

天津大学

本科生毕业论文



题目：基于自制数据集与改进的 GIOU 损失函数的不良广告审查系统

学 院 智能与计算学部

专 业 计算机科学与技术

年 级 2017 级

姓 名 吴熹之

学 号 3017216111

指导教师 庄志强

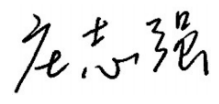
独创性声明

本人声明：所呈交的毕业设计（论文），是本人在指导教师指导下，进行研究工作所取得的成果。除文中已经注明引用的内容外，本毕业设计（论文）中不包含任何他人已经发表或撰写过的研究成果。对本毕业设计（论文）所涉及的研究工作做出贡献的其他个人和集体，均已在论文中作了明确的说明。本毕业设计（论文）原创性声明的法律责任由本人承担。

论文作者签名：

2021 年 6 月 17 日

本人声明：本毕业设计（论文）是本人指导学生完成的研究成果，已经审阅过论文的全部内容。

论文指导教师签名： 

2021 年 6 月 17 日

摘要

实时目标检测是计算机视觉的主要研究方向之一。这篇研究致力于基于深度学习，改进实时广告竞价系统中的广告审查环节，并采用一阶段实时目标检测技术^[1]，回避传统目标检测技术和两阶段目标检测技术在准确度或者速度上不足之处。

本篇文章构建了一种基于 YOLO 框架和 Darknet-53 骨干网络的实时广告图片审查系统。YOLO 算法将目标检测作为回归问题求解，速度快且通用性强，非常适合于分秒必争的实时广告竞价系统。同时本研究还提出了一种基于 GIoU 的损失函数来替换原模型的均方损失函数，在牺牲部分全类别平均精度（Mean Average Precision, mAP）与时间的情况下，增加了模型的定位精确度。

由于公司方提供的广告数据质量参差不齐，存在大量噪音。出于商业机密的考虑，互联网平台上的开源数据集不适用于本研究的广告审查任务。出于以上原因，本研究自行制作了 14912 张广告数据集用于训练和验证，这些数据集的数据来源中，有 100 张通过脚本从谷歌图片中批量下载，有 1200 张来自谷歌开源数据集 OPEN IMAGE DATASET V6，剩下 13612 张从公司方提供的 725570 张广告链接中筛选得到。

从广告审查模型的训练结果来看，效果是很可观的。广告审查模型平均审查一张图片耗时 9.15 微秒，在 17 种待检测物体中，在广告数据集多分类目标检测上的全类别平均精度（Mean Average Precision, mAP）达到 72.75%，不良广告的检测准确率达到了 90.15%。同时通过改进给予 GIoU 的损失函数，提高了定位准确度。如果应用在实时广告竞价系统中的话，可以提高审查环节的效率，节省审查任务的开支与预算。

关键词： 目标检测，YOLO 算法，深度学习，不良图片识别

ABSTRACT

Real-time object detection is one of the main research directions of computer vision. This research is dedicated to improving the advertising review link in the real-time advertising bidding system based on deep learning, and adopts one-stage real-time target detection technology to avoid the lack of accuracy or speed of traditional target detection technology and two-stage target detection technology Place.

This article builds a real-time advertising image censoring system based on the YOLO framework and the Darknet-53 backbone network. The YOLO algorithm solves object detection as a regression problem. It is fast and versatile, and is very suitable for a real-time advertising bidding system where every second counts.

Due to the uneven quality of the advertising data provided by the company, there is a lot of noise. For the sake of commercial secrets, the open source data set on the Internet platform is not suitable for my advertising review task, and there is no data set for Chinese advertising. Therefore, this research produced 14912 advertising data sets for training and verification. Among the data sources of these data sets, 100 were downloaded from Google images in batches through scripts, and 1200 were from Google's open source data set OPEN IMAGE DATASET V6. The remaining 13,612 advertisements were selected from the 725570 advertisement links provided by the company.

Judging from the training results of the advertising review model, the result is considerable. Under GPU acceleration, the advertising review model takes an average of 40.31 milliseconds to review an image. Among the 17 objects to be detected, the full-category average precision (mAP) on the multi-category target detection of the advertising data set reaches 72.75%. At the same time, by improving the GIoU loss function, the localization accuracy is improved. The detection accuracy rate of bad advertisements reached 90.15%. If applied in a real-time advertising bidding system, the efficiency of the review process can be improved, and the expenditure and budget of the review task can be saved.

KEY WORDS: Object Detection, YOLO Algorithm, Deep Learning, Image Censorship

目 录

第一章 绪论	1
1.1 国内外研究现状.....	1
1.2 主要研究内容.....	4
第二章 理论研究.....	6
2.1 目标检测模型结构.....	6
2.2 边界框.....	7
2.3 损失函数.....	9
第三章 实验与分析.....	10
3.1 实验目的.....	10
3.2 实验数据.....	10
3.2.1 数据描述.....	10
3.2.2 数据获取.....	11
3.2.3 数据清理.....	12
3.2.4 数据预处理.....	14
3.2.5 数据集制作.....	14
3.3 训练模型.....	20
3.4 验证模型的检测准确度.....	26
3.5 基于 GIoU 的定位损失函数改进.....	27
3.5.1 GIoU 与边界框回归	27
3.5.2 设计 GIoU 损失函数算法.....	28
3.5.3 训练采用 GIoU 损失函数算法的 YOLOv3 模型.....	32
3.5.4 验证模型的检测准确度.....	34
3.5.5 检验使用 GIoU 的模型的定位识别准确度.....	35
3.6 总结与展望.....	35
参考文献.....	37
附 录.....	38
致 谢.....	42

第一章 绪论

传统互联网广告一般包含三方,分别是广告主、广告代理商以及互联网媒体。而在实时竞价广告交易模式中有四个主体,包括广告主、需求方平台、广告交易平台以及互联网媒体四个主体。广告主将自己的广告需求投放到需求方平台上,互联网媒体将自己的广告流量资源放到交易平台上,需求方平台通过与广告交易平台进行技术对接完成竞价购买^[2]。当用户打开一个软件时,通常有一个进入界面的广告展示机会,出价高者会获得这个广告展示机会,并被目标用户看到。从开始竞价到完成投放,这一系列过程只有 100 毫秒,全部依托机器完成。

相应的实时竞价广告交易平台也对广告流量资源的审查提出了特定的要求。不同于以往的不良图片过滤系统专注于对硬色情内容的审查过滤,实时竞价广告系统提出了更多的要求:希望展示的广告不要出现低俗色情内容,包括硬色情、软色情以及擦边球内容;不要出现赌博借贷类金融犯罪;不要出现网页游戏网络小说广告等低质量内容;不要出现诱导用户点击内容,这会误导广告交易平台对于广告点击率以及用户喜好的判断;根据作者观察研究,审查不通过图片主要分为以下几类:网页游戏广告、成人内容擦边球广告、不良社交平台广告、金融诈骗(赌博借贷)类广告、诱导用户点击的广告。而这个广告可以根据其所携带的特定物品进行识别,比如说武器(枪支刀剑)、身体部位、关键字、赌博道具(扑克牌、麻将)、红包、金币、中央白色弹窗、关闭按钮。这些审查不通过的广告中一共可以找到 17 种可供目标检测系统识别的物体,这些物体在外形、颜色等方面都具有信息量大、有区分性的和独立的特征,可以为特征提取提供深层次的特征。

1.1 国内外研究现状

对于图片过滤技术国内外研究者做出了大量研究。因为不良图片中包含大量皮肤暴露的信息,传统的不良图片过滤基于感兴趣区域(Region of Interest, RoI)并对皮肤进行特征提取,然后采用分类算法诸如支持向量机(SVM)、决策树、K 近邻算法(KNN)等来进行分类,以此来决定是否不良图片。其中,对感兴趣区域的特征提取通常是先对目标图片中的像素进行遍历来判断这是否是皮肤的像素。Moheb 等人通过 YCbCr 颜色空间,将色度分开存储,来检测皮肤暴露程度^[3]。

但这类方法存在的问题是,不同人种肤色差异巨大,而且同一人种在不同的光照条件下差异也很大。如果在背景中出现了类似人类皮肤的物体,那么就有可能出现误判。而且在我的任务中,不只是色情内容需要被审查出来,还有页游广告、诈骗广告、广告弹窗等需要被审查出来,这些广告如果采用传统的基于感兴趣区域的特征提取,将很难实现。比如说广告弹窗大小不一,样式不一,很难找到共同点进行提取。只有通过目标检测对其中的某些要素进行识别,才能做到过滤这些不良广告的目的。

随着深度学习的兴起,基于深度学习的目标检测技术得到了极大的发展。有两种主要的目标检测器,一种是两阶段检测器,代表是 Faster R-CNN^[4]。二阶段检测器有更高的定位和识别精准度,而一阶段检测器有更高的识别速度。在 2017 年, T.-Y. Lin 等人在 Faster R-CNN 的基础上提出特征金字塔网络(FPN)^[5],在 FPN 之前,大多数基于深度学习的检测器仅运行检测在网络的顶层。虽然更深层次的 CNN 层有助于分类,但这对于定位对象有负面影响。由于一个 CNN 的前身自然形成了一个特征金字塔传播,FPN 可以检测各种大小尺度的物体。二阶段检测器的一般流程是这样的:第一阶段(RPN, Region Proposal Network)里产生候选边界框。在第二阶段,用感兴趣区域池化层(RoI Pooling Layer)从多个候选边界框中提取特征,然后进行分类和边界框回归。图 1-1 展示了二阶段检测器的基本结构。除了二阶段检测器外,另一种是一阶段检测器,代表是 YOLO^[6]和 SSD^[7]。SSD 与先前的检测器区别在于:先前的检测器只在顶层进行目标检测,而 SSD 在网络的不同层上检测不同尺度的目标。在 2017 年, T.-Y. Lin 等人发现了“一步检测”准确率不如“两步检测”的原因,并提出了 RetinaNet^[8],并提出了“焦点损失”损失函数,在提供很高速度的同时保证了精准度。同年, Redmon 提出了 YOLO9000^[9],相对于初始 YOLO 在定位和高分辨率分类器上做了很大的改进。一般情况下,一阶段检测器直接从输入图片中回归产生预测框,省去了感兴趣区域这一步骤,所以一阶段检测器时效性更好,更加适合实时性任务。本篇研究所提出的图片审查系统就是基于 YOLOv3 算法的一阶段检测器。图 1-2 展示了一阶段检测器的基本结构。

国内有不少关于不良图片过滤系统的研究。江英^[10]构建了一种基于卷积神经网络的 VGG19 与 ResNet50 图像分类的图片过滤系统,并横向比较了二者在训练准确度、损失函数和验证准确度上的优劣。首先其通过从海外色情网站截图的方法获得了 12000 张色情图片,并将这些图片作为一个类别放入 COCO2017 的数据集作为一类图片进行,一共 91 类图片,作为输入进入模型训练。本人认为这个任务实际上是一个二分类任务,这样的图片过滤系统只关心是否是不良图片,剩下的 90 个类别并没有存在的意义,作为一个二分类任务反而可以提高过滤系

统的效率。杨源^[11]提出了一种基于滑动窗口和 ResNet50 的细粒度图像分类的图片过滤系统。首先通过大小为 224×224 滑动窗口将原始图片碎片化，并对每个窗口内的图像输入 ResNet 进行分类。并在分类时按不同的身体部位进行训练，并在网络顶层加入了一个 softmax 层来实现。这么做可以提高精度，一定程度上增强了对于小物体的分类能力。

但是图片分类的方法在某些任务中并不适用。因为在本篇研究的任务中，需要识别处于图片的角落而且外形非常小的物体，像关闭按钮、金币图标等等。如果采用图像分类的话，会直接将关闭按钮这些小物体的特征忽略掉，即便采用了滑动窗口的方法，因为滑动窗口也具有一定大小，如果将滑动窗口的大小限制地过小的话，会成倍地增加任务量并降低系统的时效性。

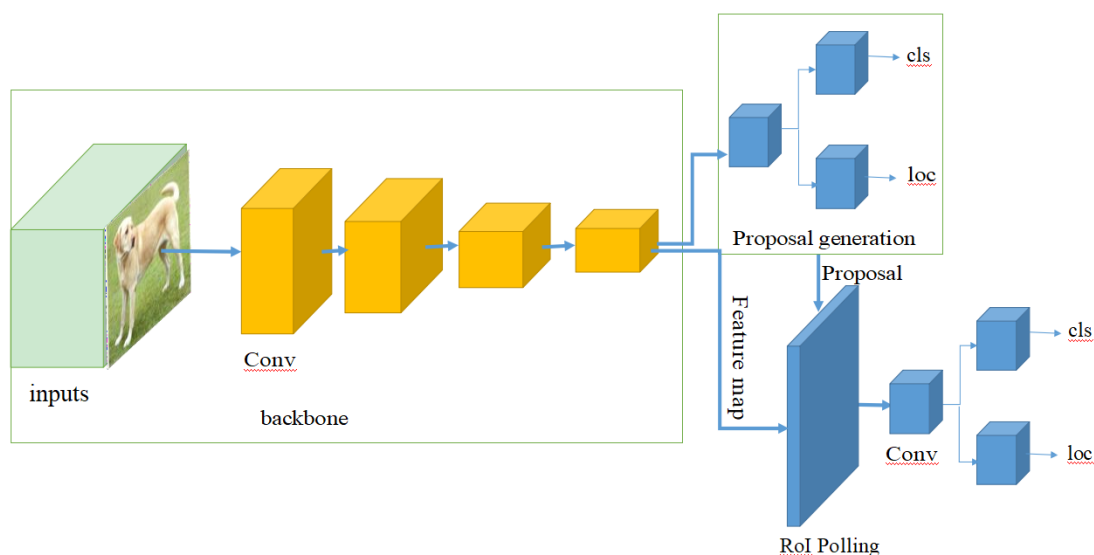


图 1-1 二阶段检测器的一般结构

王俊境^[12]提出了一种基于 R-FCN 目标检测的不良图片过滤系统，R-FCN 是一种二阶段目标检测网络，基于对 Faster R-CNN 的改进。与 Faster R-CNN 相比，R-FCN 将感兴趣区域池化层提前到 Conv4 和 Conv5 卷积层之间。王俊境在这个模型的基础上提出了改进方案，对感兴趣池化层的特征图进行补偿、对位置得分特征图进行了增强，使网络的性能提升、训练参数减少从而更适合识别不良图片的任务。不过这种二阶段目标检测模型的改进仍然缺乏时效性，在速度上表现不如一阶段目标检测模型。而且在类别增加后这一缺陷将变得尤为突出，因为其只包含两类图片，即硬色情内容图片和正常图片。

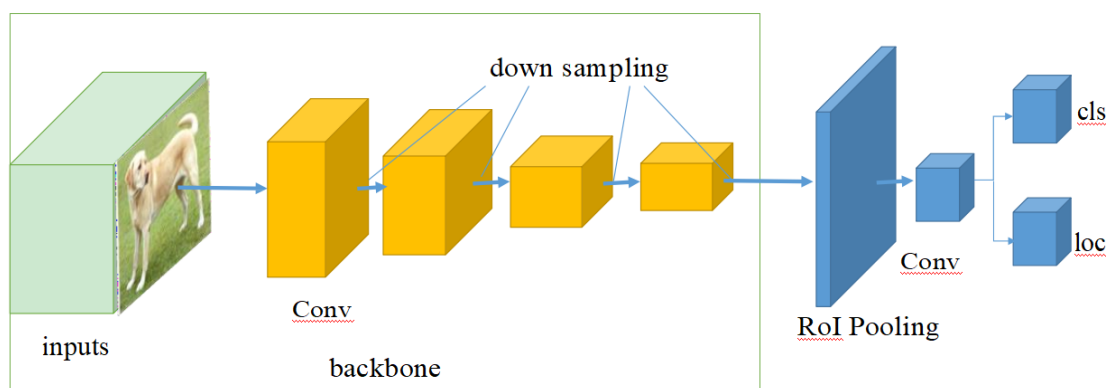


图 1-2 一阶段检测器的一般结构

而在本任务中并没有硬色情图片，只能认定为擦边球图片，也没有明显的特征，最多只到露出肚脐和肩膀，或者罩杯轮廓相对明显的地步。而且除了这些还有很多其他类的物体需要被检测，比如说枪支、刀剑等等共 17 类物体。所以如果采用二阶段检测的方法，无法达到时效性方面的要求。

1.2 主要研究内容

本篇研究基于 YOLOv3 骨干网络，实现了对 17 种物体的检测，完成了广告审查的任务。

本篇研究完成了以下任务：

(1) 首先制作了 14912 张数据即用于训练、测试和验证。

其中有 1200 张下载于 OPEN IMAGE DATASET V6，还有 100 张是直接下载于谷歌图片，其他 13612 张广告来自于公司方提供的数据，通过运行 Labellmg 工具的源代码进行数据集的标注。

(2) 进行数据清理、数据预处理。

首先需要清除数据中不能用作训练的数据。因为数据集的格式与 YOLO 接受的数据集格式不同，编写了一段 Python 代码批量转换这 1200 张数据集的格式，使其能够被 YOLO 所接受。编写了一个 Javascript 脚本向谷歌请求搜索页面所有图片的 url 链接，随后通过一段 python 代码批量下载这些图片

(3) 进行训练和调参

通过复现源码和调参摸索，最终训练得到一个表现不错的模型。达到了 72.75% 的 mAP 与 0.8593 的平均损失。

(4) 进行广告检测的验证

在 680 张“欢迎”广告和另外 680 张“不欢迎”广告验证集上进行检测，验证精确率 89.47%；验证准确率为 90.15%；验证召回率为 96.03%；验证 F1 分数

为 92.63%；与现有的不良图片过滤系统相比，本实验的不良广告审查系统识别速度更快，平均识别一张广告仅需 9 微秒；实验证明，所提出的方法可以解决目前广告审查系统存在的不足，并加以改进。

第二章 理论研究

2.1 目标检测模型结构

本实验采用 YOLOv3 模型架构，这是一个基于 Darknet-53^[13]骨干网络的深度学习神经网络，但在这里 Darknet-53 移除了最后面的全连接层，为了方便称呼仍然称为 Darknet-53，结构如下图 2-1 所示。

其中，DBL 模块是一个卷积层后面跟了一个批归一化再加上一个带泄露线性整流函数；Res Unit 是由两个 DBL 模块构成的残差网络；Resn 由一个零填充加上一个 DBL 模块再加上 n 个 Res Unit 组成；Concat 代表张量拼接。将 Darknet 不同层拼接到一起，拼接会扩张张量的维度。

Darknet-53 包含 53 个卷积层，是一个全卷积网络（FCN）。整个目标检测网络包含 106 层，其中大量使用了残差网络来跳层连接，防止训练中因为层数过深而效率降低，表现下降。同时为了减弱池化带来的负面影响，比如减少参数和特征这些负面影响，在网络中直接用卷积网络过滤器的步长来实现降采样。具体方法是将步长设置为 2，这样可以避免池化带来的负面影响。

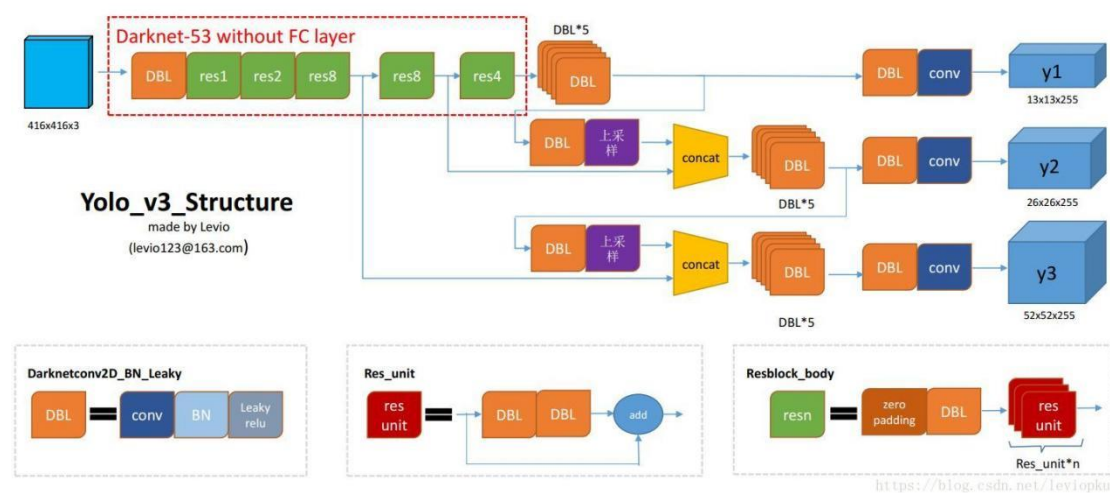


图 2-1 YOLOv3 结构图^[16]

为了加强算法对小目标检测的精确度，YOLOv3 有多种检测尺度，一共有 3 条预测路线，最终 y1、y2、y3 分别为 13×13×255、26×26×255、52×52×255，将多个尺度的特征图融合在一起，这样检测更准确。而且三条预测路线也采用的是全卷积结构，其中最后一个卷积核的个数是 66 个，因为在本实验中一共有 17 个类别： $3 \times (17 + 4 + 1) = 66$ ，3 表示一个单元格中包含 3 个边界框，4 表示框的 4 个坐标信息，1 表示物体是否存在。

2.2 边界框

输入图片被分隔成了多个单元格以及各个单元格对应的 3 个边界框。在这里对图像进行三个尺度上的检测，分别是 13×13 、 26×26 、 52×52 ，这样可以识别各个尺度上的物体，分别对应大型物体、中等大小物体、小物体。每个边界框都会预测：（1）边界框的位置（一共 4 个值，中心坐标 b_x 和 b_y ，框的高度 b_h 和宽度 b_w ）（见图 2-2）；（2）这个框内是否有物体，称为 Objectness，用一个布尔值表示，1 代表有物体，2 代表没有物体；（3）N 个类别，即框内的物体属于 N 个类别中的哪个类别。

输入图片被分隔成了多个单元格以及各个单元格对应的 3 个边界框。在这里对图像进行三个尺度上的检测，分别是 13×13 、 26×26 、 52×52 ，这样可以识别各个尺度上的物体，分别对应大型物体、中等大小物体、小物体。每个边界框都会预测：（1）边界框的位置（一共 4 个值，中心坐标 b_x 和 b_y ，框的高度 b_h 和宽度 b_w ）（见图 2-2）；（2）这个框内是否有物体，称为 Objectness，用一个布尔值表示，1 代表有物体，2 代表没有物体；（3）N 个类别，即框内的物体属于 N 个类别中的哪个类别。

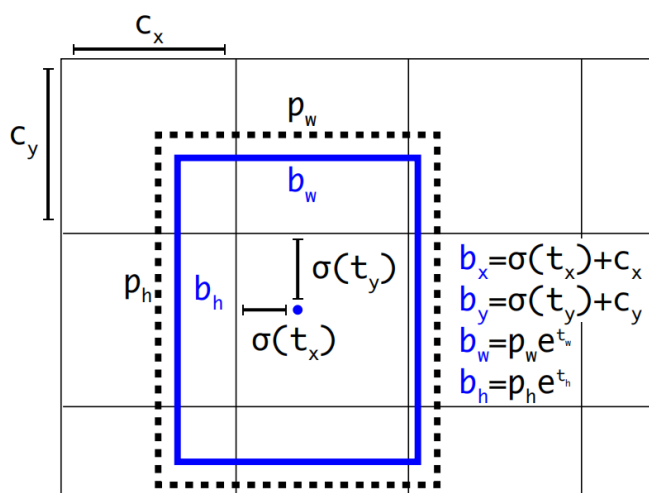


图 2-2 边界框结构示意图

通过三个尺度上的检测，32 倍的降采样得到的感受野最大，能够用来检测大型目标，当输入为 416×416 时，每个单元格的三个先验框大小为 $(116, 90)$ ； $(156, 198)$ ； $(373, 326)$ 。16 倍降采样下能够检测中等大小的物体，先验框大小分别为 $(30, 61)$ ； $(62, 45)$ ； $(59, 119)$ 。8 倍降采样有最小的感受野，

能够检测比较小的目标，因此先验框大小分别为（10,13）；（16，30）；（33，23）。

表 2-1 三个尺度下边界框所对应的感受野和先验框大小

特征图	13×13	26×26	52×52
感受野	大	中	小
	(116,90)	(30, 61)	(10,13)
先验框	(156, 198)	(62, 45)	(16, 30)
	(373,326)	(59, 119)	(33, 23)

图 2-2 中 t_x, t_y, t_w, t_h 是需要学习的参数， $\sigma(tx)$ 、 $\sigma(ty)$ 分别代表边界框中心坐标和当前单元格左上角的相对位移， σ 代表 sigmoid 函数，将偏移量限制在当前单元格当中，有利于模型收敛。这样就克服了在早期版本中 t_x, t_y 不受约束的问题，这样可以将二者的值限制在 0~1 之间，从而限制在了当前的单元格内，这使得模型训练过程中定位更加准确。

$$\begin{aligned} \Pr(\text{Class}|\text{Object}) \times \Pr(\text{Object}) \times \text{IOU}(\text{truth}, \text{pred}) = \\ \Pr(\text{object}) \times \text{IOU}(\text{truth}, \text{pred}) = \sigma(t_0) \end{aligned} \quad (2-1)$$

$$\text{IOU}(A, B) = \frac{A \cap B}{A \cup B} \quad (2-2)$$

$$\sigma(x) = \frac{1}{1+e^{-x}} \quad (2-3)$$

其中， t_x, t_y, t_w, t_h 是 YOLOv3 所作出的预测；

C_x, C_y 是先验框所在单元格的左上角坐标，即相对位置；

P_w, P_h 是先验框的宽和高；

C_x, C_y, P_w, P_h 由图片的宽和高归一化；

b_x, b_y, b_w, b_h 由边界框预测；

$\sigma(t_0)$ 是边界框置信度预测值，是当前框有目标的概率与 IOU 相乘的结果；

在 YOLOv3 中对物体进行定位时，直接预测相对位置，预测边界框中心坐标与整张图片左上角的相对坐标。这种预测采用了逻辑回归，首先，表（2-1）中第三行的 9 个先验框参数被三个张量平分之后，根据三个尺寸即：大、中、小来取出各自的先验框；然后，每个 y 输出在自己的单元格中输出三个预测边界框，假设是在大尺度下，即 13×13×66 下（因为共 17 个类别，即 $(17+4+1) \times 3=66$ ），输出了 3 个预测，三种尺度下一共输出了 9 个预测，那么逻辑回归就是用来对每个先验框所包含的内容进行一个目标存在性得分（objectness score），然后以此来选择最佳的先验框进行预测。这一步可以去掉不必要的先验框，减少计算量。不同于 Faster R-CNN，YOLOv3 只对一个最佳的边界框预测进行操作，逻辑回归

就可以从 9 个预测当中找到目标存在性得分最高的一个，相当于建立了一个目标存在性得分与最佳边界框预测的线性关系。这样做也是为了支持多标签对象。

2.3 损失函数

YOLOv3 的损失函数包括以下三个部分：

- (1) t_x, t_y, t_w, t_h 部分带来的误差
- (2) 置信度带来的误差，即目标存在性得分的预测与标注值之间的误差
- (3) 类别预测带来的误差，即预测类别与标注类别不同而产生的误差

分别对应 $lbox, lobj, lcls$ ，由于 yolo 模型并未显示定义损失函数，只在 `yolo_layer.c` 文件中定义了敏感度图函数用于后向传播过程，倒推出的公式如下：

$$lbox = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{i,j}^{obj} [(t_{xi} - \hat{t}_{xi})^2 + (t_{yi} - \hat{t}_{yi})^2] + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{i,j}^{obj} [(t_{wi} - \hat{t}_{wi})^2 + (t_{hi} - \hat{t}_{hi})^2] \quad (2-4)$$

$$lobj = \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{i,j}^{noobj} [c_i \log(\hat{c}_i) + (1 - c_i) \log(1 - \hat{c}_i)] + \lambda_{obj} \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{i,j}^{obj} [c_i \log(\hat{c}_i) + (1 - c_i) \log(1 - \hat{c}_i)] \quad (2-5)$$

$$lcls = \lambda_{class} \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{i,j}^{obj} \sum_{c \in classes} [p_i(c) \log(\hat{p}_i(c)) + (1 - \hat{p}_i(c)) \log(1 - p_i(c))] \quad (2-6)$$

$$loss = lbox + lobj + lcls \quad (2-7)$$

其中： S 代表单元格尺寸， S^2 即 13×13 、 26×26 、 52×52 ；

B 代表边界框；

$$1_{i,j}^{obj} = \begin{cases} 1, & \text{如果在第 } i \text{ 个单元格内而且被第 } j \text{ 个边界框检测到} \\ 0, & \text{其他情况下} \end{cases}$$

$$1_{i,j}^{noobj} = \begin{cases} 1, & \text{如果在第 } i \text{ 个单元格内没有物体} \\ 0, & \text{其他情况下} \end{cases}$$

二元交叉熵的定义如下：

$$BCE(\hat{c}_i, c_i) = -\hat{c}_i \times \log(c_i) - (1 - \hat{c}_i) \times \log(1 - c_i) \quad (2-8)$$

λ 常数在不同的情况下载损失函数中有不同的值，这样可以增加模型的稳定

性。比如说边界框坐标预测时最高 $\lambda_{coord} = 5$ ，而在没有物体出现的时候，

$$\lambda_{noobj} = 0.5。$$

非极大值抑制（Non-Maximal Suppression, NMS）的方法被用来检测最优边界框。首先，设定一个 NMS 阈值和一个 IOU 阈值，然后当预测的概率低于 NMS 阈值的时候，排除这个预测结果；同时当 IOU 值高于 IOU 阈值的时候，同样排除这个预测，那么这样就得到了最佳的边界框。

第三章 实验与分析

3.1 实验目的

开发出一个可以有效检测出“不欢迎图片”，即可以检测出不被公司方实时广告竞价系统认可的广告图片。在保证检测精度的情况下，检测时间应尽量缩短，最好在 100 毫秒以内。一般一张广告从竞价到展示需要 100 毫秒，但一般检测系统不会参与到这个系统当中，因为一旦审查不通过，连接很容易超时，一般广告在进入系统前审查检测完毕。先前一直采用人工审查，如果能够通过构建一个系统审查，可以节省大量人力物力。然后能在对现有方法的总结上提出创新。

在进行调研过后，决定构建一个基于 YOLOv3 的目标检测系统，在保证精度的基础上不至于影响速度。

3.2 实验数据

3.2.1 数据描述

本实验的数据来自公司方提供的广告 url 链接，每条链接指向一条广告图片。这些图片包括被“欢迎”的图片和“不被欢迎”图片，二者的 url 被一起存放在了一个 csv 文件中，可以通过“status”列下的 id 来区分两类图片，id 为 1 代表被“欢迎”，id 为 2 代表不被“欢迎”。csv 文件中一共存放了 725570 条 url。

1	redis_id	adspot_id	app_id	channel_id	iurl	text	pic text	status
2					https://			1
3					https://			1
4					https://			1
5					https://			1
6					http://			1
7					https://			1
8					https://			1
9					http://			1
10					http://			2

图 3-1 csv 文件截图，redis_id 代表图片 ID，iurl 代表图片网络地址，status 代表图片种

类。

这些图片并未明显违法法律法规，也未违反相关平台管理，但这些图片给平台的管理运营造成了负面影响，所以统称这些图片为“不欢迎”图片。

还有少量实验数据来自于开源数据集网站 OPEN IMAGE DATASET V6(网址见于图 3-2)，主要为了补充公司方缺少的广告数据类型。比如说页游广告中有一些第一人称射击类游戏（First Person Shooting, FPS），这些网页游戏广告在宣传时一般盗取电脑端 FPS 游戏或者主机端 FPS 游戏的截图，从而构成了虚假宣传。而识别这些游戏的关键在于检测画面中出现的枪支或者其他武器。而这类广告数量并不充足，无法为检测系统提供足够的特征，于是采取了从 OPEN IMAGE DATASET V6 获取针对枪支数据集的方法，但这些数据集采取了不同于 YOLO 的标注方法，需要二次加工才能使用。除了来自开源数据集网站 OPEN IMAGE DATASET V6，还有部分数据直接从谷歌图片上通过脚本批量下载，在本地进行标注。所有的公司方提供的数据也在本地进行了标注。

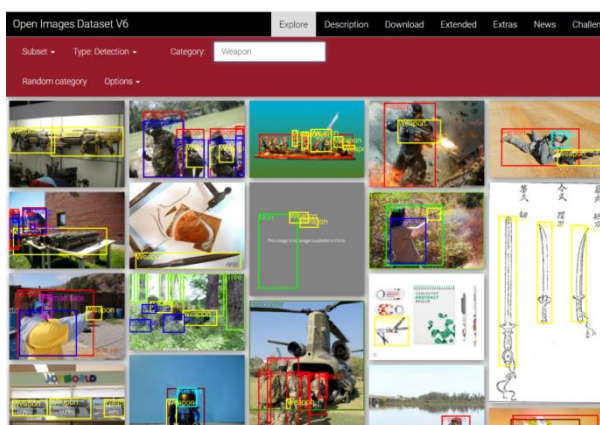


图 3-2 OPEN IMAGE DATASET.[14] 开源数据集网站，开源了多种数据，可以提供目标检测、图像分隔、图像关系研究等；

除此之外还有一小部分类型的数据集图片既不大量存在公司方提供的广告中，OPEN IMAGE DATASET 也没有提供开源数据集，只能从互联网上自行搜索下载。主要是从谷歌图片上搜索，然后批量下载 url 地址并将图片爬取下来。

3.2.2 数据获取

（一）公司方数据爬取

公司方广告数据的 url 地址全部存放在一个 csv 文件中，这里通过一个 Python 网络爬虫程序来爬取这些图片数据并保存到本地。这个程序调用了 Python

HTTP 请求库 request, Python 数据处理库 pandas 以及自带的库 os。程序的流程如下:

(1) 调用 pandas 库, 读入 csv 文件;

(2) 调用 request 库和 os 库, 编写爬虫的请求头来伪装成一个用户, 从而应对某些网站的反爬取机制。

(3) 通过一个循环开始图片爬取。在循环内部, 首先验证文件路径是否存在, 如果存在停止爬取, 不存在就爬取文件并在本地创建路径, 同时记录爬取成功的数量。如果不是上述任何一种情况, 就记录爬取失败的数量。

(4) 循环结束后输出爬取失败的数量和成功的数量。

该程序附在附录中。

(二) OPEN IMAGE DATASET 数据获取

通过运行 OIIdv4 工具源码[15]来针对性地下载对应数据集中国的图片与其标注, 并按不同类别保存到本地。

(三) 谷歌图片搜索结果数据获取

这种方式承担了一小部分的数据集获取, 为了保证下载的效率, 编写了一个 javascript 脚本来批量下载图片的 url 地址, 然后通过这些 url 爬取相关图片。

3.2.3 数据清理

本实验面对的数据规模相当得大, 需要从 72 万张数据中整理出有用的数据, 分析数据的特征, 找到可以用于目标检测的物体, 同时这些物体又能很好地代表“不欢迎”广告。而 OPEN IMAGE DATASET 的数据可以通过直接编写程序来处理, 所以主要处理的对象是公司方的数据。

(一) “不欢迎”类广告

用于作为数据集中的训练集和测试集, 所以这些广告图片被全部爬取了下来。但是由于人工审查存在失误的情况, 这些数据中有大量“无效数据”。这类“无效数据”既不能对“不欢迎”广告数据作出描述或者代表, 也不能作为数据集的一部分参与训练。因为这些数据本身是属于“欢迎”广告, 但是因为人工审查的失误而被划入了“不欢迎”广告。

在总共爬取下来的 74,413 张广告图片中, 只有 19,586 张广告图片可以用于训练。这个过程完全有人工完成, 因为没有办法让程序自动识别这些图片的区别, 只有借助这些图片制作而成的数据集进行训练, 才有可能通过深度学习的方法让程序区分这些图片。

下面是两张“不欢迎广告”图片的示例, 见图 3-3:

从图 3-3 中可以看出，第一张空调广告图片是不应该被封禁的，但第二张图片包含了一个弹窗，这会诱导用户进行点击，而不论是点击“取消”还是“重试”，都会导致打开这个广告所指向的页面，这种广告是需要被封禁的。而第一张广告被封禁有可能是人为失误，但也有可能是在人工审查的过程中采用了图像聚类等方法，但图像聚类只能提取浅层特征，所以导致空调广告中的空调被当做是弹窗了。在“不欢迎”类广告中，类似第一张图片的内容占比 73.68%，类似第二张图片的广告占比 26.32%。



图 3-3 这两张图片同属于“不欢迎”类广告

（二）“欢迎”类广告

用于作为数据集中的测试集。这类广告都比较中规中矩，但仍然存在部分广告有“违规”行为，存在擦边球暗示等等问题，也可以认为这是人工审查的失误导致的。下图 3-4 是两张“欢迎”广告的示例。

从图 3-4 可以看出，第一张图片属于典型的“欢迎”类广告，为了推销自己的商品，也没有不实或虚假宣传。而第二张图片存有疑问，有色情擦边球的嫌疑，这有可能是审查的失误导致的。在“欢迎”类广告中，类似第一张图片的广告占大约 95%以上，类似第二张图片的广告占比 5%以下。因为没有下载全部的“欢迎”类广告，只能根据现有下载的 6766 张“欢迎”类广告图片进行一个估算。



图 3-4 这两张图片同属于“欢迎”类广告

在以上两类广告中还有很多失效的广告，这些广告的 url 已经失效，后者即便登入了相应的 url 图片也已经消失了。还有一些广告因为格式问题无法呈现，这些广告也需要被手动删除清理。这些广告的初步分类清理全部通过人工手工分类完成，并在完成这一步后，将清理好的广告制作成数据集，包括训练集和测试集。

3.2.4 数据预处理

在开始制作数据集之前，首先还需要针对 OPEN IMAGE DATASET V6 (OIDv6) 目标检测框的标注格式问题进行格式转换处理。这是因为 OIDv6 为了增强数据集的通用性，标注边界框的时候，标注格式为：

[文字标签，标注框左上角 X 坐标(cord(0))，标注框左上角 Y 坐标(cord(1))，标注框右下角 X 坐标(cord(2))，标注框右下角 Y 坐标(cord(3))]
格式样例是：

[GREEN_HEART Cord(0), Cord(1), Cord(2), Cord(3)] (3-1)

坐标系原点出于左上角，详见图 3-5。

而在 YOLO 标注框中，假设 GREEN_HEART 类型标签为“0”，数据格式为：
[0 X_{中心}, Y_{中心}, X_{长度}, Y_{长度}] (3-2)

YOLO 格式可见图 3-6 这是由 YOLO 的定位算法决定的，因为 YOLO 以单元格为中心，聚类得到物体中心的坐标与标注框的长度和高度，产生 9 个标注框的预测值，并逻辑回归产生最佳边界框预测的算法所决定的。那么为了让 YOLO 网络能够利用来自 OPEN IMAGE DATASET V6 的数据集，需要对公式 3-1 中的标注格式进行处理，从而得到公式 3-2 中的格式。

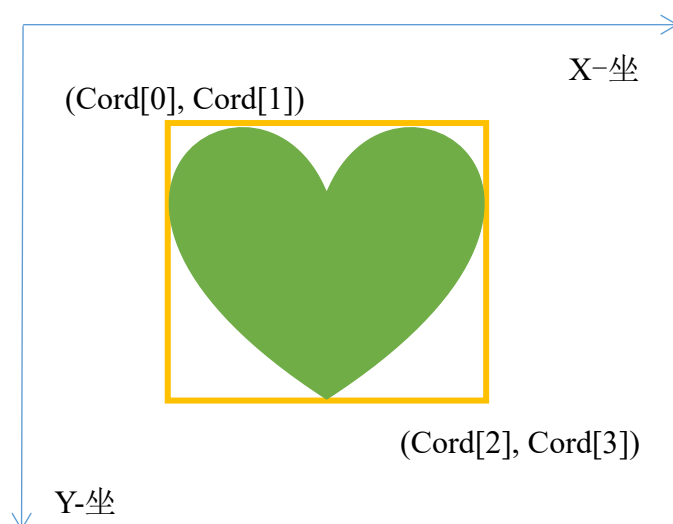


图 3-5 OPEN IMAGE DATASET 数据集标注框的一般格式

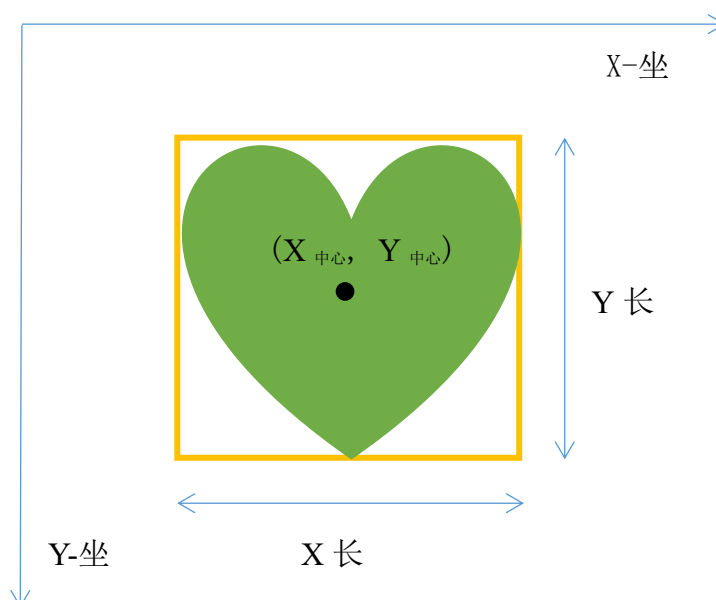


图 3-6 YOLO 数据集标注框的一般格式

处理程序的主要算法是：将 $\text{cord}[2]$ 的值减去 $\text{cord}[0]$ ，然后把二者的差赋值给 $\text{cord}[2]$ ， $\text{cord}[2]$ 除以 2 并赋给一个变量 X_{center} ，同理将 $\text{cord}[3]$ 的值减去 $\text{cord}[1]$ ，得到一个二者的差值赋给 $\text{cord}[3]$ ， $\text{cord}[3]$ 除以 2 并赋给一个变量 Y_{center} ，从而得到了新的 $\text{cord}[2]$ 和 $\text{cord}[3]$ ，此时的 $\text{cord}[2]$ 和 $\text{cord}[3]$ 分别代表边界框的长度和高度。然后将 $\text{cord}[0]$ 的值加上 X_{center} 并赋值给 $\text{cord}[0]$ ，同理将 $\text{cord}[1]$ 的值加上 Y_{center} ，并赋值给 $\text{cord}[1]$ 。这样新的 $\text{cord}[0]$ 和 $\text{cord}[1]$ 分别代表了物体中心相对坐标原点的 X 坐标和 Y 坐标。这样， $\text{cord}[0]$ ， $\text{cord}[1]$ ， $\text{cord}[2]$ ， $\text{cord}[3]$ 就符合了 YOLO 的格式要求。同时把类别的名称替换成数组元素的序号，就可以将描述类别的文

字替换成 YOLO 格式下代表类别的数字。以上就是标签格式预处理的核心算法，其他代码主要是进行文件读取和目录转换，程序名称为“convert_annotations.py”，附在附录当中。

3.2.5 数据集制作

数据集制作是整个广告审查系统最关键的部分，没有一个合适的数据集就无法有针对性地训练模型，无法完成审查任务。所以训练数据集大部分取材于公司方提供的“不欢迎”类广告，理论上只要模型能够检测出“不欢迎广告”，这个模型就能够用于公司方发布的广告审查任务。

（一）广告种类划分

首先，如果要对广告进行目标检测，要确定检测的对象是什么。而这个检测对象的选择十分关键。

A. 广告检测的“意象”

这个检测的对象，本实验称之为“意象”。类似于古诗当中的意象是被用来寄托主观情思的客观物象，举个例子，马致远的《秋思》中有“枯藤老树昏鸦，小桥流水人家”的诗句，而“枯藤”、“老树”、“昏鸦”就是这首诗中的意象，表达了作者对家乡的思念与伤感之情。

而在一张广告中，比如下图 3-7，金币图标和红包图标就是两类“意象”，只要发现了有这两类意象，就能知道广告带有诱导用户点击的意图，和向用户传达“优惠”、“折扣”等信息的意图。一般而言这只是一种正常的营销手段，但是这种广告在公司方平台上就涉嫌欺诈 APP 用户和诱导用户点击。因为这种广告点进去一般没有红包领，这就构成了欺诈；更重要的是广告的点击率对于评价这个广告十分重要，而这种广告的点击率很有可能是带有水分的，可能是用户无意识点击的；还有一点就是这可能对用户的偏好产生误导，因为用户点击的原因可能是因为用户看见了红包，而这个广告兜售的商品可能并不让用户感兴趣。

B. 根据意象对广告进行分类

根据“不欢迎”广告的内容，可以将大部分“不欢迎”广告放入一下四个类别当中：

- a. 诱导点击类
- b. 色情擦边球类
- c. 网页游戏类
- d. 赌博类

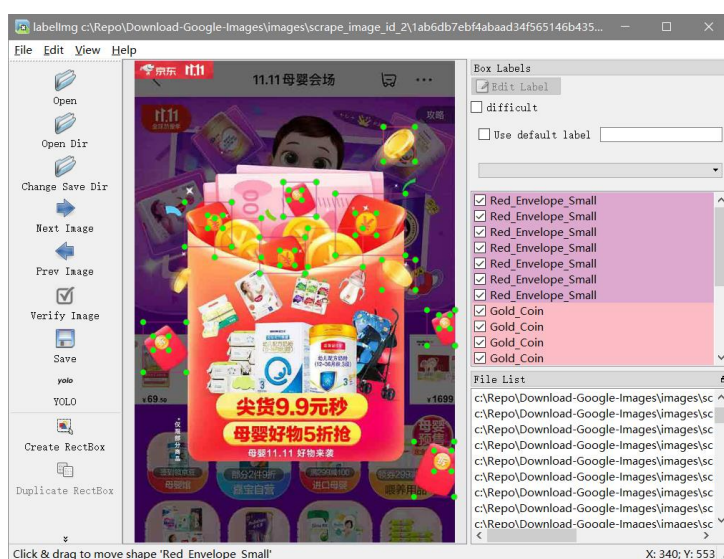


图 3-7 带有多个红包和金币意象

每一类图片包含多种不同的意象，一共发现了 17 类可供识别的意象。分别是：武器类意象、人体部位类意象（脚部、腿部、胸部、内衣、臀部、肩膀、肚脐、怀孕）、扑克牌、白色弹窗、箭头、关闭按钮、红包_小、红包_大、麻将。这些意象彼此之间在外形、颜色以及所处的背景上等特征上有较大差别，可以作为目标检测识别的对象。不同类别的广告和意象的对应关系在表 3-1 中进行了描述。其中“√”表示包含这类意象是某广告属于该类别广告的充分不必要条件，“×”表示包含这类意象是某广告属于该类别广告的不充分不必要条件。

a. 诱导点击类

该类广告主要通过一些特定意象来诱导 APP 用户出于各种原因来点击这则广告，从而达到让用户打开广告链接的目的。因为此时对于广告的任何操作，不论是点击“关闭按钮”、“取消”、“确认”等按钮，都会打开这条链接；在某些情况下，用户点击网络的任何一个地方都会打开这条链接。这么做客观上增加了这条广告的点击量，混淆了用户的真实喜好，导致用户对广告甚至是使用的 APP 产生反感。出于这些因素的考虑，该类广告应该被识别出来并在产生更多的破坏之前被封禁。具体如下图 3-7 展示。

表 3-1 四个主要“不欢迎”类别广告包含的意向

意象\广告种类	诱导点击类	色情擦边球类	网页游戏类	赌博类
武器	×	×	√	×
脚部	×	√	×	×
腿部	×	√	×	×
扑克牌	×	×	√	√

白色弹窗	√	×	×	×
箭头	√	×	×	×
胸部轮廓	×	√	×	×
内衣轮廓	×	√	×	×
臀部轮廓	×	√	×	×
肩膀轮廓	×	√	×	×
肚脐轮廓	×	√	×	×
怀孕	×	√	×	×
关闭按钮	√	×	√	×
红包_小	√	×	×	√
红包_大	√	×	×	√
麻将	×	×	√	√



图 3-7 四张诱导点击类广告；包含 5 种诱导点击类相应的“意象”，从左到右分别包含白色弹窗、关闭按钮、红包、金币、箭头。图中这些“意象”都已经标注了出来。

b. 色情擦边球类

这类广告并未包含硬色情内容，大多都是一些擦边球内容，也就是性暗示内容及其他凸显人体某些部位轮廓的内容。由于公司方提供的广告数据中不包含硬色情内容，所以硬色情不在本实验的检测范围之内。而之所以要封禁擦边球内容，主要是因为广告推销的是产品，而不是推销代言广告的模式，这样也会一定程度上导致点击率含有水分，或者混淆用户喜好。这种广告可以被认为是使用不正当手段进行竞争。如下图 3-8 所示。



图 3-8 四张色情擦边球类广告，图中包含 5 中色情擦边球类的“意象”，一类诱导点击“意象”，从左到右分别包含脚部、腿部、肩部轮廓、胸部轮廓、肚脐轮廓、关闭按钮。

判别是否有擦边球嫌疑的方法是：如果模特某些身体部位在广告中的轮廓特别的明显，那么这类广告就有明显的擦边球嫌疑。而且图片类的广告模特很少全身出境，一般都是上半身出境。该类广告的意象主要有：脚部、腿部、胸部轮廓、内衣轮廓、臀部轮廓、肩膀轮廓、肚脐轮廓和怀孕。“怀孕”类意象被封禁的主要原因是，大部分带有孕妇的广告是代孕广告，而代孕在我国是被法律禁止的。母婴产品广告一般会选用母亲和婴儿作为模特，不会选用孕妇作为模特。这些身体部位的标注不会导致一旦出现人物就检测出身体部位的情况，因为标注的数据都是轮廓特别明显，或者裸露了大量皮肤的情况下，才会被标注，普通的模特不会被标注为训练集。

c. 网页游戏类

网页游戏类广告主要包含虚假宣传的问题。因为这类游戏一般都是 flash 游戏，而这类广告使用的却是某些知名电脑端游戏和主机端游戏的截图。而這些截图一般都有十分炫酷的武器，可以作为识别的目标和依据。如下图 3-9 所示。

d. 赌博类

这类广告的意象非常简单，通常是赌博中出现的道具，比如扑克牌、麻将。如下图 3-10 所示。

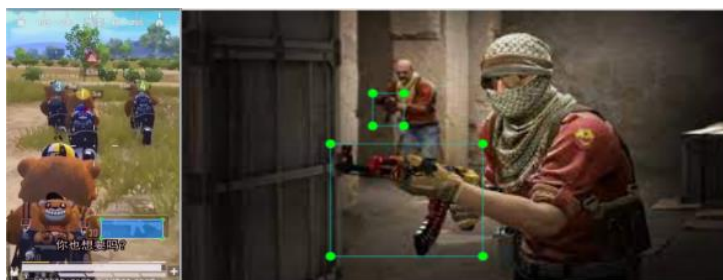


图 3-9 网页游戏类广告，图中意象为武器



图 3-10 赌博类广告，出现的意象有扑克牌、麻将

（二）数据集标注

数据集标注采用 LabelImg 工具，通过运行 LabelImg 源码来进行。图 3-7、3-8、3-9、3-10 中图片均是通过该工具标注。

标注时注意将意象标注精细化，意思是可以增加被标注意象的数量的话，就尽量多创造标注框。比如有一排扑克牌或者麻将的图片中，单独标注每张扑克牌后者每粒麻将不仅可以增加训练样本，还可以突出意象的特征。这么做还可以适应更多的情况。比如说因为拍照角度问题，擦边球类广告的模式只有一半的胸部轮廓或者一半的肩膀轮廓比较明显的话，标注两边胸部或者两边肩膀就会让模型对特征的理解下降。有些情况下模特只有一边的身体部位出镜，那么标注两边身体部位的模型训练过后，在这种情况下可能根本就识别不出来，造成漏检测。

数据集标注的工作量非常大，平均一个小时最多可以标注 170 到 180 张广告数据。

3.3 训练模型

本实验采用了 c 语言和 python 语言复现 YOLOv3 的版本，其中 c 语言复现的部分主要用于实现模型搭建，python 语言复现的部分主要用于可视化和数据处理。训练过程在远程服务器上完成，累计训练时间有 140 个小时以上。主要通过调试源代码和调参来完成训练，得到训练好的 YOLOv3 权重，然后就可以进行验证了。一共有 1000 到 1100 个可调参数。

（一）模型的配置与调参

[net]层，配置整个模型。图 3-11 是.cfg 文件中关于[net]层的参数。

A. Batch=64

不同于机器学习中的 batch，用来规定网络累计输入几个样本之后进行一次

```
[net]
batch=64
subdivisions=16
# Training
#width=512
#height=512
width=416
height=416
channels=3
momentum=0.949
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.0013
burn_in=1000
max_batches = 34000
policy=steps
steps=27200,30600
scales=.1,.1
```

图 3-11 cfg 中关于[net]层的参数

后向传播，在本实验的模型中，batch 要和 subdivisions 放在一起配合，即一个 batch 数目的图片会通过 subdivisions 次进行前向传播。在这里就是每次 4 张图片，在前向传播的时候，累加所有批次的 loss 并求平均，等到 64 张图片都已经前向传播后，再后向传播传递参数。

本次调参过程中，最佳 subdivisions 是 16，一般设置成 8 的倍数；需要注意的是在测试的时候把 batch 和 subdivisions 都设置为 1，防止在测试过程中出现报错。

B. Width=416, height=416, channels=3

输入图片的宽高一般设置为 32 的倍数，因为 Darknet 中进行了 5 次下采样，否则可能无法加载网络；本次实验的数据集是彩色图片，有三个信道。

C. Momentum

最优化方法中的动量参数，影响梯度下降到最优值的速度，经过多次试验设置为 0.949

D. Decay

权重衰减，用来防止过拟合，一直采用的是 0.0005。

E. Angle, Saturation, Exposure, Hue

这四个参数是数据增强的参数选项，分别代表通过旋转角度、调整饱和度、调整曝光量、调整色调来增强数据，产生更多训练样本。旋转角度在本实验中意义不大，因为本实验基于广告的数据集中很少有倾斜广告中物体或者场景的情况。这是为了让用户阅读方便，一般广告采取的角度和位置布局都相对固定。所以将 angle=0。而饱和度、曝光量、色调在不同质量的广告上，会产生一定的差异，所以将这三个参数分别设置为 1.5、1.5、0.1。

F. 学习率

初始学习率设置为 0.0013，在训练的过程中终止训练，调整学习率，再加载保存的模型继续训练。主要通过观察 loss 的变化调整学习率：loss 波动大，学习率过大；loss 波动小以至于几乎不变，那网络可能已经收敛或局部最优，可以适当调高学习率。

学习率与 gpu 线程数有关，在我的实验环境中 6 条 gpu 线程，学习率就为 0.0013×6 。

G. Burn_in=1000

当迭代次数大于 1000 时，采用 policy 参数中的 steps 策略来更新学习率。

H. Max_batches

最大训练批次，超过这个数目就不再训练。在这里保证每个类别的意象有足够的训练，设置为 $17 \times 2000 = 34000$ 次

I. Policy, Steps, Scales

为了让学习率在完成之前达成 100 次的衰减，在第 27200, 30600 次分别衰减 scales 倍，即衰减 10 倍。

[Convolutional]层，卷积层，同时也是一个下采样层，配置见图 3-12。
Batch_normalize=1，需要进行批归一化处理；Size=3，卷积核尺寸；Stride=2，步长，此处步长为 2 是为了降采样；Pad=1，进行补零；Filters=512，卷积核个数；Activation=leaky，泄露性线性整流函数作为激活函数

```
[convolutional]
batch_normalize=1
size=3
stride=2
pad=1
filters=512
activation=leaky
```

图 3-12 卷积层，下采样层参数配置

[Yolo]层，配置见图 3-13。

```
[yolo]
mask = 6,7,8
anchors = 12, 16, 19, 36, 40, 28, 36, 75, 76, 55, 72, 146, 142, 110, 192, 243, 459, 401
classes=17
num=9
jitter=.3
ignore_thresh = .7
truth_thresh = 1
random=1
```

图 3-13 [yolo]层的参数配置

其中, mask=6, 7, 8 和 anchors=12, 16, 19, 36, 40, 28, 36, 75, 76, 55, 72, 146, 142, 110, 192, 243, 459, 401; 因为此时位于第 79 层, 特征图刚经过 32 倍下采样, 感受野较大, 所以先验框也选取 anchors 中最后的三对; classes=17 代表类别有 17 种; num=9 指的是每个单元格预测 9 个边界框, 数目与先验框一样; jitter=3 是一种数据增强的手段, 随机协调图片长宽的比例; ignore_thresh=.5, truth_thresh=1 这两个参数代表预测框与标注框的 IOU 比值在大于 ignore_thresh 的时候, 会参与损失函数的计算。这样可以把参与损失函数计算的预测框的数量, 如果把 ignore_thresh 设置得很大, 那么会减少参与损失函数计算的预测框数目, 容易导致过拟合。而当 ignore_thresh 设置得很小, 参与的预测框数目庞大, 噪音也多, 容易欠拟合。在检测大型物体的时候, 设置 ignore_thresh 为 0.7 比较合理, 而在检测小物体的时候设置为 0.5 比较合理。一共有三层[yolo]层, 图 3-13 的[yolo]层检测大型物体, 设置为 0.7; 剩下两个[yolo]都设置为 0.5, 提高对小物体的检测准确度。

[yolo]层上面那一层的[convolutional]层配置, 见于图 3-14.

```
[convolutional]
size=1
stride=1
pad=1
filters=66
activation=linear
```

图 3-14 [convolutional]层配置

注意将卷积核数量设置为: (类别数+4(边界框的 4 个坐标)+1(有物体置信度))×预测框个数。此处有 $(17+4+1) \times 3=66$ 。

(二) 模型训练

模型训练和上一步调参配置同时进行, 没有先后关系。边训练边调整参数, 保存模型重新展开训练。如果长时间没有降低模型的损失或者没有提高模型的准确度, 那么就停止训练, 进行调参之后再重新开始训练。从开始训练到最终训练结束, 一共有三个训练阶段, 下面是三个阶段中平均准确度 (Mean Average Precision, mAP) 与损失函数的可视化, X 坐标轴正方向描述的是训练批数的增加, Y 轴正方向描述的是平均准确度的提升, Y 轴负方向描述的是损失函数的下降。

A. 模型训练第一阶段

这一阶段设置学习率相对较大, 快速降低损失函数, 提升模型学习速度。按照这个思路, 所有[yolo]层的边界框的 ignore_thrsrh 都设置为 0.7, 这样可以减

少参与损失函数计算的边界框数目，达到快速收敛的目的。训练效果如下图 3-15 所示。

从图中可以看出，mAP 曲线和损失函数曲线波动较大；当损失函数下降趋势逐渐稳定之后，mAP 曲线并未进入稳定状态，也没有继续增加，而是维持在了 67% 左右。这说明学习率过大，导致模型一直在最优解附件徘徊，却始终无法进入最优解。此时停止训练，并进行调参。

B. 模型训练第二阶段

此时调低学习率，让模型更好地收敛。同时放低前两个[yolo]层的边界框 ignore_thresh 至 0.5，从而增加参与损失函数计算的边界框，客观上增加一定的噪音，防止模型过拟合。保持权重衰减参数和动量参数保持不变，此时除了手动调低学习率外，可以维持别的参数不变来降低学习率。效果如下图 3-16 所示。

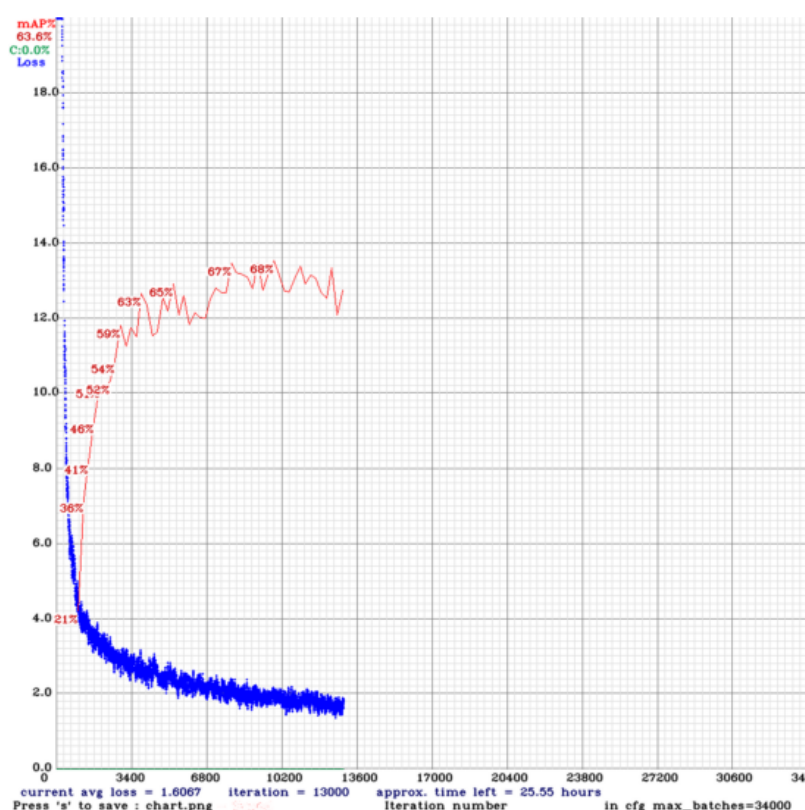


图 3-15 第一阶段 mAP 与损失函数

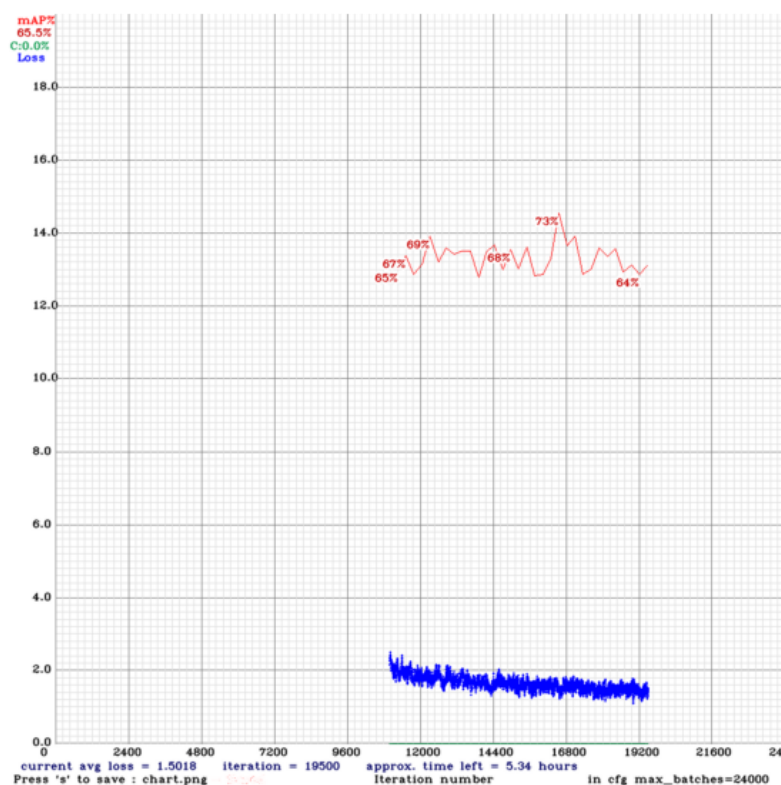


图 3-16 第二阶段 mAP 与损失函数

从图中可以看出，mAP 仍然有一定波动，损失函数下降的过程中也有一定的波动。最高曾经达到了 73% 的 mAP，将这里的权重文件保存下来供之后训练。而出现峰值之后 mAP 的表现并不理想，似乎还有一些下降的趋势。此时怀疑模型已经产生了过拟合，而且测试集可能已经经过了过多的测试，需要替换新的测试集。测试集是随机从一批“不欢迎”广告当中选出来的，因为如果特意选取测试集，无法代表广告数据集的真实情况。但随机选取带来的问题是缺乏针对性，广告中各种类型的“意象”本身分布就很不均匀，如果随机选取的话很有可能会漏掉一些广告。比如：同样对于弹窗广告，大部分弹窗广告的形式是一个白色弹窗位居广告正中央，但也有少量弹窗广告是由其他颜色的弹窗组成，这类弹窗广告也以一定数目存在于训练集中。如果测试集中没有这类特殊颜色的弹窗广告，那么特殊颜色这个特征就会被当做噪音忽略掉，而不是作为一个特征保留。所以本实验为了增强数据集的概括性，在随机选取大部分测试集广告的基础上，有针对性地向其中加入了含有特定重要特征的广告作为测试集的一部分，并重新开始训练。

C. 模型训练第三阶段

这是最后一个阶段，之所以分了三个阶段是因为本实验在重复训练的过程中，大部分的训练都遵循这种分三个阶段的方法。在这里所呈现出来的是一组连续的，

效果最好的三阶段训练过程。那么可以从图 3-17 看到，虽然由于新测试集的加入，模型 mAP 有一定的下降，但之后这一阶段的 mAP 曲线稳步上升，而且没有太大的波动。损失函数也是稳定下降。停止训练，防止过拟合。进行验证。

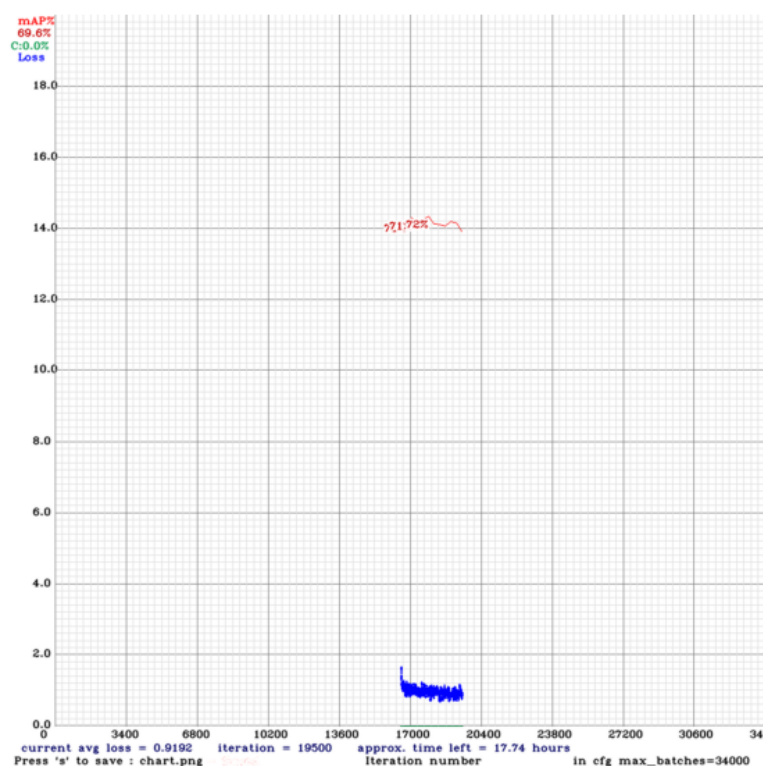


图 3-17 模型训练第三个阶段

3.4 验证模型的检测准确度

模型验证采用两组图片作为验证集。一组图片全部由“欢迎”广告组成，另一组图片由“不欢迎”广告组成。两组图片各 680 张，都是随机选取。通过检测两组图片，得到结果验证。其中加载不成功的图片一律视为“不欢迎”广告。如果识别出 17 类“意象”中的任意一种，视为“不欢迎”广告，否则为“欢迎”广告。结果如下表 3-2 所示。

因为我们的目标是要检测“不欢迎”广告，所以在这里“不欢迎”广告是正样本。通过下列公式计算验证结果的各项数据。

$$\text{Precision} = \frac{tp}{tp + fp} \quad (3-1)$$

$$\text{Recall} = \frac{tp}{tp + fn} \quad (3-2)$$

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn} \quad (3-3)$$

$$\text{F1-score} = \frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}} \quad (3-4)$$

验证精确率为 89.47%;

验证召回率为 96.03%;

验证准确率为 90.15%;

验证 F1-score 分数为 92.63%;

虽然训练过程中，mAP 的表现不够理想，但验证广告识别精度效果还是不错的。主要是因为 mAP 是对所有目标检测类别，也就是“意象”，的检测准确度的平均，有些类别的准确度可能很低，虽然被检测出来了，但是拉低了 mAP。但是在验证集中，只要能够检测出来有一种类别的“意象”，就认为这是“不欢迎”广告，相当于把检测出所有类别“意象”的平均准确度，改变成了至少检测出一样类别“意象”的检测成功率，改变了评价的标准，所以在验证时各项指标更完美。为什么改变了这个指标呢？因为作为公司方检测不良广告，不关心其中某个类别的准确度是多少，只关心这个广告是否是不良广告。所以验证准确度就通过这个评价标准来计算。

表 3-2 在 680 张原始数据上验证的结果

真实值\预测值	预测为“不欢迎”广告	预测为“欢迎广告”
“不欢迎”广告 (680 张)	653 (TP)	27 (FN)
“欢迎”广告 (680 张)	107 (FP)	573 (TN)

3.5 基于 G10U 的定位损失函数改进

3.5.1 G10U 与边界框回归

边界框的回归是 YOLOv3 模型的重要模块之一。一种对边界框回归的改进方法就是使用 IoU 计算的度量损失来更换替代回归损失，替代回归损失的例子有 L1 正则范数与 L2 正则范数。

为什么在某些情况下，使用 IoU 计算的度量损失要优于替代回归损失呢？在

这里给出 IoU 和 GIoU 的定义，为了方便阅读，再次给出 IoU 的定义：

$$IoU = \frac{|A \cap B|}{|A \cup B|} \quad (3-5)$$

$$GIoU = IoU - \frac{|C \setminus (A \cup B)|}{|C|} \quad (3-6)$$

其中，公式(3-5)代表 A、B 两个矩形区域的重叠面积除以 A、B 面积的总和；在公式(3-6)中，C 代表能涵盖 A、B 的最小矩形的面积； $|C \setminus (A \cup B)|$ 代表矩形 C 去除矩形 A、B 面积之和后的面积，设这个面积为 S。公式(3-6)代表 A、B 两片区域的 IoU 值减去 S 和 C 的比值。

根据下图 3-18。定义绿色框为真实框，三个绿色框大小相同。定义三个黑色框为预测框，固定三个预测框的左下角，预测框的右上角在虚线圆上移动。可以看出，预测框的右上角处于虚线圆的任意位置时，L2 正则范数保持不变。而在这三种情况下，IoU 和 GIoU 都各不相同，且不论预测框的右上角在虚线圆上的任意位置，IoU 和 GIoU 都不会保持不变。所以在某些情况下，L2 范数不能反映预测框与真实框的重叠程度，使用 IoU 计算的度量损失要优于替代回归损失。

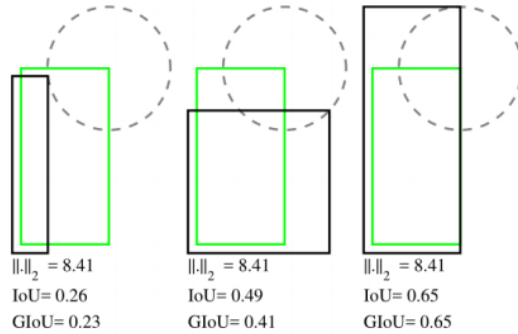


图 3-18 L2 正则损失与 IoU、GIoU^[17]

那么在本实验中，选用 GIoU 计算度量损失来替换公式(2-4)中的损失。公式(2-4)中的损失为平方误差损失。不选用 IoU 来作为计算度量损失的原因是，在真实框与预测框不重叠的情况下，二者的 IoU 值为 0。这将导致梯度为 0，从而无法进行参数的更新与优化。而在这种不重叠的情况下，GIoU 值不为 0。

GIoU 作为计算度量，具有尺寸不变性、且与 IoU 保持强相关性。那么定义：

$$L_{GIoU} = 1 - GIoU \quad (3-7)$$

这个损失函数具有诸多优点，比如非负性，不可分者同一性，对称性和三角不等式关系^[17]。

3.5.2 设计 G10U 损失函数算法

由于 YOLOv3 模型本身并没有显式地定义损失函数，而是通过计算梯度的方式在 YOLOv3 模型中进行参数更新与优化。YOLOv3 中定义的损失函数并未参与参数更新，而只是为了输出给使用者查看。

参考 YOLOv3 模型中计算边界框回归损失的梯度的函数，一个实现在 yolo_layer.c 中的 float delta_yolo_box() 函数，如下图 3-19 所示。可以发现，可以发现，YOLOv3 模型通过这种求参数偏差项的方法来完成了参数的传递，完成后向传播。而在本实验中，同样定义一个 delta_yolo_box() 函数来传递梯度，但传递的不是 YOLO 目标框的 t_x, t_y, t_w, t_h 参数，而是目标框的左、上、右、下的坐标，分别用 l、t、r、b 来指代。

那么要表示目标框的左、上、右、下的坐标，首先需要有一个转换函数来将 YOLO 目标框的 x, y, w, h 格式转换为坐标。那么这个类似于数据预处理步骤中

```
float delta_yolo_box(box truth, float *x, float *biases, int n, int index, int i, int j, int lw, int lh, int w, int h, float *delta, float scale, int stride) {
    box pred = get_yolo_box(x, biases, n, index, i, j, lw, lh, w, h, stride);
    float iou = box_iou(pred, truth);
    float tx = (truth.x*lw - i);
    float ty = (truth.y*lh - j);
    float tw = log(truth.w*w / biases[2*n]);
    float th = log(truth.h*h / biases[2*n + 1]);
    delta[index + 0*stride] = scale * (tx - x[index + 0*stride]);
    delta[index + 1*stride] = scale * (ty - x[index + 1*stride]);
    delta[index + 2*stride] = scale * (tw - x[index + 2*stride]);
    delta[index + 3*stride] = scale * (th - x[index + 3*stride]);
    return iou; }
```

图 3-19 float delta_yolo_box() 函数

将 OPEN IMAGE DATASETv6 中的图片标注框转换为 YOLO 标注框一样，只不过是反其道而行之。实现该转换算法代码如下图 3-20 所示。

```
// 将 x, y, w, h 转换成上左下右
boxabs to_tblr(box a) {
    boxabs tblr = { 0 };
    float t = a.y - (a.h / 2);
    float b = a.y + (a.h / 2);
    float l = a.x - (a.w / 2);
    float r = a.x + (a.w / 2);
    tblr.top = t;
    tblr.bot = b;
    tblr.left = l;
    tblr.right = r;
    return tblr; }
```

图 3-20 将 YOLO 边界框格式转换为本实验所需的边界框格式

然后本实验还需要计算 GIoU，那么包裹矩形 A、矩形 B 的矩形 C 的坐标也需要求出。算法是：矩形 C 的左、上坐标通过求矩形 A 和矩形 B 的最小左、上坐标求出，矩形 C 的右、下坐标通过求矩形 A 和矩形 B 的最大右、下坐标求出，程序就不再展示。

那么有了以上两个矩形框的各项参数，就可以很容易地定义求出 A、B 的 IoU 和 GIoU 值的函数了。定义为 `float box_giou(box a, box b)`，不展示程序。

接下来，需要取出预测框和真实框的 t、b、l、r 值，如下图 3-21 所示。

```
boxabs pred_tblr = to_tblr(pred);
float pred_t = fmin(pred_tblr.top, pred_tblr.bot);
float pred_b = fmax(pred_tblr.top, pred_tblr.bot);
float pred_l = fmin(pred_tblr.left, pred_tblr.right);
float pred_r = fmax(pred_tblr.left, pred_tblr.right);

boxabs truth_tblr = to_tblr(truth); //通过这个函数将xywh转换为左上右下
```

图 3-21 t、b、l、r 值得定义

其中，对预测值 `pred_t`, `pred_b`, `pred_l`, `pred_r` 需要取最大最小值来获得正确的结果，而对真实值则可以直接调用 `to_tblr` 函数。

定义浮点数 I 代表交集的面积值，浮点数 U 代表并集的面积值，浮点数 C 代表大框 C 的面积值，浮点数 X 代表预测框的面积值。

那么要对 IoU 公式(3-7)求导：

$$z = \frac{I}{U} = \frac{I(t, b, l, r)}{U(t, b, l, r)} \quad (3-7)$$

根据复合函数求导的链式法则：

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx} \quad (3-8)$$

那么(3-7)对 t 求导有：

$$z' = \frac{\dot{I}_t}{U} - \frac{I}{U^2} \dot{U}_t = \frac{\left(U \frac{dI}{dt} - I \frac{dU}{dt} \right)}{U^2} \quad (3-9)$$

那么 GIoU 的求导也类似：

$$GIoU = IoU - \frac{C - U}{C} = IoU + \frac{U - C}{C} \quad (3-10)$$

为了方便表示，定义一个 GIoU 项 q：

$$q = \frac{U - C}{C} = \frac{U}{C} - 1 \quad (3-11)$$

对 IoU 的求导如(3-9)，那么有对 q 的求导：

$$q_t' = \frac{\dot{U}_t}{C} - \frac{U}{C^2} \dot{C}_t = \frac{\left(C \frac{dU}{dt} - U \frac{dC}{dt} \right)}{C^2} \quad (3-12)$$

因此 GIoU 的导数为公式(3-9)和公式(3-12)的和。

以上推导代码实现如图 3-22 所示。p_dt, p_db, p_dl, p_dr 分别是 GIoU 对 t, b, l, r 的偏导数。

以此类推，预测框 X，预测框与真实框的交集 I，预测框与真实框的并集 U，包裹预测框与真实框的大框 C 分别对 t, b, l, r 求导，如下图 3-23 所示。

因为有：

```
float Ih = fmin(pred_b, truth_tblr.bot) - fmax(pred_t, truth_tblr.top);
float Iw = fmin(pred_r, truth_tblr.right) - fmax(pred_l, truth_tblr.left);
float I = Iw * Ih; //相交区域的面积
```

所以在图 2-23 中，I 对 t 的求导 dI_wrt_t 需要进行一个比较。

```
float p_dt = 0;
float p_db = 0;
float p_dl = 0;
float p_dr = 0;
if (U > 0) { //IoU对tblr的导数
    p_dt = ((U * dI_wrt_t) - (I * dU_wrt_t)) / (U * U);
    p_db = ((U * dI_wrt_b) - (I * dU_wrt_b)) / (U * U);
    p_dl = ((U * dI_wrt_l) - (I * dU_wrt_l)) / (U * U);
    p_dr = ((U * dI_wrt_r) - (I * dU_wrt_r)) / (U * U);
}
if (iou_loss == GIoU) {
    if (C > 0) { //q对tblr的导数
        p_dt += ((C * dU_wrt_t) - (U * dC_wrt_t)) / (C * C);
        p_db += ((C * dU_wrt_b) - (U * dC_wrt_b)) / (C * C);
        p_dl += ((C * dU_wrt_l) - (U * dC_wrt_l)) / (C * C);
        p_dr += ((C * dU_wrt_r) - (U * dC_wrt_r)) / (C * C);
    }
}
```

图 3-22 IOU 分别对 t、b、l、r 求导

```
// 预测框对tblr求导
float dX_wrt_t = -1 * (pred_r - pred_l);
float dX_wrt_b = pred_r - pred_l;
float dX_wrt_l = -1 * (pred_b - pred_t);
float dX_wrt_r = pred_b - pred_t;
// I对tblr求导 <预测>
float dI_wrt_t = pred_t > truth_tblr.top ? (-1 * Iw) : 0;
float dI_wrt_b = pred_b < truth_tblr.bot ? Iw : 0;
float dI_wrt_l = pred_l > truth_tblr.left ? (-1 * Ih) : 0;
float dI_wrt_r = pred_r < truth_tblr.right ? Ih : 0;
// U对tblr求导 <预测>
float dU_wrt_t = dX_wrt_t - dI_wrt_t;
float dU_wrt_b = dX_wrt_b - dI_wrt_b;
float dU_wrt_l = dX_wrt_l - dI_wrt_l;
float dU_wrt_r = dX_wrt_r - dI_wrt_r;
// C对tblr求导 <预测>
float dC_wrt_t = pred_t < truth_tblr.top ? (-1 * Cw) : 0;
float dC_wrt_b = pred_b > truth_tblr.bot ? Cw : 0;
float dC_wrt_l = pred_l < truth_tblr.left ? (-1 * Ch) : 0;
float dC_wrt_r = pred_r > truth_tblr.right ? Ch : 0;
```

图 3-23 X、I、U、C 分别代表预测框面积、交集面积，并集面积，大框 C 的面积

那么现在可以定义本实验的 `delta_yolo_box()` 函数来计算梯度，从而进行后向传播参数传递和参数优化。函数的实现如图 3-24 所示。

```
ious delta_yolo_box(box truth, float *x, float *biases, int n, int index, int i,
int j, int lw, int lh, int w, int h, float *delta, float scale, int stride,
float iou_normalizer, IOU_LOSS iou_loss) {
    ious all_ious = { 0 };
    // i - 宽度的步长
    // j - 高度的步长
    // 得到一个绝对坐标系下的预测框
    box pred = get_yolo_box(x, biases, n, index, i, j, lw, lh, w, h, stride);
    all_ious.iou = box_iou(pred, truth); // 计算 iou
    all_ious.giou = box_giou(pred, truth); // 计算 giou
    // 防止 nan
    if (pred.w == 0) { pred.w = 1.0; }
    if (pred.h == 0) { pred.h = 1.0; }
    if (iou_loss == MSE) // 旧的损失
    {
        float tx = (truth.x*lw - i); // truth: ground truth bounding box
        float ty = (truth.y*lh - j);
        float tw = log(truth.w*w / biases[2 * n]); // biases: anchors 锚定框
        float th = log(truth.h*h / biases[2 * n + 1]);
        delta[index + 0 * stride] = scale * (tx - x[index + 0 * stride]); // x: 1. output
        delta[index + 1 * stride] = scale * (ty - x[index + 1 * stride]); // scale = (2 - truth.w * truth.h)
        delta[index + 2 * stride] = scale * (tw - x[index + 2 * stride]); // stride: 1. w * l, h
        delta[index + 3 * stride] = scale * (th - x[index + 3 * stride]);
    }
    else {
        all_ious.dx_iou = dx_box_iou(pred, truth, iou_loss);
        // 分别代表 x, y, w, h 的梯度
        delta[index + 0 * stride] = (all_ious.dx_iou.dl + all_ious.dx_iou.dr);
        delta[index + 1 * stride] = (all_ious.dx_iou.dt + all_ious.dx_iou.db);
        delta[index + 2 * stride] = ((-0.5 * all_ious.dx_iou.dl) + (0.5 * all_ious.dx_iou.dr));
        delta[index + 3 * stride] = ((-0.5 * all_ious.dx_iou.dt) + (0.5 * all_ious.dx_iou.db));
        // 指数导数
        delta[index + 2 * stride] *= exp(x[index + 2 * stride]);
        delta[index + 3 * stride] *= exp(x[index + 3 * stride]);
        // 标准化 iou 权重
        delta[index + 0 * stride] *= iou_normalizer;
        delta[index + 1 * stride] *= iou_normalizer;
        delta[index + 2 * stride] *= iou_normalizer;
        delta[index + 3 * stride] *= iou_normalizer;
    }
    return all_ious; }
```

图 3-24 `delta_yolo_box()` 函数实现

3.5.3 训练采用 G10U 损失函数算法的 YOLOv3 模型

通过控制变量的方法，采用与章节 3.3 中同样数据，同样的调参方法，同样的验证集，来对比模型的训练效果。

训练的 mAP 曲线如下图 3-25, 3-26, 3-27 所示，同样分为三个阶段。

第一阶段如图 3-25:

mAP 曲线与 loss 曲线都波动较大。

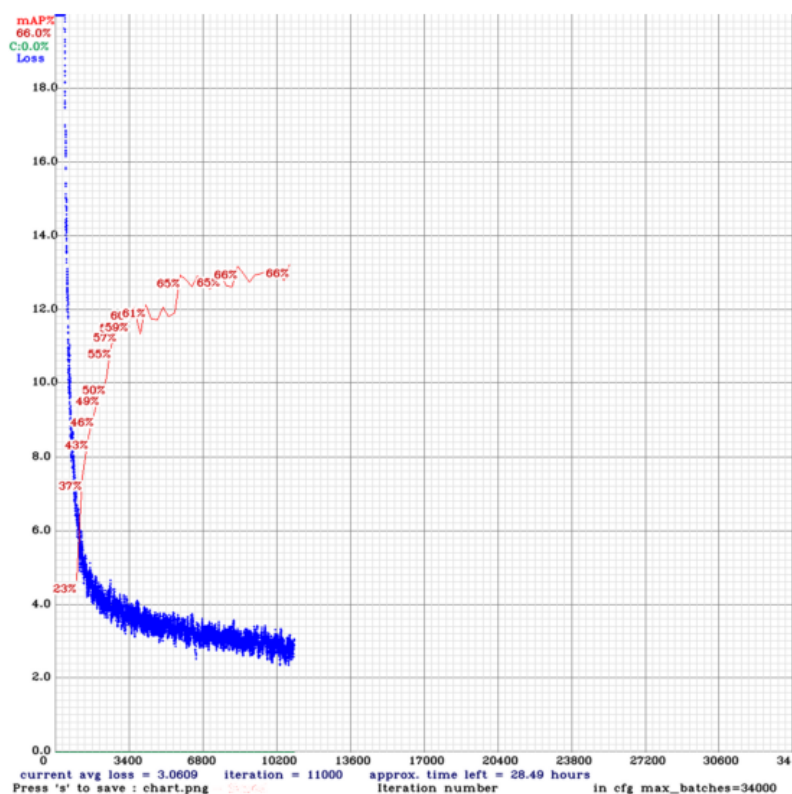


图 3-25 第一阶段 mAP 曲线与 loss 曲线

第二阶段训练 mAP 曲线与 loss 曲线如图 3-26 所示。

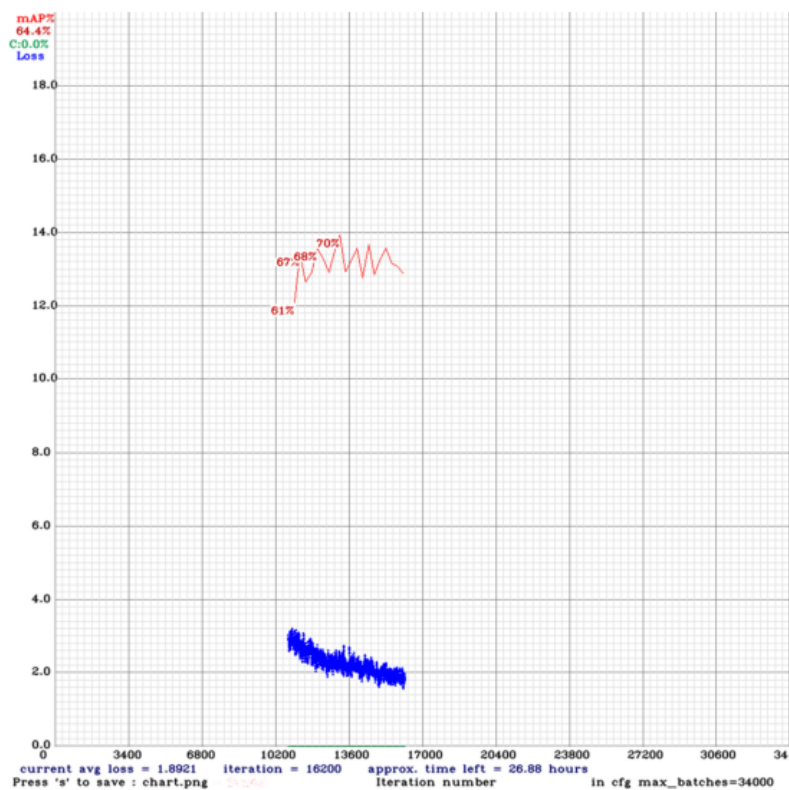


图 3-26 第二阶段 mAP 曲线与 loss 曲线

第三阶段训练 mAP 曲线与 loss 曲线如图 3-27 所示。

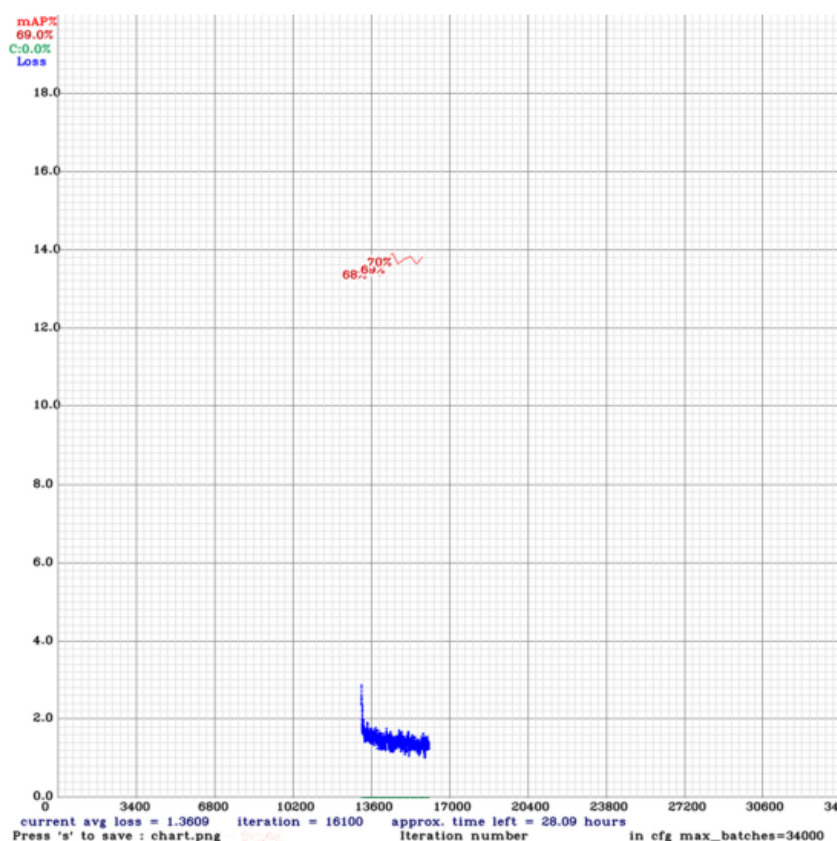


图 3-26 第二阶段 mAP 曲线与 loss 曲线

通过与 3-3 章对比发现，采用 GIoU 的 YOLOv3 模型在 mAP 的表现和 loss 表现均略逊色于原版的模型，但这可能是因为在 GIoU 的 YOLOv3 模型需要不同的调参策略来获取最优表现，而本实验为了进行对照，采用的是同样的调参策略。而且 mAP 和 loss 并非评判模型好坏的唯一标准。接下来对模型进行验证，同样从“欢迎”广告和“不欢迎”广告中各自抽取 680 张广告，与 3-3 章中抽取的是同样的广告数据集。

3.5.4 验证模型的检测准确度

与先前一样，模型验证采用两组图片作为验证集。一组图片全部由“欢迎”广告组成，另一组图片由“不欢迎”广告组成。两组图片各 680 张，都是随机选取。通过检测两组图片，得到结果验证。其中加载不成功的图片一律视为“不欢迎”广告。如果识别出 17 类“意象”中的任意一种，视为“不欢迎”广告，否则为“欢迎”广告。结果如表 3-3 所示。

表 3-3 采用 GIoU 的模型的验证结果

真实值\预测值	预测为“不欢迎”广告	预测为“欢迎广告”
“不欢迎”广告(680 张)	656 (TP)	24 (FN)
“欢迎”广告(680 张)	185 (FP)	495 (TN)

根据公式(3-1)(3-2)(3-3)(3-4)计算得到采用 GIoU 模型的精确率, 召回率, 准确率和 F1-score 分别为:

验证精确率为 78.00%;

验证召回率为 96.47%;

验证准确率为 84.63%;

验证 F1-score 分数为 86.26;

与不使用 GIoU 相比, 使用 GIoU 后除召回率外, 精确率、准确率和 F1-score 都有不同程度的下降。主要原因是, 和不使用 GIoU 相比, “欢迎”类广告被识别为“不欢迎”广告的数量增加到了 185 张。

3.5.5 检验使用 GIoU 的模型的定位识别准确度

图 3-27 是 2 组对比图片, 分别是采用 GIoU 损失函数和采用原版的 MSE 损失函数的识别效果:



图 3-27 采用不同损失函数的识别效果

可以发现在第一组图中, 左边的图中有 1 枚金币未被识别, 而采用 GIoU 作为损失函数后, 这枚金币被成功识别; 右边第二组图中, 左边的图片对弹窗的识别并不精确, 而采用 GIoU 后更加精确了。

3.6 总结与展望

基于 YOLOv3 模型，构建了广告审查系统。并制作了用于训练的数据集，有针对性地对四类“不欢迎”广告中的 17 种检测目标，即“意象”，进行了检测。最终达到了 72.75% 的 mAP 和 92.63% 验证 F1 分数。尤其是对于“不欢迎”广告的检测，效果是很理想的。

然后提出了基于 GIoU 的损失函数来替代 MSE 损失函数，取得了可以接受的效果，在牺牲一定类别精度与时间后，换来了更高的定位精度。

在本研究的基础上，之后的研究可以从两个方向进一步深入研究和改进。一个方向是提升目标检测技术的精度，通过提高 mAP 来增强模型的表现。另一个方向是从数据集入手，对检测的物体“意象”进一步细分，提高数据集的质量和针对性。

现在对于计算机视觉在商业化和工业化上面的应用已经接近饱和，本研究致力于从广告审查的角度，对计算机视觉技术进行应用，以求达到真正地将科学技术转化为生产力。但是广告审查的领域仍然有诸多值得探索的地方，有诸多具有价值并且等待被挖掘的地方。比如说广告图片中的文字符号识别，本次研究并未涉及，但这将会是一个对广告审查有很大推动促进意义的地方，如果能够通过识别文字符号来理解语义，结合计算机视觉与自然语言处理在广告审查上面的应用，那么模型对广告的理解能够更上一层楼。

参考文献

- [1] Jiao L, Zhang F, Liu F, et al. A survey of deep learning-based object detection[J]. IEEE Access, 2019, 7: 128837-128868.
- [2] Wang J, Zhang W, Yuan S. Display advertising with real-time bidding (RTB) and behavioural targeting[J]. arXiv preprint arXiv:1610.03013, 2016.
- [3] Girgis M R, Mahmoud T M, Abd-El-Hafeez T. An approach to image extraction and accurate skin detection from web pages[C]//Proceedings of World Academy of Science, Engineering and Technology. 2007: 367-375.
- [4] Ren S, He K, Girshick R, et al. Faster r-cnn: Towards real-time object detection with region proposal networks[J]. arXiv preprint arXiv:1506.01497, 2015.
- [5] Lin T Y, Dollár P, Girshick R, et al. Feature pyramid networks for object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 2117-2125.
- [6] Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real-time object detection[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 779-788.
- [7] Liu W, Anguelov D, Erhan D, et al. Ssd: Single shot multibox detector[C]//European conference on computer vision. Springer, Cham, 2016: 21-37.
- [8] Lin T Y, Goyal P, Girshick R, et al. Focal loss for dense object detection[C]//Proceedings of the IEEE international conference on computer vision. 2017: 2980-2988.
- [9] Redmon J, Farhadi A. YOLO9000: better, faster, stronger[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 7263-7271.
- [10] 江英. 图片与文本过滤技术在信息监控中的应用研究 [D]. 南昌大学, 2019.
- [11] 杨源. 基于深度学习的不良图片过滤技术研究 [D]. 北方工业大学, 2018.
- [12] 王俊境. 基于深度学习的不良图片识别研究 [D]. 华南理工大学, 2020.
- [13] Redmon J, Farhadi A. Yolov3: An incremental improvement[J]. arXiv preprint arXiv:1804.02767, 2018.
- [14] 开源数据集网站 OPEN IMAGE DATASET V6. 网址 <https://storage.googleapis.com/openimages/web/index.html>.
- [15] OIDv4 工具源码. 网址 https://github.com/EscVM/OIDv4_ToolKit
- [16] YOLOv3 结构图. (2018-09-12/). <https://blog.csdn.net/leviopku>
- [17] Rezatofighi H, Tsoi N, Gwak J Y, et al. Generalized intersection over union: A metric and a loss for bounding box regression[C]//Proceedings of the

IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019: 658-666.

附 录

以下内容来自于 680 张“不欢迎”广告的审查结果的截取片段：

data/valid02/06551700ae3b1b45565d33db4cc5b3cc.jpg: Predicted in 9.531000 milli-seconds.

Gold_Coin: 93% (left_x: 3 top_y: 1007 width: 84 height: 88)

Gold_Coin: 98% (left_x: 92 top_y: 780 width: 71 height: 73)

Red_Envelope_Big: 90% (left_x: 101 top_y: 583 width: 864 height: 760)

White_Square_Popup_Window: 42% (left_x: 157 top_y: 653 width: 758 height: 663)

Gold_Coin: 100% (left_x: 406 top_y: 605 width: 236 height: 217)

Gold_Coin: 98% (left_x: 764 top_y: 764 width: 57 height: 58)

Gold_Coin: 99% (left_x: 963 top_y: 613 width: 88 height: 89)

Enter Image Path: Detection layer: 139 - type = 28

Detection layer: 150 - type = 28

Detection layer: 161 - type = 28

data/valid02/19705168953d1aff787525adc9d8d365.jpg: Predicted in 9.604000 milli-seconds.

Leg: 100% (left_x: 2 top_y: 115 width: 295 height: 281)

Leg: 99% (left_x: 132 top_y: 214 width: 101 height: 435)

Foot: 100% (left_x: 198 top_y: 306 width: 71 height: 119)

Butt: 100% (left_x: 224 top_y: 106 width: 93 height: 103)

Leg: 96% (left_x: 536 top_y: 161 width: 156 height: 430)

Leg: 100% (left_x: 549 top_y: 152 width: 288 height: 337)

Leg: 100% (left_x: 870 top_y: 130 width: 306 height: 173)

Leg: 95% (left_x: 1019 top_y: 170 width: 203 height: 458)

Belly: 98% (left_x: 1053 top_y: -2 width: 118 height: 74)

Foot: 98% (left_x: 1111 top_y: 263 width: 123 height: 130)

Enter Image Path: Detection layer: 139 - type = 28

Detection layer: 150 - type = 28

Detection layer: 161 - type = 28

data/valid02/433880922b49df02dd25be7383cb3165.jpg: Predicted in 9.504000 milli-seconds.

White_Square_Popup_Window: 100% (left_x: 61 top_y: 239 width: 508 height: 398)

Enter Image Path: Detection layer: 139 - type = 28

Detection layer: 150 - type = 28

Detection layer: 161 - type = 28

data/valid02/169948d671f09efb7304219ddc4600ff.jpg: Predicted in 9.515000 milli-seconds.

White_Square_Popup_Window: 100% (left_x: 53 top_y: 244 width: 521 height: 386)

Enter Image Path: Detection layer: 139 - type = 28

Detection layer: 150 - type = 28

Detection layer: 161 - type = 28

data/valid02/97582204b988130c38dc81c4bc157e01.jpg: Predicted in 9.716000 milli-seconds.

White_Square_Popup_Window: 100% (left_x: 255 top_y: 347 width: 572 height: 682)

Enter Image Path: Detection layer: 139 - type = 28

Detection layer: 150 - type = 28

Detection layer: 161 - type = 28

data/valid02/756530ddb577c1cff1eb384eda17d6ab.jpg: Predicted in 9.526000 milli-seconds.

White_Square_Popup_Window: 100% (left_x: 198 top_y: 444 width: 681 height: 825)

Enter Image Path: Detection layer: 139 - type = 28

Detection layer: 150 - type = 28

Detection layer: 161 - type = 28

data/valid02/6279266b162afd8c800df767a8904965.jpg: Predicted in 9.861000 milli-seconds.

White_Square_Popup_Window: 100% (left_x: 95 top_y: 532 width: 893 height: 435)

Enter Image Path: Detection layer: 139 - type = 28
 Detection layer: 150 - type = 28
 Detection layer: 161 - type = 28
 data/valid02/29978715d6b3241d63e5272cc53db3e8.jpg: Predicted in 9.538000
 milli-seconds.

Arrow(Slide Up): 100%(left_x: 515 top_y: 833 width: 222 height:
 524)

Enter Image Path: Detection layer: 139 - type = 28
 Detection layer: 150 - type = 28
 Detection layer: 161 - type = 28
 data/valid02/0601240088b513a51e164d63cfa1f336.jpg: Predicted in 9.610000 milli-
 seconds.

Arrow(Slide Up): 100%(left_x: 515 top_y: 840 width: 225 height:
 521)

Enter Image Path: Detection layer: 139 - type = 28
 Detection layer: 150 - type = 28
 Detection layer: 161 - type = 28
 data/valid02/435459ee51fa9ad743949ca535d561b4.jpg: Predicted in 9.533000 milli-
 seconds.

White_Square_Popup_Window: 100% (left_x: 100 top_y: 158 width:
 454 height: 560)

Convert_annotations.py 核心算法片段:

```
def convert(filename_str, coords):
    os.chdir("../")
    image = cv2.imread(filename_str + ".jpg")
    coords[2] -= coords[0]
    coords[3] -= coords[1]
    x_diff = int(coords[2]/2)
    y_diff = int(coords[3]/2)
    coords[0] = coords[0]+x_diff
    coords[1] = coords[1]+y_diff
    coords[0] /= int(image.shape[1])
    coords[1] /= int(image.shape[0])
    coords[2] /= int(image.shape[1])
```

```
coords[3] /= int(image.shape[0])
os.chdir("Label")
return coords;
convert_annotations.py 核心代码
def convert(filename_str, coords):
    os.chdir("../")
    image = cv2.imread(filename_str + ".jpg")
    coords[2] -= coords[0]
    coords[3] -= coords[1]
    x_diff = int(coords[2]/2)
    y_diff = int(coords[3]/2)
    coords[0] = coords[0]+x_diff
    coords[1] = coords[1]+y_diff
    coords[0] /= int(image.shape[1])
    coords[1] /= int(image.shape[0])
    coords[2] /= int(image.shape[1])
    coords[3] /= int(image.shape[0])
    os.chdir("Label")
    return cords
```

以上为附录全部内容。

致 谢

感谢张老师和庄老师对我在毕设上孜孜不倦的指导！

感谢李爽学姐和邵宁学长对我的诸多帮助！

感谢张老师在学习和生活上对我的指导和关照！在张老师的指导和建议下，我研究生决定去读匹兹堡大学的信息与计算学院的信息科学硕士。感谢张老师给我未来的规划指点迷津！

感谢一直支持我的家人和朋友！