**一种低表面亮度星系的检测方法**

A Way of Detection of Low Surface Brightness Galaxies

作者：张丰毅

院校：山东大学

指导老师：衣振萍

**摘要**

低表面亮度星系（low-surface-brightness galaxy），或LSB星系，是一种弥漫星系，当从地球观察这些星系时，在夜空的环境中，它们的表面亮度至少比周围的背景天光低一个星等。LSB星系对于暗物质的研究具有重要意义。为此，本文提出了一种对于低表面亮度星系的识别办法。主要进行了对天文图像进行增强、通道合并；对数据表进行重复合并、异常值剔除；使用YOLOv3框架，利用darknet53卷积残差网络，在谷歌提供的Colab云计算平台上训练对LSB星系图像的识别和定位等工作。经过反复试验不同参数，最终得到mAP=93.40%（IoU\_thresh=0.5），recall = 94%（conf\_thresh = 0.25）的优良训练结果，能够较为准确的找出含有LSB星系的天文图像，并给出其中心坐标。为LSB星系的识别提供了一种思路。

**关键词**：LSB，目标检测，YOLO，Darknet-53

**ABSTRACT**

A low-surface-brightness galaxy, or LSB galaxy, is a diffuse galaxy with a surface brightness that, when viewed from Earth, is at least one magnitude lower than the ambient night sky. LSB galaxies are of great importance to the study of dark matter. Therefore, this paper presents a method to identify low surface brightness galaxies. The enhancement and channel merging of astronomical images, as well as merging and outlier elimination of data table were mainly carried out; With YOLOv3 framework, darknet53 convolution residual network was utilized to train the identification and positioning of LSB galaxy images on the Colab cloud computing platform provided by Google. After repeated experiments with different parameters, the excellent training results of mAP=93.40% (IoU\_thresh=0.5) and recall = 94% (conf\_thresh = 0.25) were finally obtained, and the astronomical images containing LSB galaxies could be more accurately marked and their central coordinates were given. It provides an idea for the detection of LSB galaxy.

**Keywords:** LSB, Target detection, YOLO, Darknet-53

目录

[一、研究意义： 4](#_Toc42598961)

[二、现有研究 4](#_Toc42598962)

[三、研究过程 4](#_Toc42598963)

[3.1 天文图像处理 4](#_Toc42598964)

[3.1.1 提取图像 4](#_Toc42598965)

[3.1.2 图像增强 7](#_Toc42598966)

[3.1.3 合并通道 10](#_Toc42598967)

[3.2 数据预处理 11](#_Toc42598968)

[3.2.1.数据去重： 11](#_Toc42598969)

[3.2.2 剔除异常值 12](#_Toc42598970)

[3.3 YOLO v3 算法综述 12](#_Toc42598971)

[3.3.1 YOLO v3基本结构 12](#_Toc42598972)

[3.3.2 Darknet-53卷积残差网络 13](#_Toc42598973)

[3.3.3 多尺度特征对象检测： 14](#_Toc42598974)

[3.3.4 9种尺度的Anchor Box 15](#_Toc42598975)

[3.3.5 Logistic对象分类 16](#_Toc42598976)

[3.3.6 网络输出 16](#_Toc42598977)

[3.4 在Colab上搭建YOLO v3环境 17](#_Toc42598978)

[3.5 制作自己的数据集进行训练 17](#_Toc42598979)

[3.5.1 制作Annotation 17](#_Toc42598980)

[3.5.2 创建train.txt和test.txt 18](#_Toc42598981)

[3.5.3 创建obj.names和obj.data 19](#_Toc42598982)

[3.5.4 修改yolov3.cfg 20](#_Toc42598983)

[3.5.5 开始训练 21](#_Toc42598984)

[3.6 训练结果 22](#_Toc42598985)

[3.6.1 绘制loss-mAP趋势图 22](#_Toc42598986)

[3.6.2 检测单张图片 23](#_Toc42598987)

[3.6.3 批量检测测试集 23](#_Toc42598988)

[3.6.4 模型性能评估 25](#_Toc42598989)

[3.6.7 模型改进 27](#_Toc42598990)

[四、研究结果 28](#_Toc42598991)

[五、总结与展望 28](#_Toc42598992)

[六、致谢 29](#_Toc42598993)

[附录：参考文献 30](#_Toc42598994)

# 一、研究意义：

低表面亮度星系（low-surface-brightness galaxy），或LSB星系，是一种弥漫星系，当从地球观察这些星系时，在夜空的环境中，它们的表面亮度至少比周围的背景天光低一个星等[1]。

对LSB星系的自转曲线测量显示出极高比率的质量-光度比（γ）。这意味着恒星和发光气体对LSB星系的整体质量平衡贡献很少。在LSB星系中心的恒星中没有大的超高密度恒星，这和普通漩涡星系的核球有很大差异。因此，即使在星系中心似乎也是由暗物质主导的。这使得它们成为研究暗物质的绝佳实验室[1]。

# 二、现有研究

B. Vollmer，M. Petremand, M. Petremand等人提出了一种基于多尺度马尔可夫模型的低面亮度星系探测算法，称为 MARSIAA (MARkovian Software for Image Analysis in astronauts)。MARSIAA可以同时应用于不同的波段。 它根据图像的面亮度和周围环境，将图像分割成用户定义的多个类——通常，一个或两个类包含 LSB 结构。他们开发了一种名为 DetectLSB的算法，它允许从 MARSIAA 选择的候选源中有效地识别 LSB 星系。这种方法与匹配滤波器的应用和 SExtractor 的优化使用是互补的，具有无标度，可以同时应用于多个波段，并且适合于太空中拥挤的区域的优点。[2]

# 三、研究过程

## 3.1 天文图像处理

### 3.1.1 提取图像

一幅数字化图像在数学上等效为一个矩阵。图像处理的方法很多,对于天文图像而言，常见的处理有图像增强，图像恢复，图像分割等。

对于fits类型的天文图像文件，由文件头和数据组成。在文件头中存储有对该文件的描述，如观测时间、观测对象、拍照温度、曝光时间等信息，而数据部分则保存其对应的的16位图像矩阵。

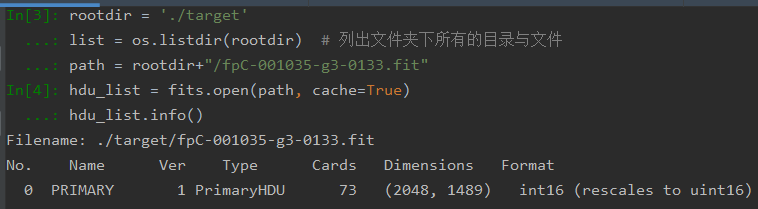


图1：fits信息

可以看到只有一个PrimaryHDU，而且第一个HDU里面就存放了数据。实验数据均为1489\*2048的矩阵，使用hdu[0].data 或者 fits.getdata()查看矩阵：

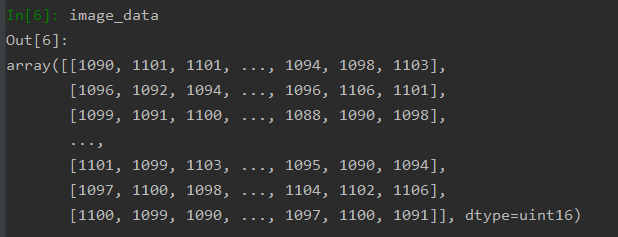


图2：灰度矩阵分布

可以看到矩阵里面为16位灰度值。

为了知道图像灰度的总体分布，进行下列四项指标的评估：

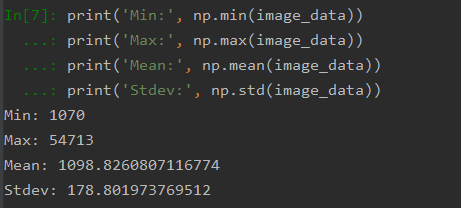


图3：数理统计指标

可以发现大量数据集中在（1098-178，1098+178）附近，而最大值54713与平均值相差了2^15/2^10 = 5 个数量级

用plt.hist(image\_data.flatten(), bins=50) 查看灰度分布的直方图

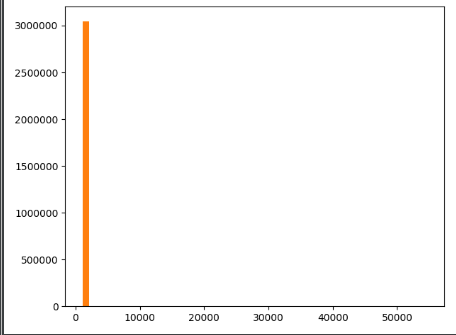


图4：图像灰度直方图

直观的说明了图像集中分布在均值附近，于是我们一定要对图像进行处理，以防归一化后退化问题。为什么要归一化？为了输出标准8位图像，我们不需要那么高的位深度。对16位图像进行归一化处理，然后再乘以255，让像素分布在[0,255]区间内

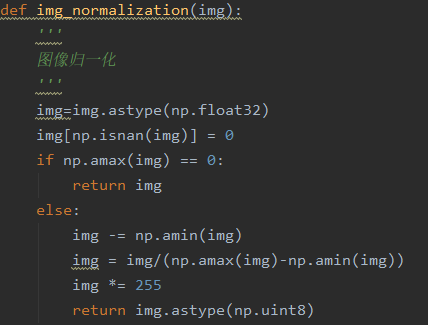


图5：归一化转8位图像

所谓退化(degeneration) ,是指图像归一化后，较小部分的值被映射为接近或等于0（即黑色）的灰度，如果原始图像的大部分像素属于这部分区间，那么这张图像将丢失暗部的细节。如果我们不加处理的输出，那么视觉效果将会这样：



图6：未处理图像

可以说，这样的图像，连亮度最高的星系也很难用肉眼辨别，更不用说LSB星系了

### 3.1.2 图像增强

#### 方案1：直方图均衡化（Histogram equalization）

直方图均衡化就是对灰度值r进行如下变换s=T(r)，使得变换后的灰度分布概率均衡 的分布在灰度区间中。这样能够发现一些原先肉眼很难发现的细节。实现代码如下：

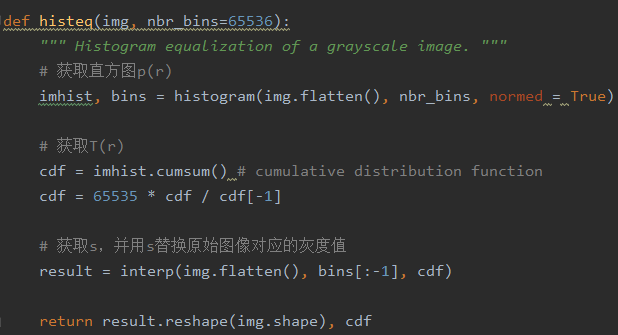


图7：直方图均衡化

处理后的直方图如下：

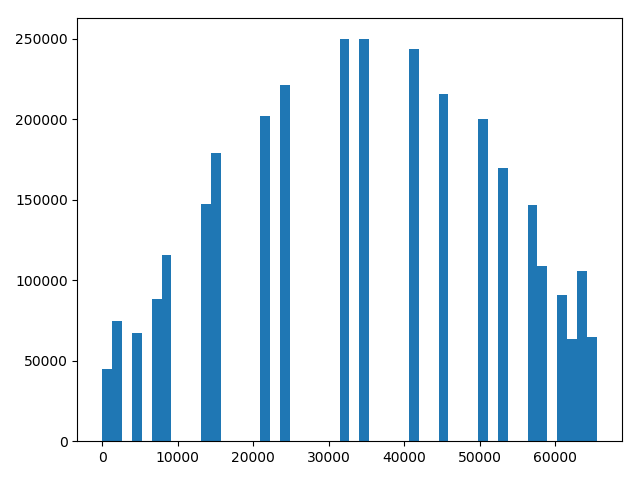


图8：均衡化后的直方图

保存为tif格式，可以看到图像的对比度增加了，原本看不见的星系也清晰可见。



图9：直方图均衡化后的图片

遗憾的是，这种办法得到的8位图像保存为jpg格式时会严重失真，且低亮度星表面星系特征不明显。

方案2：主成分增强法：

这种方法适用于数据集中分布在某值附近，而极端值相差甚远，数量又少。那我们完全可以抛弃这些值。我的方法是，取灰度分布在[mean-std,mean+std]之间的像素，超过或不够的像素改为边缘值。代码也很简单：

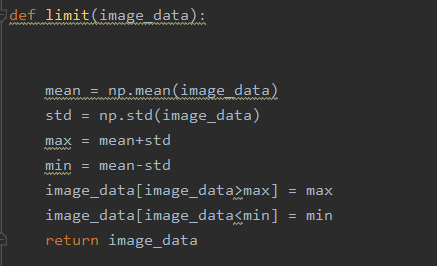


图10：主成分增强法

我们将输出的矩阵归一化后转为8位图像输出：



图11：主成分增强后的图片

可以看到，对比度比之前有所降低，保留了更多暗部的细节。

### 3.1.3 合并通道

对于给定的数据集来说，对于同一幅图像，有G（green）和R(red)两个波段的图像。为此，我们需要对相同图像的不同波段进行合并。

有两种方法可以进行通道的合并，一种是使用Numpy库中的np.dstack()进行第三维上的堆叠。第二种是使用opencv库种的cv2.merge()对不同通道进行合并。

这里使用第二种方法，由于merge的通道最少为3，对第三个通道（blue）矩阵置0

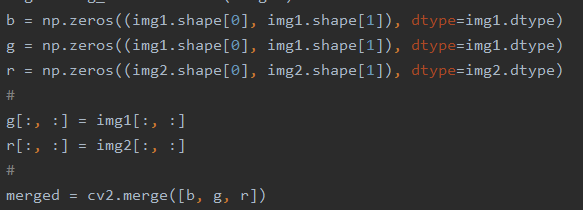


图12：合并通道

合并后效果如下：

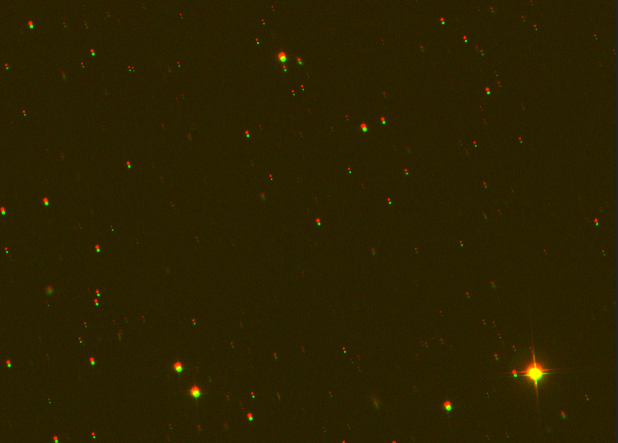


图13：合并通道后的8位RGB图像

于是我们只要将target.csv中所有出现的文件对于其对应波段的图片合并输出即可。

## 3.2 数据预处理

### 3.2.1.数据去重：

给出的csv文件中存在1129项，但是经过查找重复项，发现有9项存在重复，他们在表中的序号为（从0起）：

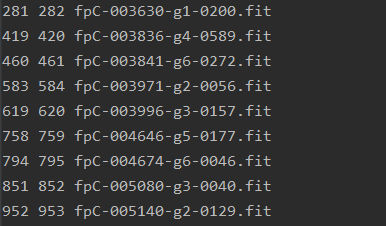


图14：重复的表项序号

这意味着，这些图片中存在2个LSB星系，于是要对其进行合并。

### 3.2.2 剔除异常值

为了防止脏数据影响训练效果，在制作label之前要对数据进行检查。包括nan值检查，超限值检查等。经检测，发现序号266-272的LSB中心坐标记录，远远超过了图像的长宽(1489\*2048)，于是将这些项及对应文件剔除。

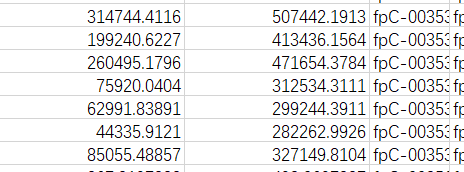


图15：异常数据

## **3.3 YOLO v3 算法综述**

对于目标检测(Object Detection)，从R-CNN到Faster R-CNN一直采用的思路是proposal+分类 （proposal 提供位置信息，分类提供类别信息）精度已经很高，但由于two-stage（proposal耗费时间过多）处理速度不行达不到real-time效果。

YOLO提供了另一种更为直接的思路： 直接在输出层回归bounding box的位置和bounding box所属的类别，把整张图作为网络的输入，把 Object Detection 的问题转化成一个 Regression 问题

关于YOLO算法，由于之前已经有所熟悉，这里不在赘述。这里着重介绍一下YOLO的第三个改进版本，YOLOv3主要调整了网络结构，利用多尺度特征进行对象检测并且对象分类用Logistic取代了softmax。原著论文见参考文献[4]。

### 3.3.1 YOLO v3基本结构

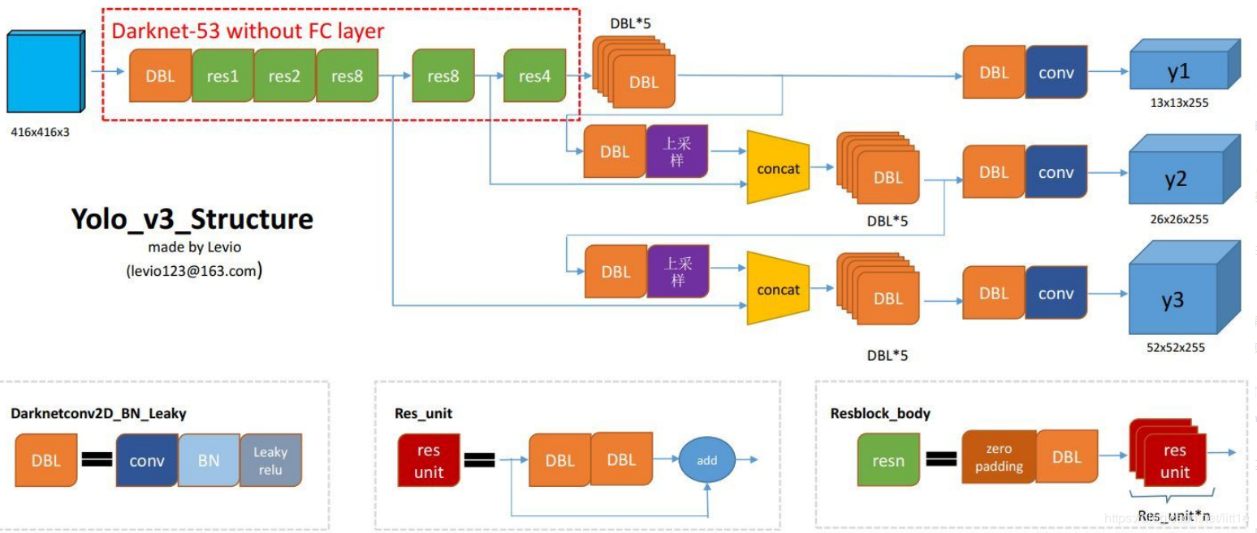


图16：YOLO\_v3结构图[5]

**DBL：**代表代码中的Darknetconv2d\_BN\_Leaky，是yolo\_v3的基本组件。就是卷积+BN+Leaky relu。

**resn**：n代表数字，有res1，res2, … ,res8等等，表示这个res\_block里含有多少个res\_unit

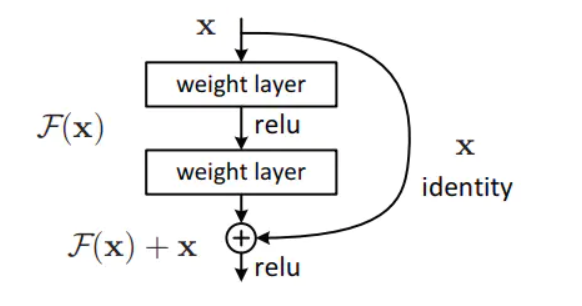


图17：一个残差组件

**concat**：张量拼接。将darknet中间层和后面的某一层的上采样进行拼接

### 3.3.2 Darknet-53卷积残差网络

为了达到更好的分类效果，作者自己设计训练了darknet-53卷积网络，大量使用残差的跳层连接，并且为了降低池化带来的梯度负面效果，没有池化层和全连接层，用conv的stride来实现降采样。在这个网络结构中，使用的是步长为2的卷积来进行降采样。每次将边长缩小一半，一共5次，即将图像缩小到原图的1/32

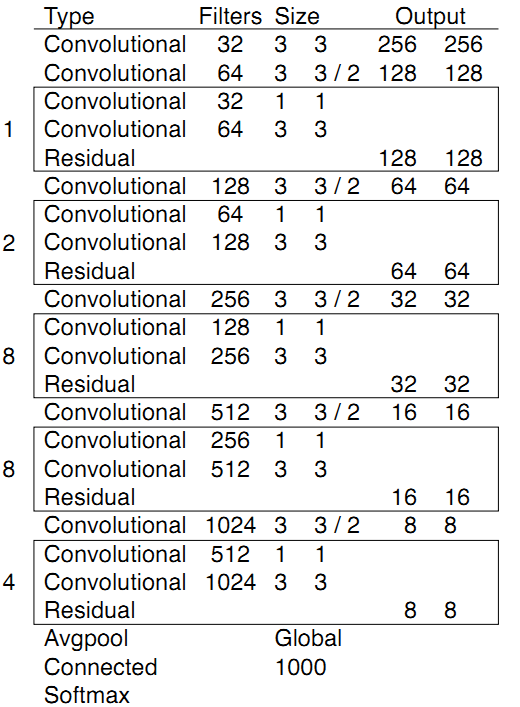


图18：Darknet-53卷积残差网络[4]

上图的Darknet-53网络采用256\*256\*3作为输入，最左侧那一列的1、2、8等数字表示多少个重复的残差组件。

### 3.3.3 多尺度特征对象检测：

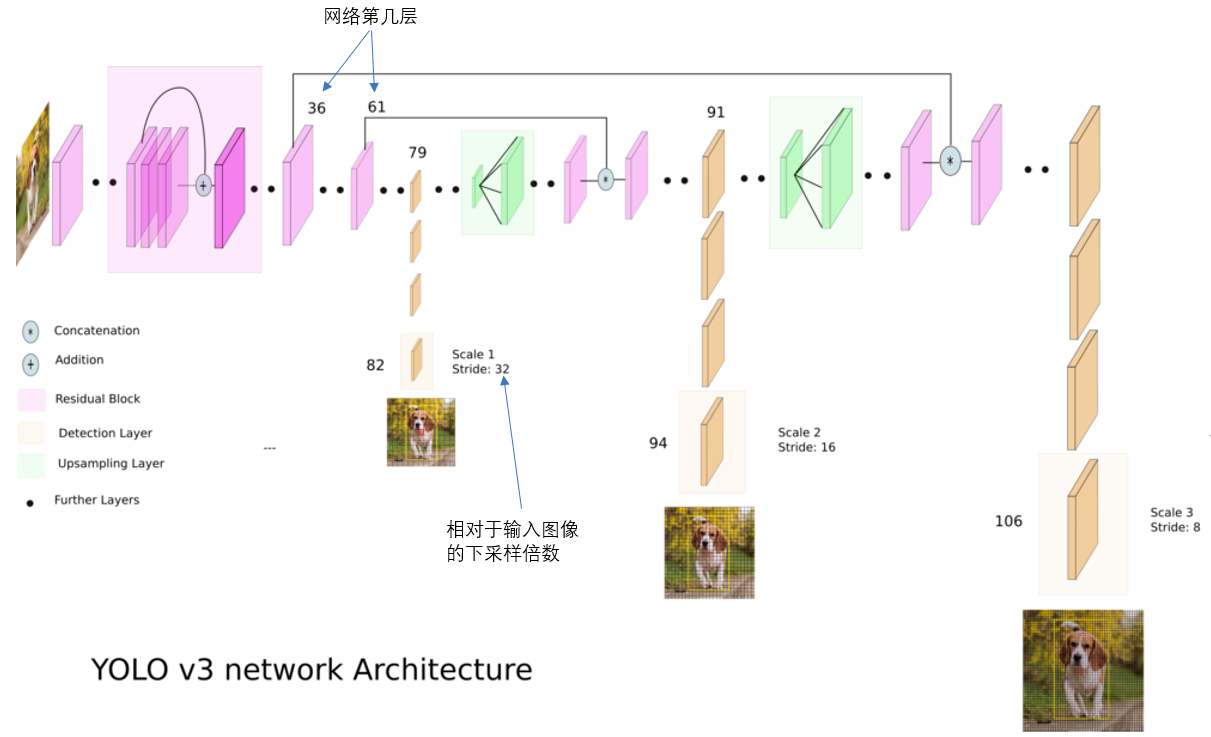


图19：YOLO v3网络结构[6]

YOLO v3采用了3个不同尺度的特征图来进行对象检测。卷积网络在79层后，经过下方几个黄色的卷积层得到一种尺度的检测结果。相比输入图像，这里用于检测的特征图有32倍的下采样。比如输入是416\*416的话，这里的特征图就是13\*13了。由于下采样倍数高，这里特征图的感受野比较大，因此适合检测图像中尺寸比较大的对象。

为了实现细粒度的检测，第79层的特征图又开始作上采样（从79层往右开始上采样卷积），然后与第61层特征图融合（Concatenation），这样得到第91层较细粒度的特征图，同样经过几个卷积层后得到相对输入图像16倍下采样的特征图。它具有中等尺度的感受野，适合检测中等尺度的对象。

最后，第91层特征图再次上采样，并与第36层特征图融合（Concatenation），最后得到相对输入图像8倍下采样的特征图。它的感受野最小，适合检测小尺寸的对象。

### 3.3.4 9种尺度的Anchor Box

随着输出的特征图的数量和尺度的变化，先验框的尺寸也需要相应的调整。YOLO\_v3采用K-means聚类得到先验框的尺寸,对应不同尺寸的特征图。

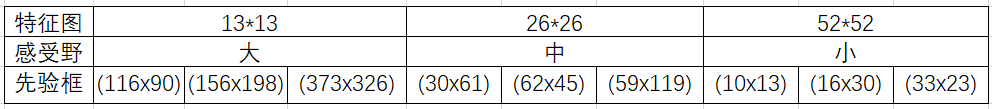


图20：特征图与Anchor Box[7]

分配上，在最小的13\*13特征图上（有最大的感受野）应用较大的先验框(116x90)，(156x198)，(373x326)，适合检测较大的对象。中等的26\*26特征图上（中等感受野）应用中等的先验框(30x61)，(62x45)，(59x119)，适合检测中等大小的对象。较大的52\*52特征图上（较小的感受野）应用较小的先验框(10x13)，(16x30)，(33x23)，适合检测较小的对象。

### 3.3.5 Logistic对象分类

预测对象类别时不使用softmax，改成使用logistic（逻辑回归）对象预测。这样能够支持多标签对象（比如一个人有Woman 和 Person两个标签）

### 3.3.6 网络输出

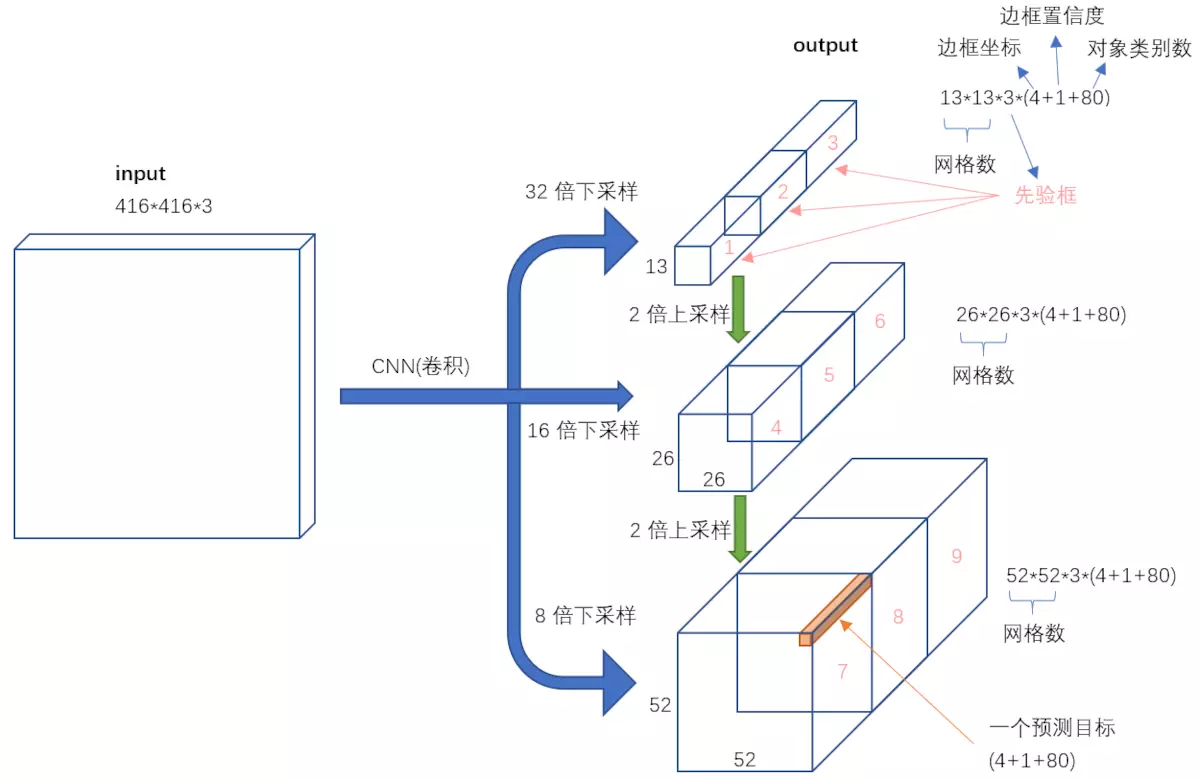


图21：输入映射到输出[7]

总的来说，对于一个输入图像，YOLO v3将其映射到3个尺度的输出张量，代表图像各个位置存在各种对象的概率。

对于一个416\*416的输入图像，在每个尺度的特征图的每个网格设置3个先验框，总共有 13\*13\*3 + 26\*26\*3 + 52\*52\*3 = 10647 个预测。每一个预测是一个(4+1+80)=85维向量，这个85维向量包含边框坐标（4个数值），边框置信度（1个数值），对象类别的概率（对于COCO数据集，有80种对象）。

对比一下，YOLO2采用13\*13\*5 = 845个预测，YOLO3的尝试预测边框数量增加了10多倍，而且是在不同分辨率上进行，所以mAP以及对小物体的检测效果有一定的提升。

综上所述，YOLOv3相比于前代的改进，不仅维持了速度快的优势（比 R-CNN 快 1000 倍，比 Fast R-CNN 快 100 倍），并且改善了对于小物体、粗粒度的检测，所以非常适合用于检测LSB星系。

## 3.4 在Colab上搭建YOLO v3环境

由于电脑性能不足以支撑如此庞大的计算量， 从而选择在谷歌的云平台Colab上远程操作vm实例训练网络。这部分涉及到的代码量和说明比较多，请参见代码文件中的”YOLOv3\_Colab.ipynb” Jupiter笔记本。

## 3.5 制作自己的数据集进行训练

感谢AlexeyAB，他的GitHub repo中仔细讲解了YOLOv3的配置、运行和改进步骤[8]

### 3.5.1 制作Annotation

关于如何制作YOLO v3格式的数据集，网上有很多教程[3]。大多数是把用LabelImg标注出的VOC格式的数据集转为YOLO v3格式的数据集。由于已经给出LSB的中心坐标，且LSB靠肉眼较难识别，所以这里直接以给出的坐标为中心点，以一定大小的矩形框作为Bounding box，制作标签。下面是YOLOv3格式的label：

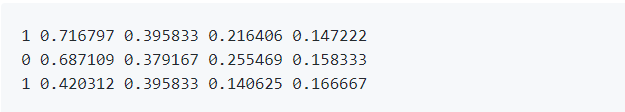


图22：YOLOv3格式的label

其中，第一个参数为类别，后面的参数依次为(x\_centre,y\_centre,width,height)，而且均为（0-1）之间的小数，为比例值

<x> = <absolute\_x> / <image\_width>

<height> = <absolute\_height> / <image\_height>

例如，对于如下LSB坐标，设bounding box的大小为整幅图像的10%，有如下label：


图23：label示例

将制作好的label和对应的图片放在一起，保持对应文件名相同，打包成zip文件。然后上传到Google Drive，解压到云服务器的content/darknet/data/obj文件夹中备用。

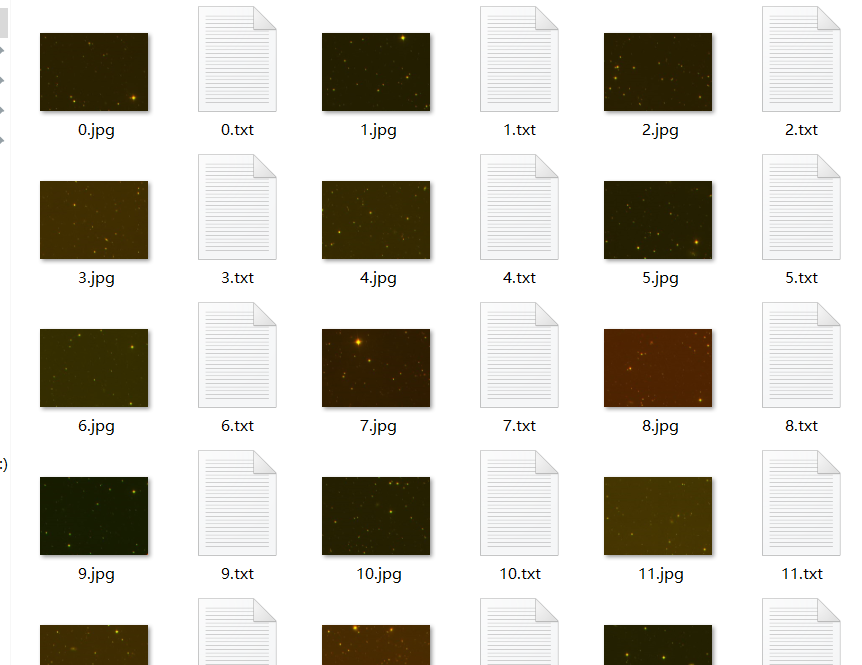


图24：obj文件夹

### 3.5.2 创建train.txt和test.txt

编写generate\_train.py，生成存放训练集和测试集的图片地址的txt文件,分配比例为9:1

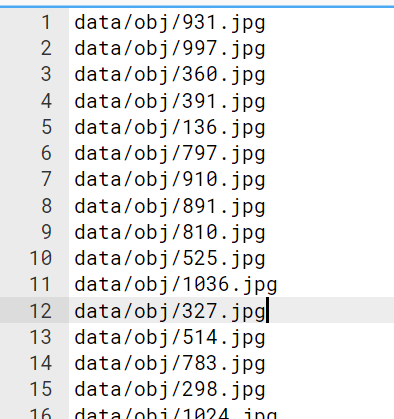


图25：train.txt

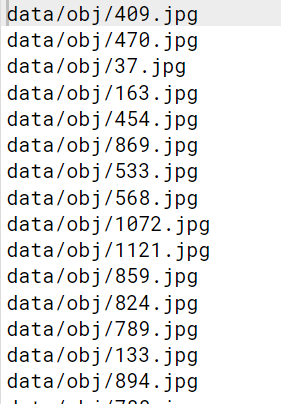


图26：test.txt

上传到云服务器darknet/data 备用

### 3.5.3 创建obj.names和obj.data

obj.names文件里面列举需要检测目标的名字，obj.data里面指明类别数，训练集地址，测试集地址，以及保存权重文件的地址

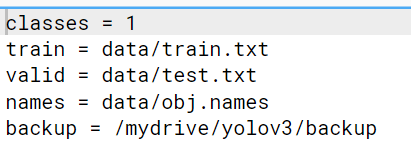


图27：obj.data



图28：obj.names

### 3.5.4 修改yolov3.cfg

原版的cfg文件是以COCO数据集的80个物体编写的，不能直接使用，主要修改下面几个参数：

batch=64

subdivisions=16

…

max\_batches = 2000//2000\*class\_num

steps=1600,1800//max\_batches\*0.8,max\_batches\*0.9

[convolutional]

…

filters=18//(5+class\_num)\*3

…

[yolo]

…

classes=1

…

random=0//random=1会改变输入图片尺寸训练，增强对不同尺寸图片识别的灵活性，但会增加训练时间

…

[convolutional]

…

filters=18//(5+class\_num)\*3

…

[yolo]

…

classes=1

…

random=0

…

[convolutional]

…

filters=18//(5+class\_num)\*3

…

[yolo]

…

classes=1

…

random=0

修改结束后,重命名为yolov3\_custom\_4上传到云服务器darknet/cfg备用

### 3.5.5 开始训练

在3.4中配置的Colab会话中键入如下代码开始训练：

!./darknet detector train data/obj.data cfg/yolov3\_custom4.cfg darknet53.conv.74 -dont\_show -map

其中，detector 后面的参数含义依次为：训练集训练，obj.data,位置 yolov3.cfg位置, 预训练权重，不显示训练中间过程，在最终形成的loss趋势图中显示mAP变化曲线。

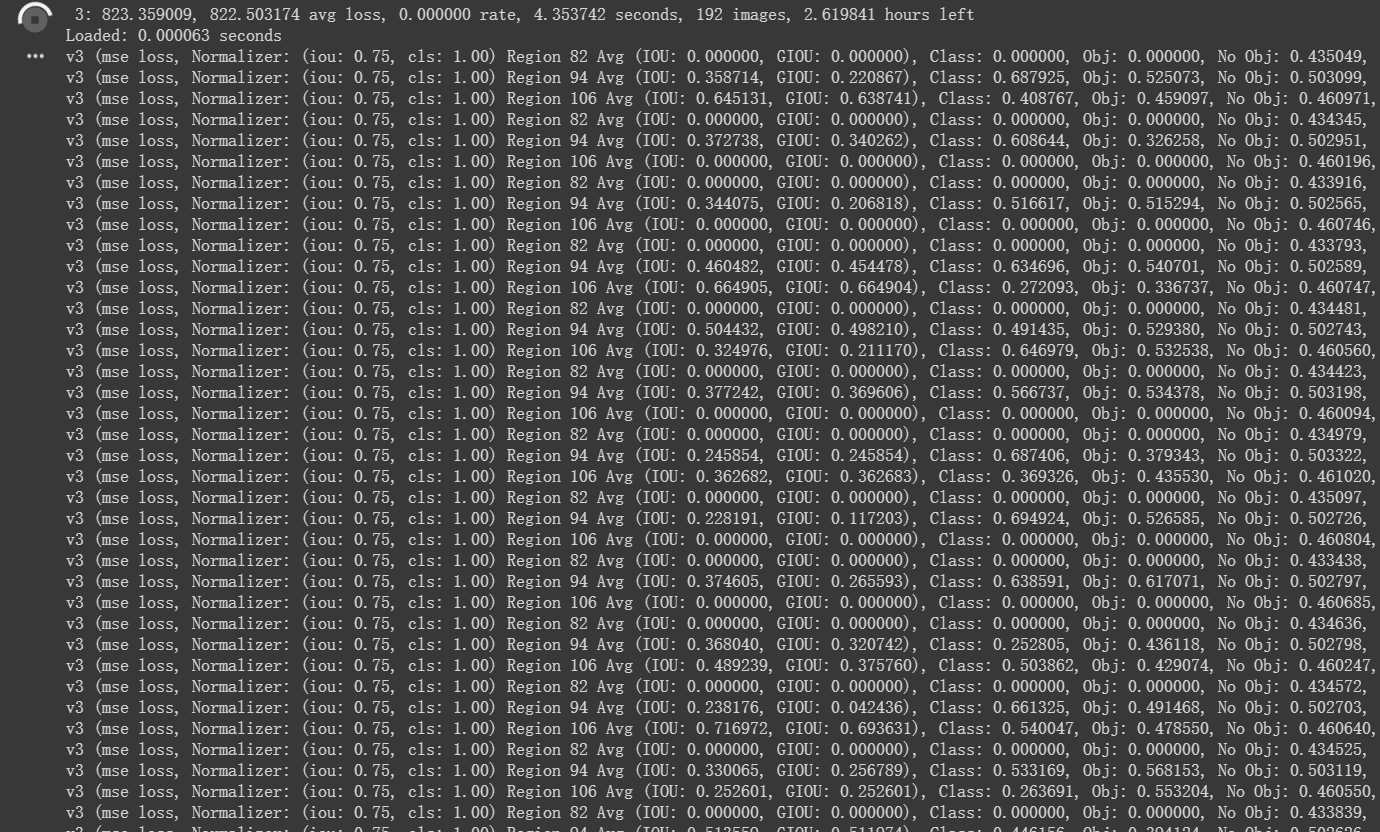


图29：训练输出

训练时长取决于喂入图片的数量、大小以及cfg文件的参数配置，这里我的配置需要7小时左右。

## 3.6 训练结果

### 3.6.1 绘制loss-mAP趋势图

根据训练过程中log文件，可以绘制出如下loss趋势图：

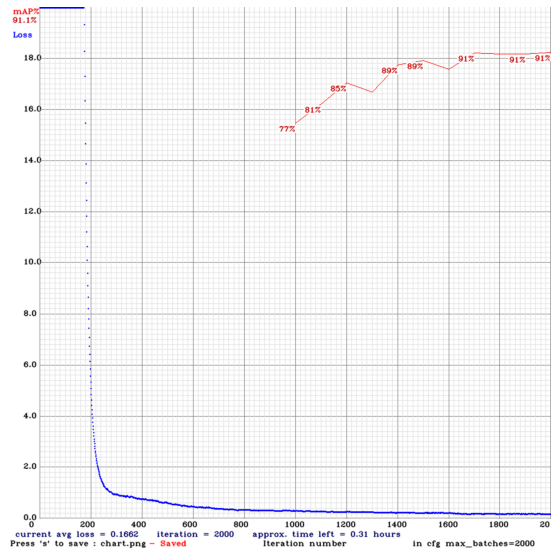


图27：loss-mAP趋势图

可以看出，经过2000次迭代，最终的loss已经接近很接近0（0.1662），mAP在1600代就开始收敛。说明训练的max\_batches还可以再往前提一提。

### 3.6.2 **检测单张图片**

使用

!sed -i 's/batch=64/batch=1/' yolov3\_custom4.cfg

!sed -i 's/subdivisions=16/subdivisions=1/' yolov3\_custom4.cfg

更改cfg文件中的配置，切换到检测模式（batch=1/subdivisons=1即单张图像一次全部喂入）

键入如下命令对单张图片进行检测（thresh=0.3即只显示概率在30%以上的预测框）

!./darknet detector test data/obj.data cfg/yolov3\_custom4.cfg /mydrive/yolov3/backup/yolov3\_custom4\_last.weights /mydrive/images/281.jpg -thresh 0.3

imShow('predictions.jpg')

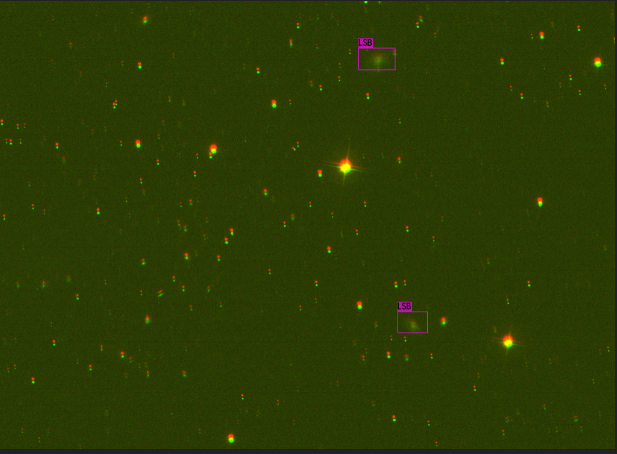




图28：单张图片识别结果

这与其在表格中的记录相对应

### 3.6.3 批量检测测试集

键入如下命令，将测试集的识别结果保存到result.txt中

!./darknet detector test data/obj.data cfg/yolov3\_custom4.cfg /mydrive/yolov3/backup/yolov3\_custom4\_final.weights -dont\_show -ext\_output < data/test.txt > result.txt

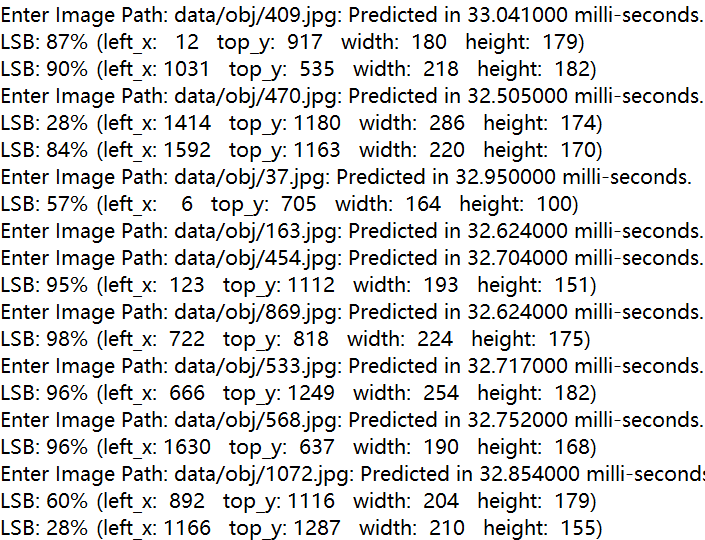


图29：批量识别结果

或使用

!./darknet detector valid data/obj.data cfg/yolov3\_custom4.cfg /mydrive/yolov3/backup/yolov3\_custom4\_final.weights -out detect\_result.txt

输出测试集全部的识别结果，每列含义依次为：图片，概率，xmin, ymin, xmax, ymax

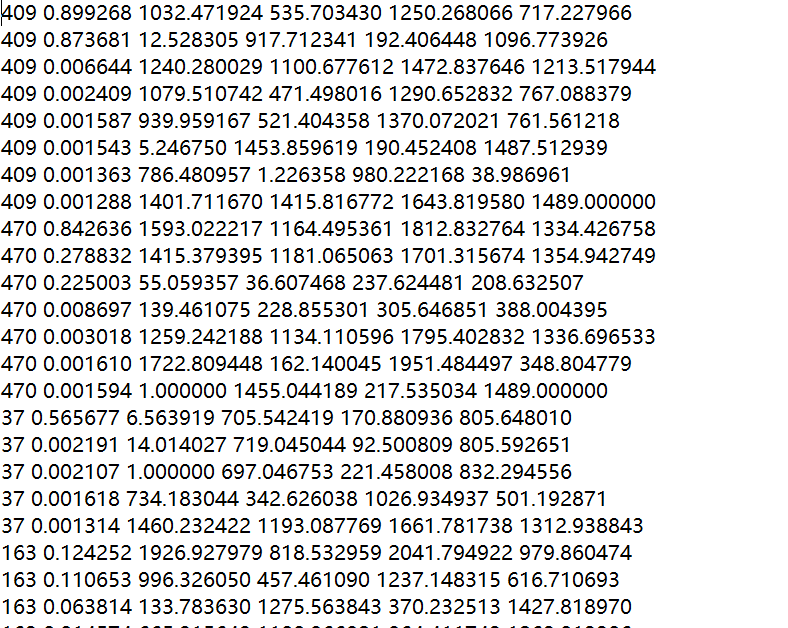


图30：全部识别结果

### 3.6.4 模型性能评估

键入

!./darknet detector recall data/obj.data cfg/yolov3\_custom4.cfg /mydrive/yolov3/backup/yolov3\_custom4\_final.weights

进行召回率的评估

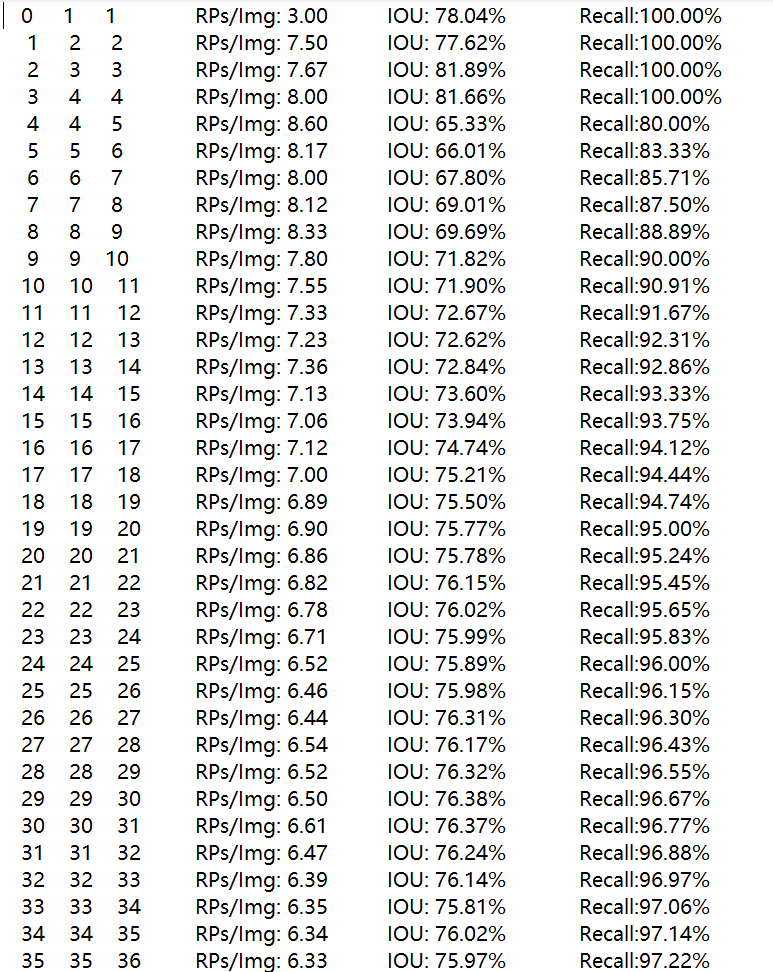


图31：召回率检测

键入

!./darknet detector map data/obj.data cfg/yolov3\_custom4.cfg /mydrive/yolov3/backup/yolov3\_custom4\_final.weights

进行mAP等参数的评估

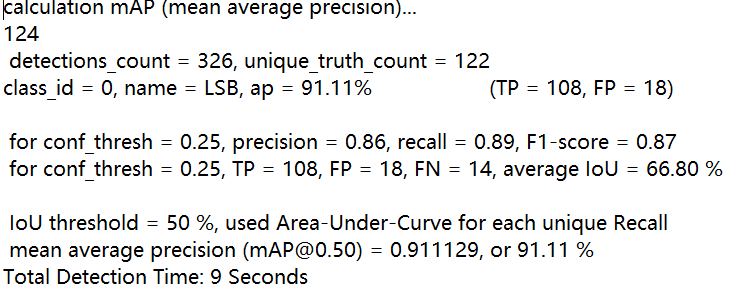


图32：评估指标结果

可以看到置信度阈值0.25，IoU阈值为0.5的情况下，精度为86%，召回率为89%，F1评分为87%，真正例108，假正例18，假负例14，平均交并比为66.80%。对于全体训练集，mAP=91.11%

### 3.6.7 模型改进

增大输入Bounding Box大小为0.1，输入resize变为更高的608\*608（保持32的整数倍）提高检测精度，并且设置random参数为1。由于这样的调整导致计算量的猛增，预计训练时间达到了24小时以上（Colab一次会话的最长时限为12小时）。于是只应用了其中某些改进，最终模型训练结果如下：

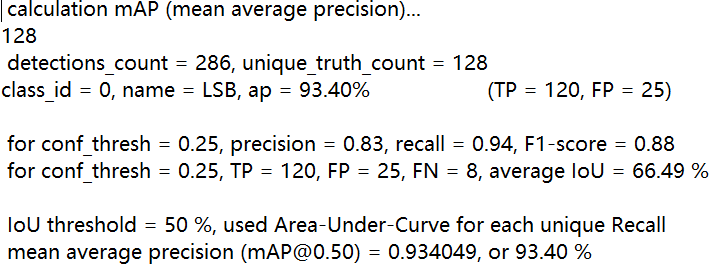
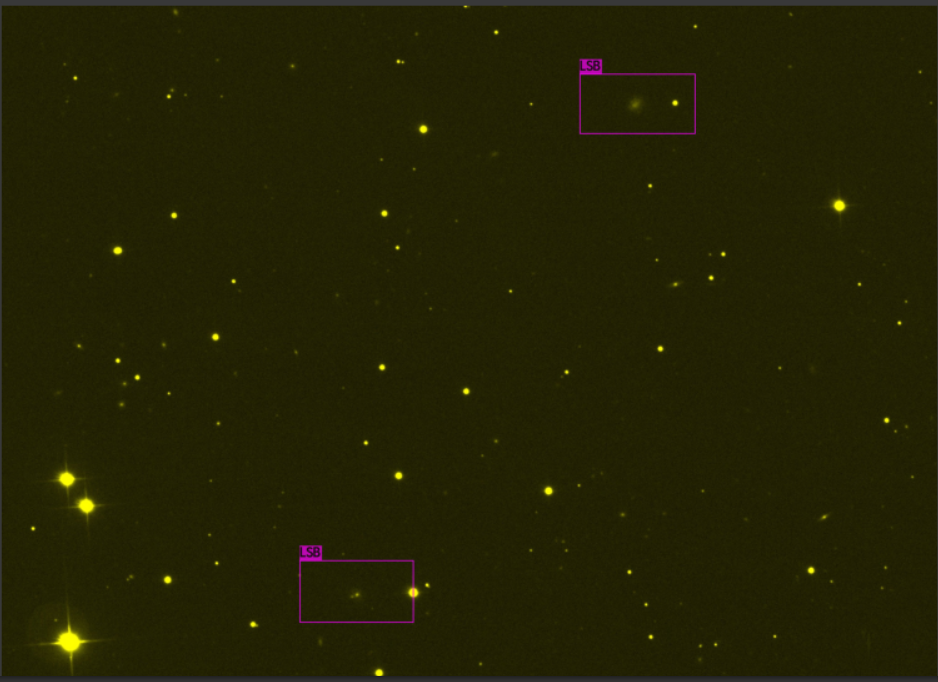


图33：最终模型性能

可以观察到，相比第一次训练，精度有所下降，但召回率和mAP均有相当的提升。对于LSB星系的识别来说，我们更看重的是召回率。故应采用最终模型。

# 四、研究结果

现在我们已经拥有训练好的模型，只要找出表中没有记载LSB坐标的图片进行预测即可。经过筛选，发现只有fpC-001035-g2-0011/ fpC-001035-r2-0011 这一组没有记录，将其喂入模型：



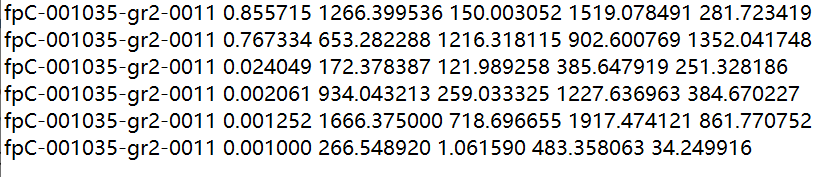


图34：LSB检测结果

显然，该幅图像的LSB星系有两个，若取检测框中心作为LSB星系的坐标，则其坐标分别为（1392.73902，215.863236），（777.94153，1284.17993）

# 五、总结与展望

对于前文所述的LSB星系的识别，出现了召回率比精确率高的情况。初步认为，是由于候选框过大，导致虽然包含LSB的概率很高，但是对于LSB的精确定位表现却不如人意。换言之，以候选框中心作为LSB位置的朴素算法还比较粗糙。囿于计算量的限制，本文也没有进行更高分辨率的输入和更合适的Bounding Box的选取。

在模型的改进方面，或许可以尝试使用最新开源的YOLOv4模型[9]，其选择更优的CSPDarkNet53作为检测模型的backbone，具有更高的输入分辨率来检测小目标；更多的层，使之具有更大的感受野，以及更多的参数，更大的模型可以同时检测不同大小的目标。在此基础上，作者添加了SPP模块，因其可以提升模型的感受野、分离更重要的上下文信息、不会导致模型推理速度的下降；与此同时，作者还采用PANet中的不同backbone级的参数汇聚方法替代FPN。

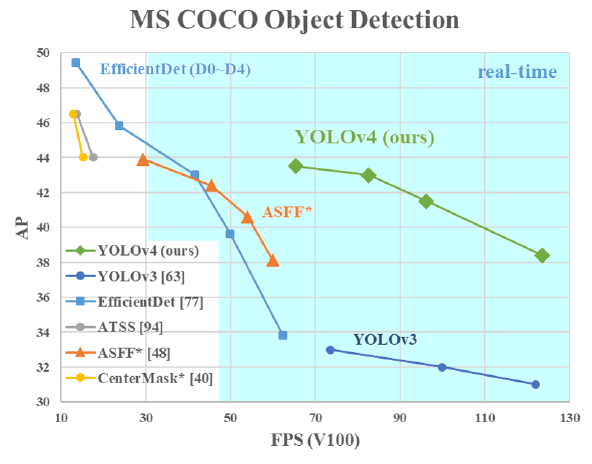


图35：YOLOv4与其他目标检测模型对比图[9]

从最终性能上看，对比当前最优目标检测器，YOLOv4 在取得与 EfficientDet 同等性能的情况下，速度是 EfficientDet 的二倍；与 YOLOv3 相比，新版本的 AP 和 FPS 分别提高了 10% 和 12%。以上种种优势，相信能在缩小BoundingBox的基础上取得更快、更精准的LSB星系识别结果。

# 六、致谢

感谢研究过程中衣振萍老师的悉心指导，StackOverflow上相关问题的解答者以及Github的开源贡献者们！

# 附录：参考文献

[1]<https://baike.baidu.com/item/%E4%BD%8E%E8%A1%A8%E9%9D%A2%E4%BA%AE%E5%BA%A6%E6%98%9F%E7%B3%BB/10853956?fr=aladdin>

[2] Vollmer, B. et al. “SIMULTANEOUS MULTI-BAND DETECTION OF LOW SURFACE BRIGHTNESS GALAXIES WITH MARKOVIAN MODELING.” The Astronomical Journal 145.2 (2013): 36. Crossref. Web.

[3] <https://blog.csdn.net/shuiyixin/article/details/82623613>

[4] Joseph Redmon and Ali Farhadi.YOLOv3: An Incremental Improvement[J].Computer Vision and Pattern Recognition (cs.CV),2018,1804.02767:.

[5]<https://blog.csdn.net/leviopku/article/details/82660381>

[6]<https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b>

[7]<https://www.jianshu.com/p/d13ae1055302>

[8] <https://github.com/AlexeyAB/darknet#how-to-train-to-detect-your-custom-objects>

[9] Alexey Bochkovskiy and Chien-Yao Wang and Hong-Yuan Mark Liao: YOLOv4: Optimal Speed and Accuracy of Object Detection[J]. Computer Vision and Pattern Recognition (cs.CV),2020