



**本科毕业论文**

**生活类爬虫系统**

**刘炜铭**

**201527010414**

|  |  |
| --- | --- |
| 指导教师 | **綦羽 副教授** |

|  |  |  |  |
| --- | --- | --- | --- |
| 学 院 名 称 | 软件学院 | 专 业 名 称 | 软件工程 |
| 论文提交日期 | 2018 年 4 月 15 日 | 论文答辩日期 | 2018 年 4 月 21 日 |

**摘 要**

网络爬虫其实是一个探测机器，它的基本行为就是模拟人去各个网站看看，然后带着我们所需的信息回来。我们使用的百度搜索引擎，它其实也是一个爬虫，每天放出无数个爬虫到各个网站，把信息带回来后，再提供给人们去检索。为了让某个网站更加容易被检索出来，我们可以通过爬虫提高它的点击率，这个过程就叫做 SEO（搜索引擎优化）。现在出现很多聚合电商，他们自己不生产任何产品，他们是搬运工，把天猫、淘宝、京东等商品聚合起来让用户筛选，成为电商中的百度，从而获益。可见网络爬虫与我们息息相关，它能极大的”优化”我们的生活。可以这样认为：每个爬虫都是你的“分身”，就像孙悟空拔了一撮汗毛，吹出一堆猴子一样。

本文将研究并开发一个生活类爬虫软件，它可以第一时间提示并且抢购附近低价美食、推送有关编程技术类型的文章、正在热播和准备上映的电影和机票价格等。为了方便使用，本文将用chrome插件作为客户端，让用户只要打开流浪器，就可以实时收到更新消息。从技术架构上，前端将使用react和material-ui框架开发。web服务后台使用python3的flask框架，为了提高系统稳定性，本文将介绍nginx加uwsgi等反向代理工具，开发一个多进程、单线程的web服务后台。爬虫系统使用crontab定时python3任务和xpath语法分析网页html结构，再使用正则表达式筛选正真所需的内容。使用mysql:5.70作为持久层，redis作为缓存，docker作为虚拟化工具，sqlalchemy作为orm框架，alembic作为数据库表版本控制。

关键词：网络爬虫 chrome插件 xpath crontab

**Implementation and Analysis web crawler application**

Liu Weiming

(College of Software Engineering, South China Agricultural University, Guangzhou 510642, China)

**Abstract:** In fact, web crawler is a detection machine. It`s base action is searching on the internet like common human do and back with some message, which is what we want. BaiDu Search Engines, a web crawler as well, BaiDu product a lot of web crawler every day, back and transfer to useful message, open for us searching. We can use web crawler to increase a website click through rate so that will be more easily retrieve, this skill is called SEO(Search Engine Optimization). Now is coming up so many polymeric e-commerce, they don`t need to produce own products, just showing commodity from TianMao, TaoBao, JinDong and so on, become the BaiDu of e-commerce. It easy to see, web crawler connected our life, it can make a hug convenient to our life. It can be considered that each web crawler is a divided yourself, like Sun Wukong pulling out a pinch of hair and blowing out a pile of monkeys.

This article is going to build a life-use web crawler application, the crawler will push and buy cheapest food around your city, showing the blog of programming, movie on show and upcoming, air ticket price and so on. For easier use, this article will build a chrome extensions for client, so user can receive message only by opening chrome browser. From the technical framework, using react and material-ui framework for client. Web server will use python3-flask framework, to more stable, this article will introduce nginx and uwsgi reverse proxy tools to build a multi process, single thread web server application. Web crawler system will use linux-crontab timing task python3 job and xpath syntax analysis website html structure, regex for select the information we want. Mysql:5.70 for database, redis for cache, docker for virtualization, sqlalchemy for orm framework, and alembic for database version control.

**Key words:** Web-crawler Chrome-extensions xpath crontab

目 录

[1 引言 5](#_Toc4086746)

[2 相关知识介绍 7](#_Toc4086747)

[3 实验与分析 16](#_Toc4086748)

[4 结论与展望 27](#_Toc4086749)

[参 考 文 献 29](#_Toc4086750)

[致 谢 31](#_Toc4086751)

# 引言

* 1. 选题的背景与意义

爬虫技术是一种最贴近人类生活的自动化技术。既可为AI自动提供有效的大数据，也可以方便人类生活。比如你的短期计划：租房、旅游、聚会。往往希望有个工具为了筛选心仪的对象。可是网上太多信息了，对于一个有其他重任的人来说，要精打细算浪费的时间（也就是金钱）是不划算的。那么很希望有这样一个软件，来帮你自动筛选，把你想要的消息推送给你。

计算机与人类息息相关，计算机的自动化特性更是取代了大多数疲惫又杂乱的工作，工人易犯错且累，计算机精准而不累。爬虫作为自动化技术的一员，对人类生活起着重要作用。

（1）搜索引擎

搜索引擎的建立需要大量的数据支撑，假如数据样本充足，人们采集信息将会极其对称且便利，人工采集显得有心无力，爬虫技术将可以代替人们源源不断采集数据。

（2）电商领域

如今电商领域，天猫、淘宝、京东、拼多多......信息五彩缤纷，价格也有高有底。如何快速搜索，快速对比成为一大难题。爬虫具有自动化的特征，可以夸张地说，爬虫可以模拟人类一切点击行为，于是聚合电商帮人类解决了这类问题。

（3）出行领域

堵车预报，天气预报，机票、动车价格，买票抢票等等，成为琐碎又不得不去关注的事情，人们每天都浪费很多时间在此。爬虫具有自动提醒功能，可以帮人们省去查看各种事儿的时间。

* 1. 国内外研究现状

自1993年初 Matthew Grays Wandered 在麻省理工学院开发出有史记载的第一个网络爬虫以来，爬虫技术已经发展了20多年，技术已日趋多样。为满足不同用户的不同需求，爬虫分为以下两大类:

（1）通用式爬虫

通用式爬虫指的是通过一系列预设的初始链接开始，根据规则提取当前网页的所需内容，然后提取当前网页里面所有的其他链接，再把这些新链接使用open-close表判重后，放去等待队列，直到到达系统预设指定条件为止的全网式爬虫技术。此类爬虫通常有两类不同的实现方式，可分为:

（1.1）深度优先搜索策略(Depth first search)，指的是从初始链接开始，选出第一个链接进行访问，处理出有用信息和新的链接集合，再从新的链接集合里面选出一个重复上诉过程，直到没有新的链接，就返回到上一层中选出一个链接进行访问。此策略具有极强的专攻性，可以爬取到最深入灵魂的隐蔽信息。

（1.2）广度优先搜索策略(Breadth first search)，又或者叫宽度优先搜索策略，指的是从初始链接开始，先把所有初始链接处理完毕，再处理从初试链接生成出来的其他链接，也就是说是一层一层的处理，这样就避免了深度优先搜索策略（1.1）会使得有些链接没机会处理的情况。

（2）倍增式爬虫(Incremental Web Crawler)，倍增式爬虫解决了僵尸链接问题，主要体现在：网站长久不更新，没有新的信息获取。这个问题对于通用式爬虫来说是一种诟病。倍增式爬虫根据网页更新的频率来自动调整网页权重，具有较高的权重的网页可以占用更高的硬件和网络资源，权重最低的只需分配剩下的计算资源即可。

* 1. 论文内容与结构安排

本文将研究并开发一个生活类爬虫软件，包括从生产到正式环境部署的过程，并且投入到实际生活使用。本文主要的工作和内容分为以下几个章节：

第一章主要介绍了网络爬虫这一领域的研究背景与意义，分析了这一领域的研究现状和应用方向，说明网络爬虫这种自动化技术已经长达20年的研究历史及人们生活对此的依赖，同时也说明爬虫与我们生活息息相关。

第二章主要介绍了整个爬虫软件所使用的相关技术知识。

第三章主要介绍了整个爬虫软件的架构与设计。

第四章主要对整个毕业设计作出总结，并分析可拓展之处。

# 相关知识介绍

* 1. React框架介绍

React 起源于 Facebook 的内部项目，因为该公司对市场上所有 JavaScript MVC 框架，都不满意，就决定自己写一套，用来架设 Instagram 的网站。做出来以后，发现这套东西很好用，就在2013年5月开源了。作为现在最火的前端框架react有着组件化，状态管理，虚拟DOM等优势，其扩展项目react-native目标更是宏大，希望用写 Web App 的方式去写 Native App。如果能够实现，整个互联网行业都会被颠覆，因为同一组人只需要写一次 UI ，就能同时运行在服务器、浏览器和手机。

React生命周期可以分为七个钩子函数，分别是：

1.componentWillMount

2.componentDidMount

3.componentWillReceiveProps

4.sholudComponentUpdate(这是一个性能优化点)

5.componentWillUpdate

6.componentDidUpdate

7.componentWillUnmount

每次钩子函数监听到变化，会重新调用render函数执行UI渲染，根据react-diff算法，不必刷新全部DOM，使得渲染效率大大提高。

React-diff算法的优势：

传统 diff 算法：传统 diff 算法通过循环递归对节点进行依次对比，效率低下，算法复杂度达到 O(n^3)，其中 n 是树中节点的总数。O(n^3) 到底有多可怕，这意味着如果要展示1000个节点，就要依次执行上十亿次的比较。这种指数型的性能消耗对于前端渲染场景来说代价太高了！

react-diff 策略，使得渲染达到O(n)复杂度。

1.Web UI 中 DOM 节点跨层级的移动操作特别少，可以忽略不计。

2.拥有相同类的两个组件将会生成相似的树形结构，拥有不同类的两个组件将会生成不同的树形结构。

3.对于同一层级的一组子节点，它们可以通过唯一 id 进行区分。

下图是React生命周期逻辑图。



* 1. Material-UI框架介绍

Material-UI是在react最流行的UI库，遵循material design，具有拟物化、质感风格、强烈的阴影和高光。

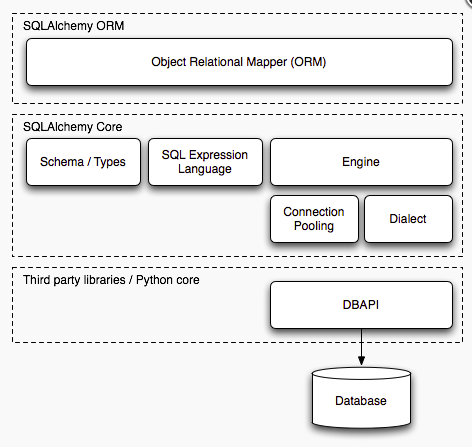
* 1. Python3-flask框架介绍

Flask是一个使用python编写的轻量级web应用框架，是基于werkzeug wsgi工具箱和jinjia2模板引擎的开源框架。由于它足够精简而且易上手，易配置扩展等特性，在github收到42731个start，相比与dingo，flask的可配置显得十分灵活。

* 1. Sqlalchemy框架介绍

Sqlalchemy是一个由python编写的orm库。提供了SQL工具包和对象关系映射工具，sqlalchemy把数据库表表达为python的list，每个list是一个truple，表示一行记录。Sqlalchemy采用的是数据映射模型，并非Active Record模型。加上python这类动态解析语言等特性，想到达到操作数据库的目的，往往只需要写一个公共使用的ORM文件即可，极其方便。SQLAlchemy首次发行于2006年2月，并迅速地在Python社区中最广泛使用的ORM工具之一。

Sqlalchemy采用两层架构，如图所示，最底层的操作数据库采用第三方库的形式完成。



* 1. Alembic框架介绍

Alembic是一个基于sqlalchemy的数据库版本迁移工具，在往后开发中，往往需要增表改表，alembic可以做到类似git一样的，保存数据库的各种历史版本，随着代码更新而部署不同的数据库结构。

* 1. MySQL:5.70数据库介绍

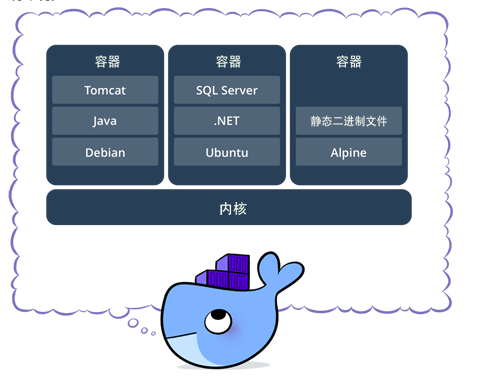
MySQL是一个关系形数据库管理系统，是最好用web-base应用数据库之一，支持事务、表级、行级锁等等。目前属于Oracle下的产品，是一个开源软件。其中，5.70是mysql一个比较稳定的版本，也是网上技术文章最多的版本，本文使用此版本作为主要存储工具。

* 1. Redis介绍

Redis是一个内存型的key-value数据库，支持hash，string, list, set，order\_set等多种数据结构。内存型数据库能火爆体现在：1、内存越来越廉价，网络速度越来越快。2、效率比落磁盘要高得多。3、redis也能极大可能保存用户存取的数据。Redis是一个开源项目，于2010年3月15日进行开发。

* 1. Docker虚拟化技术介绍

在Docker问世之前，部署项目代码视乎不是一件容易的事情。还有专门运维的团队，由于直接在服务器上安装依赖往往会弄坏系统变量，导致部署流程很繁琐。Docker提供一种虚拟化技术，支持把服务运行在docker-deam下的container中。一句话概括容器就是将软件打包成标准化单元，以用于开发、交付和部署。如下图所示，每一个container是独立的，因为每个container都有独立的PID，NET，IPC, USER, UTS, MNT，所以每一个container都是互不影响的，只有这样，我们才可以为每一个container安装不同的依赖而不影响宿主机。



相比于虚拟机，docker的优势是很明显的。如下图所示，传统的虚拟机技术首先需要虚拟出一套硬件，再在其上面运行一个完整的操作系统功能，在该系统上再运行所需的进程。而docker的容器内的应用进程则直接运行于宿主机内核中，容器没有自己的内核，而且没有进行硬件虚拟。因此容器要比传统虚拟机更为轻便。



容器与虚拟机对比总结图如下：



* 1. Linux-crontab介绍

Linux下的定时任务，crontab成为很多服务器定时任务的首选。在LINUX中，周期执行的任务一般由cron这个守护进程来处理。cron读取一个或多个配置文件，这些配置文件中包含了命令行及其调用时间。cron的配置文件称为“crontab”，是“cron table”的简写。

crontab是在后台维护一个守护进程，每一分钟调用一次，每一次调用，则去三个地方查找crontab文件，进行解析，运行对应语句，然后主程序休眠，如此反复。

1. /var/spool/cron/ 这个目录下存放的是每个用户包括root的crontab任务，每个任务以创建者的名字命名，比如vimi建的crontab任务对应的文件就是/var/spool/cron/vimi
2. /etc/crontab/ 这个文件负责安排由系统管理员制定的crontab
3. /etc/cron.d/ 这个目录用来存放任何要执行的crontab文件或脚本

* 1. xpath路径表达式语法介绍

xpath是一种选择XML文件中节点的方法，HTML文档也是XML文档的一种，利用xpath语法可以很方便地找出对应元素的对应DOM。

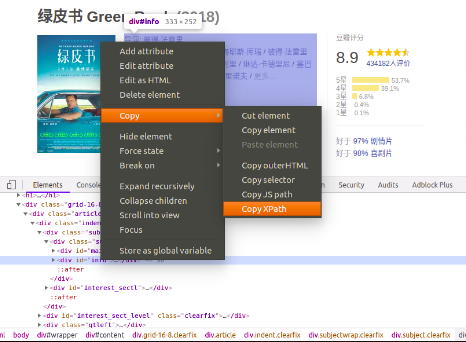
xpath语法介绍:

|  |  |
| --- | --- |
| nodename | 选取此节点的所有子节点。 |
| / | 从根节点选取。 |
| // | 从匹配选择的当前节点选择文档中的节点，而不考虑它们的位置。 |
| . | 选取当前节点。 |
| .. | 选取当前节点的父节点。 |
| @ | 选取属性。 |

谓语（Predicates）

|  |  |
| --- | --- |
| //div/dd[1] | 选取div元素的第一个dd子元素 |
| //div/dd[last()] | 选取div元素的最后一个dd子元素 |
| //title[@lang=’eng’] | 选取所有title元素，有值为end的lang元素 |
| //title[@lang] | 选取所有title元素，有lang元素 |
| \* | 匹配任何元素节点 |
| @\* | 匹配任何属性节点 |

更方便的是，对于chrome浏览器，打开调试模式可以很容易复制当前元素的xpth路径。



# 实验与分析

* 1. 实验准备

本文实验的硬件配置为 Intel Core i7-6700 处理器、NVIDIA GeForce GTX 1080 Ti 显卡、8GB RAM 的主机，软件环境为 Ubuntu 16.04 LTS 系统、GCC 5.4、cuda 9.0、OpenCV

3.4。

YOLO 所使用的深度学习框架是作者自己编写的 Darknet 框架，相比于主流的 Caffe 框架和 TensorFlow 框架，Darknet 框架有体量比较轻、源码易于阅读等优点，不过 Darknet 无法在训练中途可视化损失函数的变化过程，也不提供计算 mAP 的接口。

实验前期的环境配置比较复杂，首先要安装好最新的 NVIDIA 显卡驱动，然后安装用于 GPU 加速的 cuda 库和用于深度学习加速的 cudnn 库，最后还需要安装用于在测试时显示图片的 OpenCV。

完成环境的配置后，就可以用 git 把 Darknet 框架 clone 到本地，修改编译参数开启在编译时使用 cuda、cudnn 和 OpenCV。Darknet 框架使用 C 语言编写，直接在目录下执行

make 命令即可进行编译。编译成功后，使用 Darknet 自带的 YOLO 网络参数和权重对样例图片进行测试，成功检测到目标并且成功调用 OpenCV 进行显示。

* 1. 使用 PASCAL VOC 数据集训练 YOLO 网络

首先需要下载 VOC2007 和 VOC2012 训练数据集，一共包含 16551 张带有 PASCAL

VOC 标准标注的图片，然后使用 YOLO 提供的 voc\_label.py 脚本将 PASCAL VOC 标注转换为 Darknet 自己的标注格式。

接下来修改 Darknet 的训练数据集配置文件 cfg/voc.data：

classes= 20

train = train/train.txt valid = train/2007\_test.txt names = data/voc.names

backup = backup

其中 classes 为类别数，train 为存放训练数据集图片列表的文件，这个文件由上一步

的 voc\_label.py 脚本生成，valid 是之后用于计算 mAP 的测试数据集文件，这里暂时用不到，names 是存放全部类别名的文件，backup 是训练中保存网络权重的文件夹。

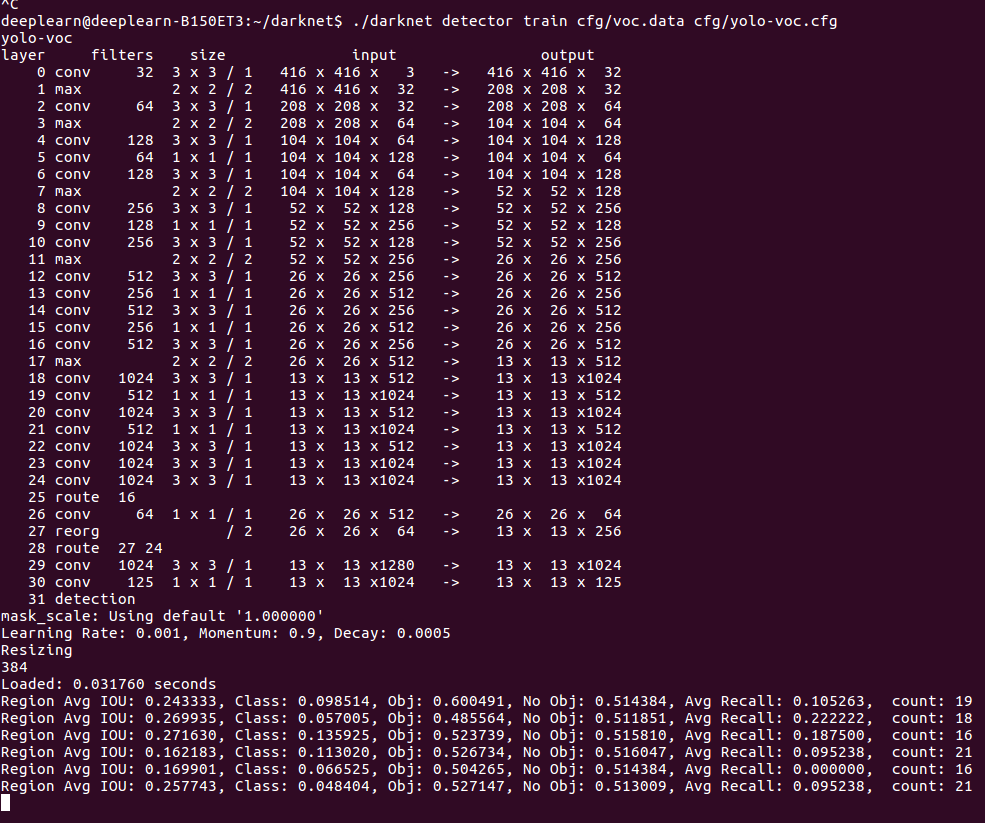


图 6 开始训练神经网络

Darknet 中已经有用于 20 类 YOLO v2 目标识别网络的配置参数，如图 6 所示直接执行训练命令就可以开始训练。随着训练的过程，可以看到 IOU、Recall 等指标逐渐升高， 代表网络的性能逐渐提高。在 NVIDIA GeForce GTX 1080 Ti 上训练的时间约为 2-3 天， 在训练中每隔一定的步数会保存一次权重文件，所以遇到以外中断的时候可以从中间继续训练，也可以直接使用中间的权重文件来进行测试，检验训练的大概效果。

|  |  |
| --- | --- |
|  |  |
|  |  |

图 7 使用 PASCAL VOC 数据集训练的模型的测试效果

网络训练好之后，就可以按照上一步去测试一些数据集中的图片，如图 7 所示，YOLO 网络能够在目标较小，有重叠的情况下准确地识别出物体的类型与位置。

* 1. 评价网络

根据性能侧重点不同，目标检测有许多评价指标，比如检测精度，检测效率，定位准确性。为了验证官方声明的 YOLO v2 在 PASCAL VOC 数据集上的性能，本文将训练好的网络用 VOC2007 测试数据集进行测试，再根据测试结果计算 mAP，VOC2007 测试数据集共包含 4952 张带标注的图片。

根据 PASCAL VOC 挑战赛对 mAP 的定义，对于每个类，首先需要设置不同的置信度阀值，求出一组准确率和召回率，以准确率为 Y 轴、召回率为 X 轴，可以画出 Precision-

Recall 曲线，Precision-Recall 曲线下边的面积就是此类的 AP，对所有类的 AP 求平均值即为 mAP，因为这里要求 20 类总的 mAP，所以本文不画出具体每一类的 Precision-Recall 曲线，只计算其 AP。

Darknet 的 valid 接口可以批量输出测试的结果，结果包括目标的类别、置信度、坐标和宽高。利用这个输出文件，首先对预测结果根据置信度由高到低排序，然后利用如下代码：

for d in range(nd):

R = class\_recs[image\_ids[d]] bb = BB[d, :].astype(float) ovmax = -np.inf

BBGT = R['bbox'].astype(float) if BBGT.size > 0:

ixmin = np.maximum(BBGT[:, 0], bb[0])

iymin = np.maximum(BBGT[:, 1], bb[1])

ixmax = np.minimum(BBGT[:, 2], bb[2])

iymax = np.minimum(BBGT[:, 3], bb[3]) iw = np.maximum(ixmax - ixmin + 1., 0.) ih = np.maximum(iymax - iymin + 1., 0.) inters = iw \* ih

uni = ((bb[2] - bb[0] + 1.) \* (bb[3] - bb[1] + 1.) +

(BBGT[:, 2] - BBGT[:, 0] + 1.) \*

(BBGT[:, 3] - BBGT[:, 1] + 1.) - inters)

overlaps = inters / uni ovmax = np.max(overlaps) jmax = np.argmax(overlaps)

if ovmax > ovthresh:

if not R['difficult'][jmax]: if not R['det'][jmax]:

tp[d] = 1.

R['det'][jmax] = 1

else:

else:

fp[d] = 1.

fp[d] = 1.

这段代码中，对于每个预测的目标边框，在对应图片中寻找与其 IOU（交叠率）最大

的未被预测的标注目标边框，假如最大的 IOU 超过阀值（0.5），就认为这个预测是成功的，并且把对应的标注目标边框设为已被预测。tp 代表正确的预测，fp 代表错误的预测。因为前边以对置信度排序，所以直接对 tp 数组和 fp 数组求前缀和再除以总预测数和

总标注目标数即可得到准确率和召回率数组，代码如下：

fp = np.cumsum(fp) tp = np.cumsum(tp) rec = tp / float(npos)

prec = tp / np.maximum(tp + fp, np.finfo(np.float64).eps)

利用准确率和召回率计算 AP 的 python 代码如下：

def voc\_ap(rec, prec):

mrec = np.concatenate(([0.], rec, [1.]))

mpre = np.concatenate(([0.], prec, [0.])) for i in range(mpre.size - 1, 0, -1):

mpre[i - 1] = np.maximum(mpre[i - 1], mpre[i])

i = np.where(mrec[1:] != mrec[:-1])[0]

ap = np.sum((mrec[i + 1] - mrec[i]) \* mpre[i + 1]) return ap

假设测试数据集中一共有 N 个目标，那么通过设置 N 个置信度阀值可以得到 N 对准

确率和召回率，对这 N 组数据根据召回率排序，然后因为 Precision-Recall 曲线在整体上会呈下降趋势（局部会存在上升，不影响），那么利用上边这段代码，设置 N 个召回率阀值（1/N，2/N，...，N/N），每次计算召回率大于阀值的最大准确率，然后对这 N 个最大准确率取平均值就是 AP。

上边这几段代码就可以合并为求单个类 AP 的脚本，那么只需另外写一个分别调用这个脚本求 20 个类的 AP 并统计求平均值的脚本，就可以计算出检测网络的 mAP。

另外，如果要对单个类求 Precision-Recall 曲线，只需要将中间得到的 N 对准确率和召回率传入 matplotlib.pyplot 库的 plot 接口，就能自动生成 Precision-Recall 曲线，再用

show 接口进行窗口显示以及保存到文件。

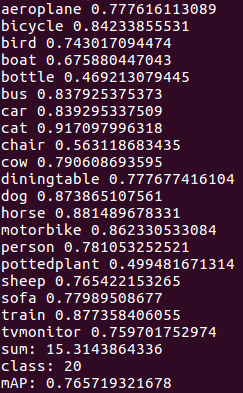


图 8 20 类的 AP 和计算平均值得到的 mAP

如图 8 所示，求出全部 20 类的 AP 之后，计算平均值即可得到 mAP，约为 76.6%， 与 YOLO 作者计算的结果（76.8%）基本一致。

* 1. 使用黑白花奶牛数据集训练 YOLO 网络

黑白花奶牛数据集只有一个类别，其训练数据集包含 10584 张较为清晰、目标明显的

黑白花奶牛图片，而测试数据集包含 4291 张存在遮挡、模糊、背光、目标较小和几个目标紧挨在一起的情况的黑白花奶牛图片。按 PASCAL VOC 标准完成数据集的标注之后， 和 3.2 中一样使用脚本转换为 Darknet 使用的标注格式，编辑相应的数据集配置文件，把

20 类修改为 1 类。因为类别数量变了，所以网络结构的配置文件中需要把最后一个卷积层的卷积核数量从 125 修改为 30（计算公式为 5×（类别数+5）），然后按和 3.2 中相同的步骤完成剩下的训练。

完成训练后，就可以按 3.3 中相同的步骤来测试以及根据测试结果计算检测网络的

mAP。

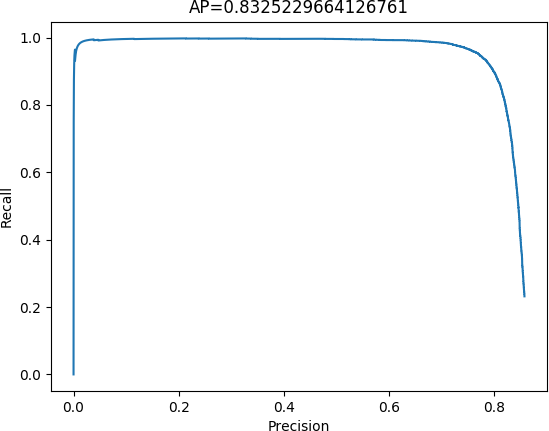


图 9 黑白花奶牛网络的 Precision-Recall 曲线

因为黑白花奶牛数据集只有一个类，其 AP 即为这个网络的 mAP。如图 9 所示，根据一组准确率和召回率用 3.3 中的 python 脚本绘制 Precision-Recall 曲线，最后计算曲线下的面积 AP=83.3%。

在单类检测网络中，这样的评分不算高。结合数据集的和 YOLO 网络的特点，本文认为评分不高的原因主要是测试数据集存在遮挡、模糊、背光、目标较小和几个目标紧挨在一起的情况，而训练集的图片绝大部分都是清晰明显的奶牛头部，这样就比较考验目标检测模型的泛化能力。

因此，接下来本文通过 Darknet 的 test 接口批量测试小部分测试数据集中的图片，并通过 OpenCV 直接进行窗口显示，对其中存在遮挡、模糊、背光、目标较小和几个目标紧挨在一起情况的图片保存下来，然后分析 YOLO 目标检测网络具体在哪种情况下表现不好。

|  |  |
| --- | --- |
| A | B |
| C | D |

图 10 黑白花奶牛网络的测试结果

如图 10 所示，从测试集选取了一些图片进行测试，发现 YOLO 模型在反光（图A）、重叠（图 B）、遮挡（图 C）的情况表现不错，但在目标非常小且重叠（图 D） 的情况下就无法准确识别出全部目标，这应该是由于 YOLO 识别时将图片划分成一定数量的网格。

* 1. 使用 SSD 模型训练网络对比与分析

在 PASCAL VOC 数据集上 YOLO v2 的精度稍微优于 SSD300，本文将用黑白花奶牛数据集对 SSD 模型进行训练，再与 YOLO 的结果进行各方面的对比。

进行YOLO 实验已配置好各种环境，要运行SSD 只需要再安装TensorFlow-GPU 1.6， 然后下载 SSD-TensorFlow 即可。

与 YOLO 类似，把数据集用脚本转换为 TensorFlow 的格式后就可以进行测试、训练和评分。本文直接使用黑白花奶牛数据集训练一个 SSD300 网络并用 TensorFlow 自带的接口进行评价，最后计算出的 mAP 为 78.4%，略低于 YOLO v2 的结果，这个差距略大于两者在 PASCAL VOC 数据集上的表现的差距（74.3%和 76.8%）。

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
| YOLO v2 | SSD300 |

图 11 YOLO 和 SSD 对小目标的测试结果对比

如图 11 所示，选取了一些存在大量小目标的图片在 YOLO 和 SSD 下进行对比，可以看出 YOLO 在小目标的识别上比 SSD 表现更优秀，但还是不能完全识别出图片上所有最小的目标，本文认为这种现象的根本原因是 YOLO 与 SSD 这类直接回归思想的模型为了提高速度，固定了输入尺寸并划分为固定数量的网格，如果多个目标挤在同一个网格中就会难以检测，总的来说是一个速度与精度的平衡问题。

|  |  |
| --- | --- |
|  |  |
| YOLO v2 | SSD300 |

图 12 YOLO 和 SSD 对侧面目标的测试结果对比

如图 12 所示，为了检验 YOLO 和 SSD 对存在角度变换的目标的鲁棒性，选取了一张从侧后方拍摄的黑白花奶牛图片，对这张图片，YOLO 以较高的置信度识别并准确定位出牛头，而 SSD 没有成功检测到牛头（或置信度小于 25%不显示）。

|  |  |
| --- | --- |
|  |  |
| YOLO v2 | SSD300 |

图 13 YOLO 和 SSD 对低分辨率图片的测试结果对比

但是 YOLO 也有不如 SSD 的地方，如图 13 所示，对于这张分辨率比较低的图片，

YOLO 的表现非常差，而 SSD 正确地预测到了全部目标。

|  |  |
| --- | --- |
|  |  |
| YOLO v2 | SSD300 |

图 14 YOLO 存在对背景的误预测情况

如图 14 所示，在某些背景占整张图片的比例比较大，并且背景中存在一定的不规则纹理的图片中，YOLO 容易把背景误识别为目标。

总的来说，YOLO 和 SSD 在设计上的大的方向以及精度与速度的平衡上是大致相同的，在实现的细节上的不同导致两者在不同方面的细节表现存在差距，在 PASCAL

VOC 数据集和黑白花奶牛数据集测试计算的 mAP 都是 YOLO 略优于 SSD。

# 结论与展望

4.1 总结

本文对目前比较热门的目标检测领域中的 YOLO 目标检测模型进行了实验和分析， 分别使用官方的 PASCAL VOC 和自己的黑白花奶牛数据集去训练了神经网络并根据

PASCAL VOC 标准进行了评价。在 PASCAL VOC 数据集上，本文测试后计算的 mAP 为

76.6%，与官方声明的 76.8%基本一致。然后，本文基于黑白花奶牛数据集分别在 YOLO

v2 模型和 SSD300 模型下训练检测网络并评价，YOLO v2 检测网络测试结果计算的 mAP 为 83.3%，SSD300 检测网络测试结果计算的 mAP 为 78.4%，这个差距略大于两者在PASCAL VOC 数据集上的表现的差距。最后，本文将在黑白花奶牛测试数据集中挑选出一些 YOLO v2 模型和 SSD300 模型检测结果差别较大的图片，更进一步分析两者之间的差别，得出了 SSD 在小目标检测上表现较差的结论。

本文也有很多不足的地方，在实验与分析方面，本文只对 2 个目标检测模型进行了实验，没有对目标检测中另一个方向的 R-CNN 系列模型进行实验。在改造模型对自己的数据集进行训练的时候，没有考虑到需要先使用聚类算法算出 anchors 的尺寸，而是直接使用了 PASCAL VOC 数据集的 anchors。实验之后也没有去思考如何去进一步训练或改进模型使评分达到更高。在实验结束后撰写论文期间，YOLO 推出了 v3 版本，由于时间不足也没有对其进行实验。

总的来说，在毕业设计和论文撰写和过程中，能够做到以认真的态度面对，认真思考和钻研问题。虽然在设计中遇到了很多问题，但是能够耐心地面对问题，解决问题，从问题中汲取经验教训，能够通过不断的努力，完成了毕业设计。

4.2 展望

在毕业设计与撰写论文的过程中，从零开始学习了深度学习的相关知识。在理解算法的过程中遇到很多问题，在配置环境和进行实验的过程中遇到了很多报错，运用了大学四年所学的知识，也在网上搜索的过程中学到了很多新的知识和技术。最终通过自己的努力，完成了项目。在此过程中也少不了老师的指导和同学的帮助，通过与其他人的沟通学习和解决问题也是作为一个软件工程师必不可少的技能。

选择基于 YOLO 模型的目标检测算法实现与分析这个课题的原因是之后的工作和目标检测相关，希望提前对这个领域进行了解，令自己的学习和工作能够相辅相成。

对于毕业设计中的 YOLO 模型和 SSD 模型，希望能够进一步理解并改进模型，而不

是止步于实验和分析性能。也希望有机会能把目标识别模型运用到实际工程中，构建一个实用的目标识别系统。

目标识别领域的进步日新月异，希望自己能够主动去了解更多新的知识和技术。

最后，希望自己能够从毕业设计中吸取经验和教训，以后的工作和学习中思考问题的方式能够更加的成熟与全面，处理问题的时候能够更加的严谨。

# 参 考 文 献

Cai Z, Fan Q Feris R S, et al. A unified multi-scale deep convolutional neural network for fast object detection[C]. European Conference on Computer Vision, 2016: 354-370.

Dai J, Li Y, He K, et al. R-FCN: Object Detection via Region-based Fully Convolutional Networks[C]. Neurl Information Processing Systems, 2016: 379-387.

Everingham M, Van Gool L, Williams C K I, et al. The PASCAL Visual Object Classes (VOC) Challenge[J]. International Journal of Computer Vision, 2010, 88(2): 303-338.

Girshick R, Donahue J, Darrell T, et al. Rich feature hierarchies for accurate object detection and semantic segmentation[C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014: 580-587.

Girshick R. Fast R-CNN[C]. Proceedings of the IEEE International Conference on Computer Vision, 2015: 1440-1448.

He K, Gkioxari G, Dollár P, et al. Mask R -CNN[J]. arXiv preprint arXiv: 1703.06870, 2017. Hinton G E, Salakhutdinov R R. Reducing the dimensionality of data with neural networks[J].

Science, 2006, 313(5786): 504-507.

Hinton G E, Osindero S, The Y W. A fast learning algorithm for deep belief nets[J]. Neural Computation, 2006, 18(7): 1527-1554.

Lecun Y, Bottou L, Bengio Y Haffner P. Gradient-Based Learning Applied to Document Recognition[J]. Proceedings of the IEEE 1998, 86(11): 2278-2324

Lin T Y, Dollár P, Girshick R, et al. Feature Pyramid Networks for Object Detection[J]. arXiv preprint arXiv: 1612.03144, 2016

Liu W, Anguelov D, Erhan D, et al. SSD: Single shot multibox detector[C]. European Conference on Computer Vision, 2016: 21-37

Redmon J, Divvala S, Girshick R, et al. You only look once: unified, realtime object detection[C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016: 779-788.

Redmon J, Farhadi A. YOLO9000: better, faster, stronger[J]. arXiv preprint arXiv: 1612.08242, 2016: 1-9.

Ren S Q, He K M, Girshick R, et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks[C]. Advance in Neural Information Processing Systems, 2015: 91-99.

Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions[C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015: 1-9.

# 致 谢

在学士学位论文完成之际，由衷地感谢高月芳老师和陈湘骥老师的悉心指导以及在毕业设计过程中提供过帮助的同学。

本论文是在高月芳老师和陈湘骥老师的共同指导下完成的，高月芳老师在深度学习的入门与目标检测算法的实验和分析方法方面给了我非常重要的意见，并且提供了实验硬件环境，陈湘骥老师在目标检测算法的实际运用以及论文撰写过程中的注意事项上给予了我很多有价值的意见，让我避免了许多错误和少走了许多弯路。在此，再次感谢高月芳老师和陈湘骥老师认真的工作和宝贵的意见。

在大学毕业之际，感谢四年来教导过我的每一位老师，感谢他们的悉心教导以及在我学业上提供的种种帮助，因为你们尽心尽责地工作和指导，我们不仅收获了大学的专业的知识，更重要的是，教给了我们学习的方法，让我们受用终生。感谢陪伴在身边四年的同学，我们一起学习，一起找工作，一起生活，感谢你们的帮助和陪伴，感谢你们充实了我的大学。

感谢家人，感谢你们这么多年来的支持和爱护，感谢你们无私的付出，感谢你们在我学习和生活上的莫大支持。

在此，衷心地祝愿每一位，生活如意，工作顺利。感谢你们。