

# ASL Detection

杨逸君\*  
201800302011  
scottyang924@163.com  
18 级人工智能班  
ShanDong University  
Qingdao, ShanDong, China

朱炳文  
201800180145  
201800180145@mail.sdu.edu.cn  
18 级人工智能班  
ShanDong University  
Qingdao, ShanDong, China

侯晓阳  
201800250012  
houxiaoyanghxy@mail.sdu.edu.cn  
18 级人工智能班  
ShanDong University  
Qingdao, ShanDong, China



## ABSTRACT

我们项目的主题是多场景目标检测，工作主要基于两大场景：一个是日常生活，使用 VOC 数据集，复现了 Faster RCNN、Mask RCNN、Cascade RCNN、YOLOv5、SSD 五个目标检测算法，实现了图片的目标检测与搜索。重点是第二个，美式手语实时检测场景，American Sign Language，包含 26 类，我们针对美式手语自制了数据集，经过不断扩充最终获得包含训练集 3200 张图片、验证集 630 张图片的 ASL 数据集，提高了模型的泛化能力，使其能够更好地应对不同背景下的实时检测。我们着重实现了 YOLOv5，在源码的基础上实现了稀疏训练与剪枝，优化了模型结构，提升了模型检测速度并保持了准确度，使其更“专注”于 ASL。我们还实现了 Cascade RCNN 和 SSD，与 YOLOv5 的效果作比较。最后我们基于这两个场景做了一个完整的前端，可视化结果，方便使用。

## KEYWORDS

Object Detection, Daily Life, American Sign Language, Faster RCNN, Mask RCNN, Cascade RCNN, YOLOv5, SSD

## 1 INTRODUCTION

我们项目的主题是多场景目标检测，分为两大场景：日常生活 Daily Life 和美式手语 American Sign Language。但由于美式手语检测是我们工作的重点，所以给项目取名 ASL Detection。

目标检测是计算机视觉领域的重要应用之一，任务是找出图像中所有感兴趣的物体，包含物体定位和物体分类两个子任务，同时确定物体的类别和位置。

在日常场景检测任务中我们使用了 VOC 数据集，它包含了我们日常生活中常见的 20 类。我们通过这个任务入手目标检测，复现了五大经典目标检测算法——Faster RCNN、Mask RCNN、Cascade RCNN、SSD、YOLOv5，实现了图片目标检测，其中 Faster RCNN 还有图片的目标搜索功能，Mask RCNN 包含

了实例分割部分。该场景可以满足于我们日常生活的基本检测工作，能够代替人力检测生活中常见的 20 类人和物。

在美式手语实时检测场景中，我们自制了 ASL (American Sign Language) 数据集，美式手语类别是 26 个大写字母，字母手势如 Figure 1，它区别于 VOC，属于小物体目标检测，不同类别之间的区别更小，比如这里的 M 和 N，更容易受背景干扰，检测难度更大。大部分都是静态手语，但也存在像 J 和 Z 这样的动态手语。我们希望实现一个完整的美式手语实时检测，既保证实时性又保持较高的准确性，让更多人关注手语，让每个人都能够读懂手语，希望这个项目使人们可以去理解聋哑人的世界。

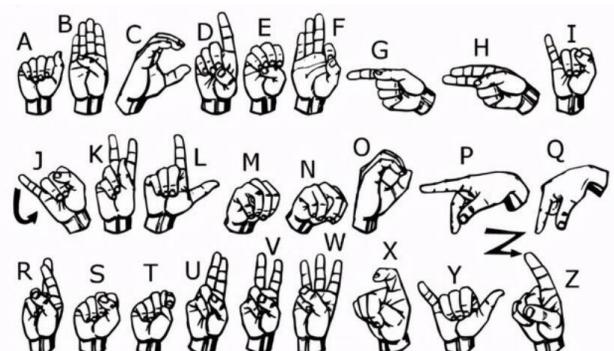


Figure 1: American Sign Language

基于之前的经验，我们从 5 个模型中挑出 Cascade、YOLO、SSD 为我们的手语目标检测服务。我们着重研究了 YOLOv5 以及稀疏训练与剪枝的优化，对这 26 个类别，自己制作了数据集，同时我们将多种增强方法同时运用到一张图片，采用更多的增强方法，最后得到训练集 3200 张图片，验证集 630 张。

用最终 ASL 数据集训练优化的 YOLOv5，具有较强的泛化能力，可以很好地应对不同背景，同时拥有很快的检测速度满足实时性要求，并保持了较高的准确度。

除了 YOLO，我们还应用了 Cascade RCNN，比较双阶段与单阶段算法，实验得出 Cascade RCNN 很慢，准确率的优势也不明显。此外我们还应用了 SSD，比较单阶段的两个算法，SSD 的效果出乎意料，可与 YOLOv5 媲美。

我们希望通过第一个日常生活场景熟悉目标检测领域的经典算法及其应用方法，以此为奠基，为我们的 ASL Detection 提供经验与教训，从中挑选适合的算法，从拍摄到标定 label，全程自主制作 ASL 数据集，训练手语检测模型，并设计完整的前后端交互，实现一个完备且能投入使用、对用户友好的美式手语实时检测系统。

## 2 RELATED WORK

### 2.1 Object Detection

Object Detection 就是在给定的图片中精确找到物体所在位置，并标注出物体的类别，所以 Object Detection 要解决的问题就是物体在哪里以及是什么的整个流程问题。然而这个问题可不是那么容易解决的：物体的尺寸变化范围很大，摆放物体的角度，姿态不定，而且可以出现在图片的任何地方，更何况物体还可以是多个类别。

目前学术和工业界出现的目标检测算法分成 3 类：

- 传统的目标检测算法：Cascade + HOG/DPM + Haar/SVM 以及上述方法的诸多改进、优化
- 候选区域/窗 + 深度学习分类：通过提取候选区域，并对相应区域进行以深度学习方法为主的分类的方案，如：
  - R-CNN: Selective Search + CNN + SVM
  - SPP-net: ROI Pooling
  - Fast R-CNN: Selective Search + CNN + ROI
  - Faster R-CNN: RPN + CNN + ROI
  - R-FCN 等系列方法。
- 区域选择基于深度学习的回归方法：YOLO/SSD/DenseBox 等方法；以及最近出现的结合 RNN 算法的 RRC detection；结合 DPM 的 Deformable CNN 等。

传统目标检测流程：

- 区域选择：穷举策略，采用滑动窗口，且设置不同的大小，不同的长宽比对图像进行遍历，时间复杂度高
- 特征提取：SIFT、HOG 等；形态多样性、光照变化多样性、背景多样性使得特征鲁棒性差
- 分类器分类：主要有 SVM、Adaboost 等

### 2.2 Traditional Algorithms

对一张图片，用各种大小的框（遍历整张图片）将图片截取出来，输入到 CNN，然后 CNN 会输出这个框的得分（classification）以及这个框图片对应的 x,y,h,w（regression）。

传统目标检测的主要问题是：

- 基于滑动窗口的区域选择策略没有针对性，时间复杂度高，窗口冗余
- 手工设计的特征对于多样性的变化没有很好的鲁棒性

### 2.3 R-CNN

预先找出图中目标可能出现的位置，即候选区域（Region Proposal）。利用图像中的纹理、边缘、颜色等信息，可以保证在选取较少窗口（几乎甚至几百）的情况下保持较高的召回率。

所以，问题就转变成找出可能含有物体的区域/框（也就是候选区域/框，比如选 2000 个候选框），这些框之间是可以互相重叠互相包含的，这样我们就可以避免暴力枚举的所有框了。有了候选区域，剩下的工作实际就是对候选区域进行图像分类的工作（特征提取 + 分类）。

2014 年，RBG (Ross B. Girshick) 使用 Region Proposal + CNN 代替传统目标检测使用的滑动窗口 + 手工设计特征，设计了 R-CNN 框架，使得目标检测取得巨大突破，并开启了基于深度学习目标检测的热潮。

R-CNN 的简要步骤如下：

- 输入测试图像
- 利用选择性搜索 Selective Search 算法在图像中从下到上提取 2000 个左右的可能包含物体的候选区域 Region Proposal
- 因为取出的区域大小各自不同，所以需要将每个 Region Proposal 缩放 (warp) 成统一的 227x227 的大小并输入到 CNN，将 CNN 的 fc7 层的输出作为特征
- 将每个 Region Proposal 提取到的 CNN 特征输入到 SVM 进行分类

R-CNN 虽然不再像传统方法那样穷举，但 R-CNN 流程的第一步中对原始图片通过 Selective Search 提取的候选框 region proposal 多达 2000 个左右，而这 2000 个候选框每个框都需要进行 CNN 提特征 + SVM 分类，计算量很大，导致 R-CNN 检测速度很慢，一张图都需要 47s。

### 2.4 Fast R-CNN

即使使用了 Selective Search 等预处理步骤来提取潜在的 bounding box 作为输入，但是 R-CNN 仍会有严重的速度瓶颈，原因也很明显，就是计算机对所有 region 进行特征提取时会有重复计算，Fast-RCNN 正是为了解决这个问题诞生的。与 R-CNN 框架图对比，Fast R-CNN 主要有两处不同：一是最后一个卷积层后加了一个 ROI pooling layer，二是损失函数使用了多任务损失函数 (multi-task loss)，将边框回归 Bounding Box Regression 直接加入到 CNN 网络中训练。

Fast R-CNN 相对于 R-CNN 的提速原因就在于：不过不像 R-CNN 把每个候选区域给深度网络提特征，而是整张图提一次特征，再把候选框映射到 conv5 上，而 SPP 只需要计算一次特征，剩下的只需要在 conv5 层上操作就可以了。

Fast R-CNN 存在的问题：速度存在瓶颈：选择性搜索，找出所有的候选框，这个也非常耗时。

## 2.5 Faster R-CNN

rgbd 在 Fast R-CNN 中引入 Region Proposal Network(RPN) 替代 Selective Search，同时引入 anchor box 应对目标形状的变化问题（anchor 就是位置和大小固定的 box，可以理解成事先设置好的固定的 proposal）。Faster R-CNN 的简要步骤如下：

- 对整张图片输入进 CNN，得到 feature map
- 卷积特征输入到 RPN，得到候选框的特征信息
- 对候选框中提取出的特征，使用分类器判别是否属于一个特定类
- 对于属于某一类别的候选框，用回归器进一步调整其位置

Faster R-CNN 的主要贡献就是设计了提取候选区域的网络 RPN，代替了费时的选择性搜索 selective search，使得检测速度大幅提高。

## 2.6 Mask RCNN

Mask R-CNN 是 ICCV2017 的 best paper，在一个网络中同时做目标检测（object detection）和实例分割（instance segmentation）。Mask RCNN 沿用了 Faster R-CNN 的思想，特征提取采用 ResNet-FPN 的架构，另外多加了一个 Mask 预测分支。该算法在单 GPU 上的运行速度差不多是 5 fps，并且在 COCO 数据集的三个挑战赛：instance segmentation、bounding-box object detecton、person keypoint detection 中的效果都要优于现有的单模型算法（包括 COCO2016 比赛的冠军算法）。

## 2.7 Cascade RCNN

Cascade R-CNN 算法是 CVPR2018 的文章，通过级联几个检测网络达到不断优化预测结果的目的，与普通级联不同的是，cascade R-CNN 的几个检测网络是基于不同 IOU 阈值确定的正负样本上训练得到的，这是该算法的一大亮点。cascade R-CNN 的实验大部分是在 COCO 数据集做的，而且效果非常出色。

目前的目标检测算法大都使用  $\text{IoU} = 0.5$  的 IoU 阈值来定义正负样本，这是相当宽松的阈值，导致 detector 产生许多干扰的 bndbox。论文提出了 Cascade R-CNN 来解决上面的问题。Cascade R-CNN 是一个顺序的多阶段 extension，利用前一个阶段的输出进行下一阶段的训练，阶段越往后使用更高的 IoU 阈值，产生更高质量的 bndbox。Cascade R-CNN 简单而有效，能直接添加到其它 R-CNN 型 detector 中，带来巨大的性能提升(2-4%)

## 2.8 YOLO

Faster R-CNN 的方法目前是主流的目标检测方法，但是速度上并不能满足实时的要求。YOLO 一类的方法慢慢显现出其重要性，这类方法使用了回归的思想，利用整张图作为网络的输入，直接在图像的多个位置上回归出这个位置的目标边框，以及目标所属的类别。YOLO 大致流程：

- 给一个输入图像，首先将图像划分成  $7 \times 7$  的网格
- 对于每个网格，我们都预测 2 个边框（包括每个边框是目标的置信度以及每个边框区域在多个类别上的概率）

- 根据上一步可以预测出  $7 \times 7 \times 2$  个目标窗口，然后根据阈值去除可能性比较低的目标窗口，最后 NMS 去除冗余窗口即可

YOLO 将目标检测任务转换成一个回归问题，大大加快了检测的速度，使得 YOLO 可以每秒处理 45 张图像。而且由于每个网络预测目标窗口时使用的是全图信息，使得 false positive 比例大幅降低。

但是 YOLO 也存在问题：没有了 Region Proposal 机制，只使用  $7 \times 7$  的网格回归会使得目标不能非常精准的定位，这也导致了 YOLO 的检测精度并不是很高。

## 2.9 SSD

SSD 结合了 YOLO 中的回归思想和 Faster R-CNN 中的 anchor 机制，使用全图各个位置的多尺度区域特征进行回归，既保持了 YOLO 速度快的特性，也保证了窗口预测的跟 Faster R-CNN 一样比较精准。SSD 在 VOC2007 上 mAP 可以达到 72.1%，速度在 GPU 上达到 58 帧每秒。

## 2.10 PASCAL VOC

PASCAL VOC 挑战赛是视觉对象的分类识别和检测的一个基准测试，提供了检测算法和学习性能的标准图像注释数据集和标准的评估系统。

PASCAL VOC 为图像识别和分类提供了一整套标准化的优秀的数据集。数据集图片包括 20 个类别：

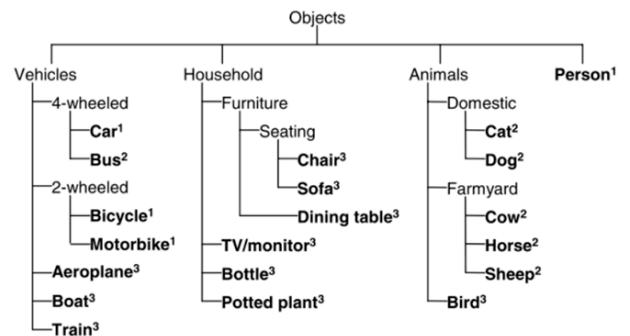


Figure 2: PASCAL VOC 数据集类别

从 2007 年开始，PASCAL VOC 每年的数据集都是这个层级结构，总共四个大类：vehicle, household, animal, person，20 个小类，预测的时候是只输出图中黑色粗体的类别。数据集主要关注分类和检测，也就是分类和检测用到的数据集相对规模较大。关于其他任务比如分割，动作识别等，其数据集一般是分类和检测数据集的子集。

Annotation 文件夹存放的是 xml 文件 (如 Figure 37)，该文件是对图片的解释，每张图片都对于一个同名的 xml 文件，包含 BBox、label 等信息。ImageSets 文件夹存放的是 txt 文件，这些 txt 将数据集的图片分成了各种集合。如 Main 下的 train.txt 中记录的是用于训练的图片集合。JPEGImages 文件夹存放的是数据集的原图片.jpg。PASCAL 的评估标准是 mAP(mean average precision)。

### 3 DATASETS

#### 3.1 VOC

针对 daily life 场景，我们从官网上下载了 VOC2012 数据集，该场景下的整个实践过程采用的都是 VOC 数据集。在 detectron2 框架上训练 Mask RCNN 和 Cascade RCNN 网络模型需要将 VOC 数据集的 xml 格式换成 COCO 数据集的 json 文件的格式（如 Figure 38）。而 YOLO v5 的数据集为另一种 txt 文件格式（如 Figure 39）。

**Mask RCNN** Mask RCNN 除了基本的目标检测，还引入了全新的实例分割。针对这一特色，我们打算利用 VOC 数据集来对我们 daily life 场景进行实例分割的尝试。因此除了从 VOC 数据集的 Annotation 中提取 bndbox 位置信息 xmin,xmax,ymin,ymax，还需要从数据集中的 2913 张 SegmentationObject 图像中获取其边界分割的信息（原始数据集针对分割图像并未给出 segmentation 的标注）。每一个 bndbox 中都有一个 object，且 SegmentationObject 图像中用不同的颜色填充了这些实例。因此可以先找到不同实例的标注颜色，再利用 opencv 中的 cv2.findContours 函数即可获得实例轮廓点的坐标。实验过程中遇到了一个问题：当某个实例的 segmentations 坐标只有 2 个时在 detectron2 框架下是不足以确定整个轮廓的，按照 github 上 detectron2 讨论区中相关问题的讨论，我在原有的 2 个坐标的基础上再加入了其中的 1 个坐标，3 个轮廓坐标输入后解决了该问题。

**Cascade RCNN** Cascade RCNN 整个网络基于 Mask RCNN，在 detectron2 框架上训练 Cascade RCNN 前的数据集处理与 Mask RCNN 类似。区别在于 Cascade RCNN 中，我们关闭其网络中的实例分割部分，仅保留其目标检测的部分。数据处理时将 Annotation 中提取到的 bndbox 位置信息 xmin,xmax,ymin,ymax 转换成 COCO 的对应格式 [x,y,w,h] 即可。

**YOLO v5** YOLO v5 的 txt 数据集格式包含 label 和 bndbox 中心坐标以及包围框的 w,h，按照上述 Cascade 的处理方法提取出 bndbox 的 [x,y,w,h] 便可容易得到前述这些信息，而由于 YOLO v5 训练时会将原始图像 resize 成固定大小，因此得到的中心坐标和 w,h 需要进行比例缩放，对应除以图像的 width 和 height。

#### 3.2 ASL

American Sign Language(ASL) 检测识别场景，起初我们的想法与 daily life 相同，尝试寻找美式手语相关数据集，但该数据集的相关资料很少，并未找到开源的数据集供我们下载训练。因此我们自制了一套 ASL 数据集。

**原图制作** 我们最初在图书馆研时间中拍摄了 75 张近距离男性手语图片，背景为朴素的地面和墙面，如 Figure 30，后续增加了 118 张远距离带人脸的男性手语图片，背景为朴素墙面，如 Figure 31。最后我们增加了女性手语和复杂背景的图片，并针对实际检测情形，对一种 label 的图像包含正反手、多角度、多距离等特征。这些图片包含 37 张女性远距离带人脸实验室背景手语图片（Figure 32），83 张女性近距离宿舍背景手语图片（Figure 33），192 张男性远距离带人脸宿舍背景手语图片（Figure 34）。所有原图共计 505 张原始图像。

**数据标注** 针对拍摄原图，我们采用了 labelImg 图像标注工具

为这些图像进行边界框标注，如 Figure 3。得到所有原图中手势标注信息 xml 文件，包含边界框的 xmin,xmax,ymin,ymax 标注信息。

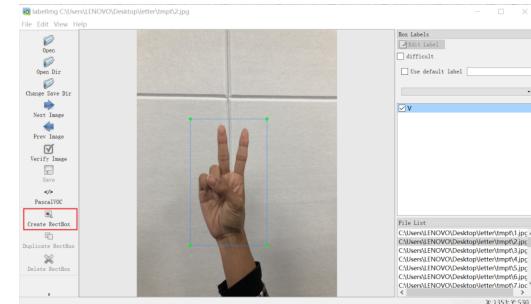
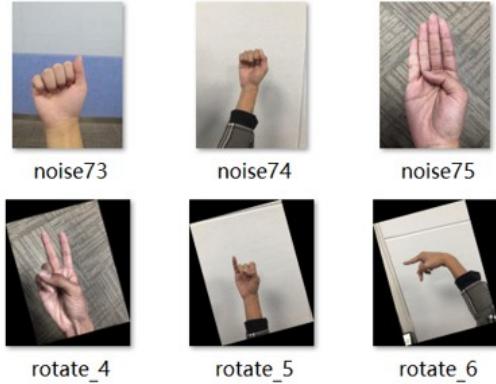


Figure 3: LabelImg image annotation

**多样性划分** 最初的实验中由于原始数据集样本的缺少，我们无法进行训练集和验证集的划分，所有样本同时作为训练集和验证集进行训练，最终的结果就是在训练样本上的检测效果很好，但对其他图像进行检测或者视频实时检测的准确率很低。当我们增加了原图数量后，具备了划分训练集和测试集的条件。505 张原始图像，我们将 400 张划分为训练集，剩下的 105 张划分为验证集。针对 105 验证集的图片，其中将近一半 50 张来自 192 张男性远距离带人脸宿舍背景手语图片（因为其背景复杂且带有人脸与现实情境更加符合），另外从 118 张男性远距离带人脸朴素墙面背景手语图片选取了 20 张，从 83 张女性近距离宿舍背景手语图片中也选取了 20 张，剩下的 15 张来自 37 张女性远距离带人脸实验室背景手语图片（由于这种类型的原图很少，将其中一大半作为验证集可以提高网络泛化能力）。其次，从每种类型原图中选取的验证集样本手语类别我们也进行了不同比例分配。对前面实验中发现的容易误检的类别（A,M,N,X,R,T），难以检测的类别（J,Z），其他手语类别，设置 3:2:1 的被选取概率。未被验证集选中的剩余 400 张图片自然划分为训练集。

**数据增强** 由于我们最终的原图数量仅有 505 张图像，只利用这些图像对网络进行训练，由于数据集过小，训练出来的网络很可能产生过拟合。因此我们采用了数据增强扩充数据集。起初我们自己实现了基本的小角度旋转、增加噪声、调整亮度和水平翻转。对最开始的 75 张图像进行了单一数据增强，每张增强图像仅仅添加一种上述增强策略，增强结果获得 300 张增强图像 +75 张原始图像，增强结果如 Figure 4。由于这些数据训练得到的网络检测效果很差，这其中不仅有原始数据的缺陷，还包括增强的不足。随后我们利用 imgaug 对图像进行了更多的增强，每张图片同时随机地进行多类增强，包括加高斯噪声、调亮度、左右旋转 15°、水平镜像、转灰度图、锐化、转换颜色空间通道增强（RGB 转 HSV）、擦除矩形框等，增强结果如 Figure 5。其中当进行旋转、镜像时，提前标注的边界框数据需要进行同步的更新。水平翻转的标注更新很容易，不涉及过多复杂变换。进行旋转增强时，利用 cv2.getRotationMatrix2D 函数根据输入的旋转角度获得绕图像中心旋转的仿射变换矩阵，更新图像的 width 和 height，并计算新的“边界框”坐标将其划入新的矩形框作为旋转后新的边界框。经过数据增强后，我们的训练集从 400 张原图，增强到了 2800 张增强图片 +400 张

原始图片，共 3200 张训练图像，如 **Figure 35**；验证集从 105 张原图，增强到 525 张增强图片 +105 张原始图片，共 630 张验证图像，如 **Figure 36**。其中验证集图片的变化(增强)程度要略低于训练集图片，为了尽可能多地保留验证集图像的准确性。



**Figure 4: Single augmentation of 75 images**



**Figure 5: Mixed augmentation of 505 images**

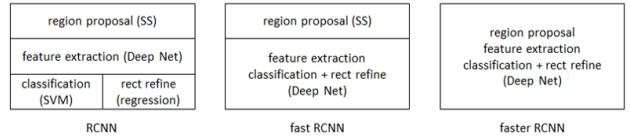
**减少误检** 针对最后的 505 张图像，我们根据前面的实验结果，增加了易误检类别样本数目(X、M、N、A、S、T)，这些手势看起来很相近，一些仅仅是一个拇指位置或者视角的区别，为了增加网络对这些手势的识别能力，我们在数据集中相对增加了这些手势不同距离的图像，让网络能够更加清楚地“认识”它们。

## 4 APPROACHES

### 4.1 Daily Life

Daily Life 场景基于 PASCAL VOC2012 数据集，包含五个算法——Faster RCNN、Mask RCNN、Cascade RCNN、SSD、YOLOv5，实现了图片的目标检测与搜索。

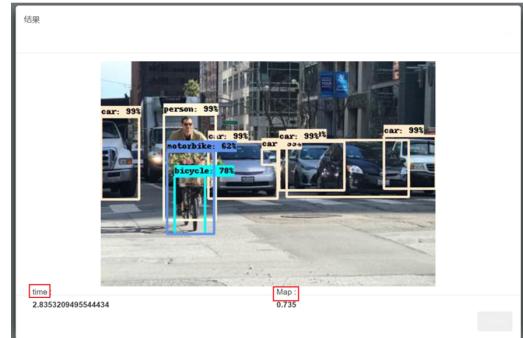
**Faster RCNN** RCNN 系列作为经典目标检测算法，将深度学习方法引入了目标检测领域，并大大提升了目标检测的准确率和速度，具有重要的历史地位。因此，我们入手目标检测，率先研读了 3 篇 RCNN 系列的经典论文，从 RCNN，到 Fast RCNN，再到 Faster RCNN，算法结构得到了逐步的优化，如 **Figure 6** 所示，同时准确率和速度都有很大的优化。



**Figure 6: RCNN 系列三大经典算法的比较**

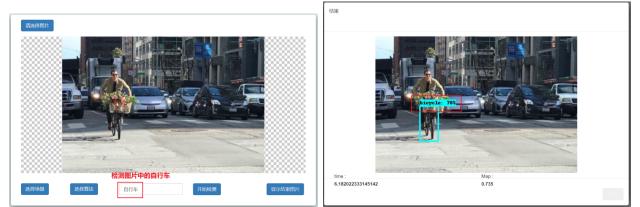
RCNN 和 Fast RCNN 在实际使用中训练分多阶段，比如 RCNN 的训练分四阶段，首先需要对 CNN 预训练，然后微调 CNN，再分别训练分类器 SVMs 和回归器 Bounding-box regressors，每个阶段之间需要存储中间结果，时间和空间上都很昂贵，笔记本电脑难以支撑其训练。Faster RCNN 相较于前两种，速度和准确率都更好，而且训练更方便，所以我们参考 Pytorch 官方代码实现了 Faster RCNN，并基于 Faster RCNN 做了一个简单的前端，实现了基于 VOC 数据集的图片目标检测与搜索，可以检测日常生活中的 20 个类别。

图片目标检测：输入要检测的图片，输出结果图片以及评价指标 mAP、Speed



**Figure 7: Faster RCNN: 图片目标检测**

图片目标搜索：输入要检测的图片和搜索对象，输出结果图片以及评价指标 mAP、Speed



**Figure 8: Faster RCNN: 图片目标搜索**

**Detectron2** 基于 PyTorch, PyTorch 可以提供更直观的命令式编程模型高度模块化；功能强大，包括如全景分割、Densepose、Mask R-CNN、Cascade R-CNN、旋转边界框、PointRend、DeepLab 等多种功能；高度模块化、可扩展，允许用户将自定义模块插入目标检测系统的几乎任何部分，我们可以将新的研究项目和核心 Detectron2 库完全分开，使用起来更加灵活；训练速度相较其他许多平台更快；安装方便快捷。

## Main Results

Implementation	Throughput (img/s)
Detectron2	62
mmdetection	53
maskrcnn-benchmark	53
tensorpack	50
simpledet	39
Detectron	19
matterport/Mask_RCNN	14

Figure 9: Main results of detectron2

**Mask RCNN** 基于 Detectron2 的 mask\_rcnn\_R\_50\_FPN\_1x 和 3x 预训练模型训练了上述已准备好的训练 Mask RCNN 的所需数据集，训练过程首先要注册数据集，并定义类别字典编号。后续就是初始化训练超参，我设定的 batchsize 为 4，学习率 lr 为 0.00025，迭代次数为 7000 次。

实验结果发现无论是在 mask\_rcnn\_R\_50\_FPN\_1x 和还是在 3x 预训练模型下，在迭代了 3000 次后其 total\_loss 的振荡幅度变得很大，且此振荡一直持续到迭代 7000 次训练结束。考虑振荡的结果，中间多迭代的几千次 total\_loss 并无明显下降。后续我们尝试将学习率 lr 调小至 1e-5，重新训练迭代 7000 次，但训练效果类似，3000 次迭代以后，total\_loss 未明显下降，且该学习率下的 total\_loss 远高于 0.00025 学习率下最终的 total\_loss。其他调参尝试的结果与上述类似，振荡问题始终存在，因此后续针对 Detectron2 的 mask\_rcnn 训练迭代次数从初始的 7000 次减少到了 3000 次。不同预训练模型和学习率下的训练指标如 Table 1；Figure 10 为 mask\_rcnn\_R\_50\_FPN\_1x，学习率为 0.00025 下 Mask RCNN 检测结果图片。

Table 1: Mask RCNN training result index

pretrained model	lr	bbox mAP	segm mAP
R_50_FPN_1x	0.00025	54.473%	23.737%
R_50_FPN_3x	0.00025	54.757%	23.354%
R_50_FPN_1x	1e-5	52.758%	21.742%
R_50_FPN_3x	1e-5	51.478%	21.836%

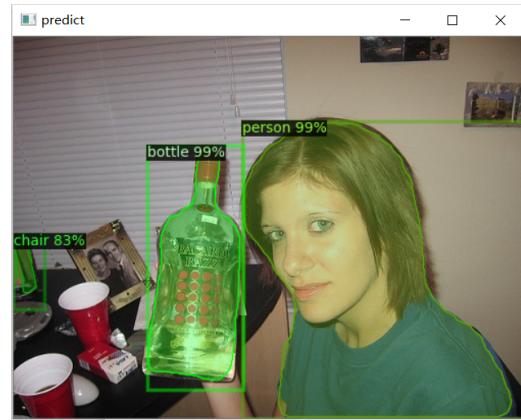


Figure 10: Ordinary image detection of Mask RCNN

**Cascade RCNN** 算法级联三个 stage 的 Mask RCNN，如 Figure 11 所示，其核心思想是使用不同的 IOU 阈值划分正负样本，让每一个 stage 的 detector 都专注于检测 IOU 在某一范围内的 proposal，因为输出 IOU 普遍大于输入 IOU，因此检测效果会越来越好。

Cascade RCNN 的实现同样基于 Facebook 的 Detectron2 框架，实现步骤与 Mask RCNN 类似，唯一不同的是关闭了实例分割，使 Cascade RCNN 更专注于目标检测，具体步骤：

- 准备数据集：将 VOC 数据集转换至 COCO 格式
- 注册数据集
- 调整网络结构：修改三阶段模型的 ROI Head 类别数；关闭了实例分割，专注于目标检测
- 训练模型与调参：调用 cascade\_mask\_rcnn\_R\_50\_FPN\_1x/3x 预训练模型训练自己的 VOC 数据集
- 使用模型预测
- 测试集评估模型

Experiments 中验证了 Cascade RCNN 效果比 Mask RCNN 更好。

RCNN 系列属于双阶段目标检测，我们应用了 Faster RCNN、Mask RCNN、Cascade RCNN，其中 Cascade RCNN 效果最优。除了图片，我们也尝试使用这些算法进行实时检测，在测试中发现他们的速度无法满足我们的要求，可能会限制之后实时检测应用（ASL）的表现。因此，我们研究了单阶段目标检测，以 SSD、YOLOv5 为主，训练 VOC 数据集。

**SSD** 虽然 SSD 这个算法出来已经两年了，但是至今依旧是目标检测中应用最广泛的算法，后面有很多基于 SSD 改进的算法，但依旧没有哪一种可以完全超越 SSD。

SSD 效果好主要有三点原因：

- 多尺度：使用 6 个不同特征图检测不同尺度的目标。低层预测小目标，高层预测大目标。
- 设置了多种宽高比的 anchor：通过 anchor 设置每一层实际响应的区域，使得某一层对特定大小的目标响应。这样检测层提取的就是 anchor 对应区域的特征了。通过 anchor 我们就可以知道这个特征对应了原图什么区域？这个区域的 label 是什么？
- 数据增强：放大与缩小

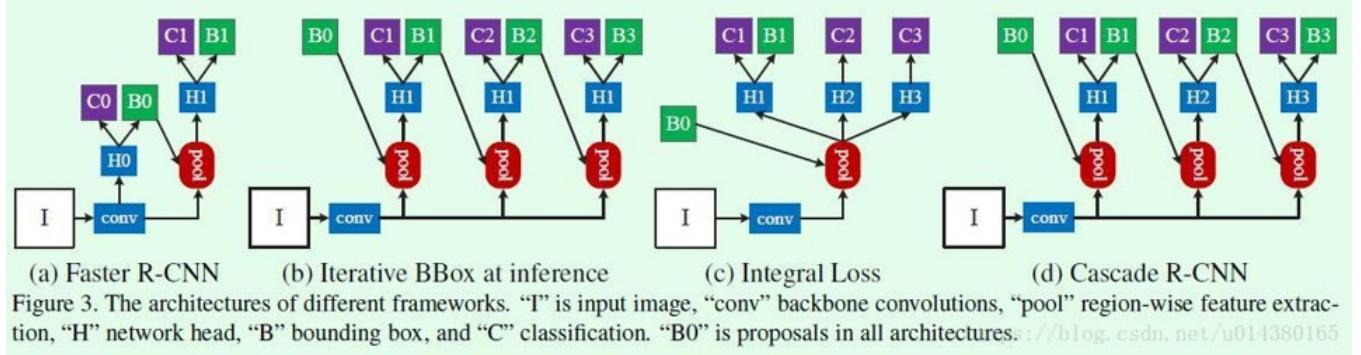


Figure 11: Cascade RCNN

在 Daily Life 的实验中，我们参考 Pytorch 官方代码实现了 SSD。SSD 在图片检测远处小物体时表现并不好，但在 ASL Detection 中，拥有与 YOLOv5 媲美的性能。

YOLOv5 作为 YOLO 系列的最新作品，于 2020 年 6 月提出，是完全基于 Pytorch 实现的。我们选择它也是看中它的新颖与轻便性。YOLOv5 的大小仅有 27MB，而使用 DarkNet 架构的 YOLOv4 有 244MB，对比之下小了近 90%，同时在准确度方面又与 YOLOv4 基准相当，速度快了近 3 倍。

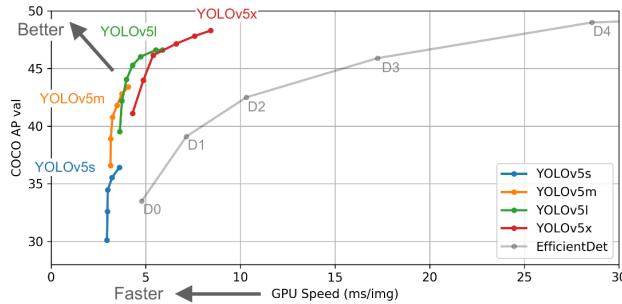


Figure 12: YOLOv5: 性能对比

我们应用 YOLOv5 官方代码，尝试了混合精度训练，将 VOC 数据集转换成 COCO 格式后训练模型。

## 4.2 American Sign Language

入手目标检测，我们使用著名的 VOC 数据集训练了 5 个算法——Faster RCNN、Mask RCNN、Cascade RCNN、SSD、YOLOv5，对不同模型进行初步探索与进一步研究，实现了 Daily Life 场景。如果说 Daily Life 是小试牛刀，那么 American Sign Language 则是我们的重头戏。基于 Daily Life 的经验，我们从 5 个模型中挑出 YOLOv5、SSD、Cascade RCNN 为我们的美式手语检测服务，其中着重实现了 YOLOv5 并在训练上做了一些优化，而 Cascade RCNN 作为 RCNN 系列（双阶段算法）的代表与之比较，SSD 作为其他单阶段的算法与之比较。而数据集方面，我们自制了美式手语数据集，类别包含 26 个大写字母，经过多阶段的处理，最终数据集包含训练集 3200 张图片，验证集 630 张图片（在 Datasets 有具体讲述）。

由于在 Daily Life 已经介绍过这三个算法，所以这里只提及不同之处。

**YOLOv5** 选择 YOLOv5 主要是看中他的新颖、检测速度快、模型轻便，同时又有不错的准确率。一开始我们直接使用原始的 YOLOv5 训练 ASL 数据集，经过 ASL 数据集的不断扩充，提高了模型的泛化能力，达到了不错的实时检测效果。但 YOLOv5 模型结构用于小小的手语检测，可能有些大材小用了，所以我们优化了训练过程，简化训练出的模型结构。新的训练过程是，首先正常训练，达到理想精度后进行稀疏训练，稀疏训练是重中之重，对需要剪枝的层对应的 bn gamma 系数进行大幅压缩，理想的压缩情况如 Figure 13，然后就可以对不重要的通道或者层进行剪枝，剪枝后可以对模型进行微调恢复精度。

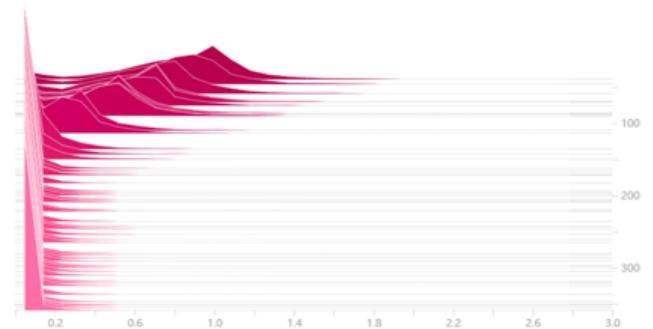


Figure 13: 稀疏训练理想的压缩情况

稀疏训练就是精度和稀疏度的博弈过程，如何寻找好的策略让稀疏后的模型保持高精度同时实现高稀疏度是值得研究的问题。scale 参数控制稀疏度的大小，大的 scale 一般稀疏较快但精度掉的快，小的 scale 一般稀疏较慢但精度掉的慢；配合大学习率会稀疏加快，后期小学习率有助于精度回升。根据 scale 的取值存在不同的策略，在我们实际的优化中，我们采用的策略是恒定的 scale，也是默认的策略。在整个稀疏过程中，始终以恒定的 scale 给模型添加额外的梯度，因为力度比较均匀，往往压缩度较高。但稀疏过程是个博弈过程，我们不仅想要较

高的压缩度，也想要在学习率下降后恢复足够的精度，不同的 scale 最后稀疏结果也不同，想要找到合适的 scale 往往需要较高的时间成本，经过几次试验后，我们确定下来 scale=0.001。

选择的剪枝策略是通道剪枝中的一种保守策略，因为 YOLOv5 中的 shortcut 连接对应的是 add 操作，通道剪枝后如何保证 shortcut 的两个输入维度一致，这是必须考虑的问题。如果对 shortcut 直连的层不进行剪枝，可以避免了维度处理问题，但同样实现了较高剪枝率，对模型参数的减小有很大帮助。虽然它剪枝率最低，但是它对剪枝各细节的处理非常优雅。经过稀疏训练后剪枝，可以大大缩小模型的大小，以 yolov5s 模型为例，试验中模型大小可以从原来 14.5MB 缩小至 8.3MB，可以极大缩短实时检测的耗时。但剪枝通常会引起精度的下降，所以我们需要有微调训练这一步来恢复精度，在提升检测速度的同时保持原有的精度。

**Cascade RCNN** 除了 YOLOv5，我们还使用 ASL 数据集训练了 Cascade RCNN，作为 RCNN 系列的代表，比较双阶段与单阶段的算法，应用 Cascade RCNN 于 ASL Detection 的过程与 Daily Life 的过程差异不大，依然基于 Detectron2 框架，只需要转换数据集格式即 VOC2COCO，重新注册数据集，调整一下模型结构的类别数即可。Cascade RCNN 的速度劣势过于明显，即使理论上有更好的准确率，但在手语实时检测的整体效果依旧不如 YOLOv5。

**SSD** 同样，我们还使用 VOC 格式的 ASL 数据集训练了 SSD，比较单阶段的两个算法。SSD 的应用与 Daily Life 中的应用过程基本相同，将 SSD 应用到手语的实时检测中，SSD 的速度优势被放大，体现了可以与 YOLOv5 媲美的检测效果。

## 5 EXPERIMENTS

### 5.1 Daily Life

针对该场景，我们对不同网络下小物体的目标检测效果进行了一系列的对比。

**Faster RCNN** 最初实现的 Faster RCNN 其对一般较大的物体检测效果还是很好的，也正如其 Faster 的名字一样，检测速度很快，一张图片的检测时间大约在两秒。此外，我们还实现了 Faster RCNN 下的图片目标搜索，当你在输入框中输入某种物体的种类名称，检测结果只会返回你输入的该种类目标，其他目标不会显示，能够满足在图片中搜索目标的需求。

**Faster RCNN vs. Mask RCNN** 对比来看，Faster RCNN 对小物体的检测效果很差，对一排图像中距离较远并排的自行车，其只能检测出少辆特征极为明显的小物体，存在大量漏检的情况，该实验结果也印证了理论下 Faster RCNN 对密集型小物体的检测效果不好的结论。而 Mask RCNN 的检测结果相比 Faster RCNN 略好，多检测出来了少量小物体，但效果仍然不佳，由于 Mask RCNN 更为复杂的网络与实例分割的过程，其检测速度要稍慢于 Faster RCNN。



Figure 14: Faster RCNN(left) vs Mask RCNN(right)

**Mask RCNN vs. Cascade RCNN** Cascade RCNN 的检测结果出人意料，其检测出了许多 Mask RCNN 未检测出来的小物体，检测结果的准确率也很高，误检目标很少，同时 Cascade 的检测速度比 Mask RCNN 快很多。导致这种结果是由于 Cascade RCNN 级联了三个 Mask RCNN，目标检测效果更好，而关闭了 Cascade 的网络结构中的实例分割部分，使得检测步骤更加简单，速度更快。



Figure 15: Mask RCNN(left) vs Cascade (right)

**SSD vs. Faster RCNN** 前面所述的网络都属于双阶段目标检测 RCNN 系列，该系列网络有一个最大的缺点就是检测速度慢，无法满足实时检测的需求。这里我们利用训练好的单阶段目标检测 SSD 网络与基础的 Faster RCNN 进行对比。SSD 检测一张图片仅需零点几秒，是 RCNN 系列网络的几十倍，但快速检测带来的后果就是误检率和漏检率都极高，检测的准确率还不如 RCNN 系列效果最差的 Faster RCNN。



Figure 16: SSD(left) vs Faster RCNN(right)

**YOLOv5 vs. Cascade RCNN** 我们实验中的另一个单阶段目标检测网络是 YOLOv5，从对比结果我们发现，YOLOv5 较 RCNN 系列表现最好的 Cascade RCNN，其对密集小物体的召回率更高，速度也更快，但检测准确率不高，存在大量的误检目

标。

在实时检测中，YOLOv5 的速度远快于 Cascade RCNN，准确率较 Cascade RCNN 要高一些。但是在一般图片的目标检测中，Cascade RCNN 的准确率还是更优，我们猜测这可能是由于 RCNN 系列网络的速度慢限制了它在实时检测中的表现。

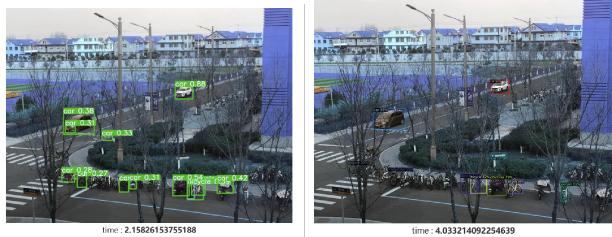


Figure 17: YOLOv5(left) vs Cascade RCNN(right)

## 5.2 American Sign Language

ASL Detection 的消融实验主要分为三大部分：数据集增强、YOLOv5 的优化、三大算法的优劣。数据集增强，主要对比经过不同的数据增强程度训练后模型的检测效果，通过扩充数据集与划分训练集、验证集，给模型带来的泛化能力的提升。YOLOv5 的优化，主要对比稀疏训练并剪枝与未剪枝的模型检测效果，这里保证使用的数据集都是最终版（即其他条件都相同）。三大算法的优劣，三个算法是 YOLOv5、Cascade RCNN、SSD，主要对比 YOLOv5 与 Cascade RCNN、YOLOv5 与 SSD。由于手语是实时检测，在报告中，用截图代替对比效果的视频，具体视频可以在答辩 PPT 中找到。

**75 张原图（未增强）** 我们最初的数据集为 75 张原图，这些图片的拍摄地点为图书馆的研讨间，且均为近距离对手拍摄，我们将这些图片送入 YOLOv5 进行训练，训练后效果很差。图片检测误检漏检现象严重，仅仅当测试图片的拍摄背景与训练样本类似为朴素墙面或者地面，且拍摄的手势清晰特征非常明显容易分辨时，才能正确检测出部分图片手势；实时视频检测几乎不能准确检测定位，检测时将许多颜色相近的物体识别为某种手势，而对正确的手势视而不见的现象层出不穷。

**75 张原图 +300 张增强图片** 对这 75 张原图单独进行加噪声、调亮度、旋转、水平镜像四种增强操作，每张图片可产生 4 张增强后的图像。将这 75 张原图 +300 张增强图片送入 YOLOv5 进行训练，由于训练集的增多，训练结果略有提升。对于较为简单特征明显的手势，基本可以识别，前提仍然是测试图像的背景与训练集图像相近，而对于容易混淆，特征不够明显的几种手势，其检测结果仍不尽人意，出错漏检的很多。而对于有人脸存在实时视频检测，效果并无明显提升。检测对比示例如 Figure 18。

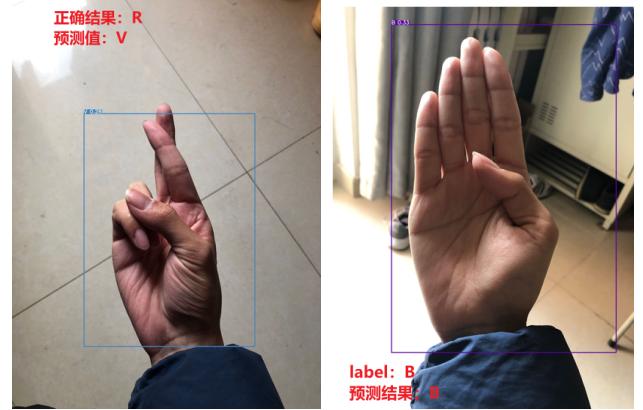


Figure 18: 75 original(left) vs 375 augmented(right)

**193 张原图 +965 张增强图片** 我们分析，原来那 75 张图，训练图片里只有手势，而我们实时检测时通常会包含我们的人脸，也不会像那 75 张原图一样离相机距离很近。所以我们对原数据集进行带人脸的训练集扩充，增加原图至 193 张，新增图像的拍摄地点仍为研讨间，背景还是朴素的墙面，再对每张原图进行单一的加噪声/调亮度/左旋转/水平镜像/右旋转，将原来 193 张原图增强到了总共 1158 张，重新训练 yolov5。利用训练结果，进行实时视频手语检测发现当我们在研讨间或其他简单朴素背景下测试时，多种手势检测效果较好。但当我们把测试地点转移到其他复杂背景下测试时效果变得较差，对稍微复杂的手势无法检测。

**3200 张训练集 +630 张测试集** 从上面的实验结果我们知道，在训练集的图像背景单一简单的情况下，测试时复杂的背景会对我们的检测造成很大干扰，因此当上面训练的网络进行手语检测时，仅在研讨间或简单背景下实时检测效果较好。因此，我们进一步扩充数据集，增加了更多复杂背景，不同性别的手势样本。同时考虑实际情况，对同一手势增加多角度多距离下的训练样本。并着重拍摄多组当前发现的几个误检频繁类别的手势，比如 X、M、N、A、S、T，共计 505 张原始图片。并根据数据集样本的上述特性，设计一定的划分比和分配比多样性分配到训练集和验证集。最终得到训练集 400 张和验证集 105 张原图。同时我们反思之前的增强过程并未将多种增强方法同时运用到一张图片，每次生成的增强图片仅包含一种单一增强。我们采用更多的增强方法，对每张图片随机进行多种类别增强，随机加噪声、调亮度、旋转、水平镜像、转灰度图、锐化、转换颜色空间、擦除矩形框。对训练集和验证集单纯进行增强，最后得到训练集 3200 张图片，验证集 630 张图片，重新训练 yolov5。实验发现，训练结果泛化能力加强，可以很好地应对不同背景下的实时手语检测，而缺陷在于对 M、N、A、T 这种区分度较低容易产生视觉混淆的类别偶尔产生误检。检测对比示例如 Figure 19。



Figure 19: 1158 augmented(left) vs 3200 augmented(right)

**Prune vs. No Prune** 本部分比较 YOLOv5 经过稀疏训练剪枝后与未剪枝的模型效果，使用的数据都是训练集 3200 张图片与验证集 630 张图片的 ASL 数据集，使用的原模型都是 yolov5s，稀疏训练的稀疏度 scale 等于 0.001，通道剪枝的策略是不对 shortcut 直连的层进行剪枝，避免维度处理。batch size=8、epochs=300 下得到稀疏训练并剪枝的结果，评测如 Figure 20。

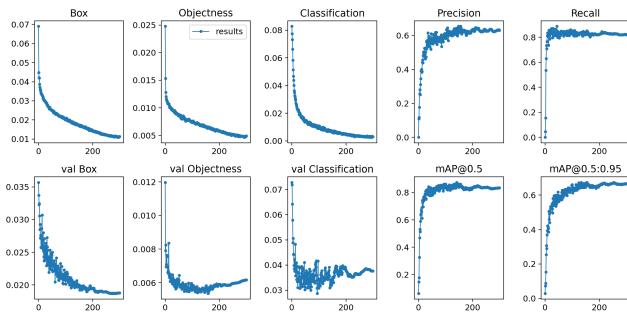


Figure 20: Evaluation of Pruned yolov5s

观察 Figure 20，模型收敛效果不错，验证集上 mAP@0.5 接近于 0.8，mAP@0.5:0.95 在 0.6 左右。mAP@0.5:0.95 指的是在不同 IoU 阈值（从 0.5 到 0.95，步长 0.05）（0.5、0.55、0.6、0.65、0.7、0.75、0.8、0.85、0.9、0.95）上的平均 mAP。这与同等条件下未剪枝的 YOLOv5s 的精度相当，两者在一些困难类别比如 MNT 等的检测上都比较普通，经常出现双边界框的情况。而在速度上未剪枝的模型处理一帧要花费 0.03s 左右，剪枝后的模型处理一帧要花费 0.02s 左右，明显速度上得到了提升。因此我们的稀疏训练及剪枝策略一定程度上提升了实时检测的速度，并保持住了精度，是有效果的。Figure 21 截取了剪枝后与未剪枝的 YOLOv5 对字母 X 的检测效果。



Figure 21: YOLOv5 with Pruned vs. YOLOv5 with No Pruned

**Cascade RCNN vs. YOLOv5** 本部分比较了包含训练集 3200 张图片、验证集 630 张图片的 ASL 数据集在 Cascade RCNN 与 YOLOv5 上的效果，YOLOv5 使用的是未剪枝的 yolov5s。Cascade RCNN 使用的预训练模型是 cascade\_mask\_rcnn\_R\_50\_FPN\_3x，学习率是 1e-4，最大迭代数是 12000 和 15000 次，训练 ASL 数据集，在 Detectron2 上评测，得到 Figure 22。

[[2020-01-01 00:00:00]] evaluation.evaluation.Evaluation results for class:					
	AP	AP50	AP75	AP95	APs
[[2020-01-01 00:00:00]] evaluation.evaluation.Per-category class AP:					
categories	AP	AP50	AP75	AP95	APs
S	0.643	0.643	0.643	0.643	0.643
I	0.363	0.363	0.363	0.363	0.363
R	0.363	0.363	0.363	0.363	0.363
O	0.363	0.363	0.363	0.363	0.363
M	0.363	0.363	0.363	0.363	0.363
N	0.363	0.363	0.363	0.363	0.363
T	0.363	0.363	0.363	0.363	0.363
P	0.363	0.363	0.363	0.363	0.363
L	0.363	0.363	0.363	0.363	0.363
D	0.363	0.363	0.363	0.363	0.363
[[2020-01-01 00:00:00]] evaluation.evaluation.Evaluation results for class:					
[[2020-01-01 00:00:00]] evaluation.evaluation.Per-category class AP:					
categories	AP	AP50	AP75	AP95	APs
S	0.643	0.643	0.643	0.643	0.643
I	0.363	0.363	0.363	0.363	0.363
R	0.363	0.363	0.363	0.363	0.363
O	0.363	0.363	0.363	0.363	0.363
M	0.363	0.363	0.363	0.363	0.363
N	0.363	0.363	0.363	0.363	0.363
T	0.363	0.363	0.363	0.363	0.363
P	0.363	0.363	0.363	0.363	0.363
L	0.363	0.363	0.363	0.363	0.363
D	0.363	0.363	0.363	0.363	0.363

Figure 22: 分别是最大迭代 12000 次和 15000 次的结果

比较上下两图，发现迭代 15000 次的 mAP 反而比 12000 次更低，说明可能是过拟合了，所以我们使用 12000 次迭代的 Cascade RCNN 与 YOLOv5 进行对比，而观察每一类的 AP，发现有几个的 AP 和其他相差很多、很低，比如 S、I，可能是数据集在这两个字母做的还不够全面。

将迭代 12000 次的 Cascade RCNN 和 YOLOv5 作比较，如图截取了对比视频



Figure 23: 分别是 Cascade RCNN 和 YOLOv5 的结果

从图上可以看出，Cascade RCNN 的 FPS 仅 2.68 帧/秒，实时性实在太差，使得它的准确率的优势在实时检测中几乎被抹灭，相比之下 YOLOv5 则很快，FPS 在 50 帧/秒以上，完全符合实时性要求。

**SSD vs. YOLOv5** 本部分比较了包含训练集 3200 张图片、验证集 630 张图片的 ASL 数据集在 Cascade RCNN 与 YOLOv5 上的效果，YOLOv5 使用的是未剪枝的 yolov5s。SSD 使用与 Daily Life 的相同的方式训练 ASL 数据集，迭代 15 个 epoch，得到结果如 Figure 24。

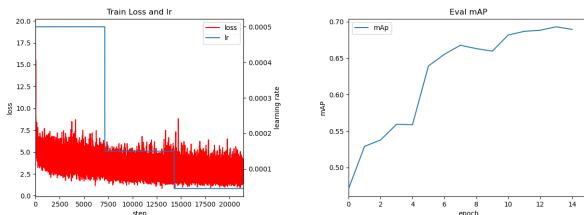


Figure 24: 分别是 SSD 迭代 15 个 epoch 的 Loss and LR 和 mAP

从上图可以看出经过 15 个 epoch 的 SSD，最佳的 mAP 可以达到 0.69 左右，比 YOLOv5 的 mAP@0.5:0.95 高一些，而在速度上和 YOLOv5 相当，FPS 在 50 帧/秒以上，如下图所示。



Figure 25: 分别是 SSD 和 YOLOv5 的结果

可见 SSD 的实时手语检测效果出乎意料地好，体现出与 YOLOv5 旗鼓相当的表现，SSD 果然是经典又有效的单阶段算法。

## 6 WEBPAGE

### 6.1 Back-end

使用 Python 语言编写，主要进行参数的传递以及实现网页之间的跳转。

这里我们选择使用 python-flask 框架。

Flask 是一个基于 Python 开发并且依赖 jinja2 模板和 Werkzeug WSGI 服务的一个微型框架，对于 Werkzeug 本质是 Socket 服务端，其用于接收 http 请求并对请求进行预处理，然后触发 Flask 框架，开发人员基于 Flask 框架提供的功能对请求进行相应的处理，并返回给用户，如果要返回给用户复杂的内容时，需要借助 jinja2 模板来实现对模板的处理，即：将模板和数据进行渲染，将渲染后的字符串返回给用户浏览器。

Flask 较其他同类型框架更为灵活、轻便、安全且容易上手。它主要面向需求简单，项目周期短的小应用，小型团队在短时间内就可以完成功能丰富的中小型网站或 Web 服务的实现，符合我们对于本课题网页的需求。同时，Flask 在网页安全、高效、稳定等方面做的都有很出色的表现。

### 6.2 Front-end

使用 html 语言编写，css 作为装饰器.js 作为功能补充。

为使得程序能够更快地回应用户的操作，网页应用能够快速地将增量更新呈现在用户界面上，而不需要重载（刷新）整个页面。我们这里使用了 ajax 方法传递前后端参数。Ajax 即“Asynchronous Javascript And XML”（异步 JavaScript 和 XML），是指一种创建交互式、快速动态网页应用的网页开发技术，无需重新加载整个网页的情况下，能够更新部分网页的技术。

通过在后台与服务器进行少量数据交换，Ajax 可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。这不仅使我们的因特网应用程序更加高效，同时也使程序更小、更快，更友好。

### 6.3 Visualization

网页主要有两部分内容：

**Daily Life:** 该部分主要是针对五个算法（Faster RCNN、Mask RCNN、Cascade RCNN、SSD、YOLOv5）实现的对普通物体的检测，对象是日常照片。

1、普通物体检测：上传一张图片，选择要使用的算法，点击开始检测，会将图片中的所有物体检测出来，并标出判断概率。

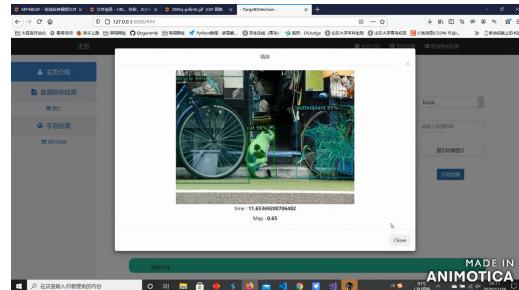


Figure 26: 普通物体检测 Mask 算法

2、特定物体检测：同样上传一张图片，选择要使用的算法，在输入框中输入需要检测出的物体，点击开始检测，得到的输出图片只会检测出我们输入的物体，其他物体不会被框出。

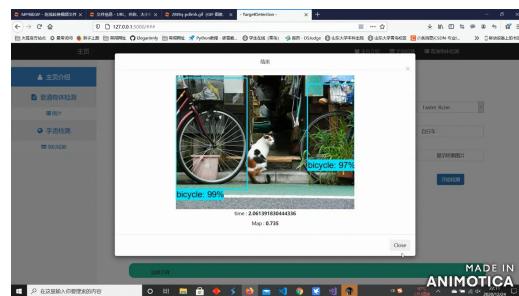


Figure 27: 检测自行车结果图 FasterRCNN 算法

**American Sign Language:** 该部分我们实现实时手语检测，目的是使得那些没有学过手语的人可以无障碍与聋哑人交流。该部分检测对象是美式手语。

根据普通物体检测结果，对不同算法检测的时间、准确性、稳定性等方面分析与对比，我们选出 Cascade、SSD、yolo v5 三个算法进行下面的手语检测。

步骤：选择手语检测模型，点击打开摄像头，启动模型，开始进行实时对美式手语的检测。

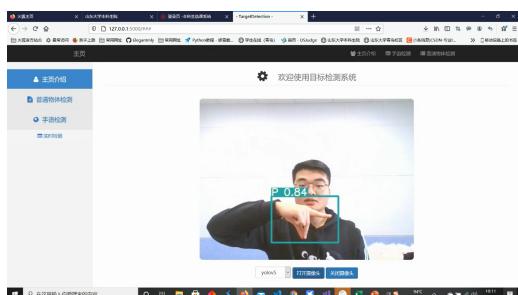


Figure 28: YOLOv5 手语检测结果

对比上述三个算法在手语检测上的效果，Cascade 在实时性和准确性等方面都要明显比 yolo v5、SSD RCNN 差一些，SSD RCNN 在稳定性方面比 Yolo v5 要稍差一些。

## 7 CONCLUSION

我们项目的主题是多场景目标检测，分为两大场景：日常生活 Daily Life 和美式手语 American Sign Language。但由于美式手语检测是我们工作的重点，所以给项目取名 ASL Detection。

日常场景检测任务中使用了 VOC 数据集，它包含了我们日常生活中常见的 20 类。我们通过这个任务入手目标检测，先读 R-CNN 系列的经典论文，从 R-CNN 到 Fast R-CNN 再到 Faster R-CNN。其中 Faster R-CNN 相较于前两种，速度和准确率都更好，而且训练更方便，所以我们实现了 Faster R-CNN，并做了一个简单的前端实现了基于 VOC 数据集的图片目标检测与搜索。

Faster R-CNN 对小物体的检测漏检误检严重，我们研究了 R-CNN 系列更新的成果，包括 Mask R-CNN 和 Cascade R-CNN。我们基于 Facebook 的 Detectron2 这个框架训练了 Mask R-CNN，R-CNN 系列属于双阶段目标检测，在测试中发现他们可能无法满足我们之后实时检测的需求。因此，我们研究单阶段目标检测，以 yolov5, SSD 为主，用 SSD 检测同一张图片，虽然速度很快，但效果都不及 R-CNN 系列的 Faster。而 YOLOv5 较 Cascade 对密集的小物体召回率更高，速度更快，但准确率不高，存在大量误检。基于 VOC 数据集的日常生活图片目标检测与搜索实现了 5 个算法，可以检测 20 个 VOC 的类别。

基于之前的经验，我们从 5 个模型中挑出 Cascade、YOLO、SSD 为我们的手语目标检测服务，着重实现了 YOLOv5，Cascade 作为 R-CNN 系列代表与之比较，SSD 作为其他单阶段的算法与之比较。

美式手语类别分 26 个大写字母，它区别于 VOC，属于小物体目标检测，不同类别之间的区别更小，比如这里的 M 和 N，更容易受背景干扰，检测难度更大。同时存在像 J 和 Z 这样的动态手语。我们着重研究了 YOLOv5，对这 26 个类别，自己制作了数据集，同时我们将多种增强方法同时运用到一张图片，采用更多的增强方法，最后得到训练集 3200 张图片，验证集

630 张。重新训练 yolov5，泛化能力加强，可以很好地应对不同背景。

除了 YOLO，我们还应用了 Cascade，比较双阶段与单阶段算法，看出 Cascade 很慢，准确率的优势也不明显。此外我们还应用了 SSD，比较单阶段的两个算法，SSD 的效果也非常好，大致不相上下。

综上，本次课题中，我们完成了既定的多场景目标检测的任务。通过对不同模型的一步步探索学习，我们逐步优化日常场景的目标检测效果，了解了每个模型的性能差异。基于此，我们的美式手语检测场景选取了适合于该场景的三个模型，并通过一步步对数据集进行扩充和增强优化，最终实现了美式手语的实时检测。基于这两大场景，我们设计了完整的前端网页以及后端接口，可视化结果，方便用户使用。

项目整体框架如下：

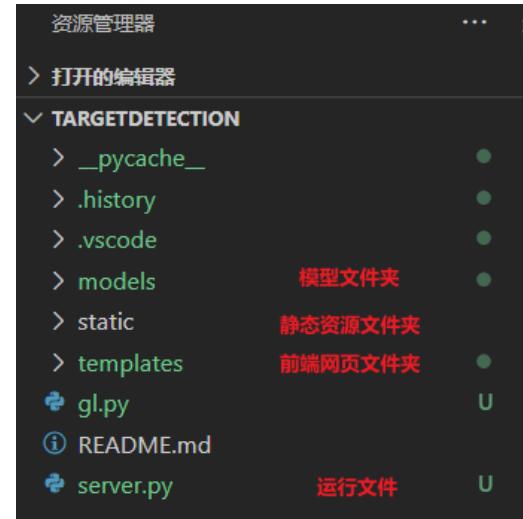
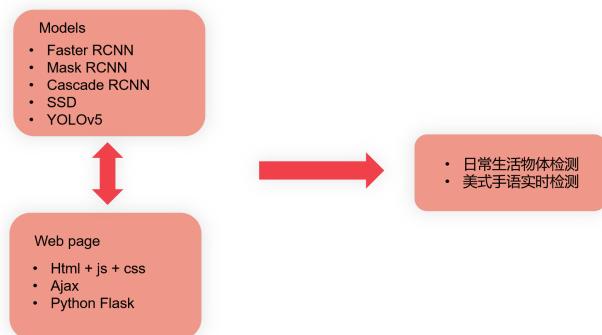


Figure 29: 项目整体框架

## 8 APPENDICES

### 8.1 References

- [1] Girshick, Ross, et al. Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR, 2014.
- [2] Girshick, Ross. Fast r-cnn. In ICCV, 2015.
- [3] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In NIPS, 2015.
- [4] K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask r-cnn. In ICCV, 2017.
- [5] Z. Cai, and N. Vasconcelos. Cascade R-CNN: Delving into High Quality Object Detection. In CVPR, 2018.
- [6] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg. SSD: single shot multibox detector. In ECCV, 2016.
- [7] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In CVPR, 2016.



Figure 31: 118 man\_lib\_with\_face original images

### 8.2 Workloads

杨逸君: YOLOv5 及优化、Faster RCNN、Cascade RCNN、SSD、制作 ASL 数据集

朱炳文: 数据集处理、制作 ASL 数据集、Mask RCNN

侯晓阳: 前后端、制作 ASL 数据集、YOLOv5

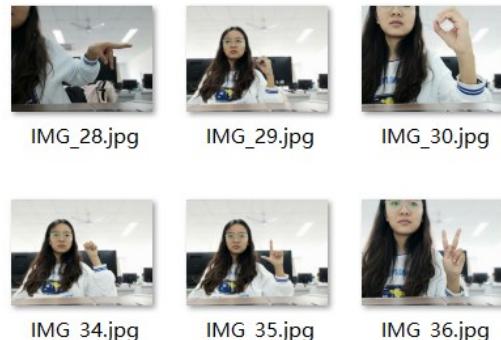


Figure 32: 37 girl\_lab\_with\_face original images

### 8.3 Results

由于篇幅有限，部分图片结果放置于此，在文中会有所引用。



Figure 30: 75 man\_lib\_close\_no\_face original images

Figure 33: 83 girl\_dor\_close\_no\_face original images



Figure 34: 192 man\_dor\_with\_face original images

```

<object>
  <name>person</name>
  <pose>Unspecified</pose>
  <truncated>0</truncated>
  <difficult>0</difficult>
  <bndbox>
    <xmin>174</xmin>
    <ymin>101</ymin>
    <xmax>349</xmax>
    <ymax>351</ymax>
  </bndbox>
  <part>
    <name>head</name>
    <bndbox>
      <xmin>169</xmin>
      <ymin>104</ymin>
      <xmax>209</xmax>
      <ymax>146</ymax>
    </bndbox>
  </part>
</object>

```

Figure 37: XML file format of VOC Annotation

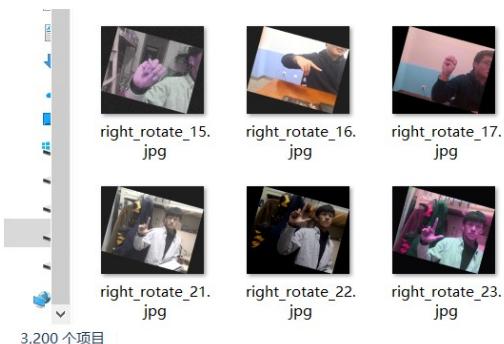


Figure 35: Final 3200 mixed augmented training images

```
{
  "file_name": "left_rotate_149.jpg",
  "width": 1671,
  "height": 1416,
  "id": 1655
},
{
  "file_name": "left_rotate_15.jpg",
  "width": 1671,
  "height": 1416,
  "id": 1656
},
{
  "file_name": "left_rotate_150.jpg",
  "width": 1671,
  "height": 1416,
  "id": 1657
},
{
  "file_name": "left_rotate_151.jpg",
  "width": 1671,
  "height": 1416,
  "id": 1658
}
```

Figure 38: json file format of COCO dataset

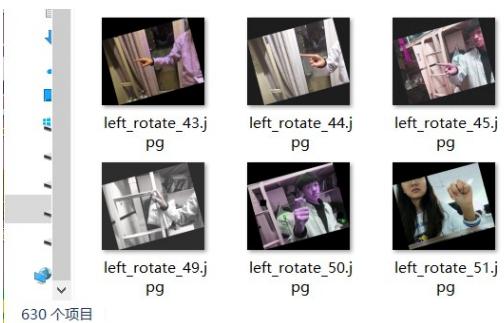


Figure 36: Final 630 mixed augmented validation images

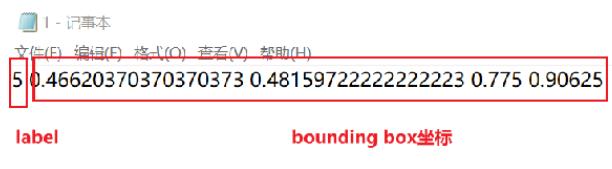


Figure 39: Txt file format in Yolov5 training

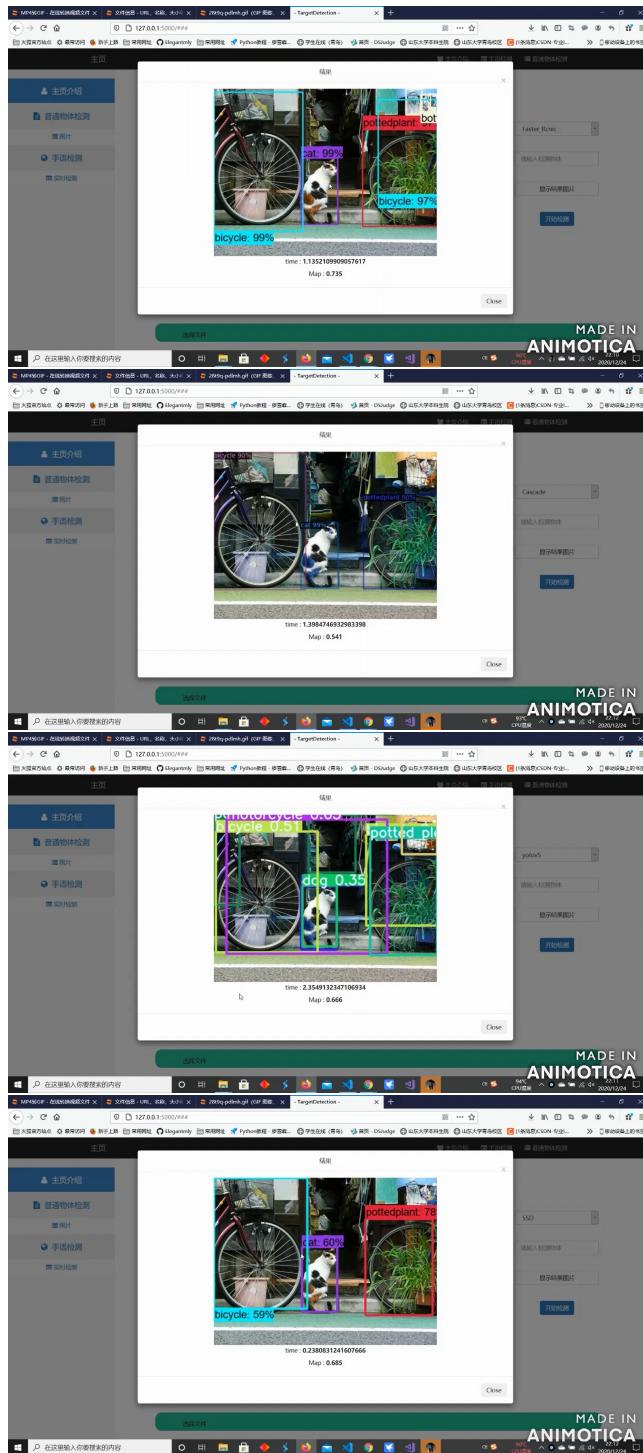


Figure 40: 普通物体检测  
从上至下依次为 Faster RCNN、Cascade、Mask、SSD

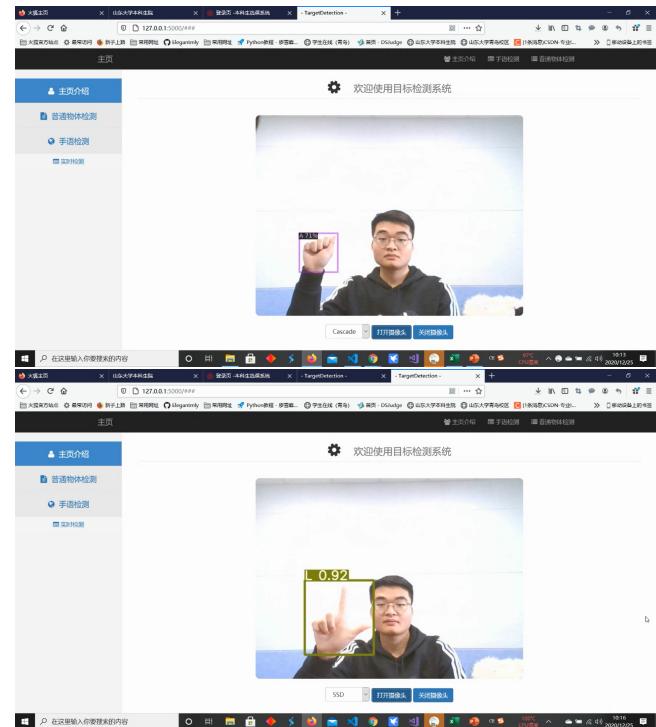


Figure 41: Cascade 和 SSD 手语检测结果