目标检测采用mobilenet\_v1网络和SSD检测框架，数据集数目从0000-0154，共155张图片，首先是生成tf\_record，调用object\_detection/datatools目录下的create\_pet\_tf\_record函数，其中这个函数需要删除mask以及face\_only部分，以及修改查找文件的路径部分，中间出现了很多问题，比如对生成tf\_record代码中的[‘filename’]、[‘object’],这些属性还不够熟悉，它们能大大提高程序运行效率；以及修改config文件时对路径的设置，还有tinymind云平台的使用上。

下面介绍一下整个流程，生成tf\_record后，下载预训练模型，修改config文件，如：num\_class（检测任务中类别数）、num\_examples(验证集的数据数，本次产生的训练集、验证集数据是按照7：3分配的，计算即可)、PATH\_TO\_BE\_CONFIGURED：/data/qq-25646217/detection (替换成tinymind上的数据集路径）、num\_steps：0、max\_evals：1、eval\_input\_reader：false。启动前需按照object\_detection的installation的教程进行必要的操作，如编译proto文件，将object\_detection/protos/下的proto文件编译成python文件。完成上述步骤后，就可以运行程序了。

对object detection过程的理解（共四个部分：候选区域、特征提取、区域分类、边框回归）：

**R-CNN**：首先经过selective search：在颜色、纹理等空间中映射特征，用到快速SIFT算法，再计算不同区域中的特征距离，合并相似区域（小区域优先合并，外接矩形时需选择与合并矩形重合度高的矩形框）；特征提取：CNN；区域分类：SVM；边框回归：拟合出x,y坐标以及宽、高。

缺点其中Selective search生成阶段很慢，并且每个候选区域都要CNN做一次inference，每个步骤单独训练，微调困难。

**Fast R-CNN**：相比上一个，进步点：将边框回归纳入到网络结构，候选区域的操作放在了特征提取中，如在卷积神经网络中某层的feature map进行截取，然后通过ROI pooling把不同大小不同长宽比的特征缩放到同样大小，再输入到网络中进行计算，回归。

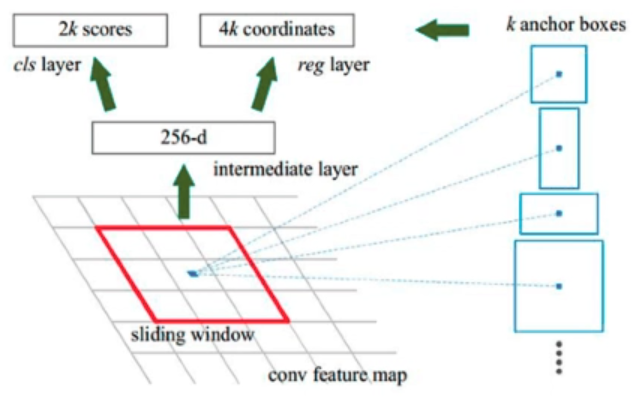
缺点：selective search还是很慢

**Faster R-CNN**:相比上一个，进步点：用神经网络来替代selective search,候选区域采用anchor的选择机制。步骤：先把数据输入到通过image net预训练的CNN模型然后抽取feature map，将这个特征向量输入到 Regon Propsal Network，这个网络会给出两个信息（Bounding-box Regression loss和Classification loss），一个是建议的候选框（包括我们希望去detect的这些物体的框），会根据特征图给出建议的bounding-box的位置；另一个是判断生成的这个区域是不是物体（前景或者背景）。然后把确定是object的框加入到ROI pooling、multitask regression等后续运算。

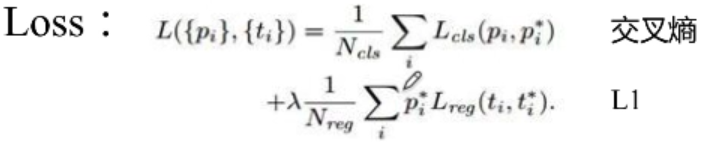
Anchor机制：锚点，即某一个框的中心点，相比与用左上和右下点确定候选框，锚点显得更有规律，在于它的中心点不变，只改变高、宽。

**RPN实现细节**：VGG网络的conv5-3的feature map(h,w,521)作为输入,经过3X3conv/relu（相当于作3X3的全连接并滑窗），再把结果同时作为两个并行branch的输入，分别经过class layer（1X1conv/sigmoid） 和regression layer(1X1conv/linear)。前者判断是前景还是背景（输出2k个scores，每个框有前景的置信度和背景的置信度）；后者输出一组坐标，代表在当前这个3X3的小区域里面输出的一些方块的坐标位置(输出4k个bonding-box)；其中k是指：以当前的3X3的卷积核的中心点去生成k个大小不同的框，论文中共生成9个，即长度分别为1、2、3，正方形、竖着、横着的框。

Ground truth：对每个anchor进行标注。每个anchor对应原图上的某一个点，anchor boxes就会对应到原图中的某一个区域，这个区域里面如果有ground truth包括的真值的块，就对ground truth的块和anchor boxes进行IoU的计算。如果IoU最大或者>threshold(0.7)则为正样本（指的是生成class layer分数的时候，positive是1，background是0），并把ground truth坐标作为这个anchor回归的ground truth输入给它；如果IoU<0.3，则作为负样本，不会生成坐标；IoU在0.3-0.7之间的不用于训练。训练时需尽量做到均匀采样。



损失：



**Non-max suppression**:抑制较小置信度的bounding box

A:将所有框按置信度排序，选择最大的一个

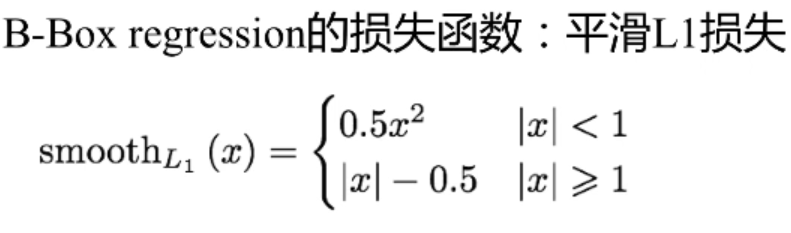
B:剩余所有框中删除与当前选中框IoU>threshold的

C:将当前选择框记录，重复A-C，直到

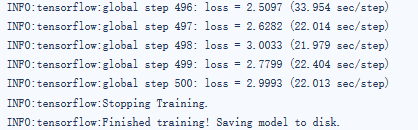
-没有剩余，或者

-达到最大值

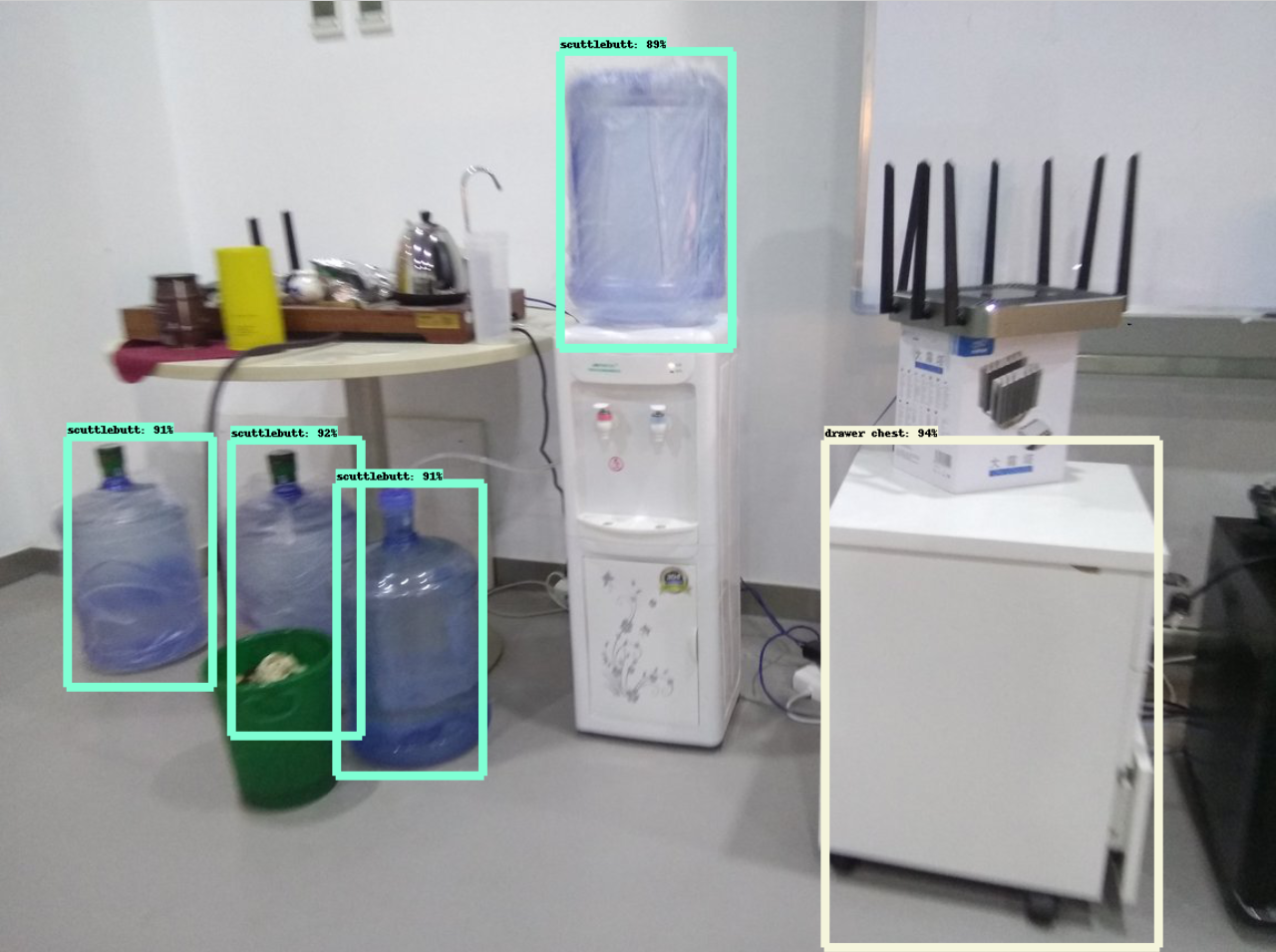
Soft NMS:步骤B的“删除”改为置信度乘以（1-IoU）



运行结果：



通过冻结权重、寄存图，进行inference，得到：



新的思路：（回归）

**YOLO**：划分为两个部分（特征提取、边框回归），思路类似faster rcnn，但是yolo没有划分区域（region）这部分操作，而是直接利用得到的特征回归出来。它通过一个很普通的卷积神经网络得到一个1X1X4096的向量，然后再次使用全连接回归出来一个7X7X30的特征向量，7X7的每一个分格在原图的感受野对应一个区域，30划分为5、5、20，5包含4个坐标和一个标量（置信度），另一个5是指：7X7某一个像素点覆盖范围的另外一个框的信息，20是pascal上的20个分类。有了这些信息可以得出它在7X7特征图的像素所代表的原图上的感受野，画出的两个小方块和他们的前景、背景信息，如果是前景，那么这两个小方块要属于同一个类别。同时，把Ground truth也人为地划分为这么多个块，用这些块去回归最后得到的feature map,通过大量的训练让模型认识到自己应该这样去做。

YOLO对整张图就只做了一次常规的inference运算，和分类比起来所需要的附加的运算并不多，甚至比一些复杂的、设计不太良好的分类网络还要更快一些。但是问题也很明显，如果在同一个小块里面包含两个不同类别的物体，是处理不了的，如果同一个小块包含的都是同类物体，但是物体数目多也不能处理。

SSD:与YOLO相比,不是每个分块预测一个类两个框，吸收了faster rcnn的anchor思想，生成不同大小的框，框的长宽比根据先验知识。在多尺度feature map上提取特征，将在不同层次的anchor提取的特征拼成一个很大的特征向量，再用这个特征向量像YOLO一样去回归。