

北京大学信息科学技术学院

# BlueBell

AQI MAP

樊乃嘉、林心宜、徐辰曦

2017-6-15

# 目录

<b>1. 背景意义</b>	<b>1</b>
1. 已有工作	1
2. 存在问题	1
3. 技术挑战	1
4. 工作内容	2
<b>2. 实验原理</b>	<b>2</b>
1. 开发环境	2
2. 系统框图	2
<b>3. 实验内容</b>	<b>4</b>
1. 技术方案	4
2. 实验步骤	4
<b>4. 实验结果</b>	<b>8</b>
1. 参数环境	8
2. 结果分析	8
<b>5. 工作总结</b>	<b>9</b>

# 1. 背景意义

## 1. 已有工作

提供天气预报，实时空气质量报告的 app 和网站很多，如中国天气网、墨迹天气、AQICN，PM2.5 监测网等等。其中，墨迹天气、中国天气网等网站会提供时景分析等图片社交功能。

## 2. 存在问题

1. 实时性较差：网站数据更新不及时，而天气、空气变幻莫测
2. 位置不精准：网站汇报的数据通常是一个地区的数据，然而我们关心的通常是自己身边环境，由于地区范围大，数据通常不精准。

## 3. 技术挑战

1. **样本偏斜**。由于需要获取图片及其对应 AQI，所以只有图片没有相应图片时间、地点、AQI 指数信息的数据不满足我们的要求，不同类别样本数量差异大。然而满足要求的图片往往是社交需要的图片，这些图片大多天气晴朗，质量优良，空气质量差的图片所占比例非常的少。这对训练分类器带来了很大的困难。
2. **图片质量差**。由于网站上图片出于社交需要，种类繁多，其中很大比例的图片与环境、场景无关（比如人物、动植物、艺术品等等），而即使是符合的图片，还面临图片后期附加效果的影响（比如滤镜、修图等等），导致样本的图片质量极差，难以使用。
3. **多因素混淆空气质量判断**。即使是符合上述所有条件的图片，也难以训练出优秀的分类器。因为通过图片判断 AQI 的任务本身难度高。图片有很多因素的影响，比如像素、拍摄水平、采光、拍摄时段、天气等等，这些都会影响一张图片的特征，从而混淆 AQI 的判断。又由于样本质量差、数量少，完全数据驱动，不增加人工特征处理是不可靠的。如何提取合适的图片特征用于 AQI 的判断是本任务的一大挑战。

## 4. 工作内容

1. 天空图片分析。上传图片，判断 AQI 等级；要求实时、位置精准到城市街道。
2. 历史分析结果展示。将 AQI 以地图形式展现；地图上会显示地图中心位置附近最近上传的时景、空气质量评级，以及历史信息的时间、地点信息；具体信息点击地图上的定制化大头针图标后，用气泡窗口呼出展示
3. 将功能包装成 iOS 系统的 App。

## 2. 实验原理

### 1. 开发环境

#### 1. 前端开发环境

Xcode 8.3.2 Objective-c

#### 2. 后端开发环境

Ubuntu 14.04.4 LTS

Python 2.7.6

Apache 2.4.7/(Ubuntu)

Django 1.11.0

Tensorflow 1.1.0

### 2. 系统框图

前端有两个页面，两个页面之间通过后台服务器的数据库完成通讯，没有直接的消息链接。表 1 是 MapView 成员之间的层次结构；最顶端的是页面 UI 和控制模块，控制模块承担页面和下层服务之间的信号传递；调用的下层服务主要有地理编码服务（GeoCodeSearch）、poi 搜索服务（poiSearch）、用户坐标服务（UserLocationinService）；控制模块向这些服务取得结果之后，再通过 Http 请求和后端服务器通信，请求相关点的具体信息。

图 1 是 UploadView 控制模块的流程图。注意，一开始页面载入时，我们默认系统时间和用户位置是将要上传图片的标签信息；一旦用户修改图片的地址

标签，我们需要重新将文字的地址信息转化为经纬度信息，这个转化需要反地理编码服务（ReverseGeoCodeSearch）来完成；同时我们选中图片后，需要向我们的后端服务器发送图片，进行分析，得到 AQI 等级；这两个服务是并行执行的。

图 2 是后端服务器的处理流程，是典型的后端分析服务器。

表1 MapView成员框架

Custom BMKMapView & MapViewController 定制化展示历史记录		
BMKGeoCodeSearch 得到中心点地址	AFNetworking(poi search) 对中心点附近搜索历史记录	BMKLocationService 维护用户坐标
BLS Cloud、Analyze Sever		

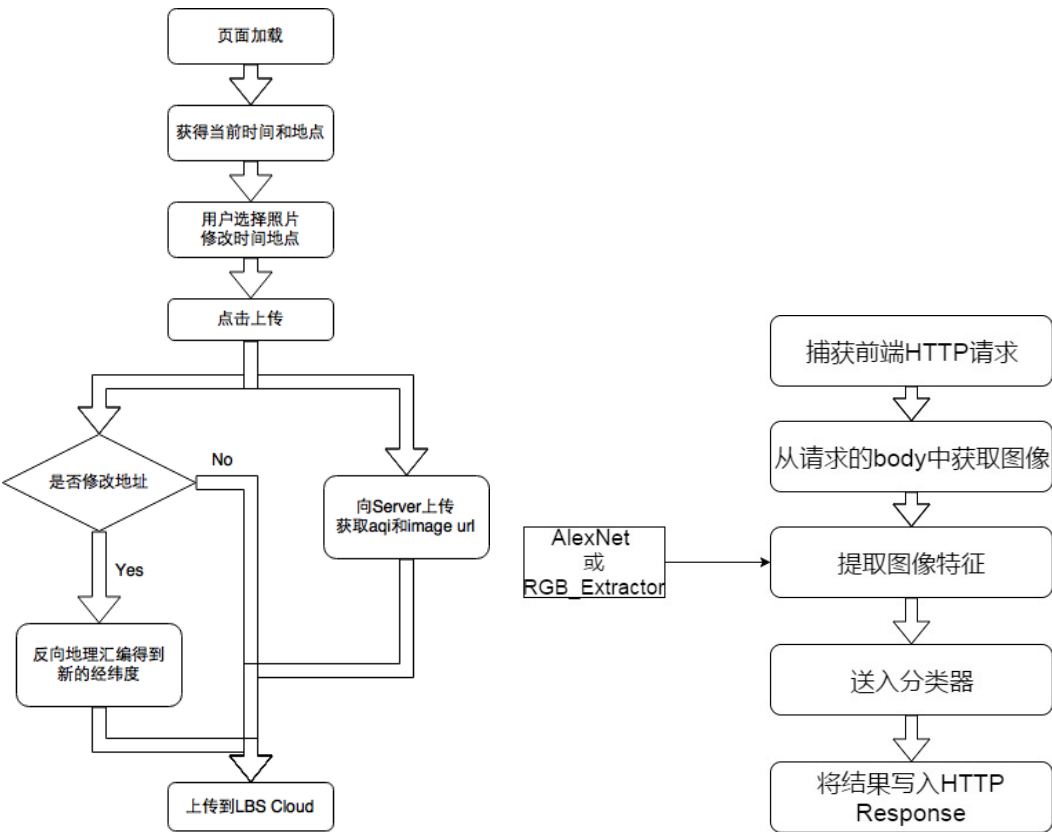


图 1 UploadView 的处理流程

图 2 后端处理流程

## 3. 实验内容

### 1. 技术方案

**样本数据获取。**通过两个数据来源分别获取图片数据和AQI数据，然后依据时间、坐标将其对应。再通过场景与物体识别获取图片标签，根据图片标签对图片进行筛选，过滤掉人物像、室内照等。

**空气质量指数预测。**我们先后尝试了两个方案。

- 方法一：一开始我们选择的方法是直接使用训练好的一个深度神经网络模型提取图片特征，再把这些特征作为输入  $x$ ，将相应的AQI作为输出  $y$ ，训练  $f: x \rightarrow y$  的一个非线性分类器。对于新的测试数据，用同样的网络进行特征提取，将提取到的特征  $x$  输入  $f$ ，得到分类结果  $y$ 。
- 方法二：考虑到已有的深度神经网络是基于目标检测模型进行训练的，而我们的空气质量检测算法不需要考虑图像中的物体种类。因而，我们改进了该模型的特征提取部分。我们肉眼判断天气状况往往基于天空的颜色，基于这种直观的感觉，这个方法将天空的颜色作为特征。

### 2. 实验步骤

**样本数据获取。**该部分代码在/src/alexnet/getdata.py 中实现。

1. 通过墨迹天气爬取时景图片；
2. 通过 face++场景与物体识别 api，获取图片标签。人工筛选一些标签，通过标签对照片进行筛选；
3. 爬取 PM2.5 历史数据网站的历史数据；
4. 根据图片附加信息中的时间、坐标，获取 PM2.5 数据库中对应的 AQI 指数；
5. 根据 AQI 所处区间，将 AQI 具体数值转化为 6 个评级。

**训练模型。**该部分代码在/src/alexnet 目录中实现。

1. 运行 feature\_extraction.py，遍历整个图片文件夹，将所有图片的 AlexNet 特征提取出来，写入 train\_data.pkl；
2. 运行 input.py，准备数据集并写入 input.pkl；
3. 运行 real\_train.py，设置要训练的参数，并开始训练。这里将数据集

随机分为训练集和交叉验证集（67%为训练集，33%为交叉验证集），用最小化交叉熵的方法训练全连接层，训练 1000 个 epoch，每个 epoch 中将数据分为大小为 128 的 batch 进行训练。训练完之后会把参数  $W$  和  $b$  写入 `weight_W` 和 `weight_b` 中；

4. 运行 `test.py` 测试模型准确率。遍历数据集中的每张图片，调用 `RGB_extract.py` 里的函数，提取特征  $x$ ，再  $Wx+b$  算出预测的结果。结果如下：

```
Epoch 3
Time: 0.206 seconds
Validation Loss = 1.22255464293
Validation Accuracy = 0.451990632877

Epoch 4
Time: 0.203 seconds
Validation Loss = 1.25018493995
Validation Accuracy = 0.466042154776

Epoch 5
Time: 0.214 seconds
Validation Loss = 1.24102621
Validation Accuracy = 0.454332552903

Epoch 6
Time: 0.199 seconds
Validation Loss = 1.25190612388
Validation Accuracy = 0.470725996642

Epoch 7
Time: 0.197 seconds
Validation Loss = 1.25480262066
Validation Accuracy = 0.447306790662

Epoch 8
Time: 0.214 seconds
Validation Loss = 1.28088870121
Validation Accuracy = 0.45199063176

Epoch 9
Time: 0.209 seconds
Validation Loss = 1.30476448659
Validation Accuracy = 0.437939111396

Epoch 10
Time: 0.211 seconds
Validation Loss = 1.33558909731
Validation Accuracy = 0.463700233634
```

图 3 AlexNet 特征分类器结果

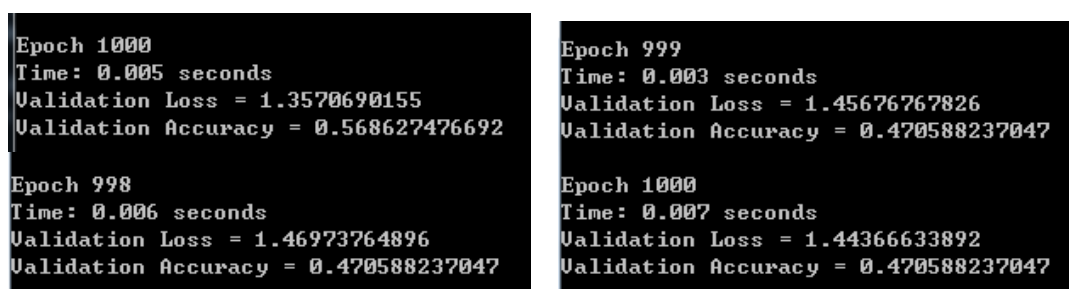
可以看到，在交叉验证集上的正确率大约为 46%，不过，这个正确率在随机的测试样本中可能效果更差，因为我们的训练数据中有一大半都是优和良，因而分类器倾向于将测试图片分类为优和良，从而最小化损失函数，而这个结果与直观感受很不一致。

上述方法的失败有两个原因：

- **第一是数据集的原因。**数据集不够随机，缺少空气质量较差的样本；样本图片质量差、干扰多。基于这个问题，我们参考了另一个小组获得的数据集。这个数据集来自同一社交账号，是基于同一设备、在同一地点、向同一方位每天拍摄的天空图。虽然只有一百多张图片，但是每张图片都很有代表性，且较为全面的覆盖了不同标签；
- **第二是我们提取的 AlexNet 特征不能够很好地表现空气质量。**结合生活经验我们知道，我们判断空气质量并不急于一张图片中拍到的天空图形如何（不同天际线可能会造成天空的图形差别很大），而更天空的颜色。新的思路是利用 RGB 特征作为抽取特征目标，将图片天空颜色的平均值用  $(R,G,B), 0 \leq R, G, B \leq 255$  这个三维向量表示。具体实现思路如下：

1. 首先，切割天空。我们统计了一下同一张图片中天空颜色的极差，发现这个极差通常不超过 70。我们假定一张图片顶部的像素必然是天空的像素，并计算一张图顶部像素的平均值，其中去掉和平均值差距大于 70 的像素点（因为这些像素点很有可能不属于天空）。
2. 接着，计算这些点的 RGB 特征。在这里我们简单地将这个用(R,G,B)表示的平均值作为这幅图像的特征。和之前 4096 维的特征相比，这个特征只有 3 维，看似表达的信息更少了，实际上是去除了大量不相干的信息。

新方法在/src/rgb 目录下训练测试，新的特征提取代码在 train.py 中实现。我们用新方法重新训练分类器，由于训练数据较少，这里训练了 1000 个 epoch。结果如下：



Epoch	Time (seconds)	Validation Loss	Validation Accuracy
1000	0.005	1.3570690155	0.568627476692
998	0.006	1.46973764896	0.470588237047
999	0.003	1.45676767826	0.470588237047
1000	0.007	1.44366633892	0.470588237047

图 4 RGB 特征分类器结果

可以看到，虽然正确率并不高，但是 loss 低了一些，说明即使判断错误，也离真实值不远；而且，这个数据集中各类数据都有，能够达到大于 20% 的正确率说明这个分类器学习到了一定的信息。在第四部分实验结果中将可以看到预测的情况，和直观感受还是很相近的。

**前端设计。**前端分为两个页面，涉及的都是基本的前端实现。下面分别从用况出发，提出一些实现的难点。具体工程为工作目录下 MapTest.xcworkspace

1. 展示页面用况概述
  - 展示地图中心坐标附近的历史数据点，用大头针标出，点击有图片、时间、地点、AQI 的展示；
  - 回到用户位置的定位按钮；
  - 地图上方动态展示当前展示地图中心坐标信息。
2. 展示页面核心难点
  - 定制化大头针的展示窗口：继承 BMKAnnotationView 实现自定义视图；
  - 地图中心位置信息展示：需要完成从经纬度到地名的查询，利用



BMKGeoCodeSearch 实现，注意要在控制类中添加对应代理；

3. 上传与分析页面用况概述：

- 用户选择拍摄一张图片或从媒体库中选择图片；
- 用户修改图片时间或地点（页面加载时预填入当前时间和地点）；
- 用户上传图片，得到 AQI 等级；
- 用户重新选择照片，开始新的上传与分析；

4. 上传与分析页面核心难点：

- 与服务器之间的多次异步通讯引发的线程同步问题：如果用户修改了图片的位置信息，我们需要向百度反向地理信息汇编请求该位置信息的地理坐标
- 因此上传过程为，首先并发地查询地理坐标（如果图片位置信息被用户改变了，即不是当前的用户坐标）、像服务器请求分析结果；再将所有结果拼接封装、上传到 LBS Cloud 上；**iOS 开发会自动地将 Http 请求做异步处理**，如果不做同步处理，可能我们还没取到结果，就将不完整的结果上传到云上了；
- 我们的解决方案是，为结果拼接封装开设新线程，并置两个结果返回的 flag；结果封装前检查 flag，若为假，则使将封装线程换下。

**后端服务器搭建。**由于我们把数据的主要信息记录在百度地图的 LBS Cloud 上，该后台存储，不能稳定的存储图片（容易出现图片转存失败），我们将数据信息中的图片存储在我们自己的服务器上。具体代码在/src/BlueBell 下实现，是个 django 项目。其中 manage.py 是 django 项目的管理文件，db.sqlite3 是后端数据库、bvlc-alexnet.npy weight 都是神经网络的参数。与我们的项目有紧密关系的代码在 AQI 目录下：

- views.py 实现对不同 url 请求的处理（关于 url 的定义在 bluebell 文件夹下的 url.py）；
- run\_prediction.py 是提取 Alexnet 特征的方法，返回 AQI 的预测结果；
- real\_prediction.py 是后来那个算 RGB 的方法，也是返回 AQI 的预测结果；
- predict.py 其实是提取 Alexnet 特征的；
- RGB\_extract.py 是提取 RGB 特征。

## 4. 实验结果

### 1. 参数环境

我们的训练样本图片来自墨迹天气<sup>1</sup>, 样本 AQI 标签来 PM2.5 科学实验专家小组数据库<sup>2</sup>, 样本过滤借口利用 face++ 的场景识别 API<sup>3</sup>。

我们采用的深度神经网络模型来自论文 Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks[C]// International Conference on Neural Information Processing Systems. Curran Associates Inc. 2012:1097-1105.

我们的前端利用了百度地图的 SDK, 以及 AFNetworking 第三方库。

最终产品后端搭载在 Apache 2.4.7/(Ubuntu) 上, 前端可在 iOS 9.2 以上版本正常运行。

### 2. 结果分析

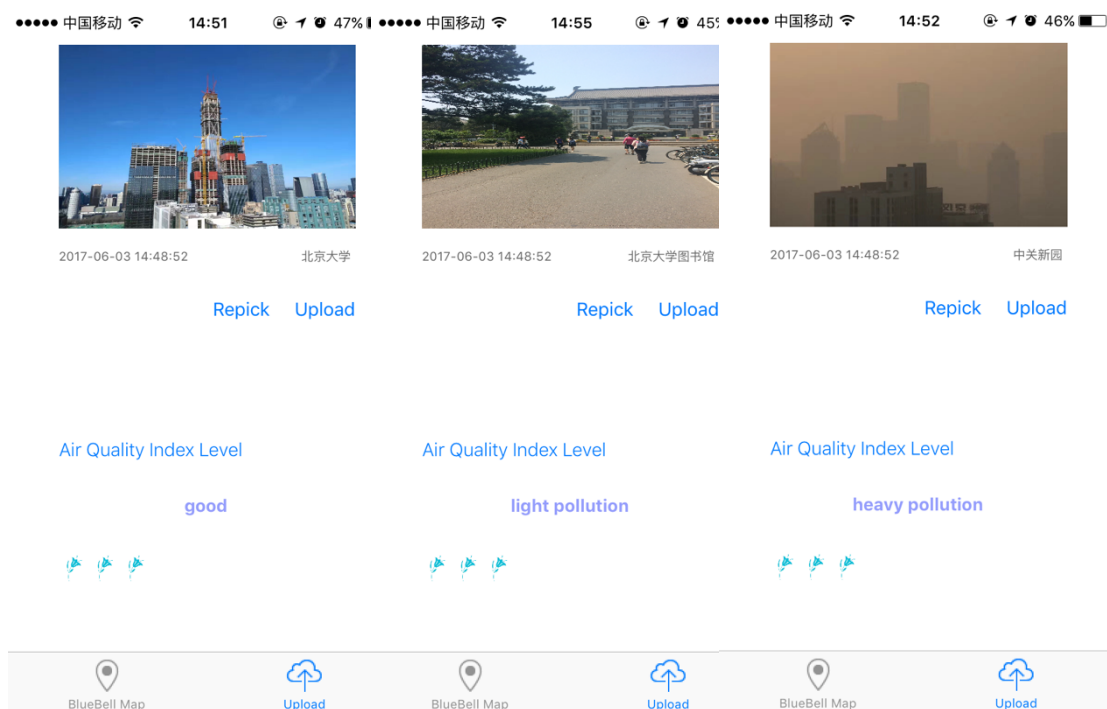


图 5 最终 AQI 分析结果

<sup>1</sup> <http://ugc.moji.com/index.php?picid=x&&cityid=1>

<sup>2</sup> <https://www.AQIstudy.cn/historydata/index.php>

<sup>3</sup> <https://console.faceplusplus.com.cn/documents/7776484>

**App demo:** <http://pan.baidu.com/s/1miBNAha> 密码:s3nb

**空气质量预测情况:** 参见图 5.

**其他实现方式:** 这一部分, 除了装这个 app, 也可以通过发送 http 请求得到结果, 具体的发送方式如下 (老师和助教可以试试), 见图 6.

**工具:** postman

**URL:** <http://162.105.175.115:8004/bluebell/comments/upload>

**方法:** POST

**参数:** addImage

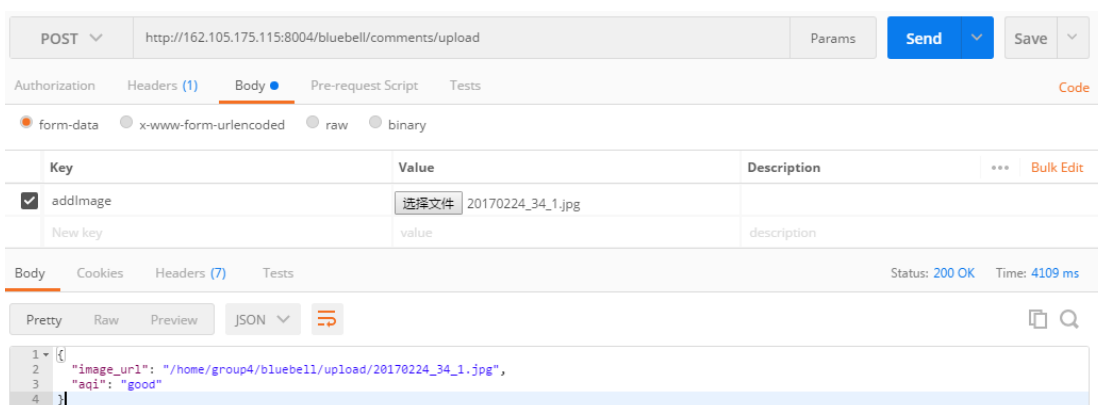


图 6 Http 请求 AQI 分析结果

## 5. 工作总结

通过这次 project 的锻炼, 我们实际应用了平时实验室学习中接触到的神经网络知识, 并作出了自己的优化尝试; 我们学习了软件工程开发的前后端技术, 应用了网络课上的相关知识, 实现了前后端之间的通信。在这次实践中, 我们除了完成老师规定的基于图片分析 AQI 的任务外, 还超额完成了可视化展示以及软件开发的工作, 不断收获挑战自我的乐趣。

关于未来工作。首先我们的图片分析还不够准确, 可以进一步优化特征提取, 和深化网络层次结构; 同时, 优化样本也是重要的一个方向。其次, 我们的 App 还很粗糙, 可以进一步支持更细节的查询; 也可以增加用户系统, 是用户上传的图片和贡献的 AQI 数据, 能够拥有自己的版权。