**机器学习纳米学位**

**猫狗大战——项目报告**

2018年9月19日

## I. 问题的定义

### 项目概述

本项目在于让计算机具备识别图片中物体（例如是猫还是狗）的能力，问题涉及到计算机视觉这个领域。而近年来深度学习的研究进展表明，基于CNN（卷积神经网络）构建的深度学习模型，对于图像识别分类，具有很高的准确率，例如在imagenet分类识别中获得很好成绩的Xception、VGG16、ResNet50等模型。

因此，本项目将构建一个CNN深度学习模型，用kaggle猫狗项目的训练数据（已经标记好猫/狗标签的训练图片），对该模型进行训练；然后让该模型对kaggle猫狗项目的测试数据（未做标记的测试图片）做出最终判断是猫还是狗；最后对判断结果进行打分，进而说明模型的性能。

### 问题陈述

本项目的问题在于如何构建一个模型，用kaggle带标签的数据进行训练，然后对无标签的数据进行二分类预测（区分猫还是狗），这是一个监督学习的过程。我们可以对任意一张图片，预测该图为某个分类的概率，然后根据概率判定该图是否为某个分类。所以这个过程是可量化，可测量，可重复的。

整个问题的处理过程是，先用imagenet的预训练模型，对kaggle的训练、测试数据进行预测，导出特征向量；然后再构建自己的相对简单的MLP模型，以特征向量为输入，进行训练，得出二分类概率；最后用自己的MLP模型，对kaggle测试数据进行预测，看结果是否符合标准，并上传至kaggle进行打分。

### 评价指标

Logistic回归损失函数，即对数损失函数，度量了真实条件概率分布与假定条件概率分布之间的差异，是常用的评价方式之一，被广泛应用于分类问题。而本项目涉及的是二分类问题，因此可以使用LogLoss作为评价标准。

评估标准采用kaggle官方指定损失函数：

参数说明

* n 为测试数据集的图片数量
* 为一张图片预测为狗的概率
* 为类别标签，1为狗，0为猫
* 为自然对数

最终函数值越小，结果越好，代表模型性能越好。

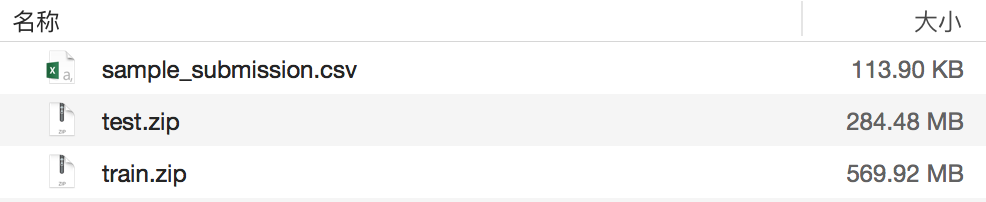
项目要求是最低要达到kaggle Public Leaderboard 前10%，即排在第 1314/10 = 131 名选手之前，其得分为0.06127，即本项目最终得分要小于该数值。

## II. 分析

### 数据的探索

#### 获取数据

从kaggle下载猫狗项目数据集，如果是在页面手动点击下载按钮，会得到一个all.zip文件，解压后会有sample\_submission.csv、test.zip、train.zip共3个文件；如果是使用kaggle API命令（kaggle competitions download -c dogs-vs-cats-redux-kernels-edition）下载，会得到test.zip、train.zip共2个文件，须单独再下载sample\_submission.csv，最终数据的目录如下：



解压train.zip，里面包含25000张训练图片（猫狗各有12500张，文件名以cat或dog为前缀）；解压test.zip，里面包含12500张测试图片（文件名为数字，无法区分猫狗）。

由于keras的api是通过不同目录来区分不同分类的，所以我们需要将train目录下的图片按cat或dog前缀划分到2个目录，test目录下的图片再全部划分到一个test子目录中，最终目录如下：

|  |
| --- |
| --train  --cat  --cat.0.jpg  --cat.1.jpg  ……  --cat.12499.jpg  --dog  --dog.0.jpg  --dog.1.jpg  ……  --dog.12499.jpg  --test  --test  --1.jpg  --2.jpg  ……  --12500.jpg |

随机展示几张图片，如下：

可以发现图片尺寸不一。

#### 检测异常数据

本项目使用keras预训练模型ResNet50[1]（imagenet权重），对训练数据集的图片进行预测（会给出图片分别属于不同种类的概率），筛选出top100概率都不是狗/猫种类的图片，然后人工判断是否合理，最后选择其中的真正异常图片进行剔除。步骤如下：

1. 先对部分图片进行检测，刚开始使用top5，发现结果里面正常图片占比较大，则不断调整top的种类数量，当为100时，发现异常图片的漏测率、误测率都较低，就选了这个参数，

然后对所有训练图片进行检测：

对[./data/train/dog]目录下的图片进行预测，筛选出top100概率都不是狗的图片，打印其文件名称。

对[./data/train/cat]目录下的图片进行预测，筛选出top100概率都不是猫的图片，打印其文件名称。

1. 对这些图片进行人工判断，保留误判的图片，例如：

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| dog.1625.jpg | cat.11879.jpg | cat.3637.jpg | cat.7009.jpg |

1. 筛选出真正异常图片，移动到invalid\_train相应子目录下，随机展示几张异常图片（包括图片本身就不是猫狗，或者是图画，或者是背景太复杂的情况）：

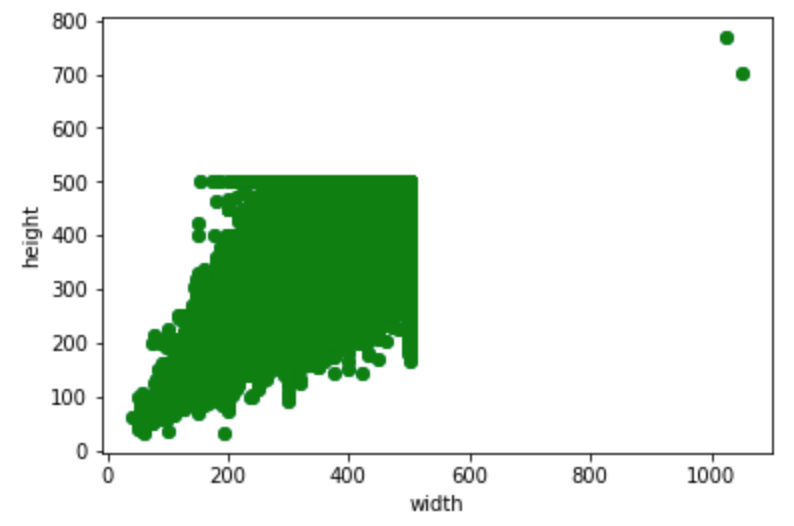
|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| dog.2614.jpg | dog.8736.jpg | dog.9517.jpg | dog.10161.jpg |
|  |  |  |  |
|  |  |  |  |
| cat.2457.jpg | cat.2939.jpg | cat.7377.jpg | cat.10712.jpg |

### 探索性可视化

1. 首先展示各分类的异常图片所占比例。

|  |  |
| --- | --- |
|  |  |
| 上图为狗的异常图片数量、正常图片数量分布比例（总共有12500张图片,其中9张异常图片） | 上图为猫的异常图片数量、正常图片数量分布比例（总共有12500张图片，其中38张异常图片） |

1. 展示正常训练图片的尺寸（宽高）分布散点图



由此可再次证明数据集的图片尺寸不一，但大致分布在一个范围内。

### 算法和技术

1. 算法

本项目实质上是处理图像分类问题，而常见的分类算法有决策树、SVM（支持向量机）、贝叶斯、KNN（K最近邻）等等，然而这些算法在以imagenet为代表的数据集上的表现并不好。

直到近年来伴随着计算机硬件的发展，深度学习神经网络开始取得显著的成绩，尤其是CNN（卷积神经网络），更是成为了图像识别中代表性的深度学习算法。CNN的局部感知野、权重共享特性，使其具有权重数量较少、更能识别位移、缩放及其它扭曲不变形图形、自动特征提取等优势。

因此，面对猫狗图像识别分类问题，本项目将采用CNN算法来搭建模型，解决问题。

典型的CNN包含以下层：

**输入层**

输入层代表整个网络的输入。在处理图像时，输入层代表的就是图片解析而成的由像素值表示的多维矩阵，如果是灰度图片，矩阵维度为1，如果RGB图片，矩阵维度为3。

**卷积层（Convolutional layer）**

卷积层主要进行卷积运算处理。对于输入数据，卷积运算以一定间隔（参数stride）滑动滤波器的窗口，将各个位置上的滤波器元素和窗口内的输入元素相乘再求和，将结果保存到对应位置[2]。扫描完整张图片后，就会得到一个特征图，因此，过滤器越多，得到的特征图就越多，从输入获得的信息也越多。

**ReLU（Rectified Linear Unit）函数**

在CNN网络中，引入非线性函数ReLU作为激活函数，使得深度神经网络的输出不再是输入的线性组合，而是可以逼近复杂函数。ReLU函数执行的操作是对输入应用max(0, input)操作，对于所有小于0的输入，输出为0。

**池化层（Pooling layer）**

池化，是将目标区域的多个元素，集约成一个元素，缩小二维空间的运算，所以也叫降采样。池化方式包括最大池化（从目标区域取出最大值）、平均池化（计算目标区域的平均值）等方式。

**Dropout layer**

Dropout层，在学习过程中，每一次传递信号，会随机删除神经元，被删除的神经元不进行此次信号的传递。以此抑制过拟合。

**全连接层（Dense layer）**

在整个网络的最后，往往会有一两个全连接层，将输入的每个神经元都与输出的每个神经元两两相连，输出一个n维结果（n分类结果）。

1. 技术

本项目使用Keras（高层神经网络API）作为开发工具，以Tensorflow为后端实现。

Keras是由Python编写的，以Tensorflow、Theano以及CNTK为后端实现的一套高层API，具有用户友好、模块化、易扩展、基于Python编写（学习调试方便）等优点。

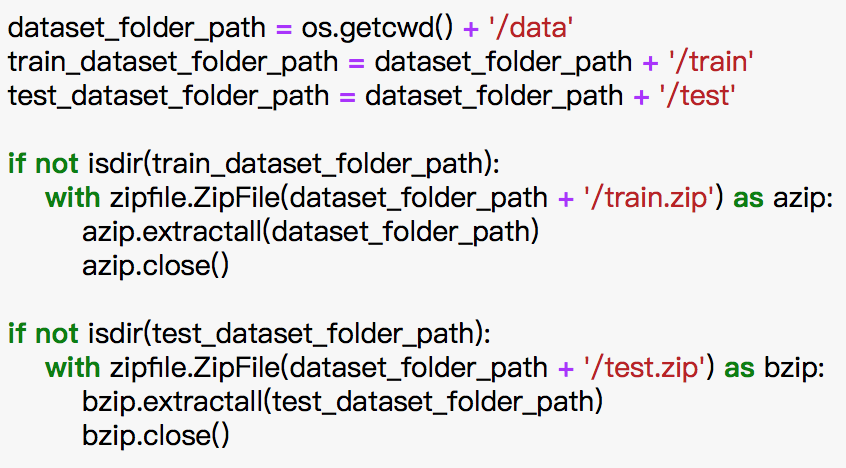
### 基准模型

项目要求是最低要达到kaggle Public Leaderboard 前10%，即排在第 1314/10 = 131 名选手之前，其得分为0.06127，即本项目最终得分要小于该数值。

## III. 方法

### 数据预处理

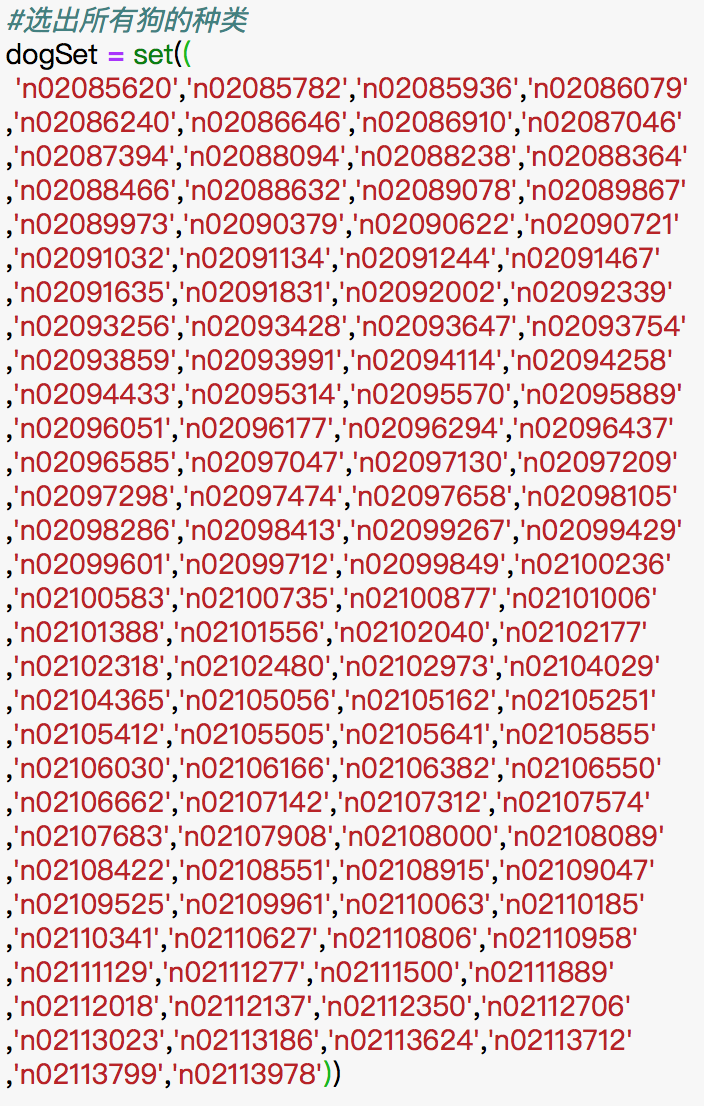
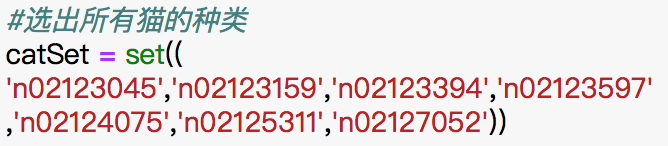
1. 调整文件目录
   1. 检查是否存在目录[./data/train]和[.data/test]，不存在的话创建目录，将训练数据集压缩包train.zip、测试数据集压缩包test.zip分别解压到对应目录



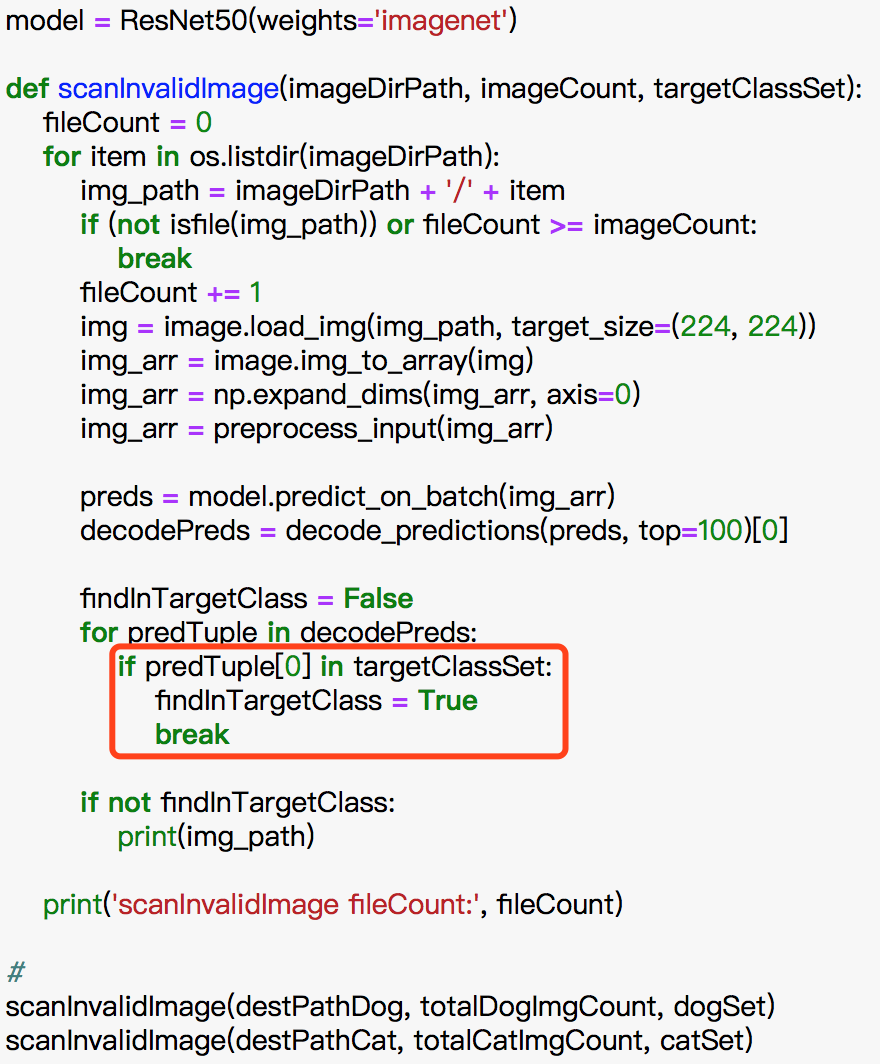
* 1. 将训练数据集解压后目录[./data/train]下的图片按照名称前缀cat或dog分别复制到[./data/train/cat]和[./data/train/dog]目录下



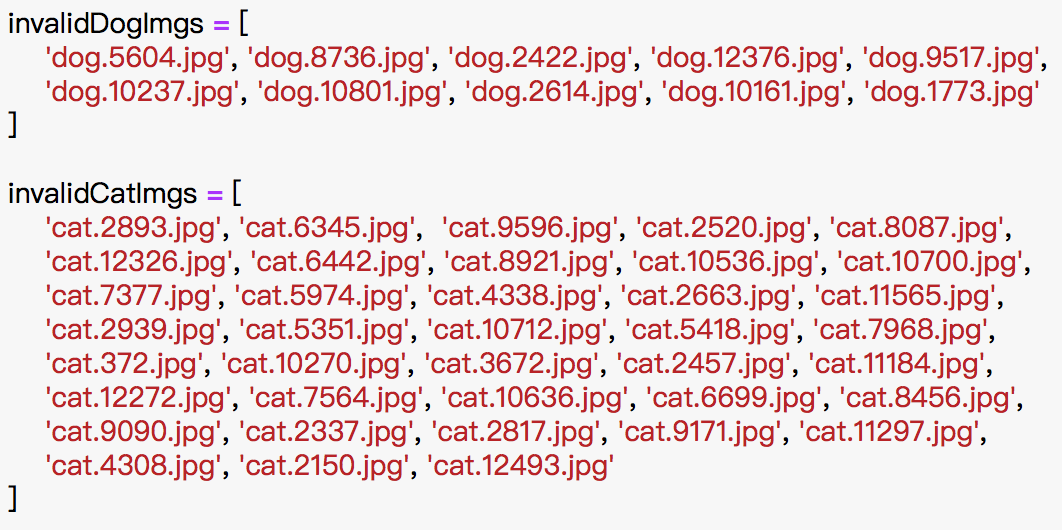
1. 剔除异常数据
   1. 取imagenet所有的种类编码及对应的种类名称，保存到文件[./imageNetClasses.txt]。从中分别选出狗、猫的所有种类，保存下来

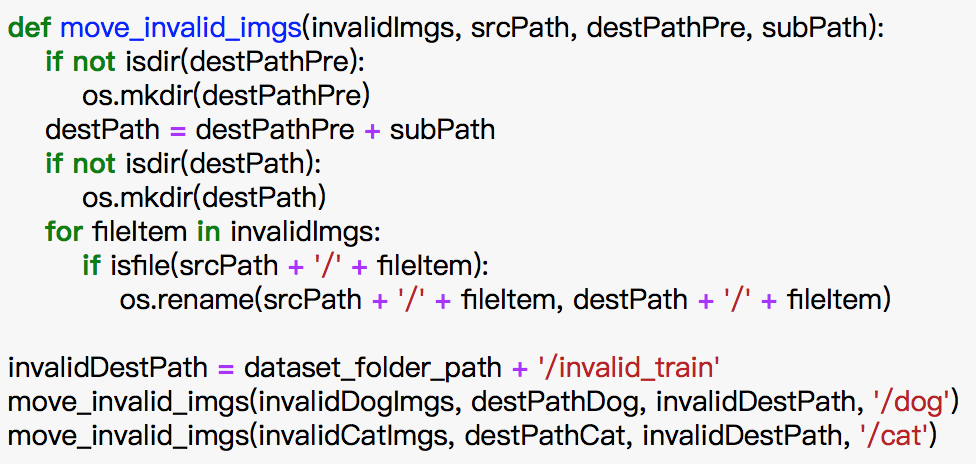
* 1. 分别从dog目录、cat目录扫描出种类不属于狗、猫的图片（红框处为判断预测的种类是否为猫/狗的种类），打印出图片名称



1. 对异常图片进行人工再次筛选，并剔除
   1. 对上面top100都不是猫狗的图片进行最终人工筛选，挑选出不是猫狗，或者有猫狗但是背景太复杂的图片，保存到临时列表中

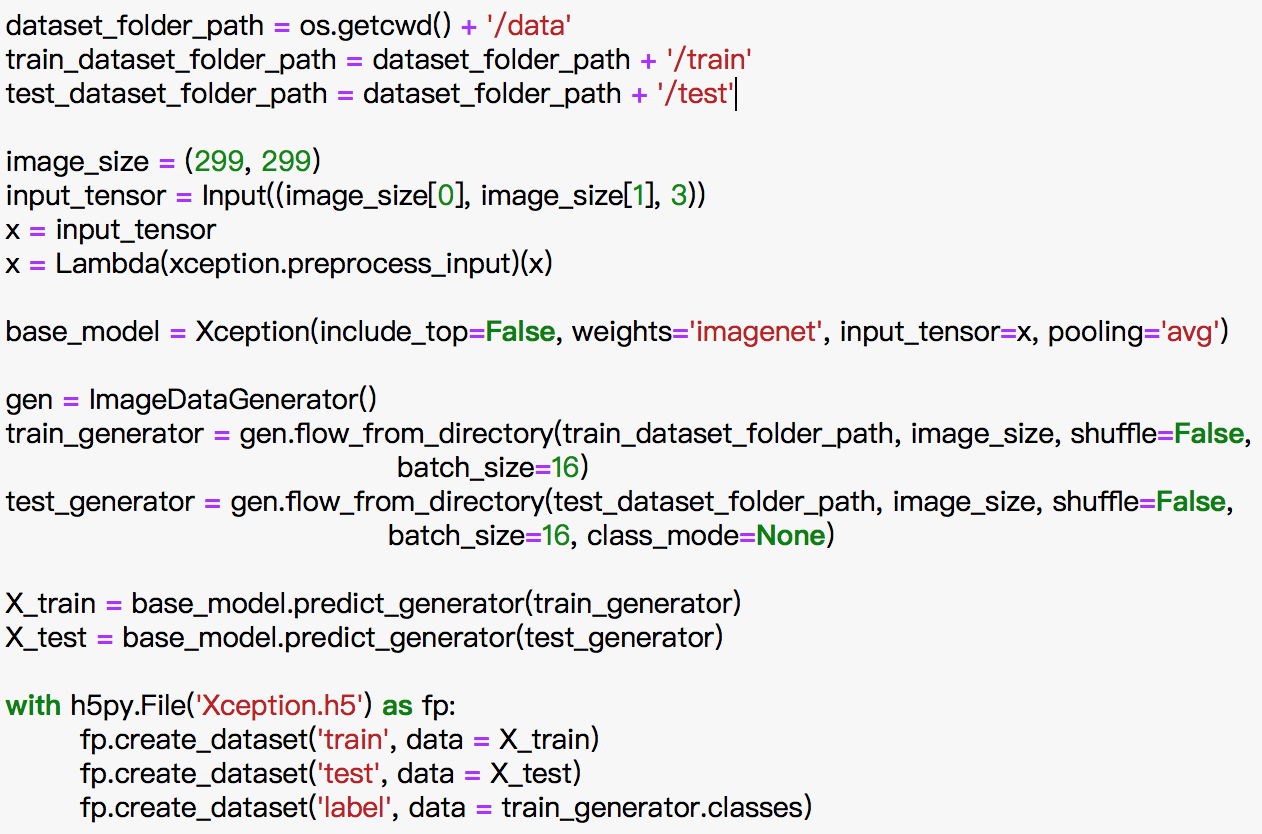


* 1. 分别移动到[./data/invalid\_train/dog]和[./data/invalid\_train/cat]，不参与接下来的训练



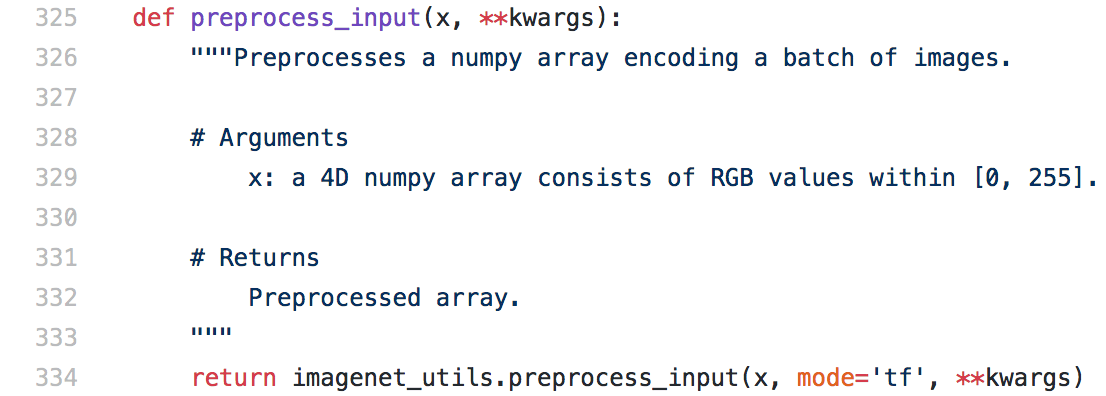
### 执行过程

1. 基于迁移学习的理念，从keras的预训练模型Xception[3]（使用预训练的imagenet权重，使用平均池化，不包含顶层的全连接网络）中，导出特征向量
   1. 代码如下：

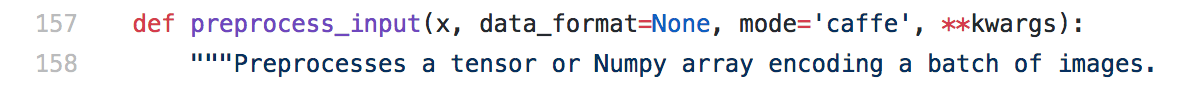


* 1. 在上图中，本项目在构建Xception模型的入参input\_tensor时，传递了预处理函数xception.preprocess\_input()，会对图片数据进行归一化（将范围为[0, 255]的像素值，以127.5为中心，映射到[-1,1]的范围）。以下为其源代码实现。

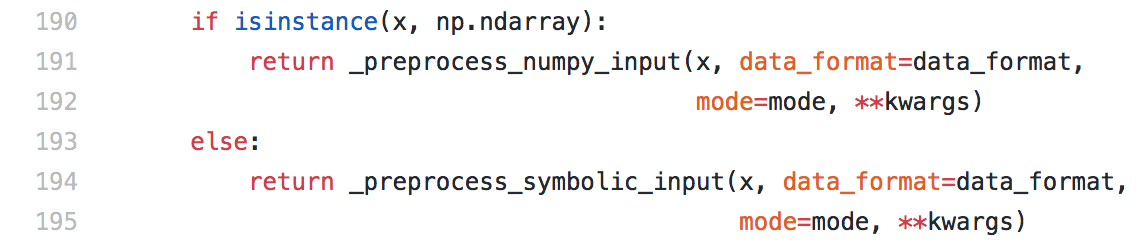
通过查看karas预训练模型Xception的源代码[4]，发现其预处理函数调用了imagenet\_utils.preprocess\_input()，参数mode=’tf’，如下图：



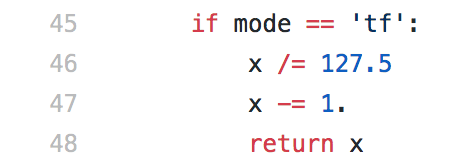
* 1. 查看imagenet\_utils.preprocess\_input()的源代码[5]



……

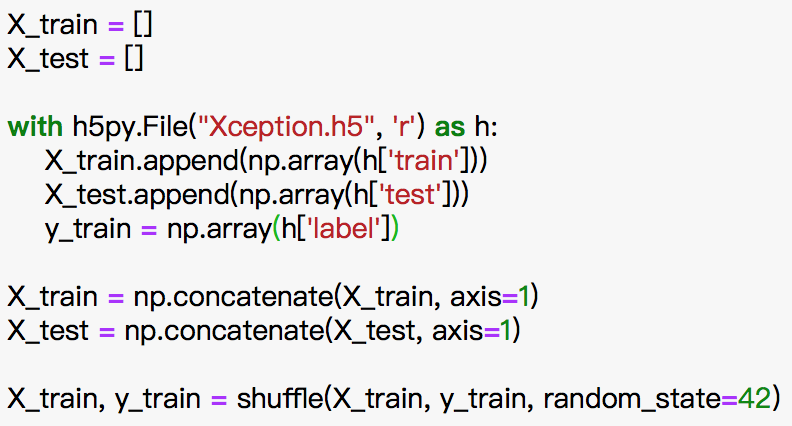


在\_preprocess\_numpy\_input()和\_preprocess\_symbolic\_input()方法中都有以下逻辑：

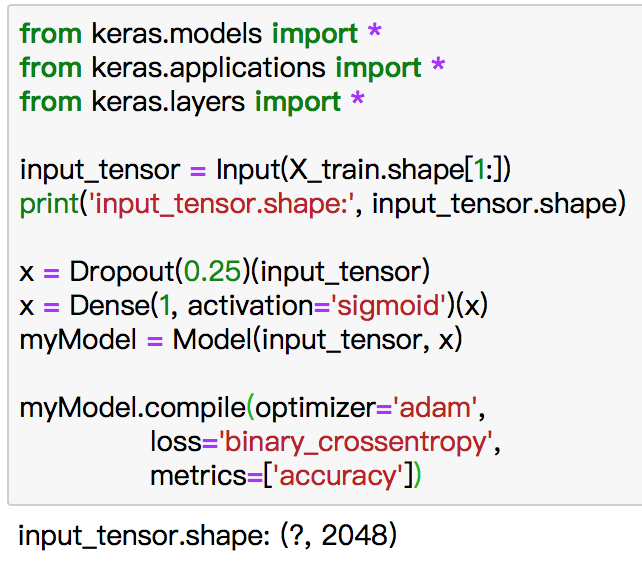


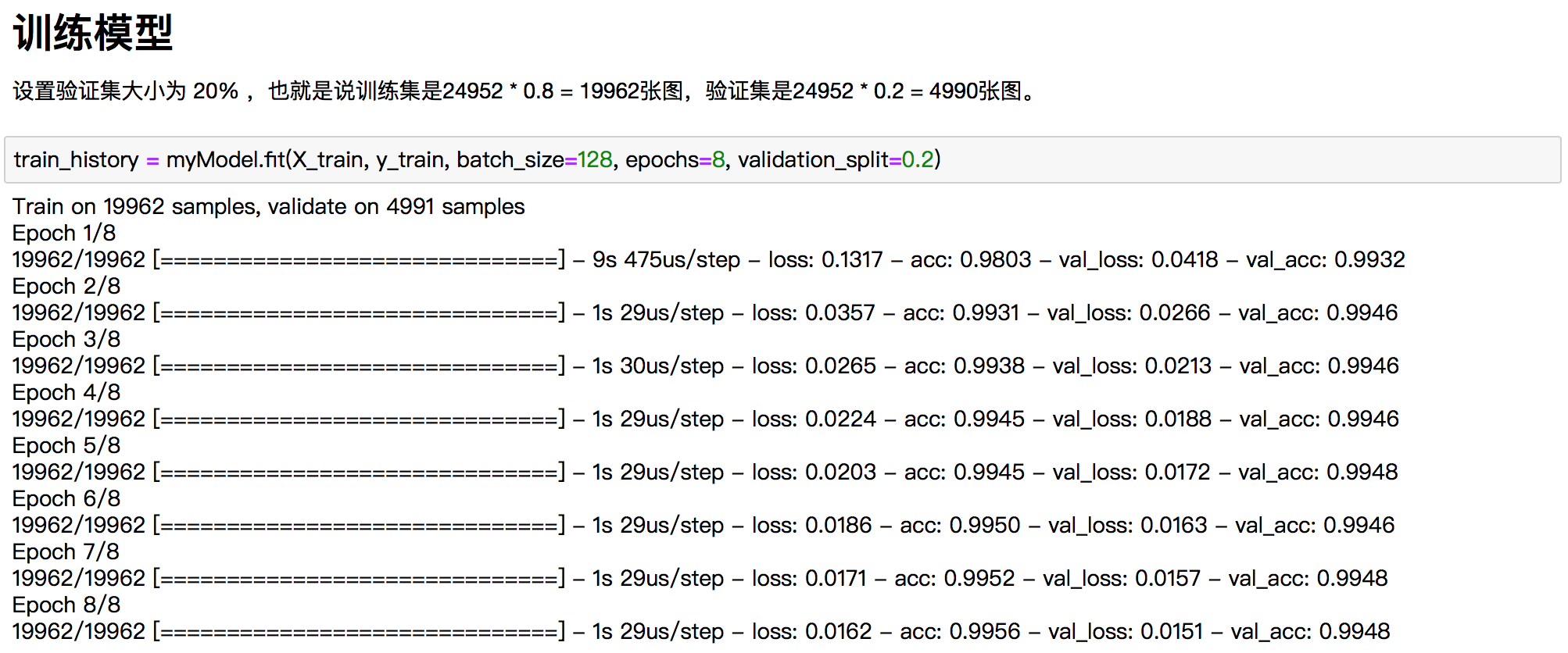
由此证明了上述归一化逻辑。

1. 导入特征向量，然后对 X 和 y 重新随机排序，以备使用



1. 以步骤二的特征向量维度（即X\_train.shape[1:]）为输入，构建自己的模型
   1. 先构建Dropout层，在学习过程中，每一次传递信号，会随机删除神经元，被删除的神经元不进行此次信号的传递。以此抑制过拟合。[6]
   2. 再构建Dense层，输出维度为1，激活函数设为sigmoid。



1. 训练模型
2. 对测试数据集进行预测



### 完善

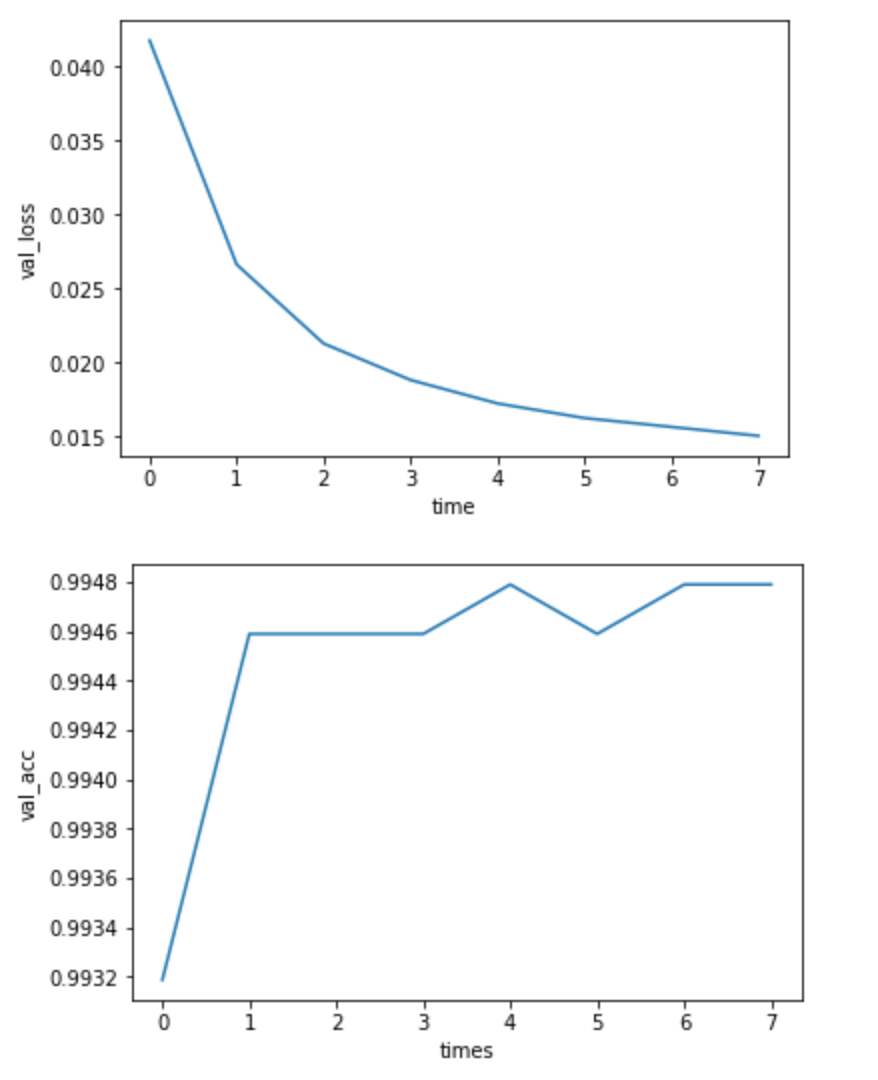
1．对于异常数据的检测，本项目使用了ResNet预训练模型来检测；可以考虑使用多个预训练模型进行检测，然后综合考虑各个模型的预测结果，更能提高对异常数据的覆盖率。

2．在导出特征向量的环节，本项目使用了Xception预训练模型，可以考虑结合多个预训练模型，来提高准确率，降低Logloss。

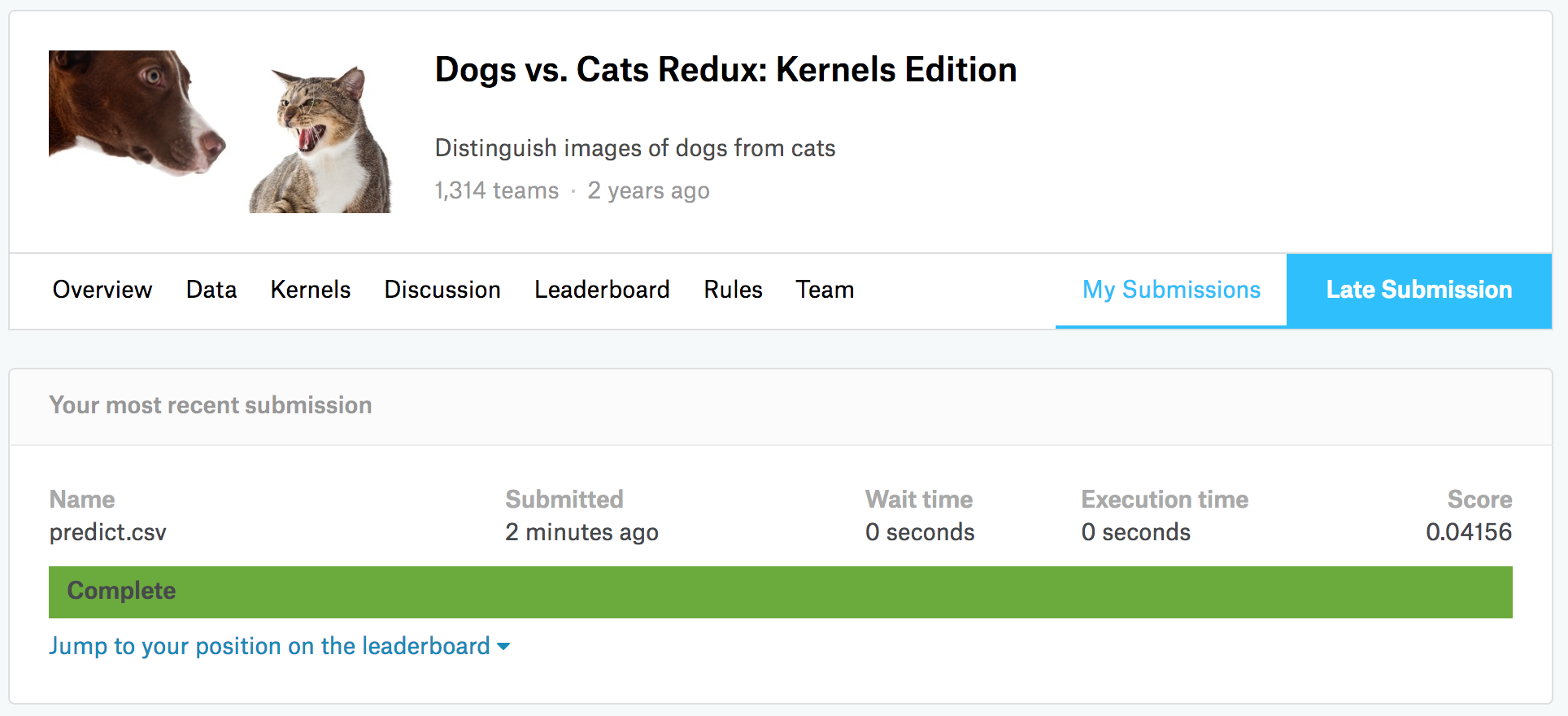
## IV. 结果

### 模型的评价与验证

1、下图为模型的学习曲线及准确率。



2、最终kaggle得分0.04156，小于0.06127，达到了项目要求。如下图：



### 合理性分析

模型最终收敛，得分比基准的效果好，达到了较高准确率，因此模型是有效的，也解决了项目中设定的问题。

## V. 项目结论

### 结果可视化

最终模型的预测结果如下，包含了图片名称，预测为某个种类及其概率：





可以看到模型的预测准确率高，即使背景复杂，也能识别出来。

### 对项目的思考

1．通过项目，对迁移学习有了一定了解，但是后续还需要增加实践，加深理解。因为迁移学习涉及到多个场景，本项目仅仅是其中的场景一：数据集小（训练集只有25000），数据相似度高(与pre-trained model的训练数据相比而言)。除此之外，还有3个场景需要去尝试处理。[7]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 场景一 | 场景二 | 场景三 | 场景四 |
| 数据集规模 | 小 | 小 | 大 | 大 |
| 数据集相似度 | 高 | 低 | 高 | 低 |

2. 本项目仅适用了Xception一个预训练模型进行特征向量输出，可考虑使用多个模型结合，提高性能。

### 需要作出的改进

1. 使用多个预训练模型进行异常数据检测，提高异常数据的检测覆盖率。
2. 结合多个预训练模型，导出特征向量，达到更好的效果。
3. 实现模型应用这一部分，将模型部署到web或ios设备，直接进行图像分类识别。

### 参考文献

1. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun.Deep Residual Learning for Image Recognition？[OL]. https://arxiv.org/abs/1512.03385.20151210
2. 斋藤康毅.深度学习入门[M].北京:人民邮电出版社,2018:203-204.
3. François Chollet. Xception: Deep Learning with Depthwise Separable Convolutions？[OL]. <https://arxiv.org/abs/1610.02357.20161007>
4. François Chollet.Keras: Deep Learning for humans? [OL].<https://github.com/keras-team/keras-applications/blob/master/keras_applications/xception.py>
5. François Chollet.Keras: Deep Learning for humans? [OL].<https://github.com/keras-team/keras-applications/blob/master/keras_applications/imagenet_utils.py>
6. 斋藤康毅.深度学习入门[M].北京:人民邮电出版社,2018:192-193.
7. 量子位.一文看懂迁移学习：怎样用预训练模型搞定深度学习？[OL].<https://mp.weixin.qq.com/s/lLoNvo2IE_Zhya8Vvqk6_w>.20170702