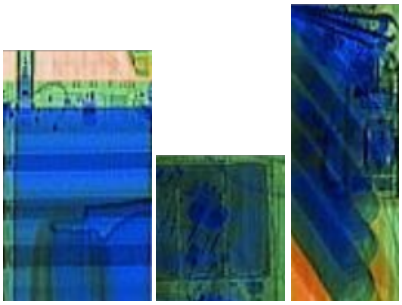
基于YOLOv3的充电宝遮挡问题检测

1 需求分析

根据项目要求，我们应该训练这样一个模型：假设有一张待检测的图片，使用我们的模型进行预测，总共检测出了X个物体，对于每个物体，至少需要输出6个值：预测框的四个坐标值、物体类别、类别置信度，于是我们的模型输出为6×X的矩阵。其中，物体类别分为带芯充电宝和不带芯充电宝。

观察数据集不难发现，对于几乎没有遮挡的充电宝，是很容易辨认的，但对于被严重遮挡的充电宝，一个经验不足的安检员就很容易漏掉，比如以下这些：



如果能针对遮挡问题改善模型，让模型在有遮挡的条件下也有很好的效果，就能帮助安检员们更好地发现（比较隐蔽的）充电宝。

经过查阅资料，我们初步将问题分成以下几个步骤：

1. 训练一个回归模型用来预测坐标，生成一些由四个坐标值组成的box，每一个box中都可能存在物体
2. 训练一个分类器，应用于步骤1产生的bounding box中，预测其中物体类别和置信度。如果一个box的置信度高于某一阈值，则认为检测到了该类物体；反之，则认为没有物体
3. 步骤2中生成了一系列包含物体的bounding box，但在其中很可能会出现同一个物体被多个box选中的情况，为了去掉多余的box，可以计算它们之间的交并比，交并比越高，说明两个框"重叠"越多。如果IOU高于某一阈值，则只留下置信度高的box，删除另一个box
4. 经过以上的步骤，应该只剩下少数box，输出它们的四个坐标、类别和置信度，作为预测结果

2 传统机器学习方法

结合课堂上的内容，我们首先尝试了传统的机器学习方法。

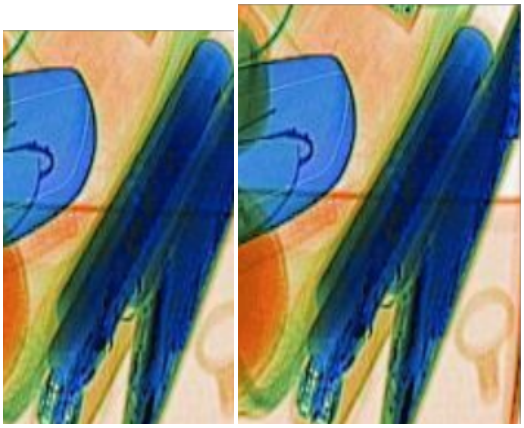
2.1 滑动窗口

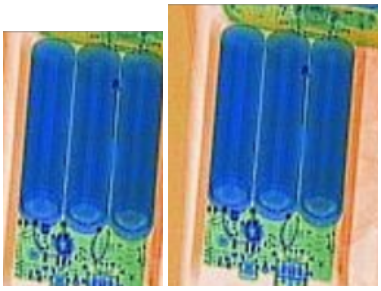
起初，我们没有想到如何通过回归模型预测box的坐标值。显然，在这里直接使用坐标数据训练一个线性回归模型是行不通的，因为充电宝的位置在图片中都是随机的。于是我们采取了一种简单的方法：设计一些固定大小的滑动窗口，这些滑动窗口按照某一步长遍历整个图像，每一次滑动都会生成一个bounding box。

只要滑动窗口的尺寸和步长设计合适，这些box中就一定会有一些包含充电宝图像，在每一个box上应用分类器，就可以得到类别和置信度的预测。

2.2 SVM分类器

根据训练数据集给出的充电宝坐标，我们提取出仅包含充电宝的小图像，将这些图片放在大小最接近的窗口中。以下图片展示了其中两个充电宝的处理效果图：





这样做加入了一些背景，但是不会太多。这样做是因为在滑动窗口检测时也是以窗口大小为图像尺寸，其中也会包含一定的背景像素，因此适当加入一点背景应该能增强分类器的适应性。

将这些图片作为训练数据，用PCA进行降维，提取出主要特征，训练SVM分类器，并在测试集上进行预测：

```
nc = N // 16 + 1
pca = PCA(n_components=nc, whiten=True).fit(X_train)
X_train = pca.transform(X_train)
X_test = pca.transform(X_test)
# 随机梯度下降更适合处理大数据集
model = SGDClassifier(loss="hinge", penalty="l2", verbose=True)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

但是训练好分类器后发现分类效果不够理想：

	precision	recall	f1-score	support
1	0.71	0.58	0.64	97
2	0.66	0.78	0.71	103
accuracy			0.68	200
macro avg	0.69	0.68	0.68	200
weighted avg	0.68	0.68	0.68	200

2.3 传统模型的不足

这种基于传统机器学习方法的模型优点是简单，容易实现，也容易调试。但经过多次调整参数、更换分类器模型后，我们发现分类效果仍然不理想，基本都和上图的结果类似，准确率维持在60%-70%之间。这可能是由于传统机器学习对于特征工程的要求较高，在图像处理中，因为我们不能手工提取特征，因此使用PCA来提取主成分，但（至少对于本数据集）这种方法并不能提取到有效的特征。

而且，就算能训练出效果较好的分类器，滑动窗口的box检测方法也会带来一些问题。通过观察数据集发现，其中充电宝的大小差别很大，最小的大约（50,70），而最大的有（300,500）。并且由于充电宝的形状一般为长方形，因此如果使用滑动窗

口，就需要设置很多不同尺寸的窗口大小，否则容易导致窗口远大于充电宝实际尺寸，包含太多背景信息，影响分类器的效果。但设置太多窗口尺寸会导致产生的窗口绝大多数都是重复而且冗余的，在全部窗口上应用分类器显然计算量非常大。

因此我们想到了使用深度学习方法，来自动学习出图像的特征。并且，使用GPU能使计算速度大幅提高。

3 深度学习模型

无论是bounding box的预测（回归问题），还是bounding box中的分类（分类问题），都是图像上的预测问题。对于图像问题，我们认为可以利用深度学习强大的特征提取能力来自动生成feature map，然后进行回归和分类，效果应该会优于传统学习方法。

因为深度神经网络的设计较为复杂，网络结构的好坏会极大程度上的影响结果的好坏。因此我们查阅资料，参考了目前表现出色的几种目标检测模型：RCNN、SSD和YOLO。目前效果较好的模型都是在日常生活的场景中检测物体，而不是在透视图中检测物体。YOLOv3对于数据集有真正的"理解能力"，对于图画中物体的检测效果也比较好，泛化能力更强，因此我们选择了YOLOv3作为基础检测模型。

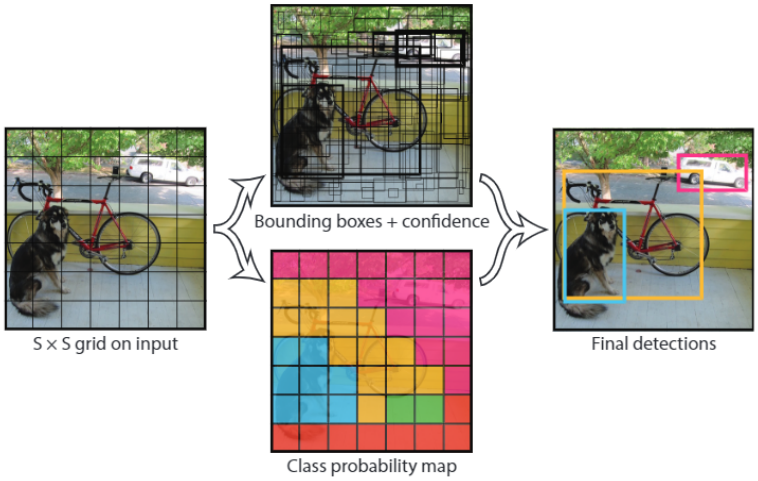
3.1 YOLO

在介绍YOLOv3之前，不得不提到YOLO。YOLOv3是在YOLO的基础上改进得到的。

YOLO（You Only Look Once）是一种基于深度神经网络的对象识别和定位算法，其最大的特点是运行速度很快，可以用于实时系统。从完成作业的角度考虑，对于较大的训练数据集，如果模型训练就要花费大量的时间，那么留给我们调整模型的时间就会被挤压。反之，如果模型本身运行速度快，我们就可以利用速度优势，更加快速的获得测试结果、迭代模型。从模型应用场景的角度考虑，速度快这个特点对于安检任务可以说是非常必要的了，毕竟这个模型的价值就在于保证质量的同时提高安检效率。不能让所有旅客安检完后还要等待模型检测出结果，实时系统能给安检员提供更多的帮助。

在YOLO之前，目标检测的主流方法都和我们的初始想法类似，分两步走，先找到物体的位置，再区分是什么物体。在RCNN中，首先用一个网络生成一些候选区域，然后做边框回归，调整候选框使其更接近ground truth，然后用另一个网络预测分类结果。

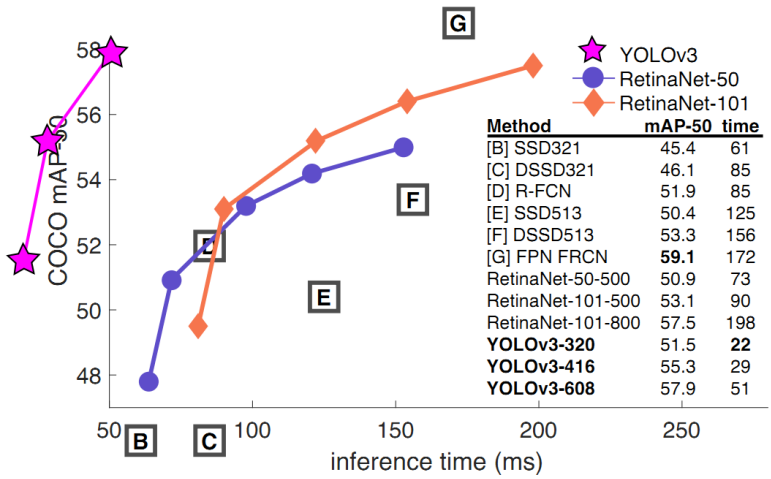
YOLO创造性的提出将bounding box的预测（回归问题）和物体的分类（分类问题）同时进行的方法，用一个深度学习网络就能同时得到位置信息和类别信息。在YOLO的设计中并不是不存在候选区域，而是省略了生成候选区域的步骤，转而用7×7的网格分割整个图像，将这些网格作为候选区域，再对这些区域做边框回归，生成bounding box。



但是正是由于网格的思想，导致YOLO对于小对象检测效果不太好（尤其是一些聚集在一起的小对象），因为每个网格中只预测两个bounding box。而在安检场景中，确实可能出现两个充电宝紧邻在一起的情况。对于YOLO的这方面不足，作者在后续的YOLOv3中做了改进。

3.2 YOLOv3

YOLOv3是在YOLOv2的基础上加入一些小Tips产生的改良版本。在论文YOLOv3: An Incremental Improvement中，作者通过比较同一mAP值下各种目标检测算法的表现，重点强调了YOLOv3的速度：



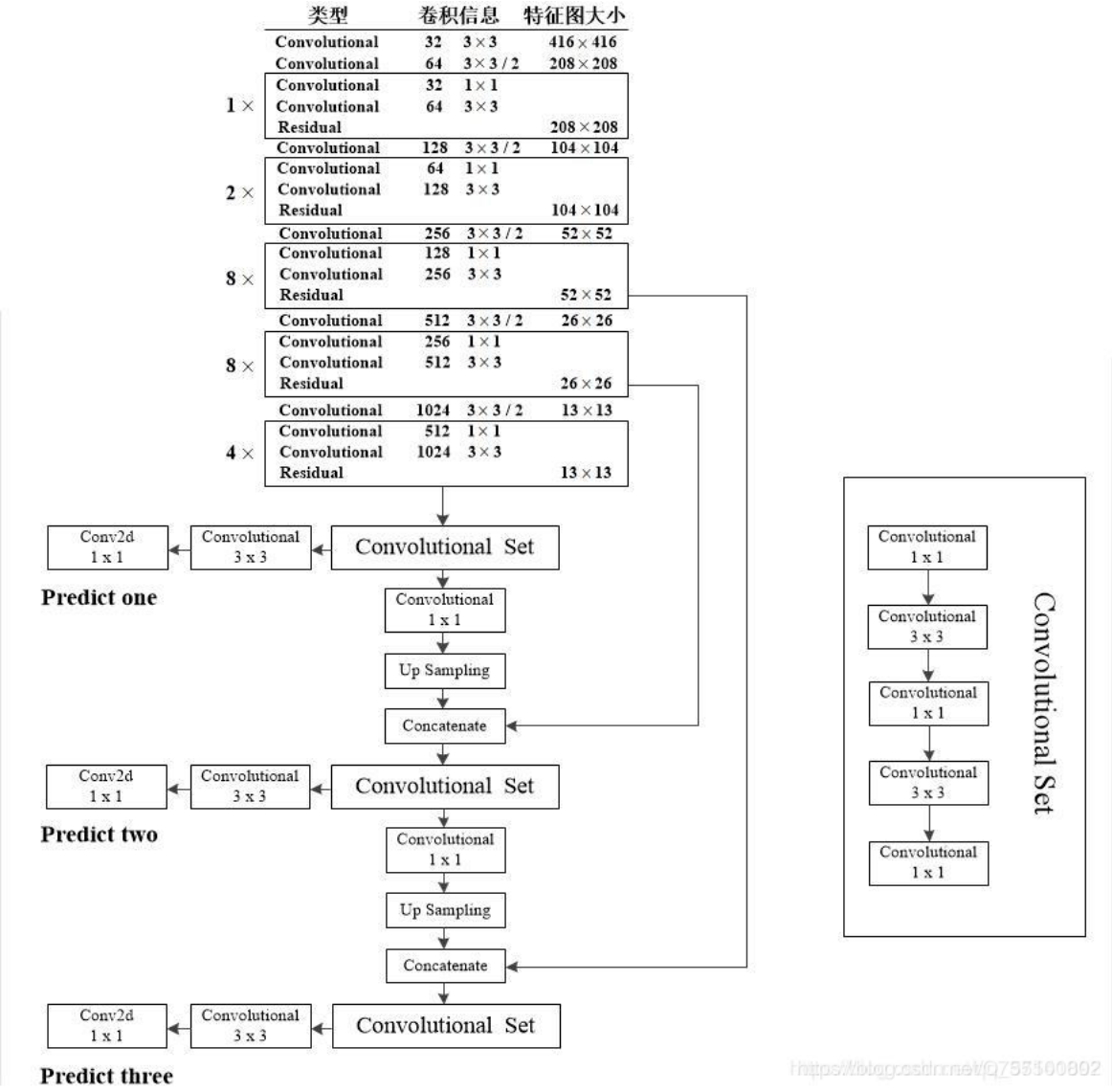
Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

```
At 320×320 YOLOv3 runs in 22 ms at 28.2 mAP, as accurate as SSD but three times faster.

It achieves 57.9 AP50 in 51 ms on a Titan X, compared to 57.5 AP50 in 198 ms by RetinaNet,
similar perfor-mance but 3.8× faster.
```

相较于YOLO，YOLOv3在保证速度的前提下，又提高了精度。YOLOv3主要的改进有：调整了网络结构，形成更深的网络层次；利用多尺度特征进行对象检测，提升了mAP及小物体检测效果；对象分类用Logistic取代了softmax。

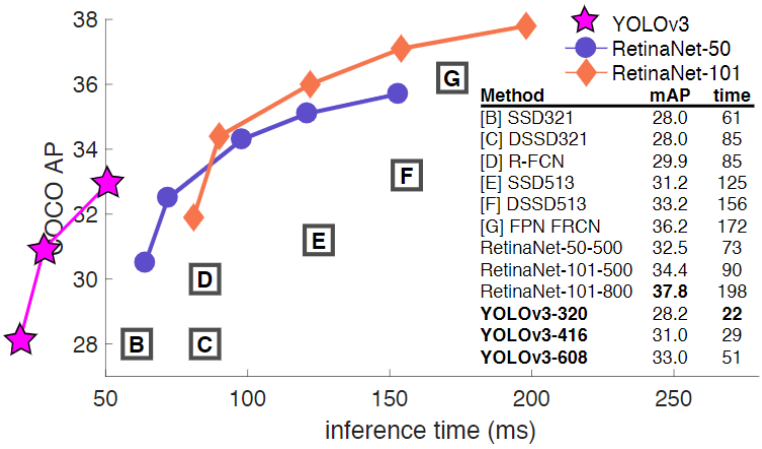
YOLOv3最重要的内容就是一个庞大而丰富的深度卷积神经网络模型，它一共有53个全连接卷积层，所以作者又将该项目称为Darknet-53，但实际上卷积层不止53层，因为特征提取也用到了大量的卷积核。它借鉴了残差网络residual network的做法，在一些层之间设置了快捷链路（shortcut connections）。下图是YOLOv3的网络结构：



网络的工作过程如下：

1. 一张416*416大小的图片输入，会经过很多层的深度卷积（图片中略过了层数）一直降维到52,26和13。
2. 在52,26和13维分别有三个全卷积特征提取器，对应的是右边的Convolutional Set，这就是特征提取器的内部卷积核结构，1×1的卷积核用于降维，3×3的卷积核用于提取特征，多个卷积核交错达到目的。每个全卷积特征层是有连接的，在图中为Concatenate标志，意味着当前特征层的输入有来自于上一层的输出的一部分。每个特征层都有一个输出Predict，即预测结果，最后根据置信度大小对结果进行回归，得到最终的预测结果。

下图表明，YOLOv3在对边框的准确性要求更高的mAP计算标准中，表现稍差。但在本项目中，我们认为边框不需要过于精准，我们的只需要帮助安检员更好的发现和定位充电宝即可。



Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

4 测试结果

为了计算最终成绩的近似值，我们手工将验证集数据分出了3个遮挡等级，以下图片分别是1级（少部分遮挡）、2级（大部分遮挡）和3级（完全遮挡）：

使用我们分出的测试集，最终mAP的计算结果：

```
In [66]: core_ap
Out[66]: 0.9851851851851852

In [67]: coreless_ap
Out[67]: 0.9711864406779661

In [68]: mAP=(core_ap+coreless_ap)/2

In [69]: print(mAP)
0.9781858129315757
```

5 分工情况

郭韵：编写测试文件、计算mAP

林雨青：模型训练、调参

王齐：模型训练、调参

范文杰：查阅资料、文档撰写

孙心怡：查阅资料、文档撰写