天津工业大学

计算机科学与技术学院

实验报告

课 程 名 称: 机器学习 .

实 习 题 目: 基于 Mask R-CNN 的物体检测 .

专 业 班 级: 计算机 S161 .

学 生 姓 名: 官亮，罗一铭，弓一凡，梁天宇

学 号: 1611640312，1611640116，1611640319，1611640324

指导教师: 刘丁 2019 年 1 月 6 日

1. 实验描述

1.1. 实验名称

基于 Mask R-CNN 的物体检测

1.2. 实验概述

物体检测是计算机视觉中的经典问题之一，其任务是用框去标出图像中物体的位置，并给出物体的类别。从传统的人工设计特征加浅层分类器的框架，到基于深度学习的端到端的检测框架，物体检测一步步变得愈加成熟。

Mask R-CNN 是何凯明基于以往的 Faster-RCNN 架构提出的新的卷积网络，主要思路就是把原有的 Faster-RCNN 进行扩展，添加一个分支来扩展更快的 R-CNN，该分支用于与现有分支并行地预测对象掩码以进行边界框识别。

1.3. 问题描述

这是在 Python 3，Keras 和 TensorFlow 上实现 Mask R-CNN。该模型为图像中对象的每个实例生成边界框和分割版。它基于 FPN 和 ResNet。实验包括：

（1）Mask R-CNN 的源代码，建立在 FPN 和 ResNet 之上。

（2）MS COCO 的训练代码

（3）MS COCO 的预训练重量

（3）Jupyter 笔记本可以在每一步都可视化检测管道

（4）ParallelModel 类用于多 GPU 培训

（5）预测 MS COCO 参数（AP）

（6）自己所提供的照片进行测试

1.4. 运行环境

Python 3.4 以上，TensorFlow 1.3，Keras 2.0.8 和其他常见软件包 requirements.txt

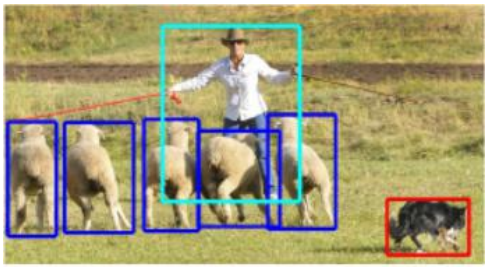
2. COCO 数据集

2.1. 数据集简介

COCO 数据集是微软团队获取的一个可以用来图像 recognition+segmentation+captioning 数据集。该数据集主要有的特点如下：（1）对象分割（2）上下文的认知（3）每个图像有多个对象（4）超过 300,000 张图片（5）超过 200 万个实例（6）80 个对象类别（7）每张图片 5 个字幕（8）10 万人的关键点。

该数据集主要用于解决 3 个问题：目标检测，目标之间的上下文关系，目标的 2 维上的精确定位。

（对象分类）



（目标定位）

（语义分割）

（实例分割）

2.2. 数据集分类

图像分类：

分类需要二进制的标签来确定目标是否在图像中。早期数据集主要是位于空白背景下的单一目标，如 MNIST 手写数据库，COIL household objects。在机器学习领域的著名数据集有 CIFAR-10 and CIFAR-100，在 32*32 影像上分别提供 10 和 100 类。最近最著名的分类数据集即 ImageNet，22,000 类，每类 500-1000

影像。

物体检测：

经典的情况下通过 bounding box 确定目标位置，期初主要用于人脸检测与行人检测，数据集如 Caltech Pedestrian Dataset 包含 350,000 个 bounding box 标签。PASCAL VOC 数据包括 20 个目标超过 11,000 图像，超过 27,000 目标 bounding box。最近还有 ImageNet 数据下获取的 detection 数据集，200 类，400,000 张图像，350,000 个 bounding box。由于一些目标之间有着强烈的关系而非独立存在，在特定场景下检测某种目标是是否有意义的，因此精确的位置信息比 bounding box 更加重要。

语义场景标注：

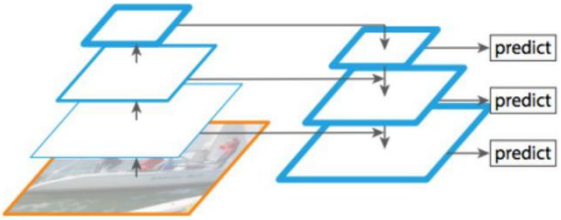
这类问题需要 pixel 级别的标签，其中个别目标很难定义，如街道和草地。数据集主要包括室内场景和室外场景的，一些数据集包括深度信息。其中，SUN dataset 包括 908 个场景类，3,819 个常规目标类(person, chair, car)和语义场景类(wall, sky, floor)，每类的数目具有较大的差别（这点 COCO 数据进行改进，保证每一类数据足够）。

2.3. COCO 展示

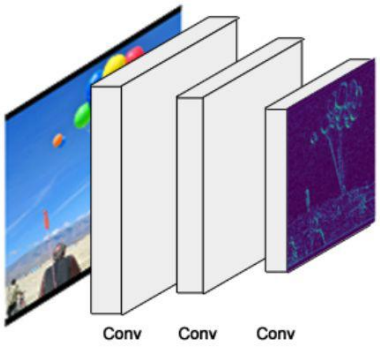
3. 网络结构描述

3. 1. ResNet-FPN：

FPN 网络：

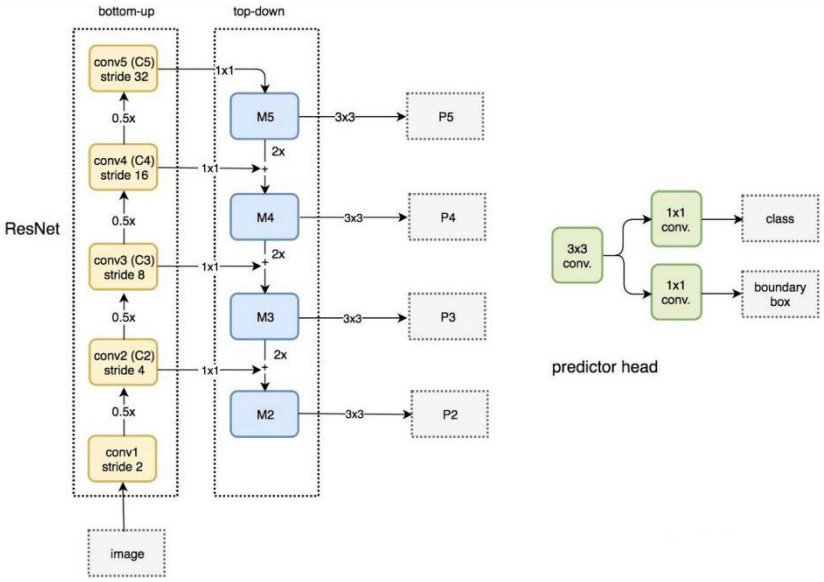


ResNet 骨架（ResNet-50 或者 ResNet-101）：



本次实验中采用 ResNet-101。

FPN 结构通过结合 ResNet 骨架网络形成了 ResNet-FPN 网络结构：



ResNet-FPN 包括 3 个部分，自下而上连接，自上而下连接和横向连接，用

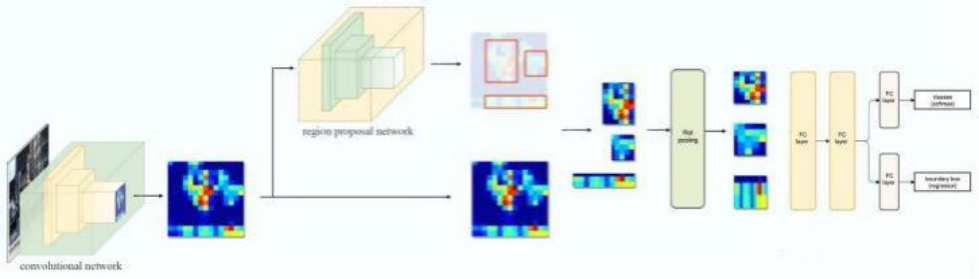
于特征提取。

3.2.Faster RCNN：

将 ResNet-FPN 和 Fast RCNN 进行结合构成 Faster RCNN 神经网络。

Faster RCNN 是两阶段的目标检测算法，包括阶段一的 Region proposal 以及阶段二的 bounding box 回归和分类。用一张图来直观展示 Faster RCNN 的整个流程：

Faster RCNN 使用 CNN 提取图像特征，然后使用 region proposal network（RPN）去提取出 ROI，然后使用 ROI pooling 将这些 ROI 全部变成固定尺寸，再喂给全连接层进行 Bounding box 回归和分类预测。

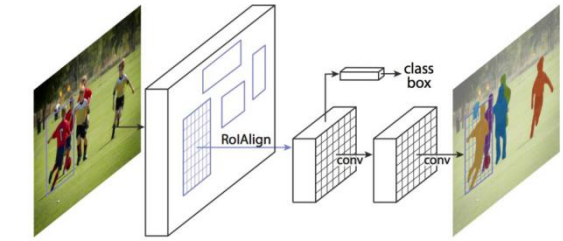


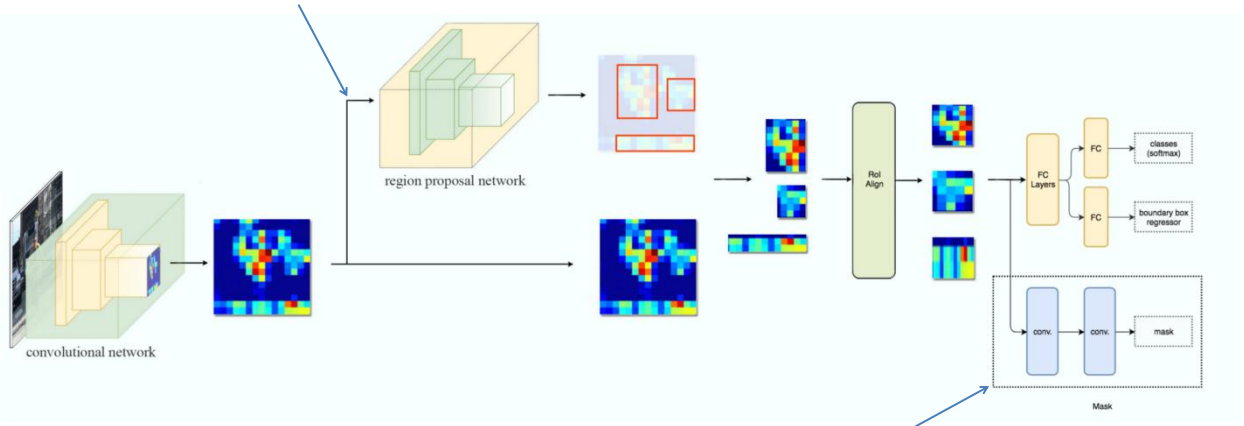
3.3.Mask R-CNN：

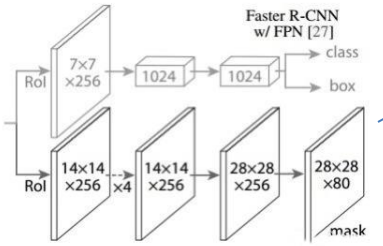
Mask R-CNN 的构建很简单，只是在 ROI pooling（实际上用到的是 ROIAlign，后面会讲到）之后添加卷积层，进行 mask 预测的任务。

Mask R-CNN 是一个两阶段的框架，第一个阶段扫描图像并生成提议（proposals，即有可能包含一个目标的区域），第二阶段分类提议并生成边界框和掩码。Mask R-CNN 扩展自 Faster RCNN，由同一作者在去年提出。Faster RCNN 是一个流行的目标检测框架，Mask R-CNN 将其扩展为实例分割框架。

两个阶段都连接到 ResNet-FPN 骨架结构。


RoIAlign
class
box
conv
conv


region proposal network
convolutional network
RoI Align
FC Layers
FC
FC
classes (softmax)
boundary box regressor
conv.
conv.
mask
Mask


Faster R-CNN
w/ FPN [27]
RoI
7×7
×256
1024
1024
class
box
RoI
14×14
×256
×4
14×14
×256
28×28
×256
28×28
×80
mask

4.ROI Align与损失函数

4.1.ROI Align：

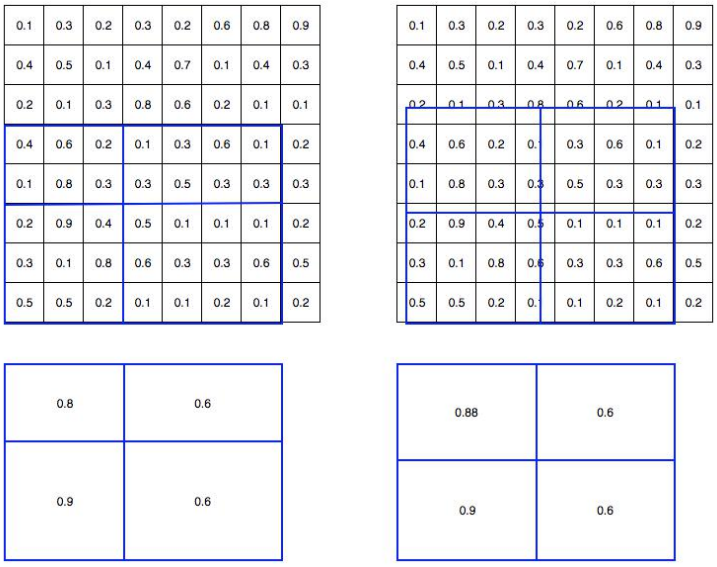
Faster RCNN 存在的问题是：特征图与原始图像是不对准的（mis-alignment），所以会影响检测精度。而 Mask R-CNN 提出了 RoI Align 的方法来取代 ROI pooling，RoI Align 可以保留大致的空间位置。

在 Faster RCNN 中，有两次整数化的过程：

1.region proposal 的 xywh 通常是小数，但是为了方便操作会把它整数化。

2.将整数化后的边界区域平均分割成 k x k 个单元，对每一个单元的边界进行整数化。

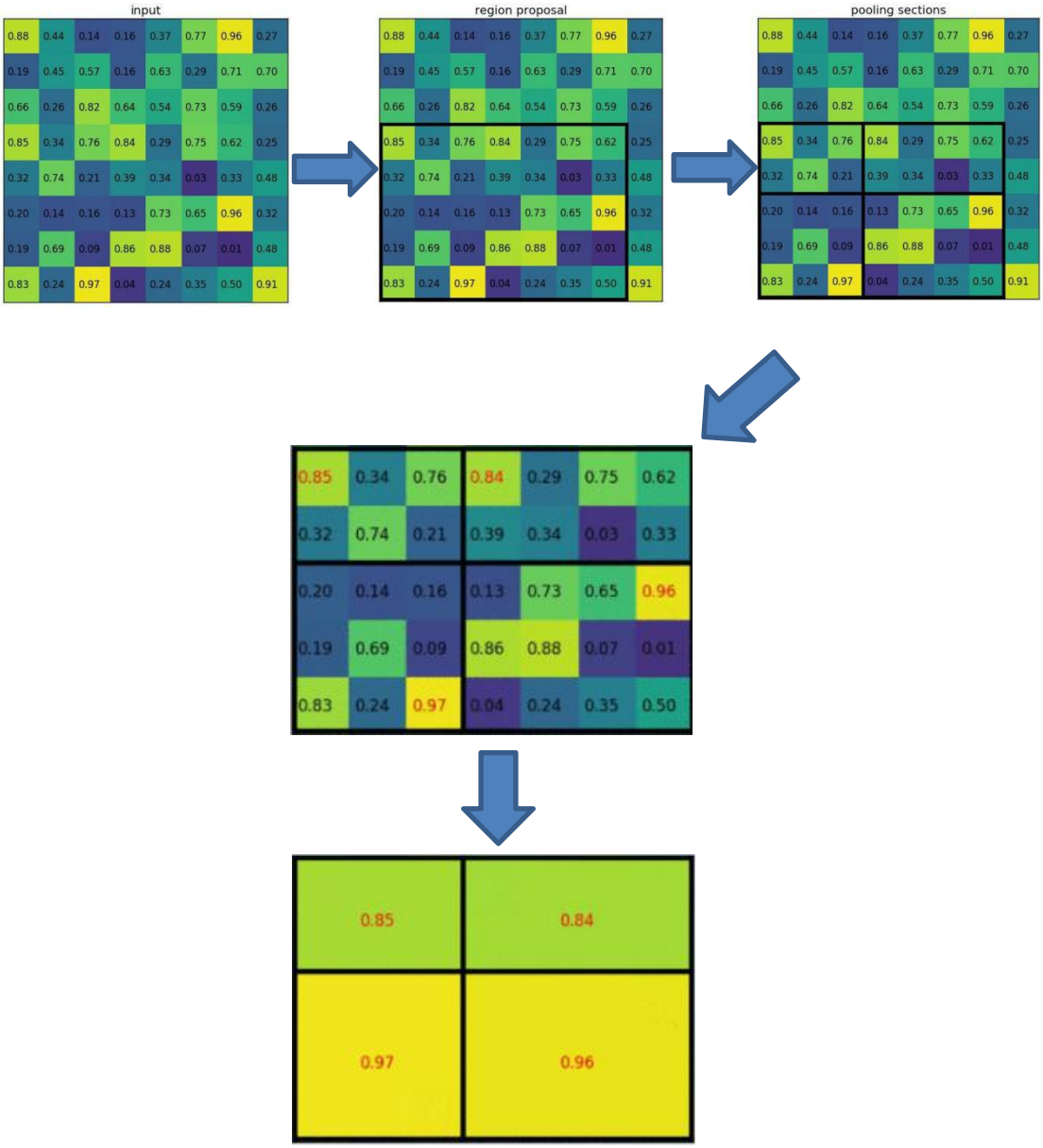
两次整数化的过程如下图所示：



经过上述两次整数化，此时的候选框已经和最开始回归出来的位置有一定的偏差，这个偏差会影响检测或者分割的准确度。解决这个问题，ROI Align 方法取消整数化操作，保留了小数，使用双线性插值的方法获得坐标为浮点数的像素点上的图像数值。

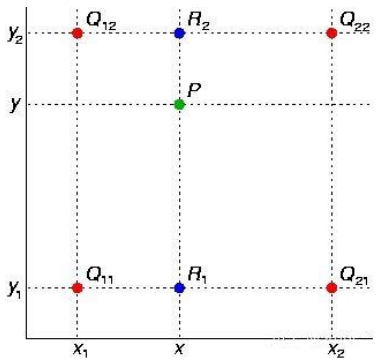
4.1.1.ROI 池化

分类器并不能很好地处理多种输入尺寸。它们通常只能处理固定的输入尺寸。但是，由于 RPN 中的边框精调步骤，ROI 框可以有不同的尺寸。因此，我们需要用 ROI 池化来解决这个问题，ROI 池化是指裁剪出特征图的一部分，然后将其重新调整为固定的尺寸。



4.1.2. 双线性插值

双线性插值本质上就是在两个方向上做线性插值



如图，假设我们想得到 P 点的插值，我们可以先在 x 方向上，对 和 之间做线性插值得到 ， 同理可得。然后在 y 方向上对 和 进行线性插值就可以得到最终的 P。其实知道这个就已经理解了双线性插值的意思了，如果用公式表达则如下（注意 前面的系数看成权重就很好理解了）。

首先在 x 方向进行线性插值，得到：

$$f(R_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \quad \text{Where} \quad R_1 = (x, y_1),$$
$$f(R_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \quad \text{Where} \quad R_2 = (x, y_2).$$

然后在 y 方向进行线性插值，得到：

$$f(P) \approx \frac{y_2 - y}{y_2 - y_1} f(R_1) + \frac{y - y_1}{y_2 - y_1} f(R_2).$$

这样就得到所要的结果 ：

$$f(x, y) \approx \frac{f(Q_{11})}{(x_2 - x_1)(y_2 - y_1)}(x_2 - x)(y_2 - y) + \frac{f(Q_{21})}{(x_2 - x_1)(y_2 - y_1)}(x - x_1)(y_2 - y)$$
$$+ \frac{f(Q_{12})}{(x_2 - x_1)(y_2 - y_1)}(x_2 - x)(y - y_1) + \frac{f(Q_{22})}{(x_2 - x_1)(y_2 - y_1)}(x - x_1)(y - y_1).$$

4. 2. 损失函数

采用 Mask R-CNN 定义多任务损失：

$$L=L_{cls}+L_{box}+L_{mask}$$

L_{cls} 和 L_{box} 与 Faster RCNN 的定义没有区别。需要具体说明的是 L_{mask}，假设一共有 K 个类别，则 mask 分割分支的输出维度是 ，对于 中的每个点，都会输出 K 个二值 mask（每个类别使用 sigmoid 输出）。

计算 loss 的时候，并不是每个类别的 sigmoid 输出都计算二值交叉熵损失，而是该像素属于哪个类，哪个类的 sigmoid 输出才要计算损失(如图红色方形所示)。并且在测试的时候，我们是通过分类分支预测的类别来选择相应的 mask 预测。

5. 实验结果展示

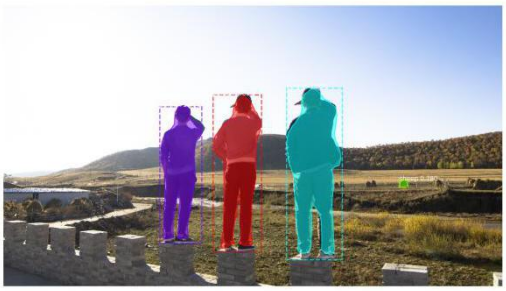
5.1. 实验参数展示：

```
(tensorflow) G:\Mask_RCNN-master\samples>python demo.py
Using TensorFlow backend.

Configurations:
BACKBONE                       resnet101
BACKBONE_STRIDES               [4, 8, 16, 32, 64]
BATCH_SIZE                     1
BBOX_STD_DEV                   [0.1 0.1 0.2 0.2]
COMPUTE_BACKBONE_SHAPE         None
DETECTION_MAX_INSTANCES        100
DETECTION_MIN_CONFIDENCE       0.7
DETECTION_NMS_THRESHOLD        0.3
FPN_CLASSIF_FC_LAYERS_SIZE     1024
GPU_COUNT                      1
GRADIENT_CLIP_NORM             5.0
IMAGES_PER_GPU                 1
IMAGE_MAX_DIM                  1024
IMAGE_META_SIZE                93
IMAGE_MIN_DIM                  800
IMAGE_MIN_SCALE                0
IMAGE_RESIZE_MODE              square
IMAGE_SHAPE                    [1024 1024    3]
LEARNING_MOMENTUM              0.9
LEARNING_RATE                  0.001
LOSS_WEIGHTS                   {'rpn_class_loss': 1.0, 'rpn_bbox_loss': 1.0, 'mrcnn_class_loss': 1.0, 'mrcnn_bbox_loss': 1.0, 'mrcnn_mask_loss': 1.0}
MASK_POOL_SIZE                 14
MASK_SHAPE                     [28, 28]
MAX_GT_INSTANCES               100
MEAN_PIXEL                     [123.7 116.8 103.9]
MINI_MASK_SHAPE                (56, 56)
NAME                           coco
NUM_CLASSES                    81
POOL_SIZE                      7
POST_NMS_ROIS_INFERENCE        1000
POST_NMS_ROIS_TRAINING         2000
ROI_POSITIVE_RATIO             0.33
RPN_ANCHOR_RATIOS              [0.5, 1, 2]
RPN_ANCHOR_SCALES              (32, 64, 128, 256, 512)
RPN_ANCHOR_STRIDE              1
RPN_BBOX_STD_DEV               [0.1 0.1 0.2 0.2]
RPN_NMS_THRESHOLD              0.7
RPN_TRAIN_ANCHORS_PER_IMAGE    256
STEPS_PER_EPOCH                1000
TOP_DOWN_PYRAMID_SIZE          256
```

5.2. 实验结果展示：

```
2019-01-08 14:59:57.198180: I T:\src\github\tensorflow\tensorflow\core\platform\cpu_feature_guard.cc:14
not compiled to use: AVX2
Processing 1 images
image                    shape: (1441, 1920, 3)       min:    0.00000  max:  255.00000  uint8
F:\anaconda\envs\tensorflow\lib\site-packages\skimage\transform\_warps.py:110: UserWarning: Anti-aliasin
rtifacts when down-sampling images.
  warn("Anti-aliasing will be enabled by default in skimage 0.15 to "
molded_images            shape: (1, 1024, 1024, 3)    min: -123.70000  max:  149.10000  float64
image_metas              shape: (1, 93)               min:    0.00000  max: 1920.00000  float64
anchors                  shape: (1, 261888, 4)        min:   -0.35390  max:    1.29134  float32
person 0.967
person 0.961
person 0.825
save success!
Processing 1 images
image                    shape: (2916, 5184, 3)       min:    0.00000  max:  255.00000  uint8
molded_images            shape: (1, 1024, 1024, 3)    min: -123.70000  max:  151.10000  float64
image_metas              shape: (1, 93)               min:    0.00000  max: 5184.00000  float64
anchors                  shape: (1, 261888, 4)        min:   -0.35390  max:    1.29134  float32
person 1.000
person 1.000
person 1.000
sheep 0.780
save success!
```



6. 心得与体会

本次大作业是一次从选题开始就由我们自主发挥的实验，刚开始对于选题有些纠结，不过在小组成员与老师一番讨论后决定把目标物体检测作为本次大作业的课题。相较前两次大作业而言，这一次的课题所涵盖的知识点更多，许多地方都需要上网查阅资料去完成，同时请教别的同学。这也让我们学到了许多书本上没有的东西，并且在实际操作中将这些知识充分地巩固吸收。我们也深刻体会到机器学习这门课程只靠考前加班、背考点应付考试是没有任何意义的，必须要实践出真知，多做实验、多写代码才能有长进。除此以外，我们也体会到了合作的重要性，同一小组的成员之间必须分工明确，多沟通，彼此之间互补空缺，才能让课题以最快速度完成。总的来说，这次大作业虽然过程中也遇到了一些问题，但都一一顺利解决，对我们小组的成员都是一次非常好的锻炼机会，相信这一次的课题中得到的经验对我们后面几个学期的学习乃至未来工作也都会有很大的帮助。

附（分工）：

- 理论分析：弓一凡、梁天宇
- 网络搭建：弓一凡、梁天宇
- 框架连接：官亮、罗一铭
- 代码调试：官亮、罗一铭
- 实验报告：弓一凡、梁天宇、官亮、罗一铭