

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：

$$\text{LogLoss} = -\frac{1}{n} \sum_{i=1}^n [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

参数说明

- n 为测试数据集的图片数量
- \hat{y}_i 为一张图片预测为狗的概率
- y_i 为类别标签，1 为狗，0 为猫
- log() 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：

名称	大小
 sample_submission.csv	113.90 KB
 test.zip	284.48 MB
 train.zip	569.92 MB

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。

2. 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

3. 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)：



Adopted

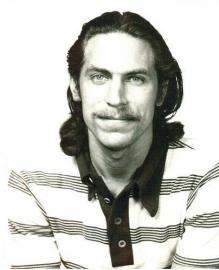


dog.2614.jpg

dog.8736.jpg

dog.9517.jpg

dog.10161.jpg



cat.2457.jpg

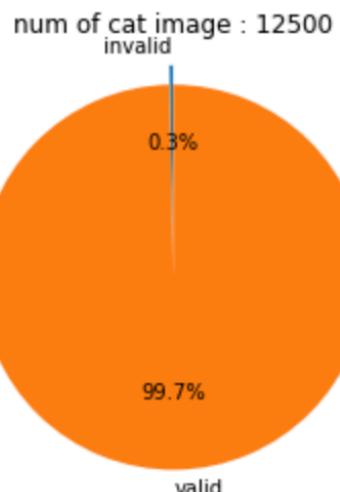
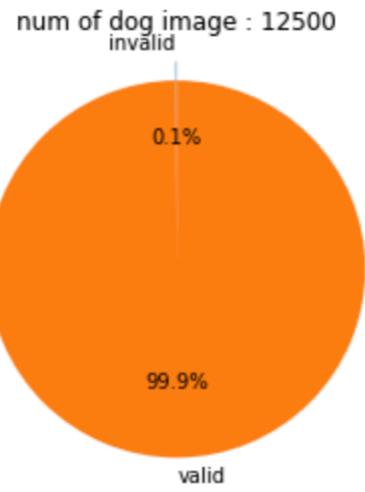
cat.2939.jpg

cat.7377.jpg

cat.10712.jpg

探索性可视化

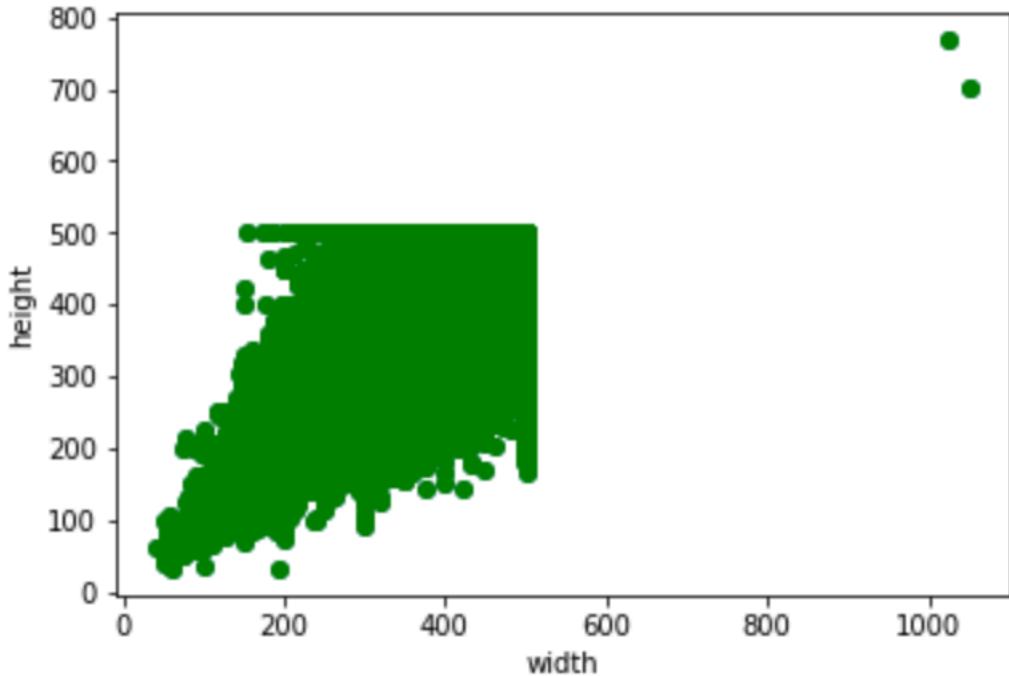
1. 首先展示各分类的异常图片所占比例。



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）

2. 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 $\max(0, \text{input})$ 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1两个全连接层，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）。

2. 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。将使用 Keras 预训练模型 Xception 导出特征向量。

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。

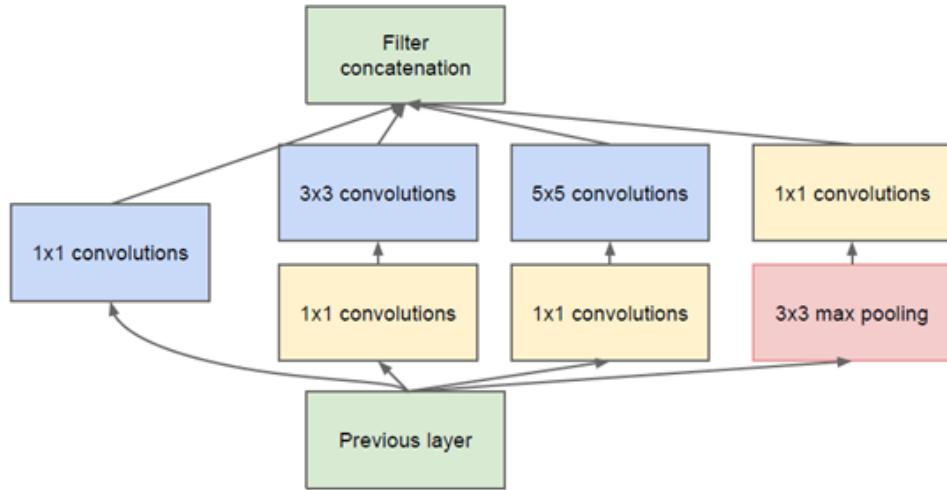
Xception 模型简介：

在 2014 年的 ImageNet 挑战赛中，GoogLeNet 和 VGG 两个模型大放异彩。VGG 更多地继承了以前的 LeNet 以及 AlexNet 模型的结构，而 GoogLeNet 则大胆引入了 Inception 模块结构，达到了减少参数，提升性能的目的。

在 GoogLeNet 之前的深度学习网络，基本都是通过直接加深、加宽网络实现，但带来了参数太多，容易过拟合、计算量大、容易梯度消失等问题。而 Google

提出的 Inception V1 结构，打破原有模型的串联模式，采用将 1×1 , 3×3 , 5×5 的卷积层和 3×3 的 pooling 池化层并联组合后 concatenate 组装在一起的设计思路，在增加网络宽度的同时，也增加了对尺度的适应性。错误！未找到引用源。

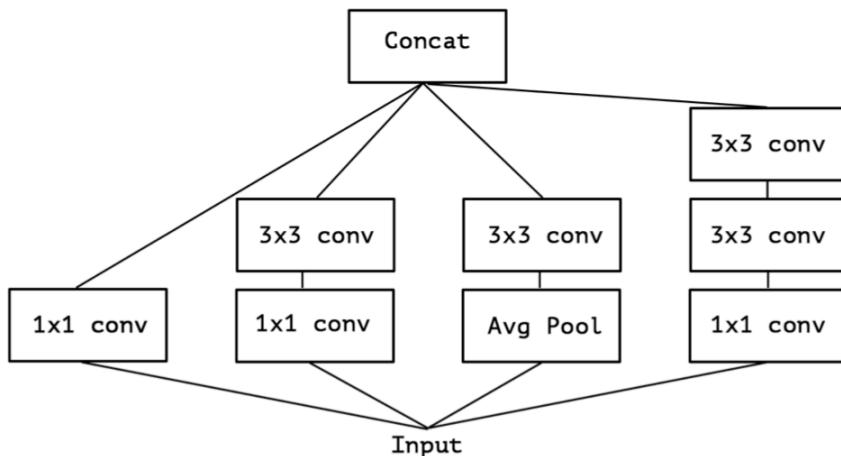
其结构如下图：



2015 年，Google 继续提出了 Inception V3 结构，对比 Inception V1 主要是将 5×5 卷积换成两个 3×3 卷积层的叠加。[3]

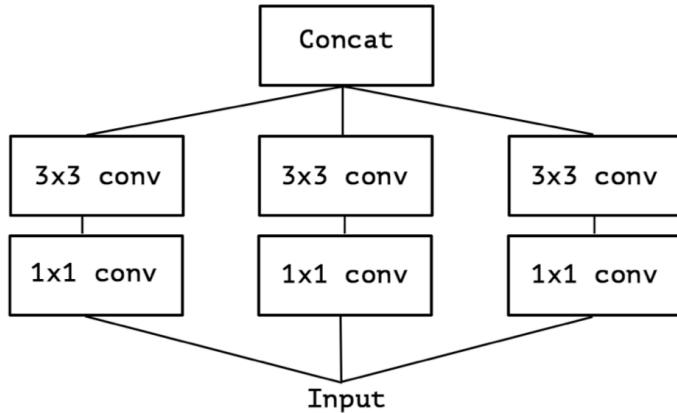
其结构如下： [4]

Figure 1. A canonical Inception module (Inception V3).



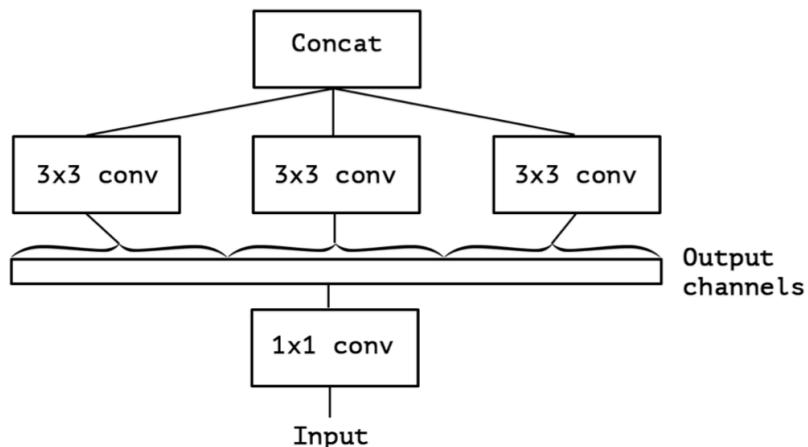
对 Figure 1 的结构，改为仅使用 3×3 卷积，去除平均池化部分，就得到下面的 Figure 2 结构：

Figure 2. A simplified Inception module.



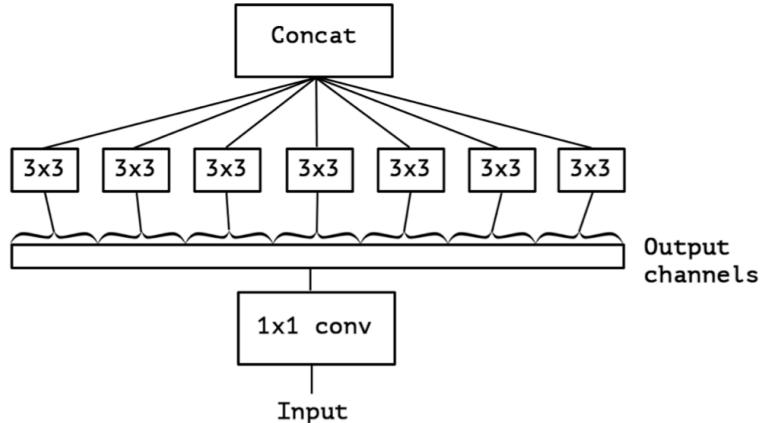
将 Figure 2 中 1×1 卷积提取为公共部分，得到下面的 Figure 3 结构：

Figure 3. A strictly equivalent reformulation of the simplified Inception module.



对 Figure 3 的结构进行极限化推演，先进行 1×1 卷积，然后对卷积后的每个 channel 进行 3×3 卷积，最后将结果 Concat，得到 Figure 4 结构：

Figure 4. An “extreme” version of our Inception module, with one spatial convolution per output channel of the 1x1 convolution.

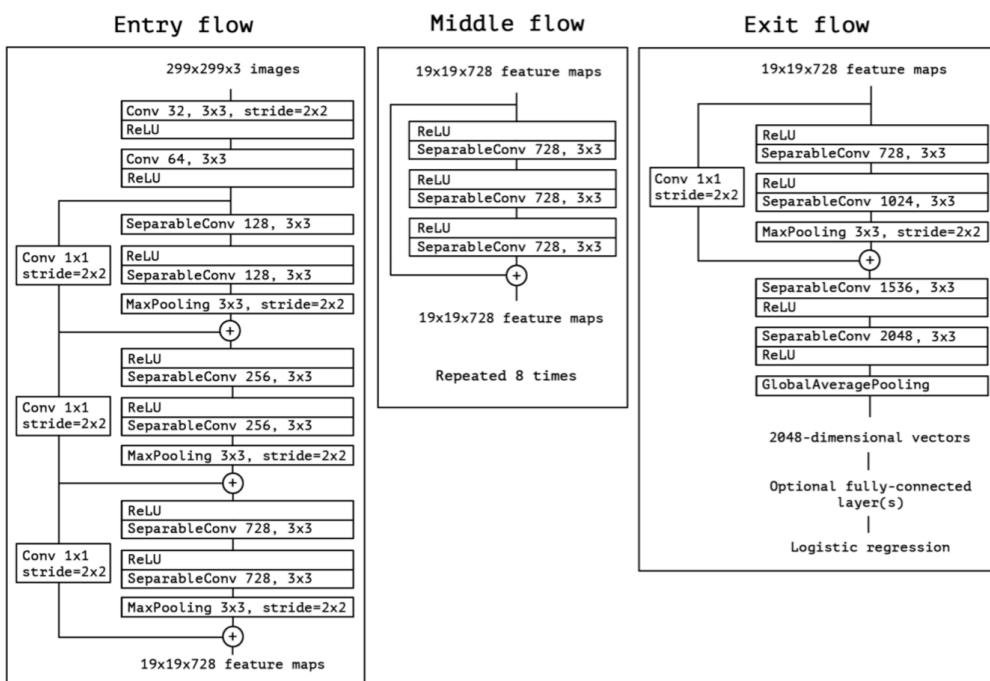


Google 在 2016 年提出的 Xception 网络模型，便是将 Inception 模块结构，换成了与 Figure 4 类似但又有所区别的 Depthwise Separable Convolution。

Depthwise Separable Convolution 主要包含两个步骤：（1）depthwise 卷积——对输入的每个 channel，分别进行 3×3 卷积操作，并将结果 Concat；（2）pointwise 卷积——第 1 步的结果，进行 1×1 卷积操作。

可以看到 Depthwise Separable Convolution 与 "extreme" version of Inception Module 的操作顺序相反，除此之外还有个区别就是，它在 depthwise 卷积之后一般不添加 ReLU，而 Inception 在两次卷积后都会使用 ReLU。

Xception 模型结构如下：[5]



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

- a) 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

```
dataset_folder_path = os.getcwd() + '/data'
train_dataset_folder_path = dataset_folder_path + '/train'
test_dataset_folder_path = dataset_folder_path + '/test'

if not isdir(train_dataset_folder_path):
    with zipfile.ZipFile(dataset_folder_path + '/train.zip') as azip:
        azip.extractall(dataset_folder_path)
        azip.close()

if not isdir(test_dataset_folder_path):
    with zipfile.ZipFile(dataset_folder_path + '/test.zip') as bzip:
        bzip.extractall(test_dataset_folder_path)
        bzip.close()
```

- b) 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

```

destPathDog = train_dataset_folder_path + '/dog'
if not isdir(destPathDog):
    os.mkdir(destPathDog)

destPathCat = train_dataset_folder_path + '/cat'
if not isdir(destPathCat):
    os.mkdir(destPathCat)

for root, dirs, files in os.walk(train_dataset_folder_path):
    for fileName in files:
        filePathName = train_dataset_folder_path + '/' + fileName
        if not isfile(filePathName):
            continue
        if ('dog' == fileName.split('.')[0]):
            os.rename(filePathName, destPathDog + '/' + fileName)
        elif('cat' == fileName.split('.')[0]):
            os.rename(filePathName, destPathCat + '/' + fileName)

```

2. 剔除异常数据

- a) 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

```

#选出所有狗的种类
dogSet = set([
    'n02085620','n02085782','n02085936','n02086079',
    'n02086240','n02086646','n02086910','n02087046',
    'n02087394','n02088094','n02088238','n02088364',
    'n02088466','n02088632','n02089078','n02089867',
    'n02089973','n02090379','n02090622','n02090721',
    'n02091032','n02091134','n02091244','n02091467',
    'n02091635','n02091831','n02092002','n02092339',
    'n02093256','n02093428','n02093647','n02093754',
    'n02093859','n02093991','n02094114','n02094258',
    'n02094433','n02095314','n02095570','n02095889',
    'n02096051','n02096177','n02096294','n02096437',
    'n02096585','n02097047','n02097130','n02097209',
    'n02097298','n02097474','n02097658','n02098105',
    'n02098286','n02098413','n02099267','n02099429',
    'n02099601','n02099712','n02099849','n02100236',
    'n02100583','n02100735','n02100877','n02101006',
    'n02101388','n02101556','n02102040','n02102177',
    'n02102318','n02102480','n02102973','n02104029',
    'n02104365','n02105056','n02105162','n02105251',
    'n02105412','n02105505','n02105641','n02105855',
    'n02106030','n02106166','n02106382','n02106550',
    'n02106662','n02107142','n02107312','n02107574',
    'n02107683','n02107908','n02108000','n02108089',
    'n02108422','n02108551','n02108915','n02109047',
    'n02109525','n02109961','n02110063','n02110185',
    'n02110341','n02110627','n02110806','n02110958',
    'n02111129','n02111277','n02111500','n02111889',
    'n02112018','n02112137','n02112350','n02112706',
    'n02113023','n02113186','n02113624','n02113712',
    'n02113799,'n02113978'))

```

```

#选出所有猫的种类
catSet = set([
    'n02123045','n02123159','n02123394','n02123597',
    'n02124075','n02125311','n02127052'])

```

- b) 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

```

model = ResNet50(weights='imagenet')

def scanInvalidImage(imageDirPath, imageCount, targetClassSet):
    fileCount = 0
    for item in os.listdir(imageDirPath):
        img_path = imageDirPath + '/' + item
        if (not isfile(img_path)) or fileCount >= imageCount:
            break
        fileCount += 1
        img = image.load_img(img_path, target_size=(224, 224))
        img_arr = image.img_to_array(img)
        img_arr = np.expand_dims(img_arr, axis=0)
        img_arr = preprocess_input(img_arr)

        preds = model.predict_on_batch(img_arr)
        decodePreds = decode_predictions(preds, top=100)[0]

        findInTargetClass = False
        for predTuple in decodePreds:
            if predTuple[0] in targetClassSet:
                findInTargetClass = True
                break

        if not findInTargetClass:
            print(img_path)

    print('scanInvalidImage fileCount:', fileCount)

#
scanInvalidImage(destPathDog, totalDogImgCount, dogSet)
scanInvalidImage(destPathCat, totalCatImgCount, catSet)

```

3. 对异常图片进行人工再次筛选，并剔除

- a) 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

```

invalidDogImgs = [
    'dog.5604.jpg', 'dog.8736.jpg', 'dog.2422.jpg', 'dog.12376.jpg', 'dog.9517.jpg',
    'dog.10237.jpg', 'dog.10801.jpg', 'dog.2614.jpg', 'dog.10161.jpg', 'dog.1773.jpg'
]

invalidCatImgs = [
    'cat.2893.jpg', 'cat.6345.jpg', 'cat.9596.jpg', 'cat.2520.jpg', 'cat.8087.jpg',
    'cat.12326.jpg', 'cat.6442.jpg', 'cat.8921.jpg', 'cat.10536.jpg', 'cat.10700.jpg',
    'cat.7377.jpg', 'cat.5974.jpg', 'cat.4338.jpg', 'cat.2663.jpg', 'cat.11565.jpg',
    'cat.2939.jpg', 'cat.5351.jpg', 'cat.10712.jpg', 'cat.5418.jpg', 'cat.7968.jpg',
    'cat.372.jpg', 'cat.10270.jpg', 'cat.3672.jpg', 'cat.2457.jpg', 'cat.11184.jpg',
    'cat.12272.jpg', 'cat.7564.jpg', 'cat.10636.jpg', 'cat.6699.jpg', 'cat.8456.jpg',
    'cat.9090.jpg', 'cat.2337.jpg', 'cat.2817.jpg', 'cat.9171.jpg', 'cat.11297.jpg',
    'cat.4308.jpg', 'cat.2150.jpg', 'cat.12493.jpg'
]

```

- b) 分别移动到[./data/invalid_train/dog]和[./data/invalid_train/cat]，不参与接下来的训练

```

def move_invalid_imgs(invalidImgs, srcPath, destPathPre, subPath):
    if not isdir(destPathPre):
        os.mkdir(destPathPre)
    destPath = destPathPre + subPath
    if not isdir(destPath):
        os.mkdir(destPath)
    for fileItem in invalidImgs:
        if isfile(srcPath + '/' + fileItem):
            os.rename(srcPath + '/' + fileItem, destPath + '/' + fileItem)

invalidDestPath = dataset_folder_path + '/invalid_train'
move_invalid_imgs(invalidDogImgs, destPathDog, invalidDestPath, '/dog')
move_invalid_imgs(invalidCatImgs, destPathCat, invalidDestPath, '/cat')

```

执行过程

1. 基于迁移学习的理念，从 keras 的预训练模型 Xception[5]（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
 - a) 代码如下：

```

dataset_folder_path = os.getcwd() + '/data'
train_dataset_folder_path = dataset_folder_path + '/train'
test_dataset_folder_path = dataset_folder_path + '/test'

image_size = (299, 299)
input_tensor = Input((image_size[0], image_size[1], 3))
x = input_tensor
x = Lambda(xception.preprocess_input)(x)

base_model = Xception(include_top=False, weights='imagenet', input_tensor=x, pooling='avg')

gen = ImageDataGenerator()
train_generator = gen.flow_from_directory(train_dataset_folder_path, image_size, shuffle=False,
                                         batch_size=16)
test_generator = gen.flow_from_directory(test_dataset_folder_path, image_size, shuffle=False,
                                         batch_size=16, class_mode=None)

X_train = base_model.predict_generator(train_generator)
X_test = base_model.predict_generator(test_generator)

with h5py.File('Xception.h5') as fp:
    fp.create_dataset('train', data = X_train)
    fp.create_dataset('test', data = X_test)
    fp.create_dataset('label', data = train_generator.classes)

```

- b) 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 keras 预训练模型 Xception 的源代码[6]，发现其预处理函数调用了 imagenet_utils.preprocess_input()，参数 mode='tf'，如下图：

```
325     def preprocess_input(x, **kwargs):
326         """Preprocesses a numpy array encoding a batch of images.
327
328         # Arguments
329         x: a 4D numpy array consists of RGB values within [0, 255].
330
331         # Returns
332         Preprocessed array.
333         """
334     return imagenet_utils.preprocess_input(x, mode='tf', **kwargs)
```

c) 查看 `imagenet_utils.preprocess_input()` 的源代码[7]

```
157     def preprocess_input(x, data_format=None, mode='caffe', **kwargs):
158         """Preprocesses a tensor or Numpy array encoding a batch of images.
159
160         .....
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190     if isinstance(x, np.ndarray):
191         return _preprocess_numpy_input(x, data_format=data_format,
192                                         mode=mode, **kwargs)
193     else:
194         return _preprocess_symbolic_input(x, data_format=data_format,
195                                         mode=mode, **kwargs)
```

在 `_preprocess_numpy_input()` 和 `_preprocess_symbolic_input()` 方法中都有以下逻辑：

```
45         if mode == 'tf':
46             x /= 127.5
47             x -= 1.
48         return x
```

由此证明了上述归一化逻辑。

2. 导入特征向量，然后对 `X` 和 `y` 重新随机排序，以备使用

```

X_train = []
X_test = []

with h5py.File("Xception.h5", 'r') as h:
    X_train.append(np.array(h['train']))
    X_test.append(np.array(h['test']))
    y_train = np.array(h['label'])

X_train = np.concatenate(X_train, axis=1)
X_test = np.concatenate(X_test, axis=1)

X_train, y_train = shuffle(X_train, y_train, random_state=42)

```

3. 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
 - a) 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。[8]
 - b) 再构建 `Dense` 层，输出维度为 1，激活函数设为 `sigmoid`。

```

from keras.models import *
from keras.applications import *
from keras.layers import *

input_tensor = Input(X_train.shape[1:])
print('input_tensor.shape:', input_tensor.shape)

x = Dropout(0.25)(input_tensor)
x = Dense(1, activation='sigmoid')(x)
myModel = Model(input_tensor, x)

myModel.compile(optimizer='adam',
                 loss='binary_crossentropy',
                 metrics=['accuracy'])

```

input_tensor.shape: (?, 2048)

4. 训练模型

训练模型

设置验证集大小为 20%，也就是说训练集是 $24952 * 0.8 = 19962$ 张图，验证集是 $24952 * 0.2 = 4990$ 张图。

```

train_history = myModel.fit(X_train, y_train, batch_size=128, epochs=8, validation_split=0.2)

Train on 19962 samples, validate on 4991 samples
Epoch 1/8
19962/19962 [=====] - 9s 475us/step - loss: 0.1317 - acc: 0.9803 - val_loss: 0.0418 - val_acc: 0.9932
Epoch 2/8
19962/19962 [=====] - 1s 29us/step - loss: 0.0357 - acc: 0.9931 - val_loss: 0.0266 - val_acc: 0.9946
Epoch 3/8
19962/19962 [=====] - 1s 30us/step - loss: 0.0265 - acc: 0.9938 - val_loss: 0.0213 - val_acc: 0.9946
Epoch 4/8
19962/19962 [=====] - 1s 29us/step - loss: 0.0224 - acc: 0.9945 - val_loss: 0.0188 - val_acc: 0.9946
Epoch 5/8
19962/19962 [=====] - 1s 29us/step - loss: 0.0203 - acc: 0.9945 - val_loss: 0.0172 - val_acc: 0.9948
Epoch 6/8
19962/19962 [=====] - 1s 29us/step - loss: 0.0186 - acc: 0.9950 - val_loss: 0.0163 - val_acc: 0.9946
Epoch 7/8
19962/19962 [=====] - 1s 29us/step - loss: 0.0171 - acc: 0.9952 - val_loss: 0.0157 - val_acc: 0.9948
Epoch 8/8
19962/19962 [=====] - 1s 29us/step - loss: 0.0162 - acc: 0.9956 - val_loss: 0.0151 - val_acc: 0.9948

```

5. 对测试数据集进行预测

预测并生成预测结果文件，用于上传kaggle。

```
In [10]: import pandas as pd  
  
y_pred = myModel.predict(X_test, verbose=1)  
y_pred = y_pred.clip(min=0.005, max=0.995)  
  
df = pd.read_csv("./data/sample_submission.csv")  
  
for i, fname in enumerate(test_generator.filenames):  
    index = int(fname[fname.rfind('/')+1:fname.rfind('.')])  
    df.set_value(index-1, 'label', y_pred[i])  
  
df.to_csv('predict.csv', index=None)  
df.head(10)
```

12500/12500 [=====] - 0s 33us/step

```
/home/ubuntu/anaconda3/envs/tensorflow_p36/lib/python3.6/site-packages/ipykernel_launcher.py:10:  
ill be removed in a future release. Please use .at[] or .iat[] accessors instead  
# Remove the CWD from sys.path while we load stuff.
```

Out[10]:

	id	label
0	1	0.995
1	2	0.995
2	3	0.995
3	4	0.995
4	5	0.005
5	6	0.005
6	7	0.005
7	8	0.005
8	9	0.005
9	10	0.005

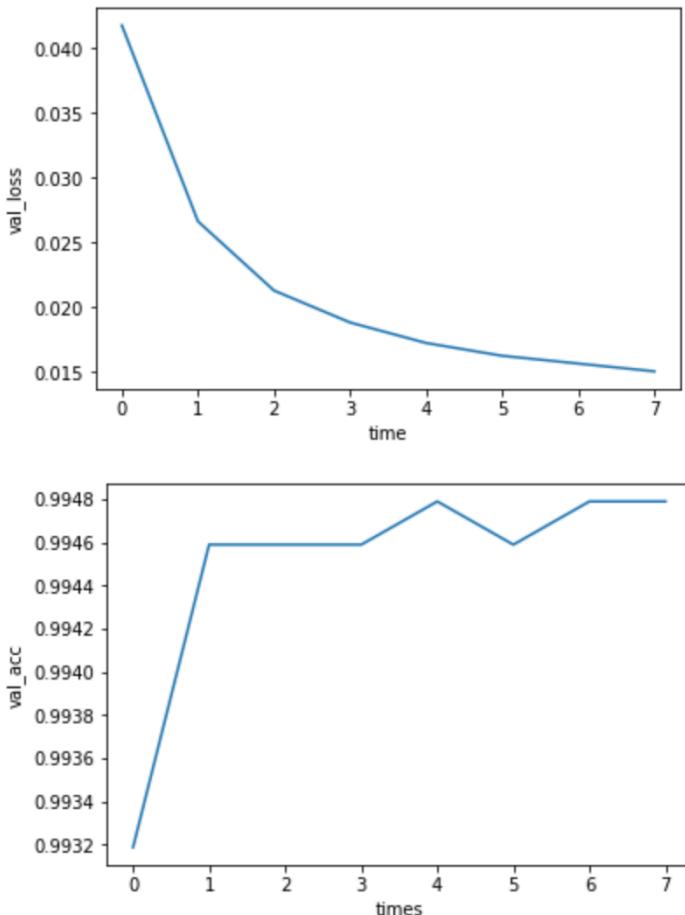
完善

- 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。
- 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

- 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：

Dogs vs. Cats Redux: Kernels Edition

Distinguish images of dogs from cats
1,314 teams · 2 years ago

Overview Data Kernels Discussion Leaderboard Rules Team My Submissions Late Submission

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
predict.csv	2 minutes ago	0 seconds	0 seconds	0.04156

Complete

Jump to your position on the leaderboard ▾

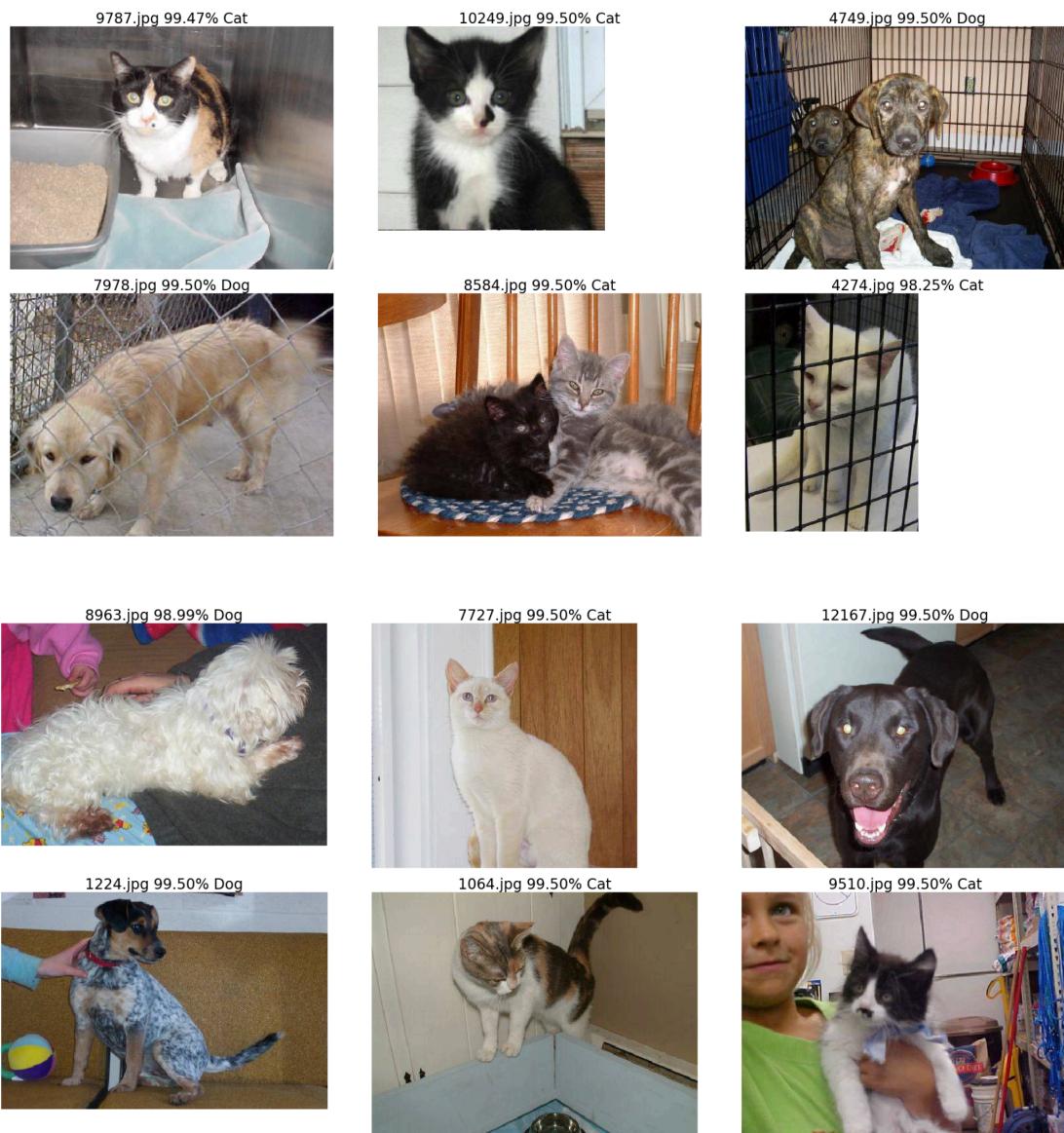
合理性分析

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

- 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。[9]

	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] AI 之路. Xception 算法详解 [OL]. <https://blog.csdn.net/u014380165/article/details/75142710>. 20170715
- [4] 清欢守护者. 精读深度学习论文(10) Xception [OL]. <https://zhuanlan.zhihu.com/p/33778384>. 20180212
- [5] François Chollet. Xception: Deep Learning with Depthwise Separable Convolutions [OL]. <https://arxiv.org/abs/1610.02357>. 20161007
- [6] François Chollet. Keras: Deep Learning for humans? [OL]. https://github.com/keras-team/keras-applications/blob/master/keras_applications/xception.py
- [7] François Chollet. Keras: Deep Learning for humans? [OL]. https://github.com/keras-team/keras-applications/blob/master/keras_applications/imagenet_utils.py
- [8] 斋藤康毅. 深度学习入门[M]. 北京:人民邮电出版社, 2018:192-193.

[9] 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