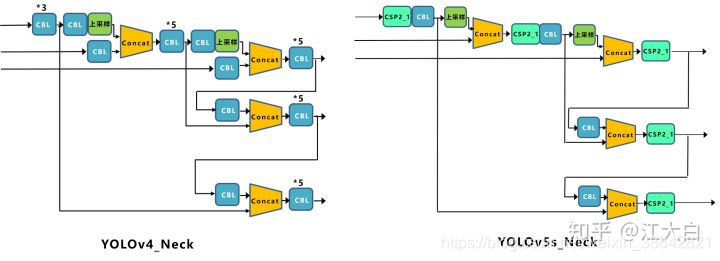
CSPNet和PRN都是一个思想，将feature map拆成两个部分，一部分进行卷积操作，另一部分和上一部分卷积操作的结果进行concate。

### neck部分

Yolov5的Neck和Yolov4中一样，都采用FPN+PAN的结构。

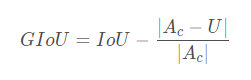


Yolov4的Neck结构中，采用的都是普通的卷积操作。而Yolov5的Neck结构中，采用借鉴CSPnet设计的**CSP2**结构，加强网络特征融合的能力。

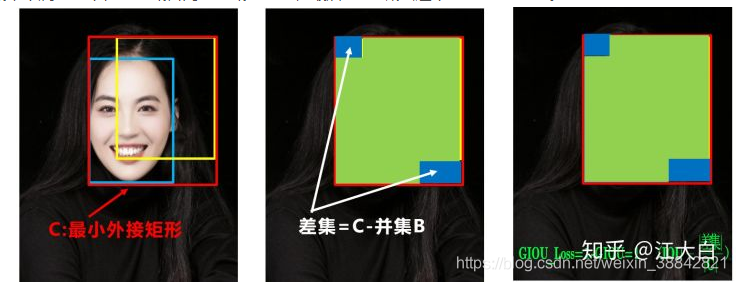


### 输出端

Yolov5中采用其中的GIOU\_Loss做Bounding box的损失函数。

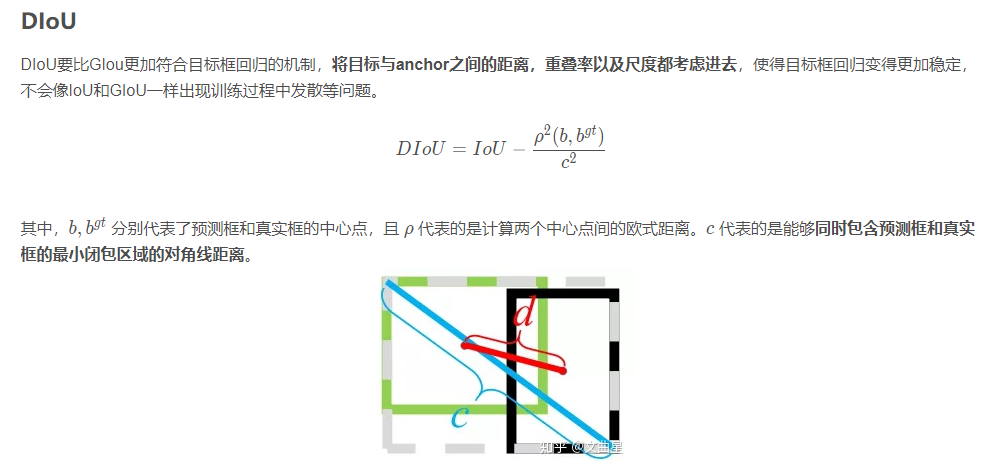


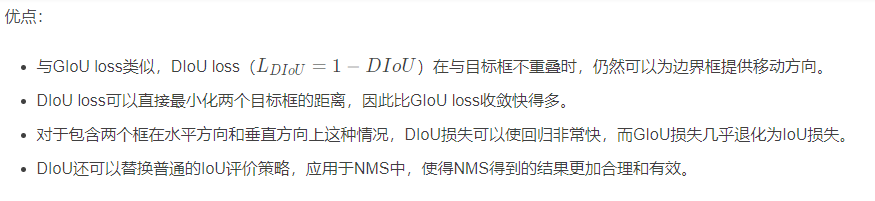
上面公式的意思是：先计算两个框的最小闭包区域面积 [公式] (通俗理解：同时包含了预测框和真实框的最小框的面积)，再计算出IoU，再计算闭包区域中不属于两个框的区域占闭包区域的比重，最后用IoU减去这个比重得到GIoU。

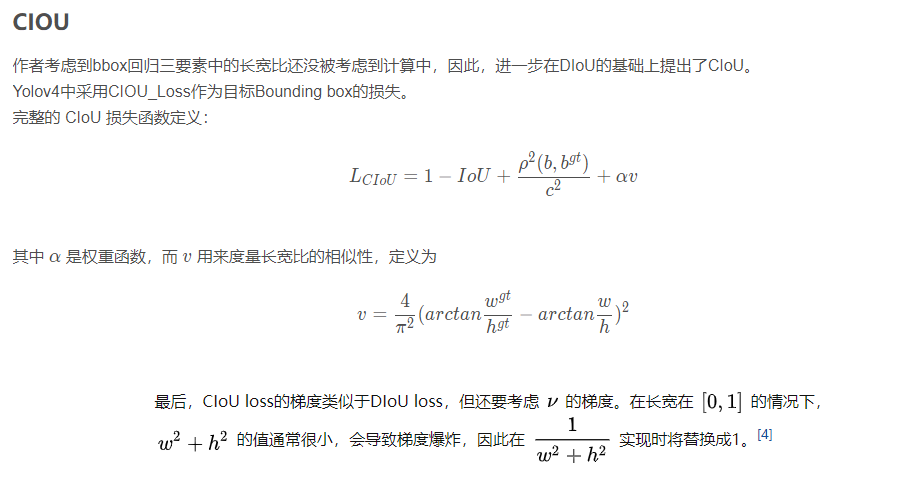


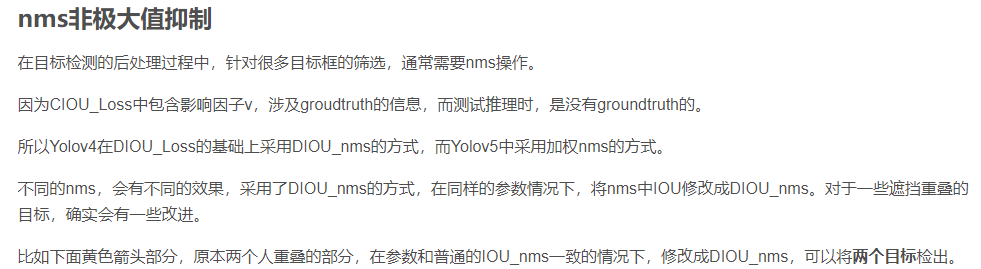
用图片来进行理解就是：

* 两个框的最小闭包区域面积 = 红色矩形面积
* IoU = 黄色框和蓝色框的交集 / 并集
* 闭包区域中不属于两个框的区域占闭包区域的比重 = 蓝色面积 / 红色矩阵面积
* GIoU = IoU - 比重



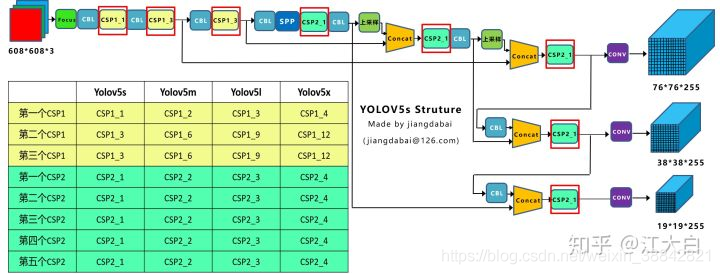




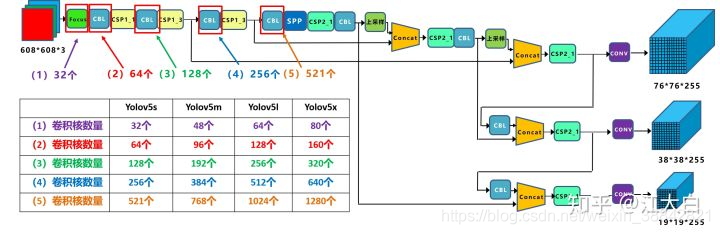




### Yolov5四种网络的深度



### Yolov5四种网络的宽度



# 人体行为识别

## CenterTrack

### Github地址

<https://github.com/xingyizhou/CenterTrack>

## AlphaPose

### GitHub地址

<https://github.com/MVIG-SJTU/AlphaPose/tree/pytorch>

### 项目主页

<http://www.mvig.org/research/alphapose.html>

## OpenPose

### OpenPose项目Github链接

<https://github.com/CMU-Perceptual-Computing-Lab/openpose>

### Github开源人体姿态识别项目OpenPose中文文档

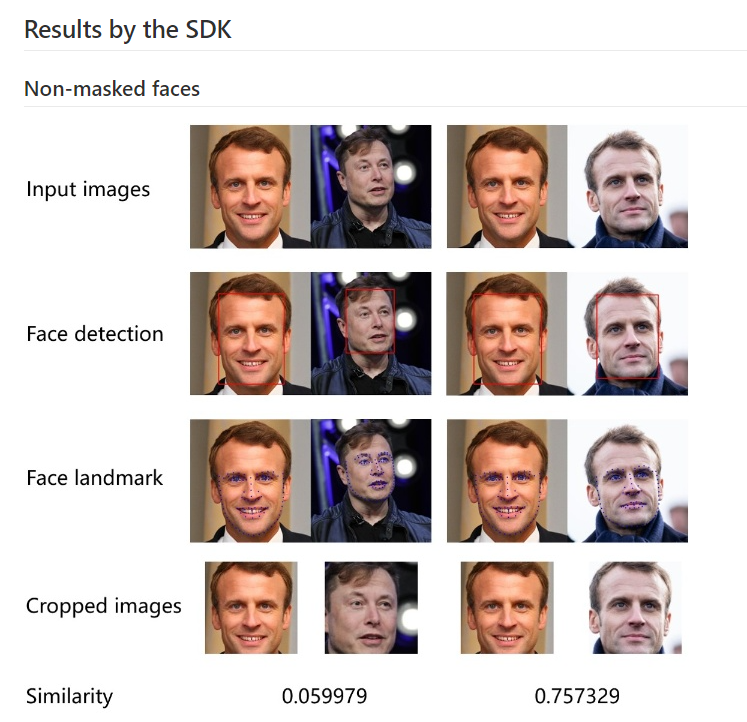
<https://www.jianshu.com/p/3aa810b35a5d>

# 人脸识别

## FaceX-Zoo

### GitHub地址

### 功能



# OpenCV实例教程在线学

## 网址

<https://learnopencv.com/>

# 口罩检测

## GitHub地址

<https://github.com/AIZOOTech/FaceMaskDetection>

## 体验地址

<https://aizoo.com/face-mask-detection.html>

## 项目介绍

<https://blog.csdn.net/dQCFKyQDXYm3F8rB0/article/details/104743982>

# CUDA 实现神经网络

## CUDA 实现BP网络

<https://github.com/Windsrain/MNIST_BPNet>

# Docker常用命令

## Docker教程

<https://www.runoob.com/docker/ubuntu-docker-install.html>

## Docker入门

<https://docs.docker.com/get-started/>

## 构建容器:docker build

<https://docs.docker.com/engine/reference/commandline/build/>

为了构建容器镜像，我们需要创建一个名为Dockerfile的文件. Dockerfile 只是一个基于文本的指令脚本，用于创建容器映像。

Usage: docker build [OPTIONS] PATH | URL | -

### -t

Name and optionally a tag in the 'name:tag' format

## 启动容器docker run

docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

运行成功后，会出现一个很长的字符串，其为容器编号，其前12位，也可作为容器编号在后面的命令中使用。

## 显示容器:docker ps

docker ps [OPTIONS]

## 开始被停止的容器：docker start

Start one or more stopped containers

## 停止运行中的容器: docker stop

docker stop [OPTIONS] CONTAINER [CONTAINER...]

## 查看运行的容器的状态:docker stats

