**场景人流识别**

|  |  |
| --- | --- |
| 组 别： | 第二小组 |
| 小组成员： | 1211002008 黄俊达 |
|  | 1211002001 白家运 |
|  | 1211002011 雷万年 |
|  | 1211006001 果 航 |

目 录

**[一、 项目研究背景 1](#_Toc23529)**

[1． 研究现状 1](#_Toc29652)

[2． 问题定义 2](#_Toc27086)

[3． 数据采集 2](#_Toc1312)

**[二、 环境搭建 2](#_Toc22554)**

[1． Anaconda软件的下载 2](#_Toc23112)

[2． 更换国内源方便下载软件包 4](#_Toc25757)

[3． Pytorch安装 5](#_Toc19567)

[4． Yolov5安装 6](#_Toc5175)

**[三、 算法模块 8](#_Toc17432)**

[1． Yolov5 8](#_Toc20470)

[（1） 输入端 8](#_Toc1611)

[（2） Backbone骨干网络 8](#_Toc4143)

[（3） Neck颈部结构 9](#_Toc31689)

[（4） Prediction 10](#_Toc11609)

[2． Deepsort 12](#_Toc19203)

[（1） 传统sort算法的流程 13](#_Toc27818)

[（2） Deepsort算法流程 14](#_Toc8881)

**[四、 整体系统演示及结果 15](#_Toc20178)**

**[五、 问题及相关解答 17](#_Toc16958)**

# 项目研究背景

## 研究现状

人群计数一直是计算机视觉中热门的研究领域之一，主要是在图像或者人群中准确地统计人群数量。尤其在近几年，随着全球各地重大踩踏事故的频发，人群计数有着更为重要的意义。通过人群计数，可以通过管控降低人群密度，减小踩踏事件发生的概率。

近年来，随着深度学习技术的不断发展，有着越来越多的人群计数算法被不断提出并持续升级优化，有如基于卷积神经网络(CNN)的人群计数算法、多阵列卷积神经网络(Multi-scale CNN,即MSCNN)、Point-Query Quadtree for Crowd Counting(PET)、CSRNet、ic-CNN等。

CNN是一种深度学习的算法，其基本原理是通过多层的非线性变换来提取图像中的特征。与传统的计算机视觉方法相比，CNN能够自动学习和提取图像中的特征，而不需要手动设计和选择特征，此外，CNN还具有强大的鲁棒性和泛化能力，能够处理各种复杂场景下的图像。基于CNN 的人群计数算法通常可以分为以下几个步骤：数据预处理、模型构建和训练、预测和评估。但是，基于CNN的人群计数算法也存在一些缺点，如数据标注成本高、模型训练时间长、对大规模和密集的人群估计效果不佳。

PET通过点查询四叉树结构，允许数据相关的分裂，能够自适应地生成查询点以解决密集人群预测问题。PET还采用了渐进式矩形窗口注意力机制，提高了效率。尽管PET在某些方面有所改进，但人群计数的根本性挑战如遮挡处理、极端密集人群的计数准确性等依然存在。与此同时的是，该方法可能增加了模型的复杂性，需要更多的研究和实验来验证其广泛适用性。

MSCNN通过不同尺度的卷积核来提取不同大小的人群特征，增强了模型对不同密度人群的适应性。但与之对应的是，这种多阵列结构的模型通常参数较多，计算量大，可能无法实现实时的人群计数预测，有些低效的分支结构可能未能有效提取特征。

CSRNet模型抛弃了Multi-column框架，使用预训练的VGG16网络和空洞卷积(Dilated Convolution)，能够更有效地获取人头的边缘信息，从而提高计数精度。该模型虽然相比多阵列结构有所简化，但依然存在一定的计算复杂度，可能影响实时性能。

ic-CNN模型采取由低分辨率到高分辨率逐步细化的策略，得到高质量的密度图，从而提升计数的准确性。但是逐步细化的策略可能会增加计算步骤，影响整体速度。

## 问题定义

正如前文所提及的，随着深度学习技术的不断发展，人群计数算法在准确性、实时性和适应性都取得了显著进展，但让面临以下问题：

1. 数据标注成本高。尽管人群计数是近年来计算机视觉的热门研究领域之一，其数据集也在不断增多和提高精度，但是当在进行模型训练时，如若不能找到适配的数据集，仍需要根据特点的目标去进行专门的数据标注。这使得人群计算这一研究领域，数据标注成本仍是非常大的。
2. 计算资源需求大。深度学习是需要进行多次训练以达到更佳目的。加之近年提出的新型算法模型，其计算难度与量级无疑是巨大的，并且部分模型如PET还需大量实验和研究才能验证其广泛适用性。
3. 极端密集人群处理困难。这个问题是目前提出的人群计数模型都面临的问题，其仍是难以攻克的难关。在当前画面出现极端密集人群时，模型都会出现漏检、误检、准确率和实时率下降等各方面问题。
4. 遮挡重检准确率低。这同样是目前人群计数模型面临的问题，当人群被遮挡后重新出现时，其模型能否正确分配ID，正确识别同一个人，准确率能否提高，这些都是当前人群计数领域面临的难题。

## 数据采集

在本次大作业中，我们共采取两种方法进行数据集的建立。

第一种方法是，根据选取好的视频，进行视频帧提取图片，进行手动标记并分好训练集和验证集，这样就可以进行训练了。优点是方便易上手，适合新手入门并深刻理解模型训练。缺点是工程量大，如果视频时长过长，会导出大量图片，导致数据标注时间成本急剧上升，而且由于是手动标记，会存在人为的误差。

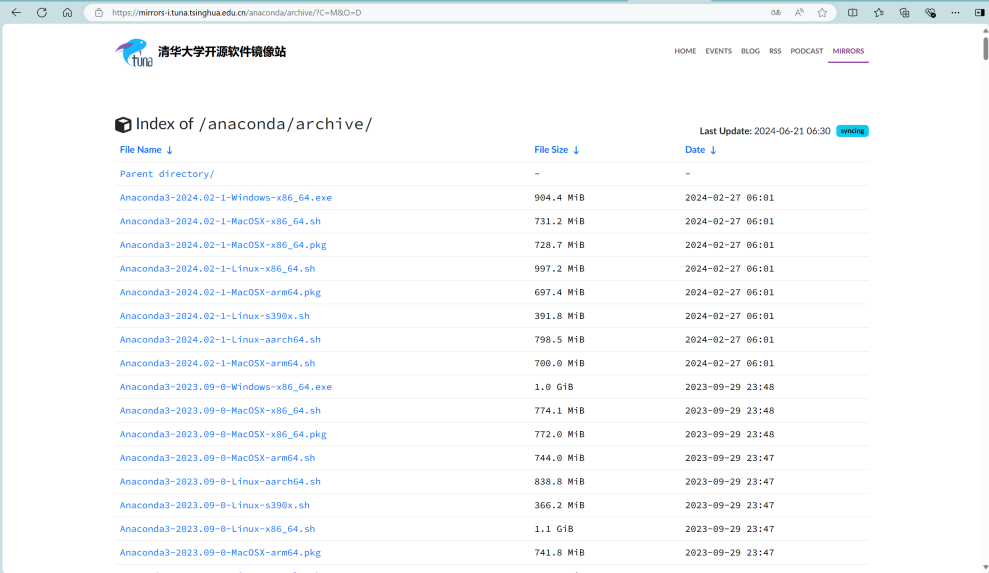
第二种方法是在网上收集好的别人标注好的训练集，而后对我们自己的模型进行训练。优点是耗时短，训练集中已经标注好且将训练集和验证集分类好了。缺点是网络上的数据集质量参差不齐，需要仔细辨别，还有可能和我们的目标差别较大，需要重新标注。

# 环境搭建

在本次大作业中，我们小组使用的是Yolov5+DeepSort相结合来进行人群计数的。故而本文档中所提及的相关环境均是在上述模型算法中适应搭建的。

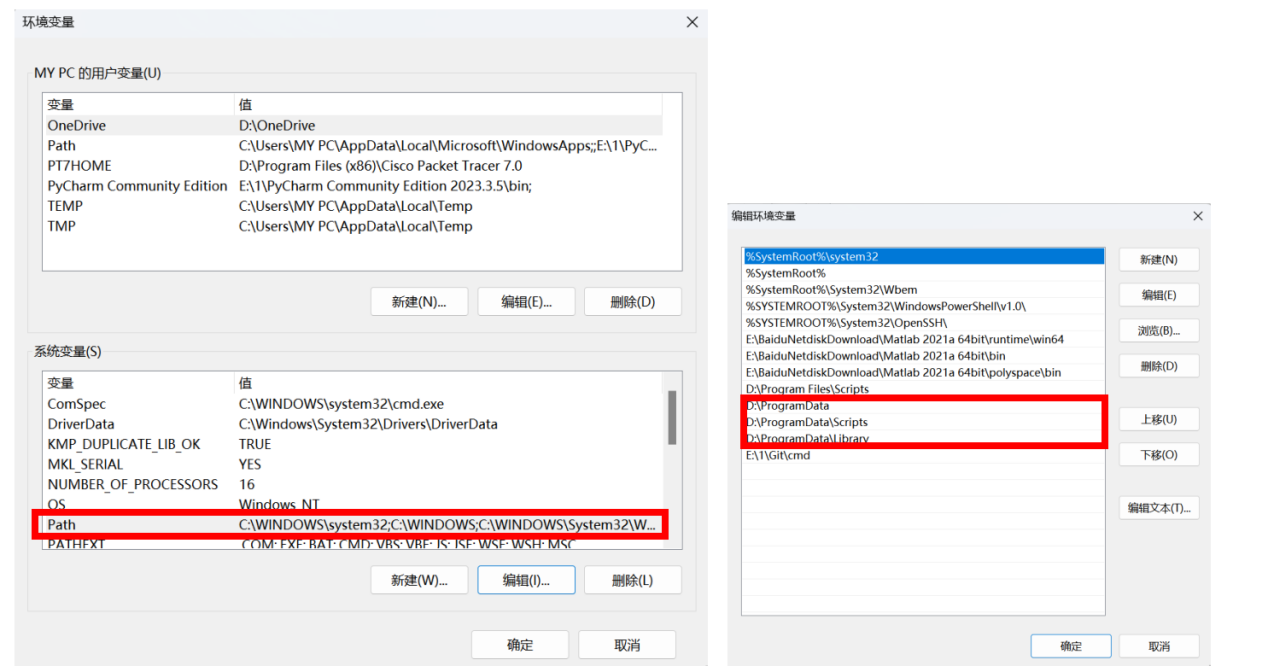
## Anaconda软件的下载

首先，我们需要先去下载Anaconda软件，方便后续进行虚拟环境的搭建以及管理。对于Anaconda软件，我们推荐去国内的镜像网站下载，如清华开源软件镜像站(https://mirrors-i.tuna.tsinghua.edu.cn/legacy\_index)。而后进入，有如下界面，根据使用的要求和电脑的版本进行选择下载即可。



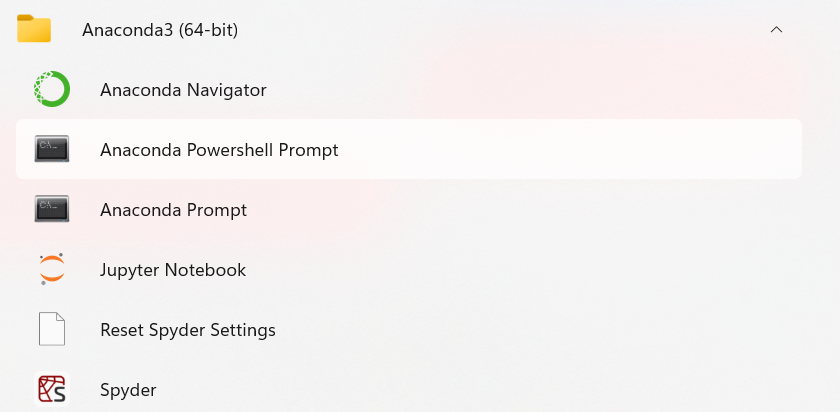
**图 1 Anaconda清华源网站下载界面**

在这里需要注意的是，需要在安装Anaconda软件，将其添加到环境变量中。直接在系统的搜索栏搜索环境变量，打开后，选择系统变量(S)-Path，双击进入，将刚才安装Anaconda软件的目录路径，及路径下的Scripts和Library添加到环境变量中。



**图 2 Anaconda环境变量配置**

随后打开Anaconda软件，可直接在系统搜索栏中搜索cmd，第一个谈弹出的就是Anaconda软件。也可以去应用中找，如下图所示。



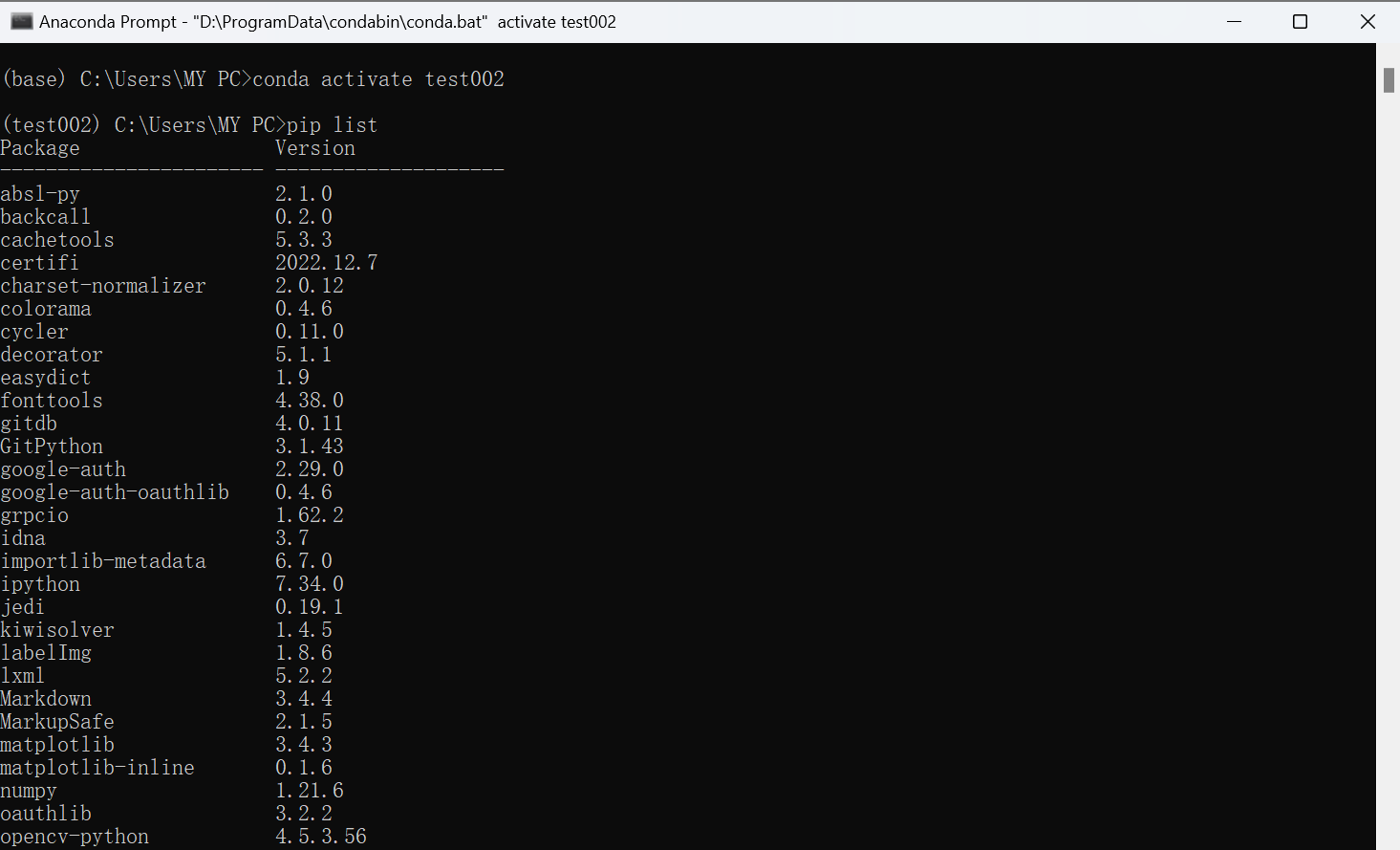
**图 3 Anaconda软件**

随后在命令行中，输入命令“conda create -n yourname python=3.8”即可设置自己的虚拟环境。在上述命令行中，yourname是指你创建的虚拟环境的名字，可自行设置；python=3.8是指定虚拟环境中，使用的python解释器为3.8版本，可根据项目需要进行自主选择调整。（在创建过程中，会弹出“Proceed ([y]/n?)”，回车键或者键入“y”皆可继续执行。）

创建完虚拟环境后，可通过命令“conda activate yourname”激活虚拟环境。我们小组创建的虚拟环境的名字为“test002”，以下皆用“test002”代替“yourname”。可以通过命令“conda env list”来展示本电脑中共有多少个虚拟环境。

## 更换国内源方便下载软件包

我们创建好虚拟环境之后，可以通过命令“pip list”来展示当前虚拟环境下，共安装了多少软件包。



**图 4 展示虚拟环境内安装的软件包**

因为我已经安装好了，所以此处展示的图片内有许多已经安装好的软件包。但是最开始里面的软件包是很少的。我们如果要运行代码需要软件包的话，是需要重新下载。而默认下载一般是从国外的网站开始下载的，由于是外网所以下载速度会非常慢，故而此处我们进行换源处理。国内的源有很多，有清华源、阿里源、豆瓣源等等，每个源基本上都一样，可自行选择。我们小组选择的是清华源(https://mirrors-i.tuna.tsinghua.edu.cn/help/pypi/)。

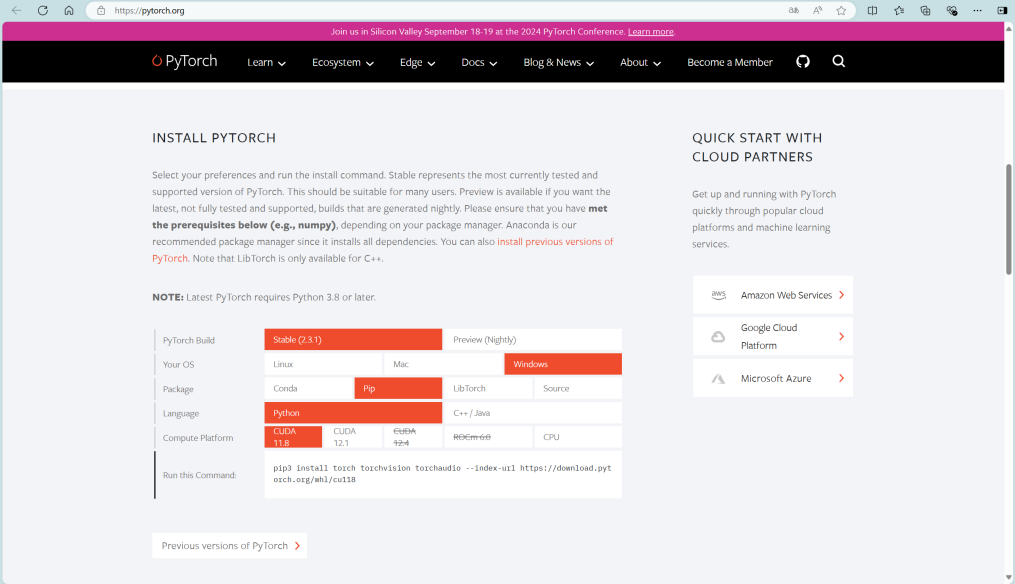


**图 5 清华源更换pypi源界面**

命令“pip config set global.index-url https://pypi.tuna.tsinghua.edu.cn/simple”填入anaconda的终端窗口中，即可完成替换。

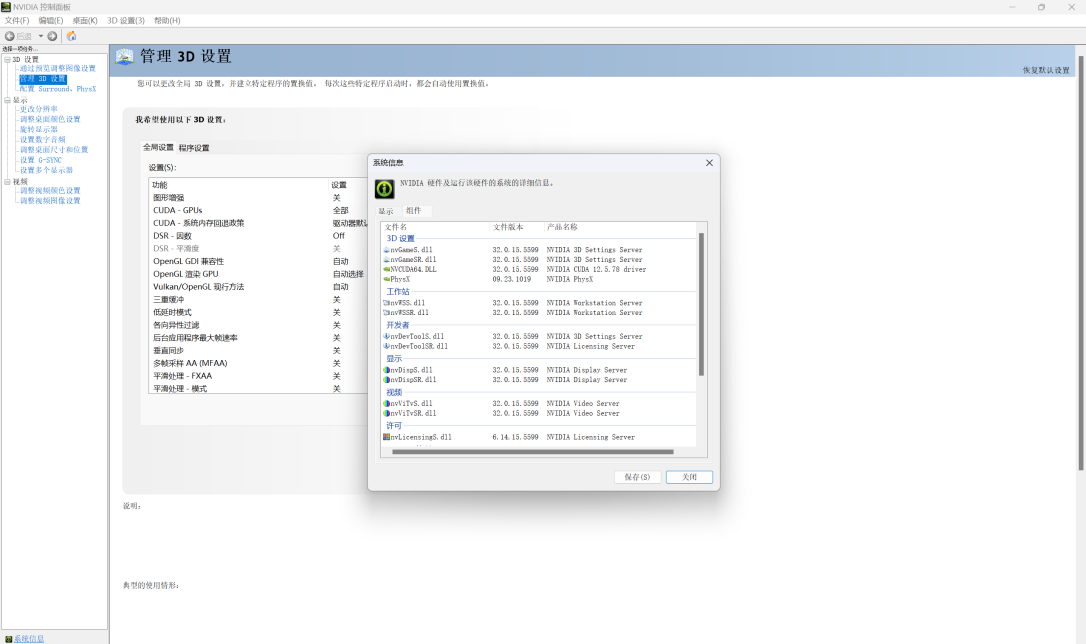
## Pytorch安装

换源之后我们需要进行Pytorch的安装，我们直接前往官网下载即可，可以通过浏览器搜索栏直接搜索“Pytorch”或者网址“https://pytorch.org/”前往。而后根据我们的电脑版本进行选择安装。



**图 6 Pytorch官网界面**

由上图我们根据自己电脑版本进行选择安装何种版本的Pytorch。一般是通过pip命令来安装。对于cuda和cpu的选择，如果电脑是NVDIA显卡，可在主屏幕右键进入NVDIA控制面板，打开系统组件，得到如下界面，选择自己对应版本的cuda安装即可。如果cuda版本过低，我们可以打开pytorch官网下面的“previous version of Pytorch”，查找过往版本，找到对应的版本即可。注意，不一定要一一对应cuda版本，可以选择比自己cuda版本低且稳定的。



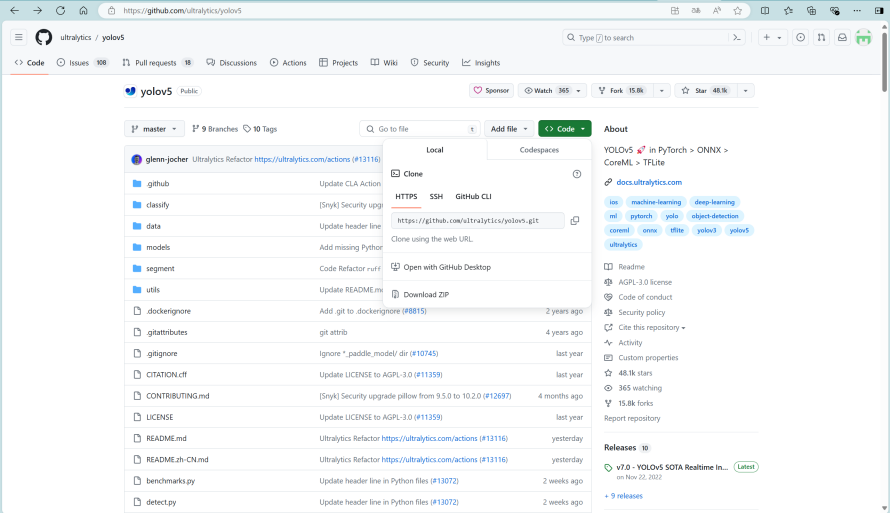
**图 7 cuda版本查找**

如果电脑是AMD显卡，则需进行cpu安装。直接选择cpu安装即可。我们小组就是进行cpu安装。

将得到的命令去anaconda创建的虚拟环境里键入，即可进行pytorch的下载。

## Yolov5安装

完成了上述的操作，即可前往官方Yolov5网站进行安装。点击右上方的code即可下载压缩包。(https://github.com/ultralytics/yolov5)

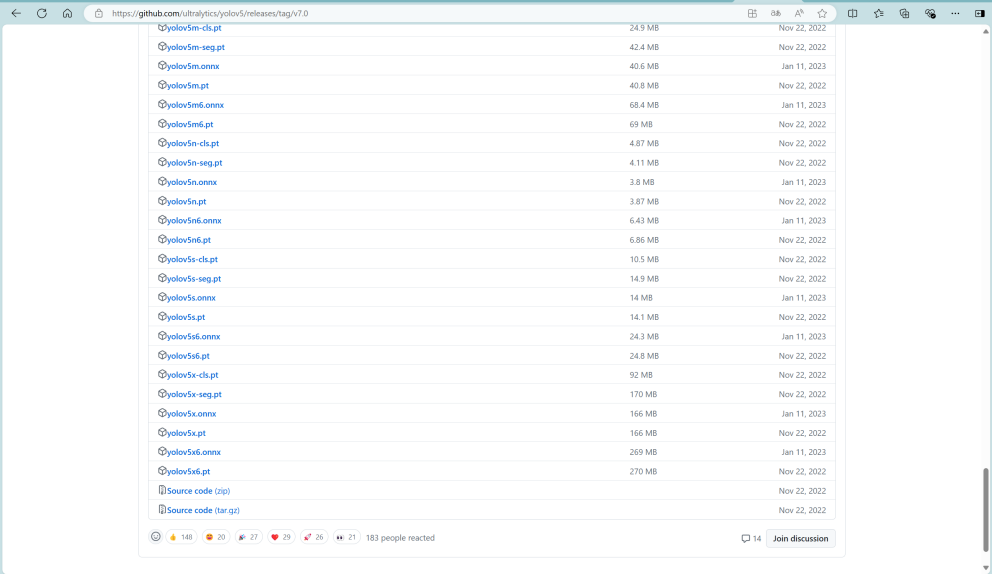


**图 8 Yolov5官方界面**

下载完压缩包后，选择好文件夹将压缩包拖入其中进行解压就好了。Yolov5所需要运行的环境在“requirements.txt”中已经表明了。根据自己的项目需求可进入“requirement.txt”进行软件包版本的调整。

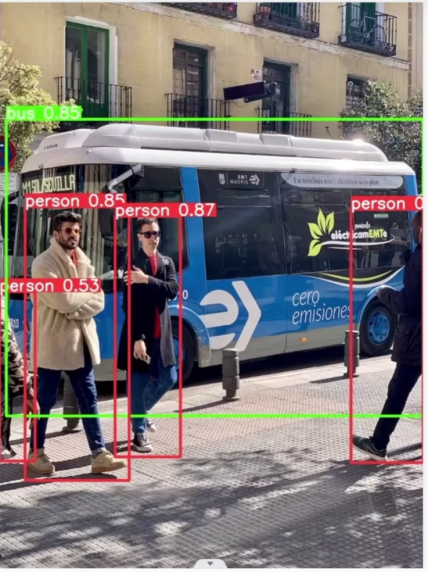
而后，使用Anaconda软件cd命令进入之前选择好的文件夹，使用命令“pip install -r requirement.txt”进行环境安装。当然，如果选择的路径比较繁琐，我们也可以进入Yolov5解压后的当前文件夹，在路径处直接输入cmd进入系统终端，激活之前设置的虚拟环境，然后使用同一条命令即可安装。

而后可以在终端或者Anaconda软件输入命令“python detect.py”进行检测是否安装无误。此处会提示出正在下载yolov5相关模型，如果下载进度比较慢，我们可以去官网下载并将其拖动到文件夹中。在**图8**右下角有Releases，点击进入。



**图 9 yolov5模型下载**

选取适合当前项目的模型下载，并将其复制粘贴到yolov5当前文件夹下，再次输入命令“python detect.py”即可得到结果。



**图 10 检测结果图**

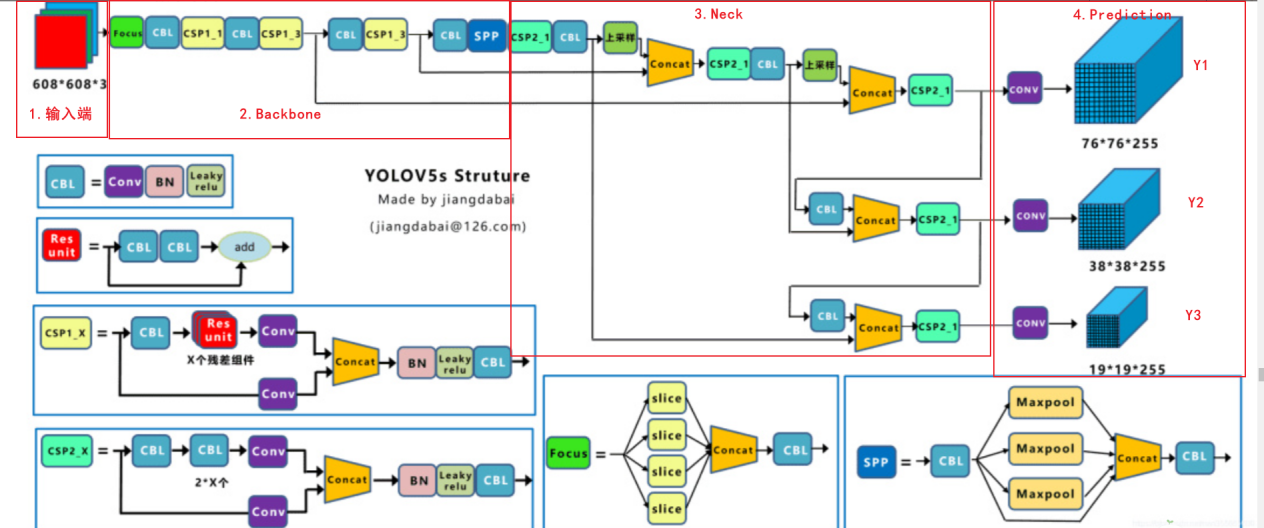
至此，yolov5所需环境已经全部安装完成了。

# 算法模块

本次大作业我们小组使用的是yolov5+deepsort相结合进行人群计数。Yolov5共提供了Yolov5s、Yolov5m、Yolov5l、Yolov5x等四个版本的目标检测网络，我们小组使用的是Yolov5s的目标检测网络。以下对于Yolov5的介绍均是以Yolov5s为主线来进行介绍的。

## Yolov5

Yolov5的网络结构如下图所示，主要包括Input、Backbone骨干网络、Neck颈部结构和Head头部结构(Prediction)四个模块，分别负责输入图片预处理、特征提取、特征融合、输出检测信息。其中，Backbone模块使用BottleNeckCSP结构从输入图像中提取丰富的信息特征。输入图像尺寸是608x608，特征图变化的规律是：608->304->152->76->38->19。



**图 11 yolov5s网络结构**

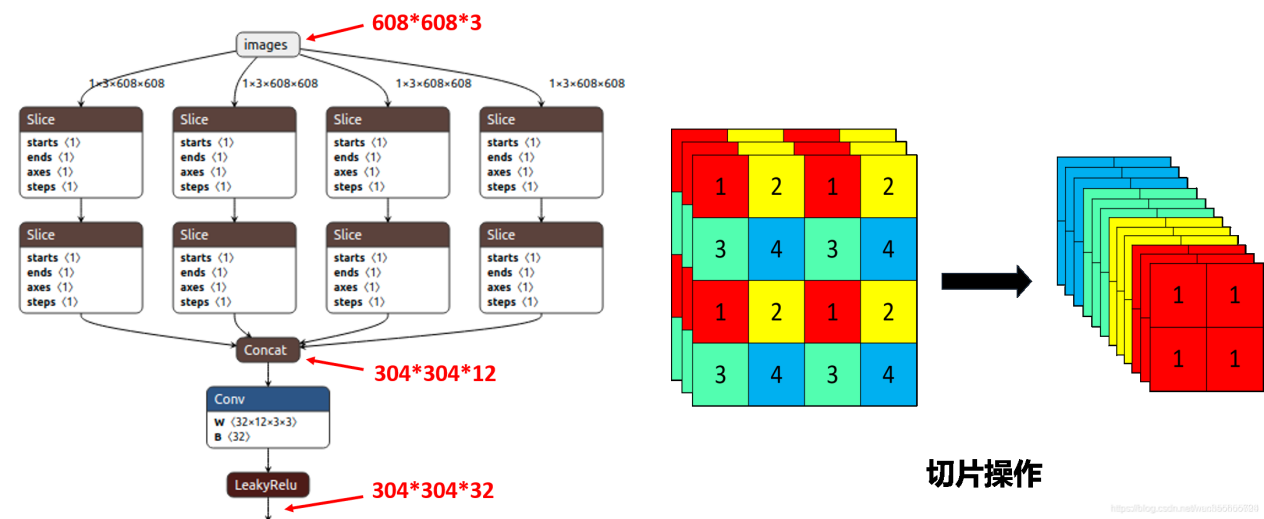
### 输入端

输入端主要对输入图片进行预处理。该网络输入图像大小为608\*608，预处理主要是将输入图像缩放至网络的输入大小，并进行归一化等操作。在网络训练阶段，Yolov5与Yolov4同样使用了Mosaic数据增强操作以提升模型的训练速度和网络精度；并提出了一种自适应锚框计算与自适应图片缩放方法。

### Backbone骨干网络

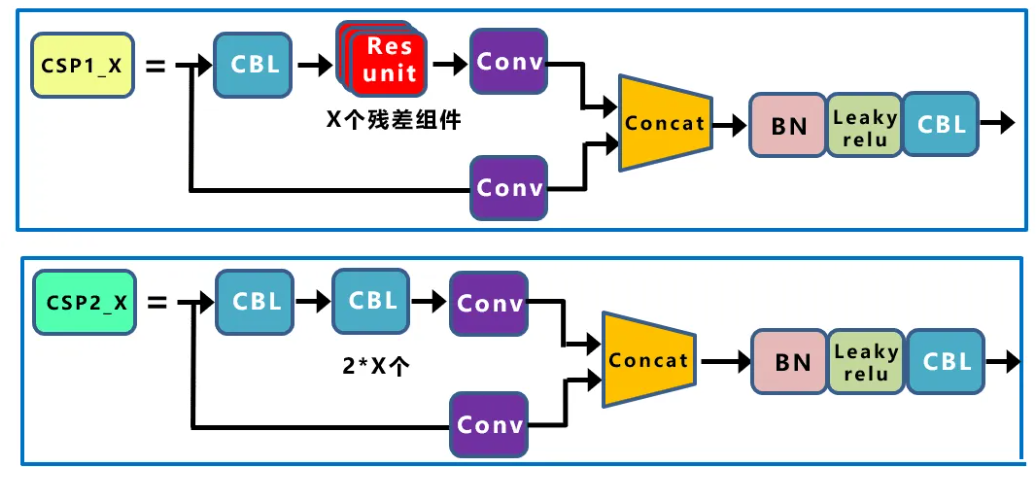
Yolov5与Yolov4相比，添加了Focus结构。其中最关键的是切片操作。例如下图（右）所示，在每张图片中每隔一个像素采集一个值，这样可组合为4张互补且无信息丢失的图片，同时将宽高信息集中到通道空间，输入通道扩充4倍，拼接后的图片相对于原图片的RGB三通道模式变为12通道，即4x4x3的图像经切片后变为了2x2x12的特征图。

而在Yolov5s网络结构中，608x608x3的图像输入经Focus结构，进行切片操作，先变为304x304x12的特征图，再经过一次32个卷积核的卷积操作，最终变为304x304x32的特征图。



**图 12 Focus结构**

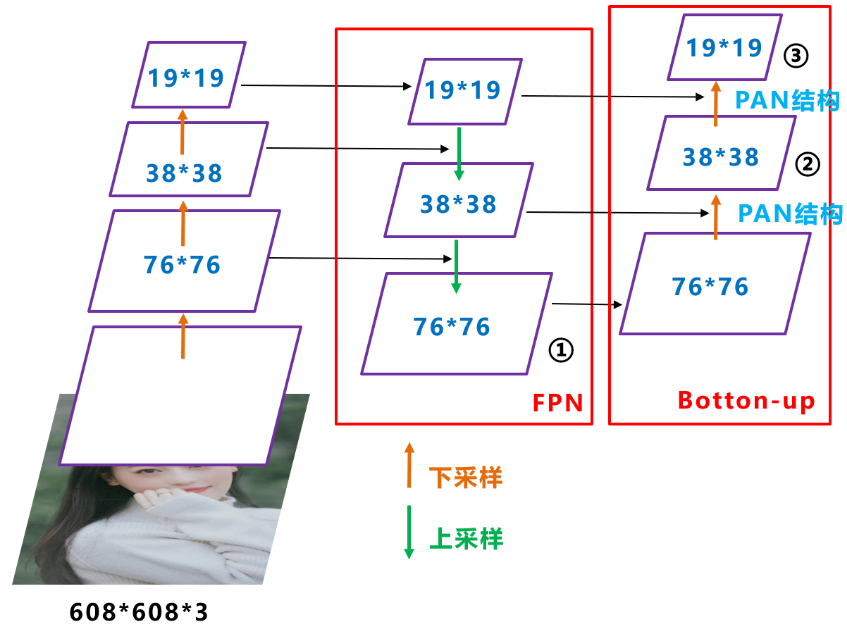
早在Yolov4网络结构中，便借鉴了CSPNet的设计思路，在主干网络中使用了CSP结构。Yolov5则是设计了两种CSP结构，其中CSP1\_x结构应用于Backbone主干网络，另一种CSP2\_x结构应用于Neck中。



**图 13 CSP结构**

### Neck颈部结构

Yolov5更新后的Neck与Yolov4同采用了FPN+PAN结构，如下图所示。



**图 14 FPN+PAN结构**

整个网络的输入为608x608，然后经过CSP模块，生成一个76x76大小的特征映射，再经过两次下采样操作之后生成19x19的特征映射。

接着传入FPN结构，依次对19x19、38x38、76x76执行融合操作，即先对比较小的特征映射层执行上采样操作，将其调整成相同大小，然后将两个同等大小的特征映射叠加起来。通过FPN操作可以将19x19大小的特征映射调整为76x76大小，这样不仅提升了特征映射的大小，可以更好的解决检测中尺度问题，而且增加了网络的深度，提升了网络的鲁棒性。

最后将其传入PAN结构，PANet网络的PAN结构是将两个相同大小的特征映射执行按位加操作，YOLOv5与Yolov4中同样使用Concat操作来代替它。经过两个PAN结构，可将76x76大小的特征映射重新调整为19x19大小，这样可以在一定程度上提升该算法的目标定位能力。

FPN层自顶向下可以捕获强语义特征，而PAN则通过自底向上传达强定位特征，通过组合这两个模块，可以很好的完成目标定位的功能。

### Prediction

Yolov5延用了Yolov3和Yolov4的多尺度融合检测方法，与Yolov4采用CIOU\_Loss作为Bounding box损失函数不同的是，Yolov5中采用GIOU\_Loss作为Bounding box的损失函数。

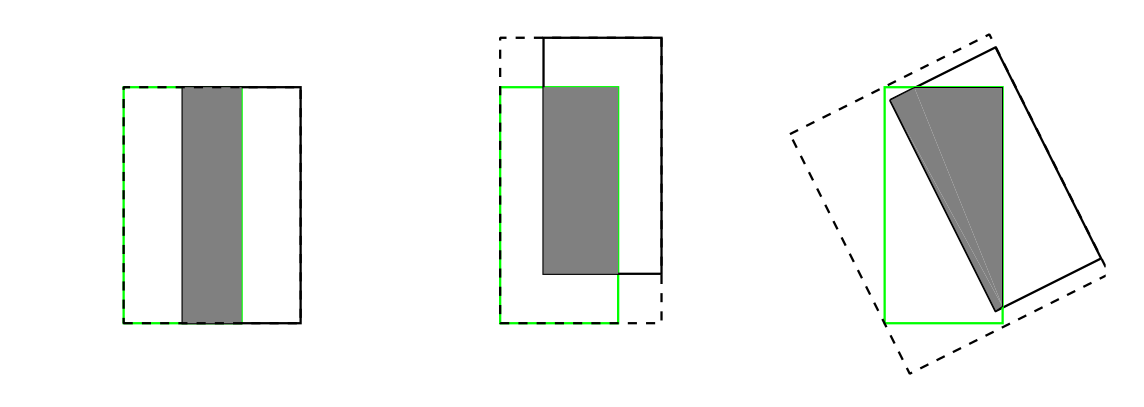
IoU即为交并比，用于度量目标检测中预测框与真实框的重叠程度。计算公式如下：

显而易见，IoU的值越高也说明预测框与真实框重合程度越高，模型预测越准确。反之，IoU越低模型性能越差。

IoU Loss计算公式如下：

由下图可以看出，如果两个目标没有重叠，IoU将会为0，并且不会反应两个目标之间的距离，在这种无重叠目标的情况下，如果IoU用作于损失函数，梯度为0，无法优化。

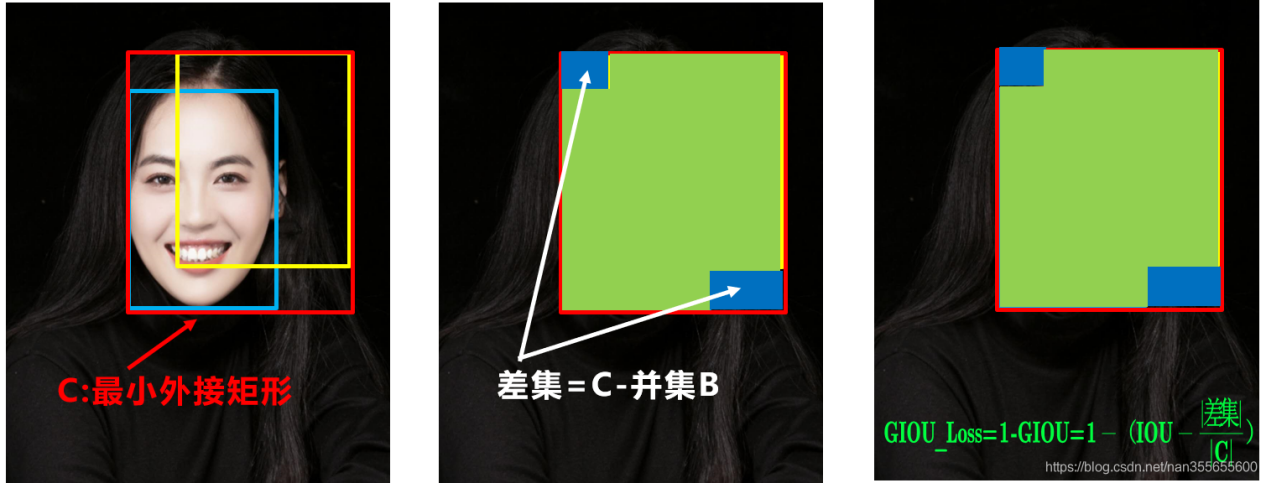
并且，IoU无法区分两个对象之间不同的对齐方式。更通俗的理解就是，不同方向上有相同交叉级别的两个重叠对象的IoU会完全相等。



**图 15 IoU**

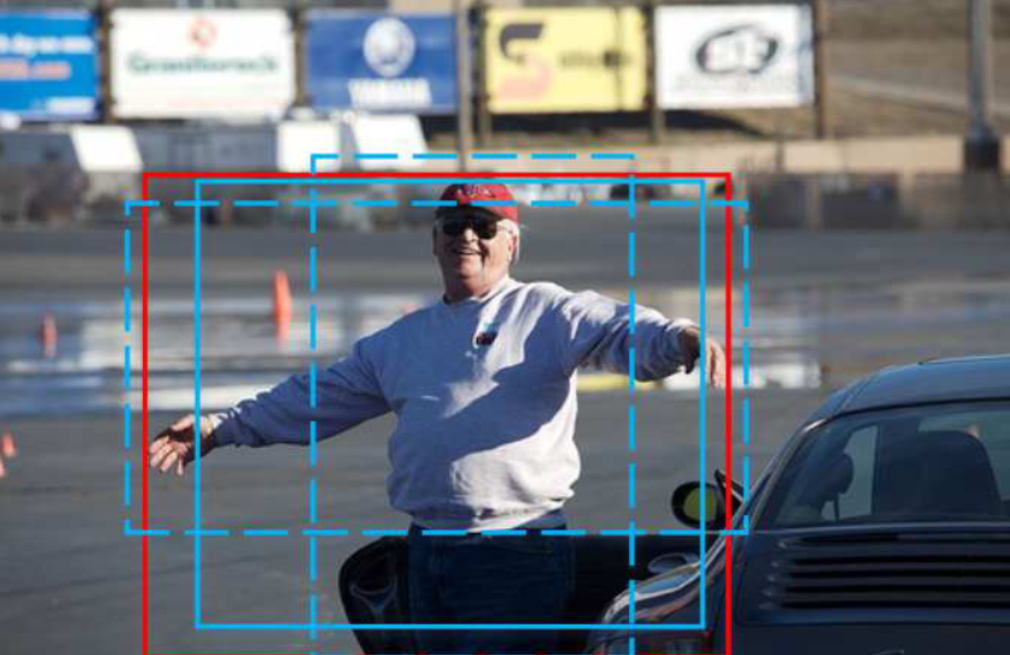
GIoU加入了C检测框（C检测框是包含了检测框和真实框的最小矩形框），这样就可以解决检测框和真实框没有重叠的问题。但是当检测框和真实框之间出现包含的现象的时候GIoU就和IoU loss是同样的效果了。GIoU Loss计算如下：

其中，C为能包含预测框和真实框的最小box。



**图 16 GIoU具体表现**

在目标检测的后处理过程中，针对很多目标框的筛选，通常需要进行NMS操作。Yolov4在DIOU\_Loss的基础上采用DIOU\_nms的方式，而Yolov5中仍然采用加权NMS的方式。



**图 17 加权NMS**

加权NMS通过分类置信度和IOU对同类物体所有的边框坐标进行加权平均，并归一化。其中加权对象包括目标物自身和IOU>阈值的相邻框。

假定所有的Box来自相同的物体，通过考虑非极大结果充分考虑了目标的信息，提出了如下的非极大权重（Non-Maximum Weighting）：

至此，关于Yolov5算法原理梳理基本完成。

## Deepsort

DeepSORT（Deep learning based SORT）是一种基于深度学习的视觉目标跟踪算法，它结合了深度学习和传统的目标跟踪算法SORT（Simple Online and Realtime Tracking）。

DeepSORT基于目标检测器（如YOLO、Faster R-CNN等）检测每一帧图像中的目标，并使用多特征融合（Muti-feature Fusion）技术对目标进行表示和描述，然后使用SORT算法对目标进行跟踪。在SORT算法的基础上，DeepSORT引入了Re-IDentification（Re-ID）模型来解决目标ID的确定问题，Re-ID模型通过计算目标在多个帧图像中的相似度来确定目标的唯一ID。

DeepSORT算法的优点是：精度高，鲁棒性强，对于目标的遮挡、形变等情况具有很好的适应性。它已经被广泛应用于行人、车辆等目标的跟踪和智能视频监控等领域。

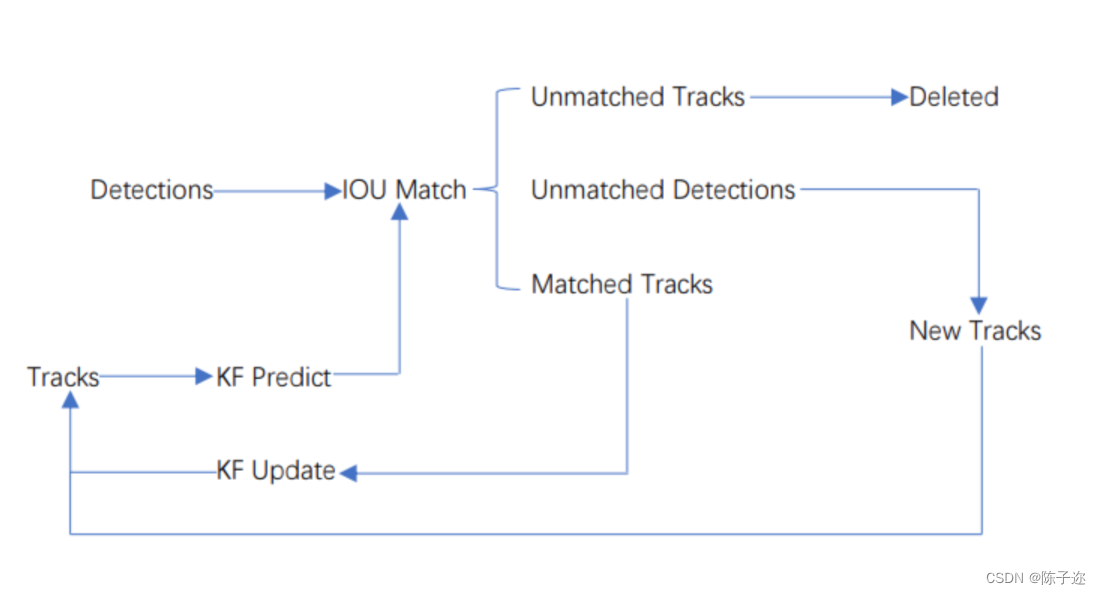
### 传统sort算法的流程

Deepsort的前身是sort算法，sort算法的核心是卡尔曼滤波算法和匈牙利算法。

卡尔曼算法作用：该算法的主要作用就是当前的一系列运动变量去预测下一时刻的运动变量，但是第一次的检测结果用来初始化卡尔曼滤波的运动变量。

匈牙利算法的作用：简单来讲就是解决分配问题，就是把一群检测框和卡尔曼预测的框做分配，让卡尔曼预测的框找到和自己最匹配的检测框，达到追踪的效果。

Sort工作流程如下图：



**图 18 传统sort工作流程**

Detections是通过目标检测到的框框。Tracks是轨迹信息。

第一步，将第一帧检测到的结果创建其对应的Tracks。将卡尔曼滤波的运动变量初始化，通过卡尔曼滤波预测其对应的框框。

第二步，将该帧目标检测的框框和上一帧通过Tracks预测的框框一一进行IOU匹配，再通过IOU匹配的结果计算其代价矩阵（cost matrix，其计算方式是1-IOU）。

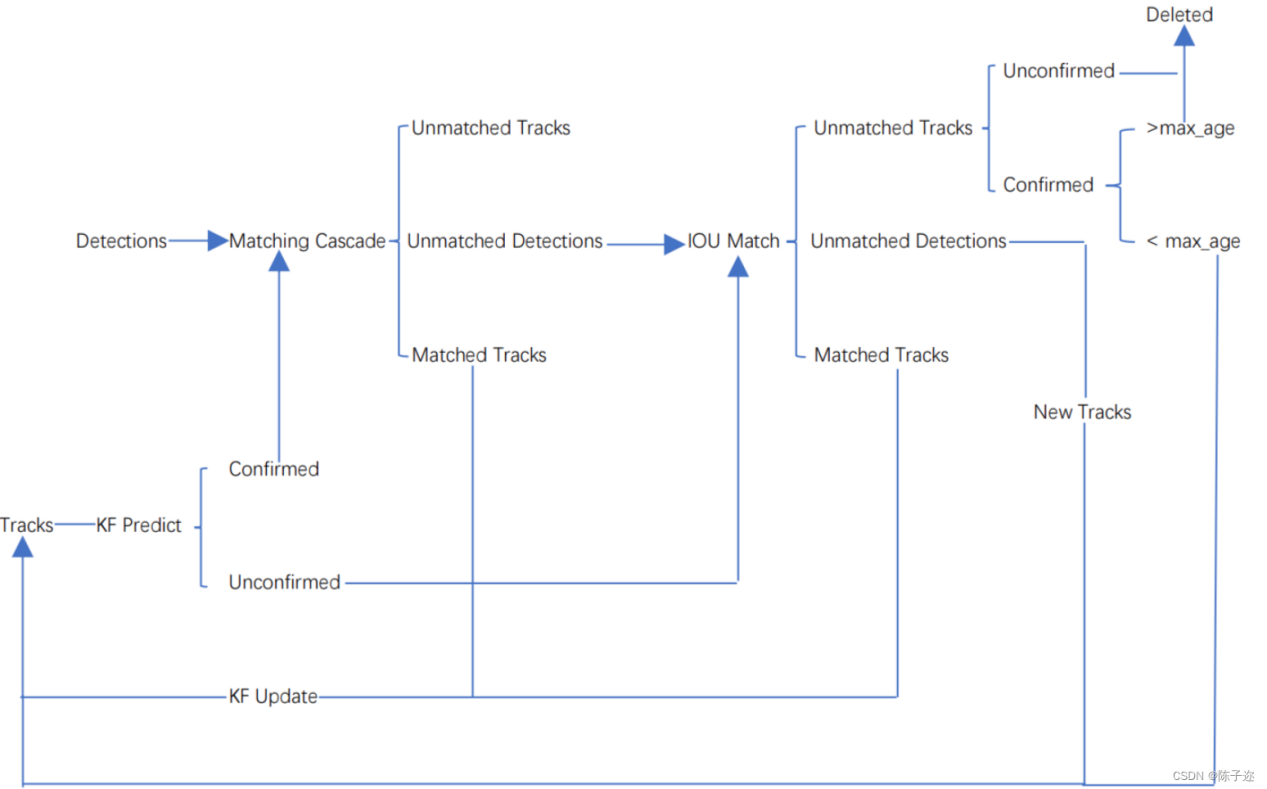
第三步，将第二步中得到的所有的代价矩阵作为匈牙利算法的输入，得到线性的匹配的结果，这时候我们得到的结果有三种，第一种是Tracks失配（Unmatched Tracks），我们直接将失配的Tracks删除；第二种是Detections失配（Unmatched Detections），我们将这样的Detections初始化为一个新的Tracks（new Tracks）；第三种是检测框和预测的框框配对成功，这说明我们前一帧和后一帧追踪成功，将其对应的Detections通过卡尔曼滤波更新其对应的Tracks变量。

第四步，反复循环第二步和第三步，直到视频帧结束。

### Deepsort算法流程

由于sort算法还是比较粗糙的追踪算法，当物体发生遮挡的时候，特别容易丢失自己的ID。而Deepsort算法在sort算法的基础上增加了级联匹配（Matching Cascade）和新轨迹的确认（confirmed）。Tracks分为确认态（confirmed），和不确认态（unconfirmed），新产生的Tracks是不确认态的；不确认态的Tracks必须要和Detections连续匹配一定的次数（默认是3）才可以转化成确认态。确认态的Tracks必须和Detections连续失配一定次数（默认30次），才会被删除。

Deepsort算法的工作流程如下图所示：



**图 19 deepsort工作流程**

整个算法的工作流程如下：

第一步，将第一帧次检测到的结果创建其对应的Tracks。将卡尔曼滤波的运动变量初始化，通过卡尔曼滤波预测其对应的框框。这时候的Tracks一定是unconfirmed的。

第二步，将该帧目标检测的框框和第上一帧通过Tracks预测的框框一一进行IOU匹配，再通过IOU匹配的结果计算其代价矩阵（cost matrix，其计算方式是1-IOU）。

第三步，将第二步中得到的所有的代价矩阵作为匈牙利算法的输入，得到线性的匹配的结果，这时候我们得到的结果有三种，第一种是Tracks失配（Unmatched Tracks），我们直接将失配的Tracks（因为这个Tracks是不确定态了，如果是确定态的话则要连续达到一定的次数（默认30次）才可以删除）删除；第二种是Detections失配（Unmatched Detections），我们将这样的Detections初始化为一个新的Tracks（new Tracks）；第三种是检测框和预测的框框配对成功，这说明我们前一帧和后一帧追踪成功，将其对应的Detections通过卡尔曼滤波更新其对应的Tracks变量。

第四步，反复循环第二步和第三步的步骤，直到出现确认态（confirmed）的Tracks或者视频帧结束。

第五步，通过卡尔曼滤波预测其确认态的Tracks和不确认态的Tracks对应的框框。将确认态的Tracks的框框和是Detections进行级联匹配（之前每次只要Tracks匹配上都会保存Detections其的外观特征和运动信息，默认保存前100帧，利用外观特征和运动信息和Detections进行级联匹配,这么做是因为确认态（confirmed）的Tracks和Detections匹配的可能性更大）。

第六步，进行级联匹配后有三种可能的结果。第一种，Tracks匹配，这样的Tracks通过卡尔曼滤波更新其对应的Tracks变量。第二第三种是Detections和Tracks失配，这时将之前的不确认状态的Tracks和失配的Tracks一起和Unmatched Detections一一进行IOU匹配，再通过IOU匹配的结果计算其代价矩阵（cost matrix，其计算方式是1-IOU）。

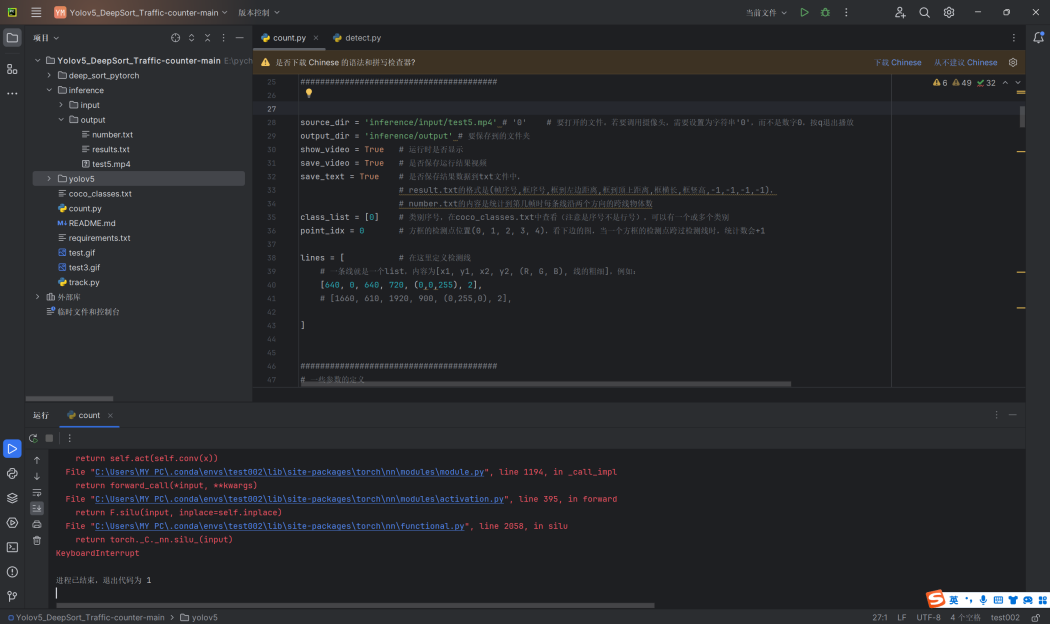
第七步，将第六步中得到的所有的代价矩阵作为匈牙利算法的输入，得到线性的匹配的结果，这时候我们得到的结果有三种，第一种是Tracks失配（Unmatched Tracks），我们直接将失配的Tracks（因为这个Tracks是不确定态了，如果是确定态的话则要连续达到一定的次数（默认30次）才可以删除）删除；第二种是Detections失配（Unmatched Detections），我们将这样的Detections初始化为一个新的Tracks（new Tracks）；第三种是检测框和预测的框框配对成功，这说明我们前一帧和后一帧追踪成功，将其对应的Detections通过卡尔曼滤波更新其对应的Tracks变量。

第八步，反复循环第五步到第七步步骤，直到视频帧结束。

# 整体系统演示及结果

下面是关于代码运行及系统演示结果。

首先根据上述步骤配置好环境，而后进行调参。选取好要计数的视频，选择是否保存输出结果，是否在运行代码时实时显示视频，画线确定方向等。



**图 20 调整参数**

而后点击右上角的运行，或者在终端命令行输入“python count.py”即可。

以下是结果展示。在输出文件夹中共有三个文件，分别是number.txt，result.txt，test5.mp4。Number文档记录输出视频左上角人数的变化，result.txt内容为(帧序号,框序号,框到左边距离,框到顶上距离,框横长,框竖高,-1,-1,-1,-1)，test5.mp4为结果输出视频。



**图 21 演示视频结果截图**

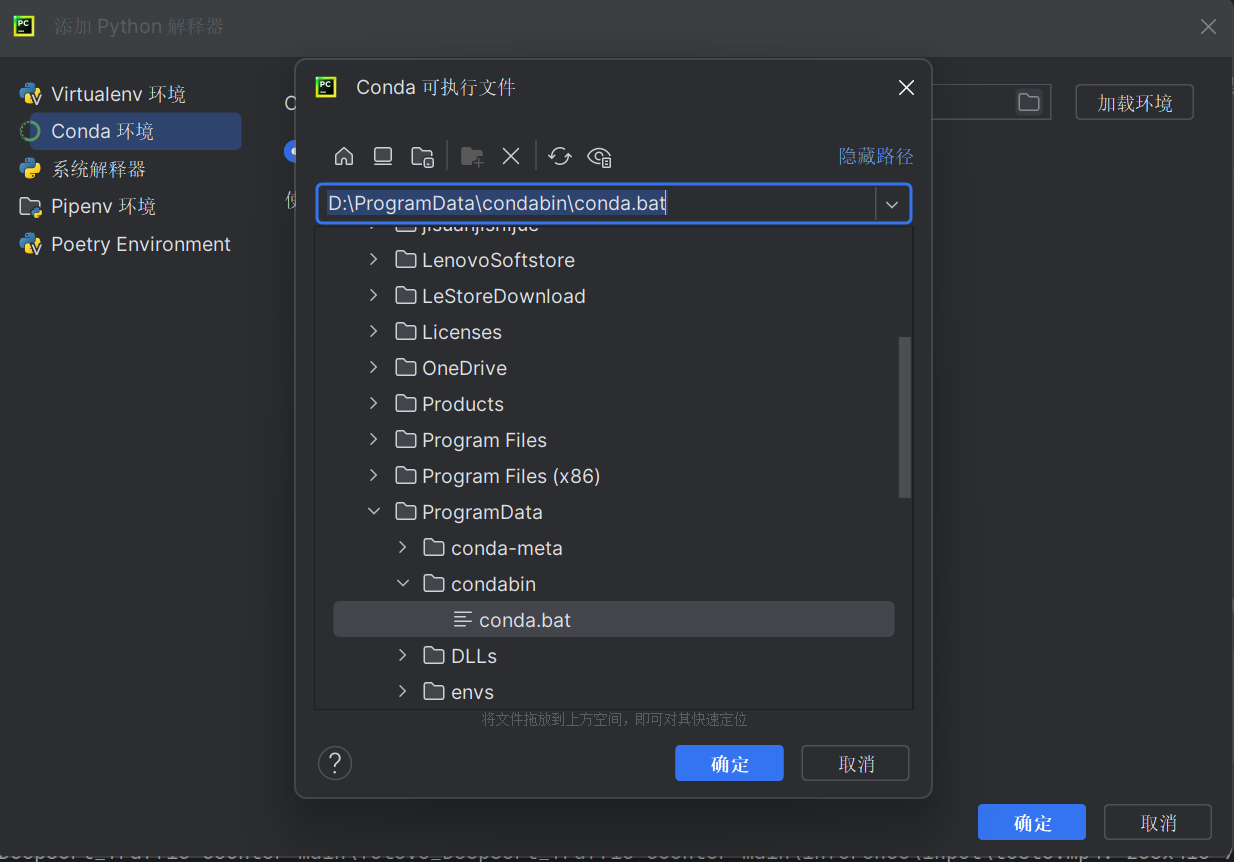


**图 22 输出文件夹**

# 问题及相关解答

* 1. Pycharm使用conda创建的虚拟环境时找不到python.exe。

可能是condaba版本不一样，新版本选不到.exe文件，在anaconda软件的安装目录下选择condabin——>conda.bat，然后加载环境，就可以选择创建的虚拟环境了。



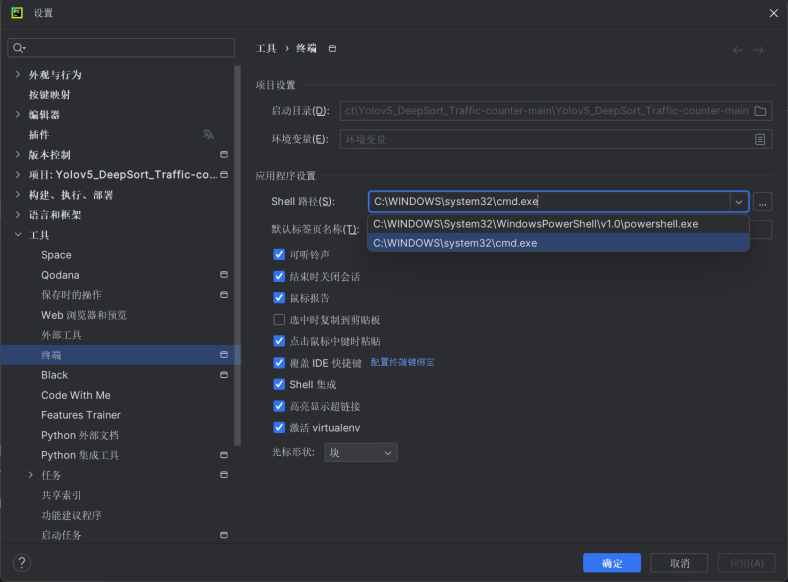
**图 23 选择conda.bat使pycharm获得创建的虚拟环境**

* 1. pip命令无法执行Python问题Unable to create process using...

前往本电脑conda创建的虚拟环境下，即envs目录下，删除\Lib\site-packages\pip-19.1.dist-info，删除\Scripts\pip\*.exe（所有pip开头的），在anaconda软件输入命令“python -m pip install --upgrade pip”。

* 1. pycharm 终端无法激活conda环境

原因是pycharm默认的终端是Windows PowerShell，将其改成cmd.exe即可。

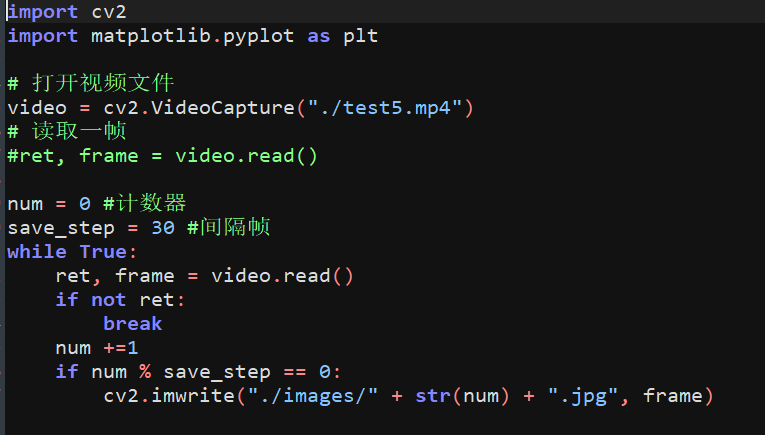


**图 24 pycharm终端设置界面**

* 1. 如何创建自己的训练集。

此处介绍较为简略，更详细地介绍在随word上传的**问题表格**中有链接。

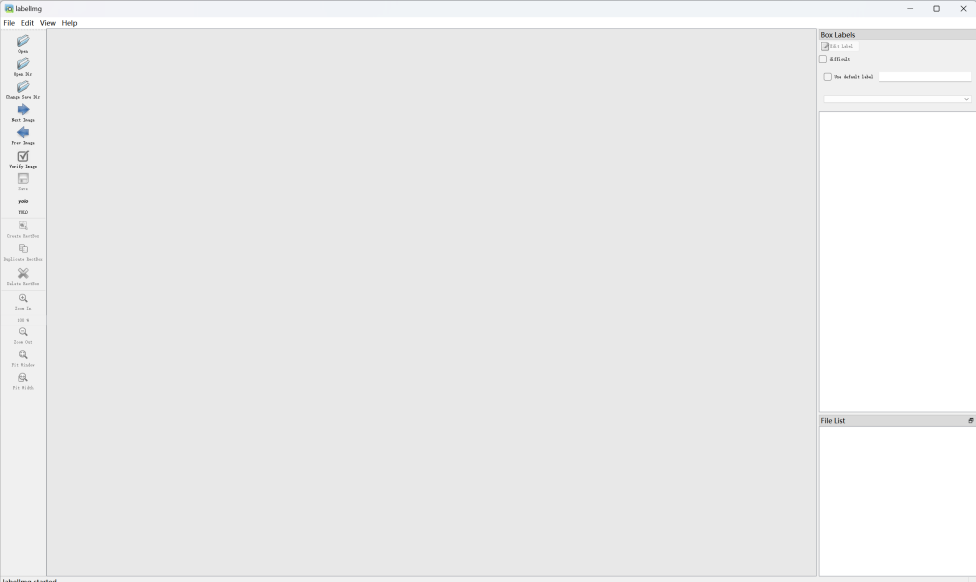
首先运行视频帧提取代码，此代码较为简单，使用的是cv2库中的函数。



**图 25 提取视频帧代码**

而后我们将得到根据设定好的帧数提取出来的图片。

随之打开anaconda软件，输入命令“pip install labelimg”，安装labelimg软件。注意此处应该与之前pypi换源相联系，不然下载进度会非常慢甚至失败。如未进换行源，也可在anaconda软件通过输入命令“pip install labelimg -i https://pypi.tuna.tsinghua.edu.cn/simple ”进行安装。而后我们直接打开输入命令“labelimg”即可打开进行标注。



**图 26 labelimg界面**

可以选取存储路径，存储格式等等。存储格式应选择yolov5能识别的格式。保存好之后，我们可以得到下列.txt文件。



**图 27 标注好之后得到的txt文件**

而后根据所需要设置的训练集和验证集进行区分即可。

至此，数据集的搭建已经完成。