

机器学习纳米学位

猫狗大战毕业项目

张斌

2019年3月6日

I. 问题的定义

项目概述

本项目为kaggle竞赛项目，是要训练一个机器学习模型，最终输入一张图片来分辨图像中是猫还是狗。本项目是一个典型的计算机视觉问题中的图像二分类问题。本项目使用卷积神经网络CNN，卷积神经网络是深度学习技术中极具代表的网络结构之一，在图像处理领域取得了很大。本项目的输入集是由“Dogs vs. Cats Redux: Kernels Edition”项目提供的训练集和测试集，通过利用kaggle下载的训练集图片，利用CNN网络对图片进行多次积层和池化层处理，在输出层进行sigmoid计算得到两个类别各自的概率，通过损失函数的值对模型的各个参数进行调整，不断的评分，不断的调整，最终训练出优秀的模型。利用最终的模型，对测试集中的每个图像，预测图像是狗的概率（1 = 狗，0 = 猫）。

问题陈述

这次需要解决的问题是：通过利用计算机图像识别技术，对kaggle提供的训练集数据利用CNN的方式进行分析学习训练出优秀的图像识别模型，利用图像识别模型对测试集中的图片进行预测，预测图像是狗或猫的概率，并将结果存入sample_submission.csv文件中，上传至kaggle进行打分。

- 1、首先将对数据集中的图片进行筛选，利用箱型图原理，剔除异常的图片，主要包含低分辨率和错误的图片。
- 2、建立训练集和验证集。
- 3、由于神经网络输入点图像的维度和像素是固定的，通过对图像进行预处理，统一图像的维度。
- 4、利用迁移学习，使用开源的优秀的图像识别模型并加载权重作为固定的图像特征提取器，利用多个具有优秀权重的识别模型对数据进行预处理，观察各自的预测结果的准确率和多个模型联合起来的预测结果准确率。
- 5、结合迁移学习的模型，建立CNN的数据模型。
- 6、对CNN模型进行训练，得到最优的模型。
- 7、利用模型预测测试集中的图像是狗的概率，将结果按照要求存入sample_submission.csv文件中。

期望的结果是训练出的模型可以准确的识别出图像是狗还是猫，并在kaggle上得到优秀评分。

评价指标

kaggle的评估标准是log损失函数，以下为表达式：

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

其中：

n 是测试集中图片数量。

\hat{y}_i 是图片预测为狗的概率。

y_i 如果图像是狗，则为1，如果是猫，则为0。

$\log()$ 是自然对数。

对数损失越小，代表模型的性能越好。

交叉熵是分类问题中常用的损失函数，被广泛应用。我使用的评估指标是kaggle提出了，首先保证的评估指标的适用性。以上计算公式摘自kaggle“Dogs vs. Cats Redux: Kernels Edition”项目。

II. 分析

数据的探索

此数据集来自kaggle“Dogs vs. Cats Redux: Kernels Edition”项目。全部解压后包括：train、test、sample_submission.csv。

cat.1534.jpg	cat.4347.jpg	cat.7158.jpg	cat.9979.jpg	dog.1543.jpg	dog.4358.jpg	dog.7161.jpg	dog.997.jpg
cat.1535.jpg	cat.4348.jpg	cat.7159.jpg	cat.997.jpg	dog.1544.jpg	dog.4359.jpg	dog.7162.jpg	dog.9980.jpg
cat.1536.jpg	cat.4349.jpg	cat.715.jpg	cat.9980.jpg	dog.1545.jpg	dog.435.jpg	dog.7163.jpg	dog.9981.jpg
cat.1537.jpg	cat.434.jpg	cat.7160.jpg	cat.9981.jpg	dog.1546.jpg	dog.4360.jpg	dog.7164.jpg	dog.9982.jpg
cat.1538.jpg	cat.4350.jpg	cat.7161.jpg	cat.9982.jpg	dog.1547.jpg	dog.4361.jpg	dog.7165.jpg	dog.9983.jpg
cat.1539.jpg	cat.4351.jpg	cat.7162.jpg	cat.9983.jpg	dog.1548.jpg	dog.4362.jpg	dog.7166.jpg	dog.9984.jpg
cat.153.jpg	cat.4352.jpg	cat.7163.jpg	cat.9984.jpg	dog.1549.jpg	dog.4363.jpg	dog.7167.jpg	dog.9985.jpg
cat.1540.jpg	cat.4353.jpg	cat.7164.jpg	cat.9985.jpg	dog.154.jpg	dog.4364.jpg	dog.7168.jpg	dog.9986.jpg
cat.1541.jpg	cat.4354.jpg	cat.7165.jpg	cat.9986.jpg	dog.1550.jpg	dog.4365.jpg	dog.7169.jpg	dog.9987.jpg
cat.1542.jpg	cat.4355.jpg	cat.7166.jpg	cat.9987.jpg	dog.1551.jpg	dog.4366.jpg	dog.716.jpg	dog.9988.jpg
cat.1543.jpg	cat.4356.jpg	cat.7167.jpg	cat.9988.jpg	dog.1552.jpg	dog.4368.jpg	dog.7170.jpg	dog.9989.jpg
cat.1544.jpg	cat.4357.jpg	cat.7168.jpg	cat.9989.jpg	dog.1553.jpg	dog.4369.jpg	dog.7171.jpg	dog.9990.jpg
cat.1545.jpg	cat.4358.jpg	cat.7169.jpg	cat.998.jpg	dog.1554.jpg	dog.436.jpg	dog.7172.jpg	dog.9991.jpg
cat.1546.jpg	cat.4359.jpg	cat.716.jpg	cat.9990.jpg	dog.1555.jpg	dog.4370.jpg	dog.7173.jpg	dog.9992.jpg
cat.1547.jpg	cat.435.jpg	cat.7170.jpg	cat.9991.jpg	dog.1556.jpg	dog.4371.jpg	dog.7174.jpg	dog.9993.jpg
cat.1548.jpg	cat.4360.jpg	cat.7171.jpg	cat.9992.jpg	dog.1557.jpg	dog.4372.jpg	dog.7175.jpg	dog.9994.jpg
cat.1549.jpg	cat.4361.jpg	cat.7172.jpg	cat.9993.jpg	dog.1558.jpg	dog.4373.jpg	dog.7176.jpg	dog.9995.jpg
cat.154.jpg	cat.4362.jpg	cat.7173.jpg	cat.9994.jpg	dog.1559.jpg	dog.4374.jpg	dog.7177.jpg	dog.9996.jpg
cat.1550.jpg	cat.4363.jpg	cat.7174.jpg	cat.9995.jpg	dog.155.jpg	dog.4375.jpg	dog.7178.jpg	dog.9997.jpg
cat.1551.jpg	cat.4364.jpg	cat.7175.jpg	cat.9996.jpg	dog.1560.jpg	dog.4376.jpg	dog.7179.jpg	dog.9998.jpg
cat.1552.jpg	cat.4365.jpg	cat.7176.jpg	cat.9997.jpg	dog.1561.jpg	dog.4377.jpg	dog.717.jpg	dog.9999.jpg
cat.1553.jpg	cat.4366.jpg	cat.7177.jpg	cat.9998.jpg	dog.1562.jpg	dog.4378.jpg	dog.7180.jpg	dog.9999.jpg
cat.1554.jpg	cat.4367.jpg	cat.7178.jpg	cat.9999.jpg	dog.1563.jpg	dog.4379.jpg	dog.7181.jpg	dog.9990.jpg
cat.1555.jpg	cat.4368.jpg	cat.7179.jpg	cat.999.jpg	dog.1564.jpg	dog.437.jpg	dog.7182.jpg	dog.9991.jpg
cat.1556.jpg	cat.4369.jpg	cat.717.jpg	cat.99.jpg	dog.1565.jpg	dog.4380.jpg	dog.7183.jpg	dog.9992.jpg
cat.1557.jpg	cat.436.jpg	cat.7180.jpg	cat.9.jpg	dog.1566.jpg	dog.4381.jpg	dog.7184.jpg	dog.9993.jpg
cat.1558.jpg	cat.4370.jpg	cat.7181.jpg	dog.0.jpg	dog.1567.jpg	dog.4382.jpg	dog.7185.jpg	dog.9994.jpg
cat.1559.jpg	cat.4371.jpg	cat.7182.jpg	dog.10000.jpg	dog.1568.jpg	dog.4383.jpg	dog.7186.jpg	dog.9995.jpg
cat.155.jpg	cat.4372.jpg	cat.7183.jpg	dog.10001.jpg	dog.1569.jpg	dog.4384.jpg	dog.7187.jpg	dog.9996.jpg

\$

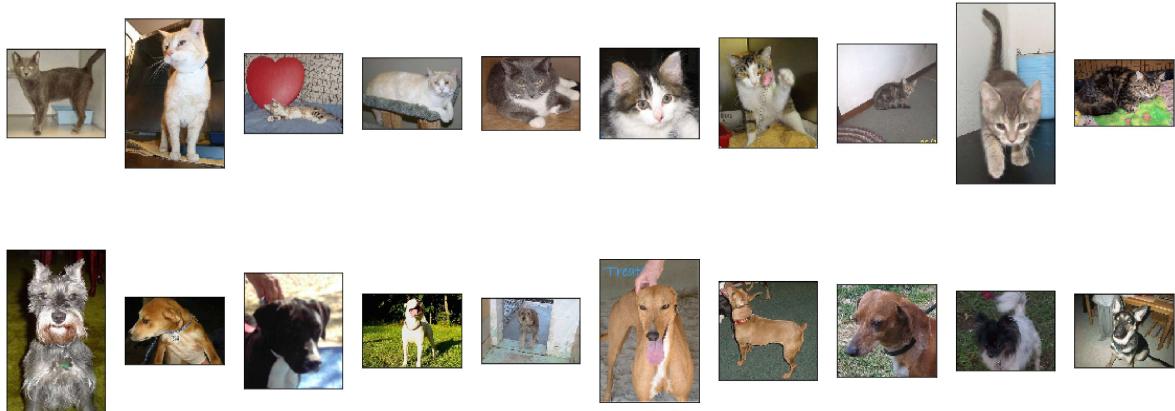
train文件夹中包含25000张猫狗的图片，每张图片的名字为“类型+序号.jpg”，此文件夹为训练集，用训练集图片对模型进行训练，根据图片的名称可以定义图片中的内容为猫或狗。

1099.jpg	12021.jpg	1795.jpg	2818.jpg	3841.jpg	4865.jpg	5889.jpg	6911.jpg	7935.jpg	8959.jpg	9982.jpg
109.jpg	12022.jpg	1796.jpg	2819.jpg	3842.jpg	4866.jpg	588.jpg	6912.jpg	7936.jpg	8965.jpg	9983.jpg
10.jpg	12023.jpg	1797.jpg	2821.jpg	3843.jpg	4867.jpg	5890.jpg	6913.jpg	7937.jpg	8960.jpg	9984.jpg
11000.jpg	12024.jpg	1798.jpg	2820.jpg	3844.jpg	4868.jpg	5891.jpg	6914.jpg	7938.jpg	8961.jpg	9985.jpg
11001.jpg	12025.jpg	1799.jpg	2821.jpg	3845.jpg	4869.jpg	5892.jpg	6915.jpg	7939.jpg	8962.jpg	9986.jpg
11002.jpg	12026.jpg	179.jpg	2822.jpg	3846.jpg	4861.jpg	5893.jpg	6916.jpg	793.jpg	8963.jpg	9987.jpg
11003.jpg	12027.jpg	17.jpg	2823.jpg	3847.jpg	4870.jpg	5894.jpg	6917.jpg	7940.jpg	8964.jpg	9988.jpg
11004.jpg	12028.jpg	1800.jpg	2824.jpg	3848.jpg	4871.jpg	5895.jpg	6918.jpg	7941.jpg	8965.jpg	9989.jpg
11005.jpg	12029.jpg	1801.jpg	2825.jpg	3849.jpg	4872.jpg	5896.jpg	6919.jpg	7942.jpg	8966.jpg	998.jpg
11006.jpg	12020.jpg	1802.jpg	2826.jpg	384.jpg	4873.jpg	5897.jpg	691.jpg	7943.jpg	8967.jpg	9990.jpg
11007.jpg	12030.jpg	1803.jpg	2827.jpg	3850.jpg	4874.jpg	5898.jpg	6920.jpg	7944.jpg	8968.jpg	9991.jpg
11008.jpg	12031.jpg	1804.jpg	2828.jpg	3851.jpg	4875.jpg	5899.jpg	6921.jpg	7945.jpg	8969.jpg	9992.jpg
11009.jpg	12032.jpg	1805.jpg	2829.jpg	3852.jpg	4876.jpg	589.jpg	6922.jpg	7946.jpg	896.jpg	9993.jpg
1100.jpg	12033.jpg	1806.jpg	283.jpg	3853.jpg	4877.jpg	58.jpg	6923.jpg	7947.jpg	8970.jpg	9994.jpg
11010.jpg	12034.jpg	1807.jpg	2830.jpg	3854.jpg	4878.jpg	5900.jpg	6924.jpg	7948.jpg	8971.jpg	9995.jpg
11011.jpg	12035.jpg	1808.jpg	2831.jpg	3855.jpg	4879.jpg	5901.jpg	6925.jpg	7949.jpg	8972.jpg	9996.jpg
11012.jpg	12036.jpg	1809.jpg	2832.jpg	3856.jpg	487.jpg	5902.jpg	6926.jpg	794.jpg	8973.jpg	9997.jpg
11013.jpg	12037.jpg	180.jpg	2833.jpg	3857.jpg	4880.jpg	5903.jpg	6927.jpg	7950.jpg	8974.jpg	9998.jpg
11014.jpg	12038.jpg	1810.jpg	2834.jpg	3858.jpg	4881.jpg	5904.jpg	6928.jpg	7951.jpg	8975.jpg	9999.jpg
11015.jpg	12039.jpg	1811.jpg	2835.jpg	3859.jpg	4882.jpg	5905.jpg	6929.jpg	7952.jpg	8976.jpg	999.jpg
11016.jpg	12030.jpg	1812.jpg	2836.jpg	385.jpg	4883.jpg	5906.jpg	692.jpg	7953.jpg	8977.jpg	99.jpg
11017.jpg	12040.jpg	1813.jpg	2837.jpg	3860.jpg	4884.jpg	5907.jpg	6930.jpg	7954.jpg	8978.jpg	9.jpg
11018.jpg	12041.jpg	1814.jpg	2838.jpg	3861.jpg	4885.jpg	5908.jpg	6931.jpg	7955.jpg	8979.jpg	
11019.jpg	12042.jpg	1815.jpg	2839.jpg	3862.jpg	4886.jpg	5909.jpg	6932.jpg	7956.jpg	897.jpg	
11010.jpg	12043.jpg	1816.jpg	283.jpg	3863.jpg	4887.jpg	590.jpg	6933.jpg	7957.jpg	8980.jpg	
11020.jpg	12044.jpg	1817.jpg	2840.jpg	3864.jpg	4888.jpg	5910.jpg	6934.jpg	7958.jpg	8981.jpg	
11021.jpg	12045.jpg	1818.jpg	2841.jpg	3865.jpg	4889.jpg	5911.jpg	6935.jpg	7959.jpg	8982.jpg	
11022.jpg	12046.jpg	1819.jpg	2842.jpg	3866.jpg	488.jpg	5912.jpg	6936.jpg	795.jpg	8983.jpg	
11023.jpg	12047.jpg	181.jpg	2843.jpg	3867.jpg	4890.jpg	5913.jpg	6937.jpg	7960.jpg	8984.jpg	

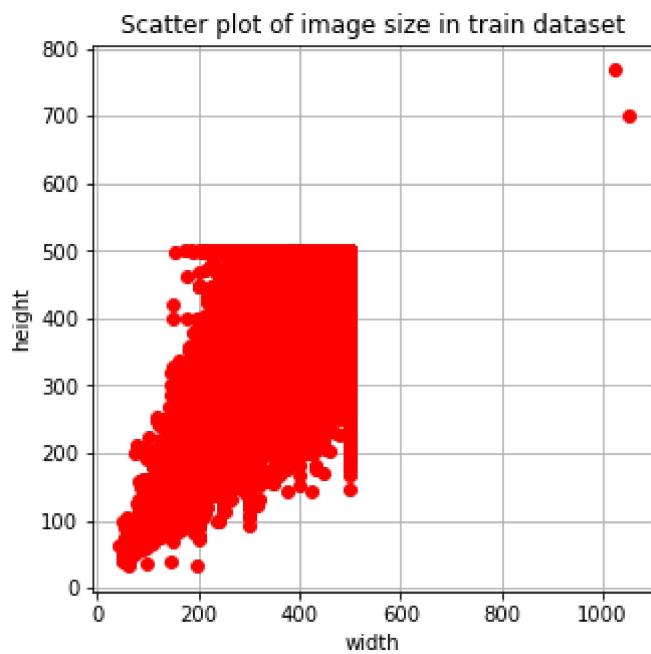
\$

test文件夹中包含12500张猫狗的图片，每张图片的名字为“序号.jpg”，此文件夹为测试集，利用算法模型预测测试集中图片的内容（即图片中是狗的概率），并将结果存入sample_submission.csv。

sample_submission.csv文件内需要将模型预测的测试集中图片是狗的概率和图片序号——对应填入，最终上传至kaggle打分。



以上为kaggle“Dogs vs. Cats Redux: Kernels Edition”项目提供的部分数据内容。

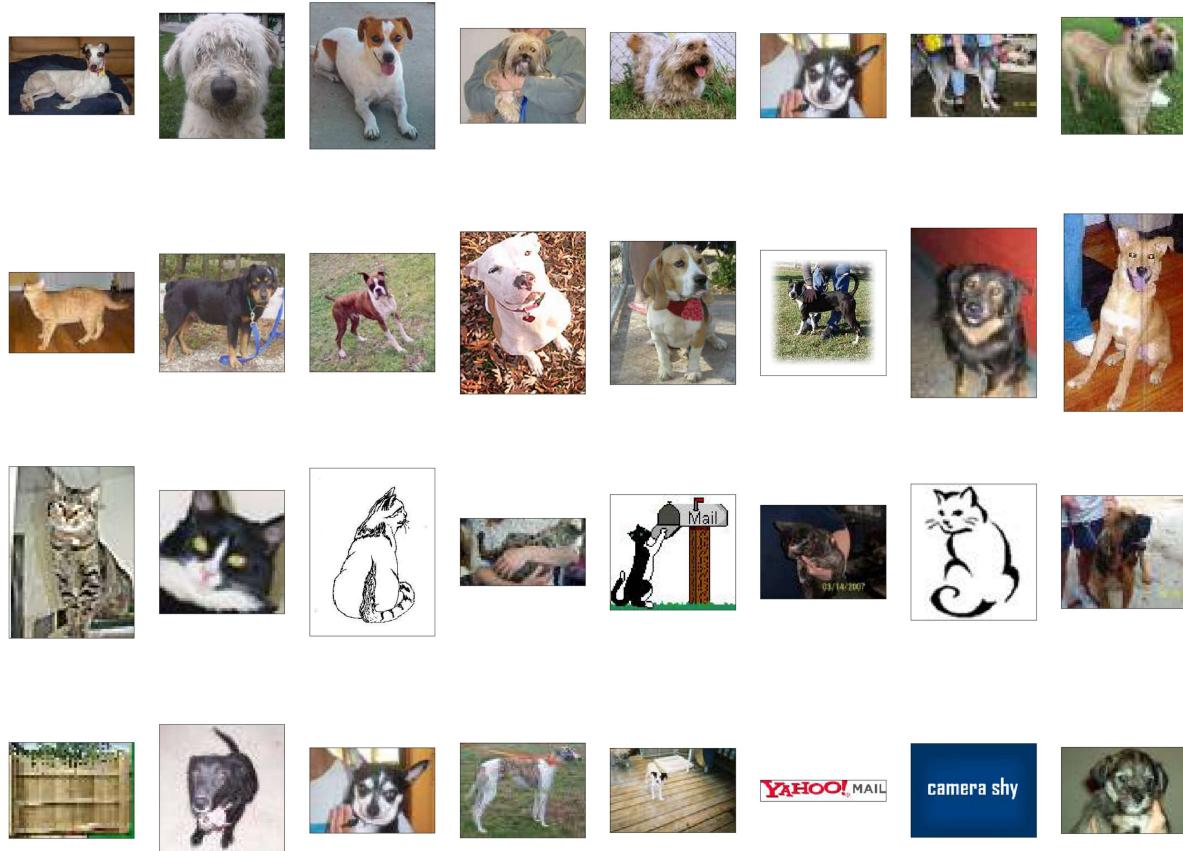
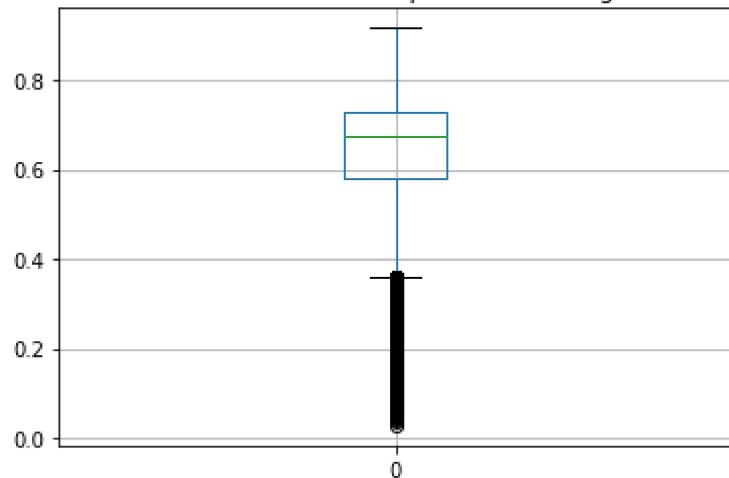


kaggle提供的train和test中的图片，以上是图片长宽的散点图，上图说明图片的大小有一定差别，需要在输入模

型前统一大小，并且有些图片的像素相比其他图片是过于低的，这个也应引起注意。

kaggle提供的数据集中，应重点关注训练集train中异常的图片，主要包括分辨率过低，图片大小太小等。对train中所有的图片进行图像直方图分析，对图片中包含的色彩与图片的像素的比值进行分析，此次利用箱型图原理，显示一组数据分散情况，箱形图为我们提供了识别异常值的一个标准：异常值被定义为小于 $Q1 - 1.5IQR$ 或大于 $Q3 + 1.5IQR$ 的值，从而筛选出异常值，并剔除。以下是箱型图和异常值，圆圈代表异常值，我们应该对异常值进行剔除。通过箱型图判断的异常值总共220张。

The ratio of color to pixel of the image



以上为判断为异常值的部分图片，其中也包含一些正常的图片，但是由于220张在25000张图片中的比例较小，所以为了方便，将220张图片全部剔除。

算法和技术

对给定的图片，计算机识别图片的内容，预测出图像属于预先定义类别中的哪一类，属于计算机视觉领域，本项目是二分类的问题，目前解决这类问题的核心技术框架是深度学习，针对图像类型的数据，深度学习中的卷积神经网络架构CNN又性能突出，卷积神经网络是一类包含卷积计算且具有深度结构的前馈神经网络，卷积神经网络能够进行平移不变分类，因此也被称为“平移不变人工神经网络”。CNN网络层级结构主要有5个层级结构：

- 1. 输入层
- 2. 卷积层
- 3. 激活层
- 4. 池化层
- 5. 全连接层

输入层

卷积神经网络的输入层可以处理多维数据，接收外部传入数据的一层，由于使用梯度下降进行学习，卷积神经网络的输入特征需要进行标准化处理。

卷积层

使用卷积核进行特征提取和特征映射。

激活层

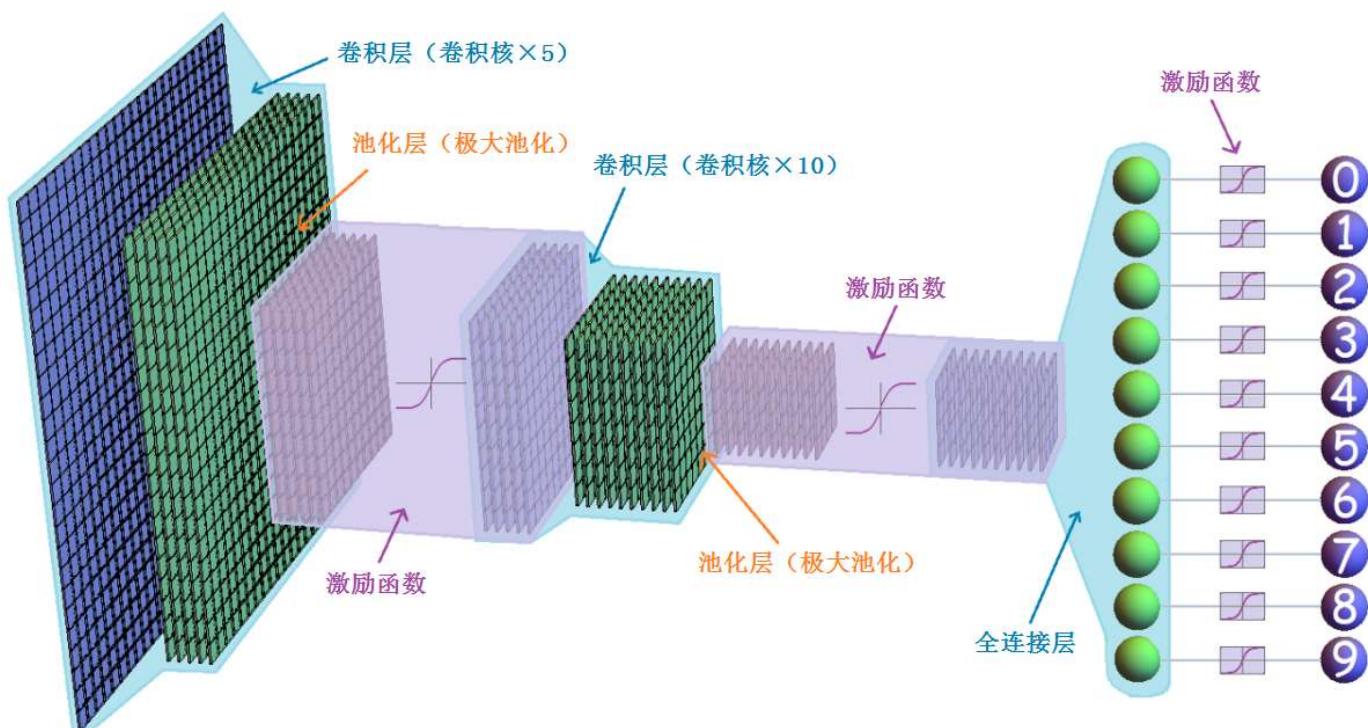
所谓激活，实际上是对卷积层的输出结果做一次非线性映射。如果不用激活函数，这种情况下，每一层的输出都是上一层输入的线性函数。

池化层

在卷积层进行特征提取后，输出的特征图会被传递至池化层进行特征选择和信息过滤。池化层包含预设定的池化函数，其功能是将特征图中单个点的结果替换为其相邻区域的特征图统计量，池化的功能是不断降低维数，以减少网络中的参数和计算次数。这缩短了训练时间并控制过度拟合。

全连接层

卷积神经网络中的全连接层等价于传统前馈神经网络中的隐含层。全连接层通常搭建在卷积神经网络隐含层的最后部分，并只向其它全连接层传递信号。特征图在全连接层中会失去3维结构，被展开为向量并通过激活函数传递至下一层或输出层。



输入层读入图像作为网络的数据输入，经过多个卷积层、激活层、池化层卷积后，神经网络主要是对图片的特征向量进行抽取，最后进入全连接层判断对象的类别，因此可以利用已经训练好的卷积神经网络提取图片中复杂的几何特征，即将原始图片用已经训练好的卷积神经网络处理之后的输出，作为新的输入，然后加上定制的全连接层，去进行分类。在模型训练的过程中，只改变新加的全连接层的权重，对模型进行优化。卷积神经网络是一种特殊的神经网络结构，可以通过卷积操作实现对图像特征的优化提取，提炼视觉特征提供图像分类的准确率。卷积神经网络在计算机图像识别技术中优势突出，是一种优秀的神经网络，在图像识别的工作中，选择最适合的卷积神经网络模型将能保证极高的准确率。在选择成熟的神经网络卷积层的过程中，由于不同模型的优缺点不同，计划比较InceptionResNetV2, InceptionV3, ResNet50, Xception四个高准确率的模型的优劣，并将组合四个神经网络模型或三个神经网络模型进行拼接，传入定制全连接层，利用交叉熵损失函数训练出最优模型，最终比较单独选用InceptionResNetV2, InceptionV3, ResNet50, Xception之中一种模型和使用以上四个模型进行拼接或其中三个进行拼接的模型中，那个模型预测图片分类的准确率高。依据Udacity学习的经验，本次计划使用Keras框架实现以上功能。

模型的选择

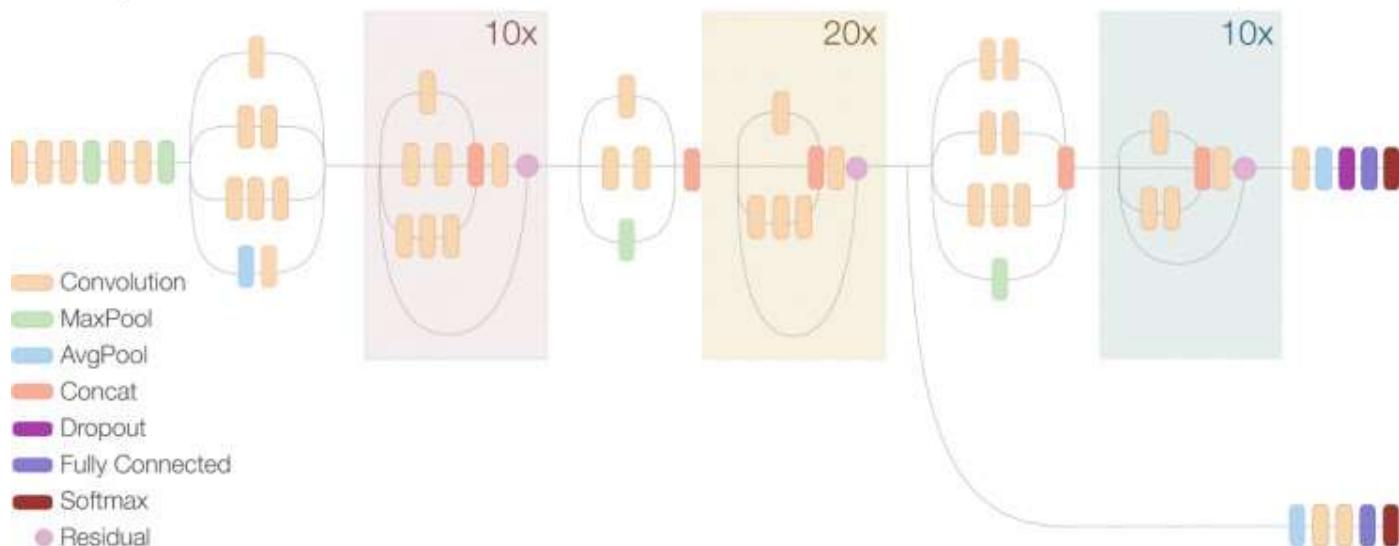
InceptionResNetV2

它在ILSVRC图像分类基准测试中实现了当下最好的成绩。Inception-ResNet-v2是早期Inception V3模型变化而来，从微软的残差网络（ResNet）论文中得到了一些灵感。

Inception Resnet V2 Network



Compressed View



Inception-ResNet-v2图中最上部分，你能看到整个网络扩展了。注意该网络被认为比先前的Inception V3还要深一些。在图中主要部分重复的残差区块已经被压缩了，所以整个网络看起来更加直观。另外注意到图中inception区块被简化了，比先前的Inception V3种要包含更少的并行塔。

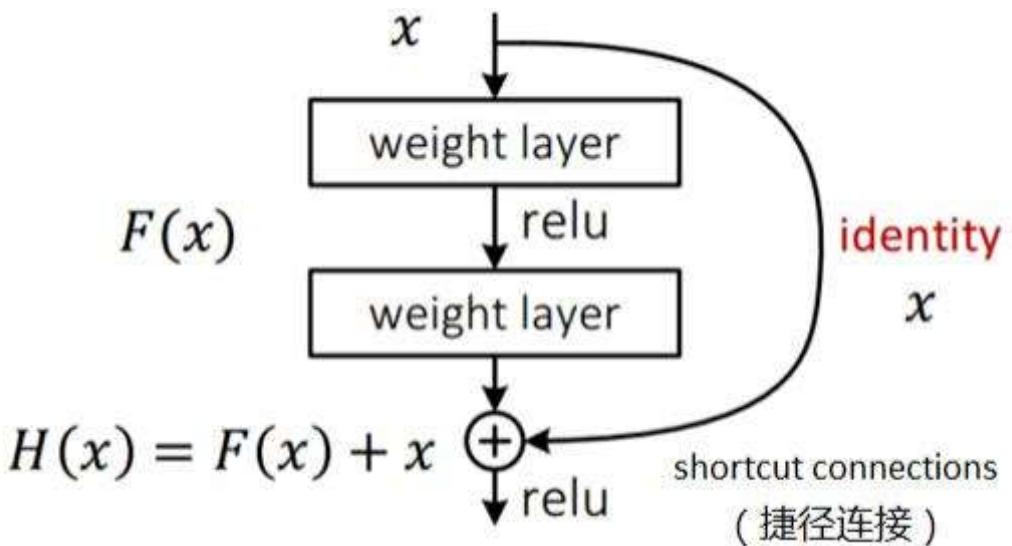
如下方图表所示，Inception-ResNet-v2架构的精确度比之前的最优模型更高，图表中所示为基于单个图像的ILSVRC 2012图像分类标准得出的排行第一与排行第五的有效精确度。此外，该新模型仅仅要求两倍于Inception v3的容量与计算能力。

Model	Architecture	Checkpoint	Top-1 Accuracy	Top-5 Accuracy
Inception-ResNet-v2	Code	inception_resnet_v2_2016_08_30.tar.gz	80.4	95.3
Inception V3	Code	inception_v3_2016_08_28.tar.gz	78.0	93.9
ResNet 152	Code	resnet_v1_152_2016_08_28.tar.gz	76.8	93.2
ResNet V2 200	Code	TBA	79.9*	95.2*

(*): Results quoted in ResNet paper.

ResNet50:

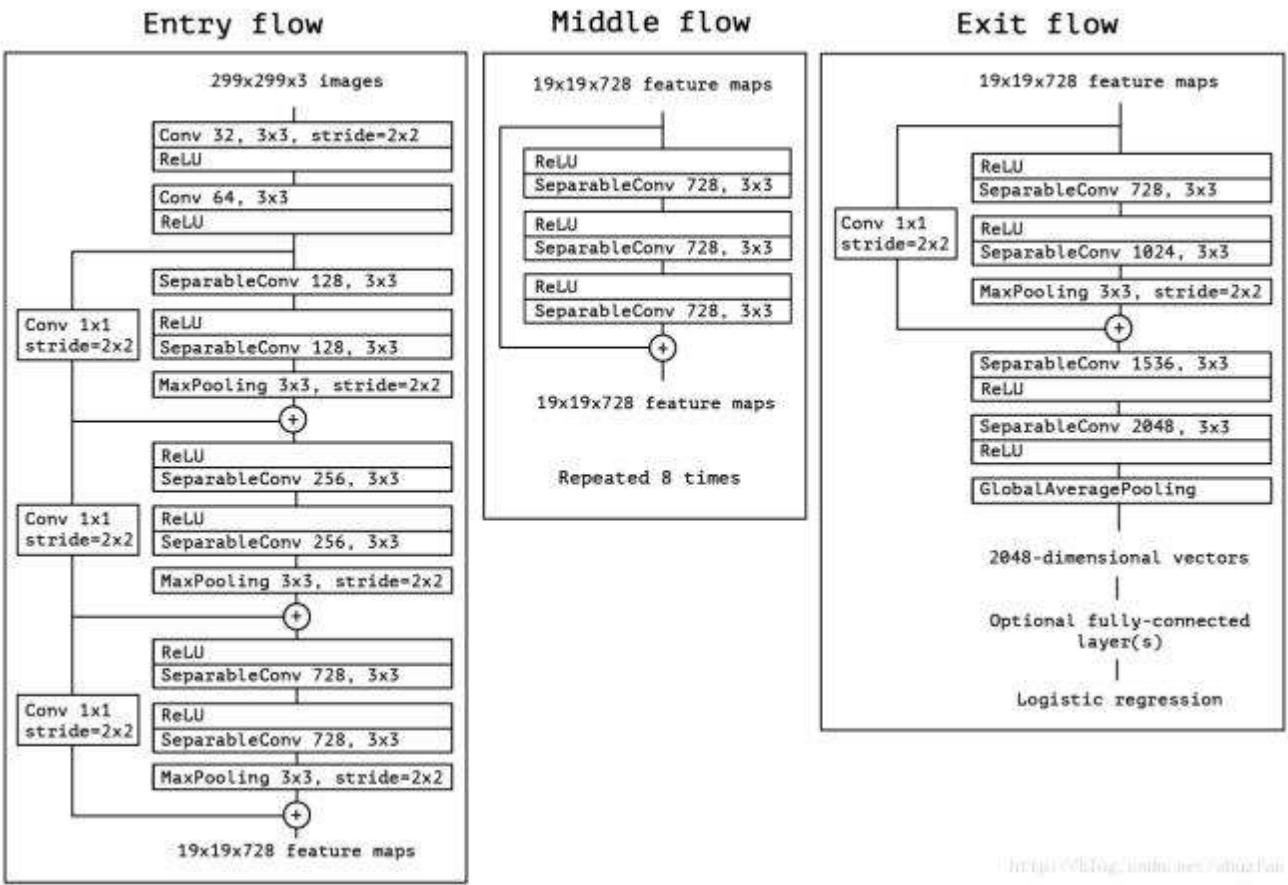
ResNet (Residual Neural Network) 由微软研究院的Kaiming He等四名华人提出，通过使用ResNet Unit成功训练出了152层的神经网络，并在ILSVRC2015比赛中取得冠军，在top5上的错误率为3.57%，同时参数量比VGGNet低，效果非常突出。ResNet的结构可以极快的加速神经网络的训练，模型的准确率也有比较大的提升。同时ResNet的推广性非常好，甚至可以直接用到InceptionNet网络中。ResNet的主要思想是在网络中增加了直连通道，即Highway Network的思想。此前的网络结构是性能输入做一个非线性变换，而Highway Network则允许保留之前网络层的一定比例的输出。ResNet的思想和Highway Network的思想也非常类似，允许原始输入信息直接传到后面的层中。这样的话这一层的神经网络可以不用学习整个的输出，而是学习上一个网络输出的残差，因此ResNet又叫做残差网络。



这种残差跳跃式的结构，打破了传统的神经网络n-1层的输出只能给n层作为输入的惯例，使某一层的输出可以直接跨过几层作为后面某一层的输入，其意义在于为叠加多层网络而使得整个学习模型的错误率不降反升的难题提供了新的方向。在此之前，深度神经网络常常会有梯度消失问题的困扰，即来自误差函数的梯度信号会在反向传播回更早的层时指数级地下降。本质上讲，在误差信号反向回到更早的层时，它们会变得非常小以至于网络无法学习。但是，因为ResNet 的梯度信号可以直接通过捷径连接回到更早的层，所以一下子就可以构建 50 层、101 层、152 层甚至1000 层以上的网络了，而且它们的表现依然良好。那时候，这在当时最佳的基础上实现了巨大的飞跃（这个22层的网络赢得了 ILSVRC 2014 挑战赛）。

Xception:

Xception 实际上采用了类似于 ResNet 的网络结构，主体部分采用了模块化设计。Xception 结构如下图所示，分为Entry flow; Middle flow; Exit flow; Entry flow 包含 8个conv；Middle flow 包含 $3 \times 8 = 24$ 个conv；Exit flow包含4个conv，所以Xception共计36层。



Xception 是对 Inception v3 的改进，是一种 Extreme Inception，因而得名 Xception，其主要是借鉴（非采用） depthwise separable convolution 来替换原来 Inception v3 中的卷积操作。

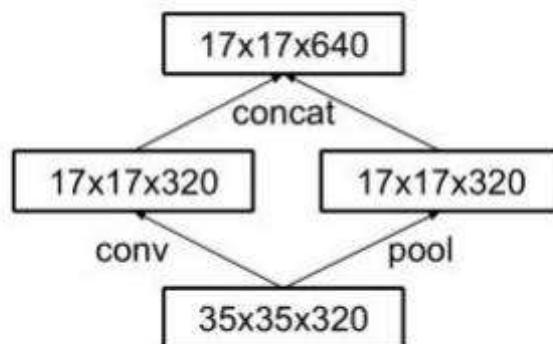
Inception V3:

Inception 网络是 CNN 分类器发展史上一个重要的里程碑。在 Inception 出现之前，大部分流行 CNN 仅仅是把卷积层堆叠得越来越多，使网络越来越深，以此希望能够得到更好的性能。Inception v3 整合了前面 Inception v2 中提到的所有升级，还使用了：

- RMSProp 优化器；
- Factorized 7x7 卷积；
- 辅助分类器使用了 BatchNorm；
- 标签平滑。

Inception V3 避免网络表达瓶颈，尤其在网络的前端。feature map 急剧减小，这样对层的压缩过大，会损失大量信息，模型训练困难；高维特征的局部处理更困难；在较低维度空间聚合，不会损失表达能力；平衡网络的宽度和深度。

在 2015 年 12 月提出的 Inception V3 结构借鉴 Inception 的结构设计了采用一种并行的降维结构，如下图：



经过优化后的inception v3网络与其他网络识别误差率，如表所示：

Network	Crops Evaluated	Top-5 Error	Top-1 Error
GoogLeNet [20]	10	-	9.15%
GoogLeNet [20]	144	-	7.89%
VGG [18]	-	24.4%	6.8%
BN-Inception [7]	144	22%	5.82%
PReLU [6]	10	24.27%	7.38%
PReLU [6]	-	21.59%	5.71%
Inception-v3	12	19.47%	4.48%
Inception-v3	144	18.77%	4.2%

以下为各个模型比较情况

模型	大小	Top1准确率	Top5准确率	参数数目	深度
Xception	88MB	0.790	0.945	22,910,480	126
VGG16	528MB	0.715	0.901	138,357,544	23
VGG19	549MB	0.727	0.910	143,667,240	26
ResNet50	99MB	0.759	0.929	25,636,712	168
InceptionV3	92MB	0.788	0.944	23,851,784	159
InceptionResNetV2	215MB	0.804	0.953	55,873,736	572
MobileNet	17MB	0.665	0.871	4,253,864	88
MobileNetV2	14MB	0.713	0.901	3,538,984	88
DenseNet121	33MB	0.750	0.923	8,062,504	121
DenseNet169	57MB	0.762	0.932	14,307,880	169
DenseNet201	80MB	0.773	0.936	20,242,984	201
NASNetMobile	23MB	0.744	0.919	5,326,716	-
NASNetLarge	343MB	0.825	0.960	88,949,818	-

从以上图中的比较，完全可以说明利用以上模型的合理性，本次选取的多个模型均已在实战中得到了优异的成绩。InceptionResNetV2, InceptionV3, ResNet50, Xception四个模型的关键参数设定为：

(weights='imagenet', include_top=False) , weights='imagenet'代表加载预训练权重, include_top=False表示不保留顶层的3个全连接网络。

基准模型

本项目的最低要求是kaggle Public Leaderboard前10%。

在kaggle上，总共有1314只队伍参加了比赛，前10%为131位之前，131位的得分是0.06127，所以模型预测结果分数要小于0.06127。

III. 方法

数据预处理

首先将对train数据集中进行箱型图分析后的异常值进行分析，异常值共计220张，删除后train数据集中图片为24780张，test测试集中图片为12500张，清理的图片已在上文可视化了部分内容。之后利用train中的数据随机抽取20%建立验证集。

由于神经网络输入点图像的维度和像素是固定的，通过对图像进行预处理，统一图像的维度和像素。

```
# 移动异常图片到'abnormal_pic'文件夹
# 记录移动pic的数量rmove_pic_num
rmove_pic_num = 0
for pic_file in abnormal_pic_list:
    shutil.move( pic_file, 'data/abnormal_pic/' + pic_file[11:-4])
    rmove_pic_num = rmove_pic_num + 1
print('共移动图片' + str(rmove_pic_num) + '张。')
```

```
# 利用train中的数据随机抽取20%建立验证集
index_valid = np.random.choice(train_files.shape[0], int(len(train_files)*0.2), replace=False)
index_train_files = np.arange(train_files.shape[0])
index_train = np.delete(index_train_files, index_valid)
```

```
# 将图片转为数字，并规范图片的大小
def path_to_tensor(img_path, size):
    img = image.load_img(img_path, target_size=size)
    x = image.img_to_array(img)
    return np.expand_dims(x, axis=0)
# 批量转换
def paths_to_tensor(img_paths, size):
    list_of_tensors = [path_to_tensor(img_path, size) for img_path in img_paths]
    return np.vstack(list_of_tensors)
train_mode = paths_to_tensor(train_files_rem, (299, 299))
valid_mode = paths_to_tensor(valid_files_rem, (299, 299))
test_mode = paths_to_tensor(test_files, (299, 299))
```

执行过程

生成迁移学习特征向量

InceptionResNetV2,Xception, InceptionV3和ResNet50这四个模型对于输入图片都有各自的默认值，InceptionResNetV2、Xception 和 InceptionV3 默认输入图片大小是(299,299)，ResNet50 默认输入图片大小是(224,224)；在将图片输入模型前，首先使用每个模型的预处理函数 preprocess_input，将数据处理成该模型的标准输入。

常见的卷积神经网络结构在前面的若干层都是卷积池化层及其各种变种，后面几层都是全连接层，这些全连接层之前的网络层被称为瓶颈层 (bottleneck). 将新的图片通过训练好的卷积神经网络直到瓶颈层的过程可以看做是对图像进行特征提取的过程。一般情况下，为了减少内存的消耗，加快计算的过程，再将瓶颈层的结果输入全连接层之前，做一次全局平均池化，为数据下一步进入定制层进行降维，加快速度的同时，也降低过拟合的程度。在

经过全局平均池化后，每个模型都将图片处理成一个行向量。在得到提取的迁移学习特征向量后，将数据存入npz文件中，待后续使用。

以下为生成迁移学习特征向量相关函数代码：

```
def Image_feature_extractor(input_model, image_size, train_mode, valid_mode, test_mode,
lambda_func=None):
    # 建立预训练的模型
    print('建立模型')
    width = image_size[0]
    height = image_size[1]
    input_tensor = Input((height, width, 3))
    x = input_tensor
    if lambda_func:
        x = Lambda(lambda_func)(x)
    base_model = input_model(input_tensor=x, weights='imagenet', include_top=False)
    # 创建新的model
    model = Model(base_model.input, GlobalAveragePooling2D()(base_model.output))
    print('完成')
    # 提取特征向量
    train = model.predict(train_mode)
    print('实例，查看提取特征向量情况' + str(train[0]))
    print('完成train预处理')
    valid = model.predict(valid_mode)
    print('完成valid预处理')
    test = model.predict(test_mode)
    print('完成test预处理')
    train_target = train_targets
    valid_target = valid_targets
    #为了后续使用test_name, 所以此次将test_files出入npz文件中
    test_files_name = test_files

    # 概览模型
    # model.summary()

    # print resule
    print('There are %d training images.' % len(train))
    print('There are %d validation images.' % len(valid))
    print('There are %d test images.' % len(test))

    # 将提取的特征向量存入文件
    np.savez("data/pre_data/cat_dog_%s.npz"%input_model.__name__, train = train, valid = valid,
test = test, train_target = train_target, valid_target = valid_target, test_files_name =
test_files_name)
```

载入特征向量

经过上面的代码运算，得到特征向量，分别是：

- cat_dog_Xception.npz;
- cat_dog_InceptionResNetV2.npz;
- cat_dog_ResNet50.npz;
- cat_dog_InceptionV3.npz;

在下一步载入各个特征向量，分别6次运行以下函数，每次运行需清除之前的数据。

预测6次分别是：

- 单独载入cat_dog_Xception特征向量；
- 单独载入cat_dog_InceptionResNetV2特征向量；
- 单独载入cat_dog_ResNet50特征向量；

- 单独载入cat_dog_InceptionV3特征向量；
- 载入将四个特征向量通过np.concatenate函数进行连接生成的特征向量。
- 载入将三个特征向量cat_dog_Xception、cat_dog_InceptionResNetV2、cat_dog_InceptionV3通过np.concatenate函数进行连接生成的特征向量。

```
#加载预处理的数据进行拼接，建立训练、验证、测试数据集
X_train = []
X_valid = []
X_test = []

y_train = []
y_valid = []

#加载预处理的数据
for filename in ['data/pre_data/cat_dog_Xception.npz',
                 'data/pre_data/cat_dog_InceptionResNetV2.npz',
                 'data/pre_data/cat_dog_ResNet50.npz',
                 'data/pre_data/cat_dog_InceptionV3.npz']:
    mid = np.load(filename)
    X_train.append(mid['train'])
    X_valid.append(mid['valid'])
    X_test.append(mid['test'])
    test_files_name = mid['test_files_name']

y_train = mid['train_target']
y_valid = mid['valid_target']
```

通过运行X_train, y_train = shuffle(X_train, y_train)将训练集中的数据打乱。

构建模型

构建定制的模型，载入预处理的特征向量数据，对数据进行概率为 0.5 的'dropout'，然后连接一层输出为X_train.shape[1]/2、激活函数为tanh的连接层，再对数据进行概率为0.5的'dropout'，最后连接输出层，激活函数为Sigmoid，输出2个类别各自可能的概率，本次的优化器采用为'rmsprop'，损失函数为'binary_crossentropy'，评判使用'accuracy'。

```
# 建立CNN模型
model = Sequential()
model.add(Dense(X_train.shape[1], input_shape=X_train.shape[1:]))
model.add(Dropout(0.5))
model.add(Dense(int(X_train.shape[1]/2), activation='tanh'))
model.add(Dropout(0.5))
model.add(Dense(2, activation='sigmoid'))
# 模型概括
model.summary()
model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

训练模型

以下是训练模型，对模型进行10次训练，将最优模型保存在'data/model.weights.best.hdf5'中。

```
#训练模型
epochs = 10

checkpointer = ModelCheckpoint(filepath='data/model.weights.best.hdf5',
                               verbose=1, save_best_only=True)

model_history = model.fit(X_train, y_train, validation_data=(X_valid, y_valid), epochs=epochs,
                           batch_size=128, callbacks=[checkpointer], verbose=1)
```

在以上的训练中，分别使用每组特征向量和四个特征向量组合的特征向量和三个特征向量组合的特征向量训练模型，比较损失函数值，比较最优的模型，如下：

InceptionV3特征向量情况

```
Epoch 4/10
19824/19824 [=====] - 3s 131us/step - loss: 0.0191 - acc: 0.9940 - val_loss: 0.0194 - val_acc: 0.9945

Epoch 00004: val_loss improved from 0.01950 to 0.01941, saving model to data/model.weights.best.hdf5
```

InceptionResNetV2特征向量情况

```
Epoch 9/10
19824/19824 [=====] - 2s 91us/step - loss: 0.0100 - acc: 0.9976 - val_loss: 0.0200 - val_acc: 0.9952

Epoch 00009: val_loss improved from 0.02043 to 0.02002, saving model to data/model.weights.best.hdf5
```

ResNet50特征向量情况

```
Epoch 3/10
19824/19824 [=====] - 2s 101us/step - loss: 0.0391 - acc: 0.9871 - val_loss: 0.0320 - val_acc: 0.9899

Epoch 00003: val_loss improved from 0.04350 to 0.03203, saving model to data/model.weights.best.hdf5
```

Xception特征向量情况

```
Epoch 5/10
19824/19824 [=====] - 2s 100us/step - loss: 0.0179 - acc: 0.9946 - val_loss: 0.0190 - val_acc: 0.9960

Epoch 00005: val_loss improved from 0.01981 to 0.01904, saving model to data/model.weights.best.hdf5
```

组合4个特征向量情况

```
Train on 19824 samples, validate on 4956 samples
Epoch 1/10
19824/19824 [=====] - 6s 312us/step - loss: 7.4239 - acc: 0.5150 - val_loss: 7.9
634 - val_acc: 0.5032

Epoch 00001: val_loss improved from inf to 7.96337, saving model to data/model.weights.best.hdf5
Epoch 2/10
19824/19824 [=====] - 5s 260us/step - loss: 8.0289 - acc: 0.4991 - val_loss: 7.9
634 - val_acc: 0.5032

Epoch 00002: val_loss did not improve from 7.96337
Epoch 3/10
19824/19824 [=====] - 5s 258us/step - loss: 8.0289 - acc: 0.4991 - val_loss: 7.9
634 - val_acc: 0.5032

Epoch 00003: val_loss improved from 7.96337 to 7.96337, saving model to data/model.weights.best.hdf5
Epoch 4/10
19824/19824 [=====] - 5s 265us/step - loss: 8.0289 - acc: 0.4991 - val_loss: 7.9
634 - val_acc: 0.5032

Epoch 00004: val_loss did not improve from 7.96337
Epoch 5/10
19824/19824 [=====] - 5s 259us/step - loss: 8.0289 - acc: 0.4991 - val_loss: 7.9
634 - val_acc: 0.5032

Epoch 00005: val_loss did not improve from 7.96337
Epoch 6/10
19824/19824 [=====] - 5s 258us/step - loss: 8.0289 - acc: 0.4991 - val_loss: 7.9
634 - val_acc: 0.5032

Epoch 00006: val_loss did not improve from 7.96337
Epoch 7/10
```

组合3个特征向量情况

```
Train on 19824 samples, validate on 4956 samples
Epoch 1/10
19824/19824 [=====] - 16s 808us/step - loss: 0.0128 - acc: 0.9957 - val_loss: 0.0169 - val_acc: 0.9967

Epoch 00001: val_loss improved from inf to 0.01690, saving model to data/model.weights.best.hdf5
```

在训练的过程中发现，四个特征向量组合获得的特征向量在训练过程中的loss一直保持在8.0289，而且没有变好的情况，可能是由于4个特征向量之间存在冲突或是复杂度过高，导致特征的提取反而劣化。在剩余的几次训练的对比中，发现三个特征向量组合获得的特征向量在训练中的loss为最优，达到0.0164。所以最终选用0.0169最低的模型（三个特征向量cat_dog_Xception、cat_dog_InceptionResNetV2、cat_dog_InceptionV3通过np.concatenate函数进行连接生成的特征向量），在确定了模型后，对定制层进行增加和修改，选择最优的层数。在确定了训练模型的层次后，对'adam'、'adadelta'、'rmsprop'几个优化器进行了比较，之间的差距也并不大，最终确定使用'rmsprop'优化器。

预测测试集

预测这里我们用到了一个小技巧，我们将每个预测值限制到了[0.005, 0.995]个区间内，这个原因很简单，kaggle官方的评估标准是LogLoss，对于预测正确的样本，0.995和1相差无几，但是对于预测错误的样本，0和0.005的差距非常大。

此处采用了[手把手教你如何在Kaggle猫狗大战冲到Top2% \(https://zhuanlan.zhihu.com/p/25978105?utm_medium=social&utm_source=weibo\)](https://zhuanlan.zhihu.com/p/25978105?utm_medium=social&utm_source=weibo)中的建议。

```

# 加载最优模型
model.load_weights('data/model.weights.best.hdf5')
# 预测test中的结果
test_targets = model.predict(X_test, verbose=1)
test_targets = test_targets.clip(min=0.005, max=0.995)
# 创建label
# label的值越接近1表示dog，越接近0表示dog
label = []
for test_target in test_targets:
    if np.argmax(test_target) == 0:
        label.append(1 - test_target[0])
    elif np.argmax(test_target) == 1:
        label.append(test_target[1])
    else:
        print('err')
# 创建ID
num = []
for test_name in test_files_name:
    num.append(re.sub("\D", "", test_name[10:]))
# 字典中的key值即为csv中列名
sample_submission = pd.DataFrame({'id': num, 'label': label})
# 将'id'转为int
sample_submission['id'] = sample_submission['id'].astype(int)
# 排序
sample_submission = sample_submission.sort_values(by='id')
# 将DataFrame存储为csv, index表示是否显示行名, default=True
# 将结果存入'data/submission.csv'文件中
sample_submission.to_csv('data/submission.csv', index=False, sep=',')

```

完善

本次从初始开始，主要完善并修改了以下几部分：

- 特征向量提取模型选择：如上所述，分6次载入不同特征向量，最终根据各自的验证集loss值，选择了有三个特征向量组成的特征向量；
- 对数据的定制层进行调整：主要初始是使用了一个0.3的'dropout'和model.add(Dense(2, activation='sigmoid'))，但是发下数据的验证机准确率总是在90%左右，并比训练集准确率低，所以考虑是过拟合的原因，将'dropout'调整为0.5后有一定改善。但是预测准确率没有到达要求，在0.5的'dropout'前增加了0.5的'dropout'层和Dense(int(X_train.shape[1]/2), activation='tanh')层，对数据预测结果有一定的优化作用；
- 对模型的优化器进行调整，选用了'nadam'/'adadelta'/'rmsprop'/'SGD'/'adam'等，最终发现'rmsprop'效果最好；
- 对模型的训练测试进行了调整，总结发现在10次性能成本最优；
- 对模型的batch_size进行了调整，分别测试了1024、512、256、128、64、32，发现不同情况下的收敛不同，如果模型发生loss不降低的情况是，需要对其进行调整，最终优选128；
- 下一步应添加图像数据增强，要对原始图片进行一些随机操作，比如旋转、剪切变换、缩放、水平翻转等，降低模型的过拟合。
- 应对train数据集异常数据的剔除根据准确，可能可以进一步的提升准确率。

IV. 结果

模型的评价与验证

```
Train on 19824 samples, validate on 4956 samples
Epoch 1/10
19824/19824 [=====] - 16s 808us/step - loss: 0.0128 - acc: 0.9957 - val_loss: 0.0169 - val_acc: 0.9967

Epoch 00001: val_loss improved from inf to 0.01690, saving model to data/model.weights.best.hdf5
Epoch 2/10
19824/19824 [=====] - 16s 808us/step - loss: 0.0133 - acc: 0.9962 - val_loss: 0.0164 - val_acc: 0.9962

Epoch 00002: val_loss improved from 0.01690 to 0.01636, saving model to data/model.weights.best.hdf5
Epoch 3/10
19824/19824 [=====] - 16s 809us/step - loss: 0.0130 - acc: 0.9960 - val_loss: 0.0200 - val_acc: 0.9947

Epoch 00003: val_loss did not improve from 0.01636
Epoch 4/10
19824/19824 [=====] - 16s 807us/step - loss: 0.0123 - acc: 0.9965 - val_loss: 0.0170 - val_acc: 0.9963

Epoch 00004: val_loss did not improve from 0.01636
Epoch 5/10
19824/19824 [=====] - 16s 806us/step - loss: 0.0122 - acc: 0.9961 - val_loss: 0.0183 - val_acc: 0.9962

Epoch 00005: val_loss did not improve from 0.01636
Epoch 6/10
19824/19824 [=====] - 16s 806us/step - loss: 0.0129 - acc: 0.9961 - val_loss: 0.0196 - val_acc: 0.9950

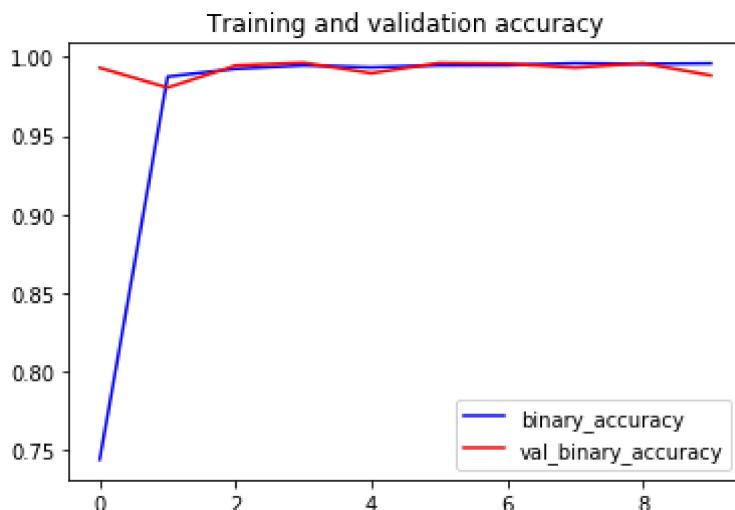
Epoch 00006: val_loss did not improve from 0.01636
Epoch 7/10
19824/19824 [=====] - 16s 806us/step - loss: 0.0134 - acc: 0.9958 - val_loss: 0.0175 - val_acc: 0.9963

Epoch 00007: val_loss did not improve from 0.01636
Epoch 8/10
19824/19824 [=====] - 16s 805us/step - loss: 0.0130 - acc: 0.9959 - val_loss: 0.0212 - val_acc: 0.9948

Epoch 00008: val_loss did not improve from 0.01636
Epoch 9/10
19824/19824 [=====] - 16s 806us/step - loss: 0.0116 - acc: 0.9963 - val_loss: 0.0186 - val_acc: 0.9956

Epoch 00009: val_loss did not improve from 0.01636
Epoch 10/10
19824/19824 [=====] - 16s 807us/step - loss: 0.0131 - acc: 0.9962 - val_loss: 0.0170 - val_acc: 0.9965
```

在验证集上达到了99.62%的准确率loss为0.0164，训练过程中的 loss 和 accuracy 如下：





将测试集的处理结果提交到kaggle上，loss为0.03891。

26 submissions for Zhang Bin		Sort by	Most recent
All	Successful	Selected	
Submission and Description		Public Score	Use for Final Score
sample_submission_3.csv 21 minutes ago by Zhang Bin		0.03891	<input type="checkbox"/>
Message			

通过以上可视化的分析，训练集和验证机的损失函数曲线变化的收敛程度是相同的，而且是稳定的，最终都到达了损失函数特别低的程度，并基本重合，所以表明模型对于这个问题是足够稳健可靠，训练数据或输入的一些微小的改变不会极大影响结果。测试集的预测结果上传至kaggle上评分，loss为0.03891，说明这个模型得出的结果是可信的。

合理性分析

单个ResNet50模型10次迭代训练结果为：

训练集loss: 0.0391，验证集loss: 0.0320。

单个Xception模型10次迭代训练结果为：

训练集loss: 0.0179，验证集loss: 0.0190。

单个InceptionV3模型10次迭代训练结果为：

训练集loss: 0.0191，验证集loss: 0.0194。

单个InceptionResNetV2模型10次迭代训练结果为：

训练集loss: 0.0100，验证集loss: 0.0200。

三个模型10次迭代训练结果为：

训练集loss: 0.0128，验证集loss: 0.0169。

四个模型10次迭代训练结果为：

训练集loss: 8.02，验证集loss: 7.96。

三种模型通过组合，优于其他的情况。

更多的层数，不同的卷积核各种各样的的组合，可以更好的抽取图片中的泛化特征，这样既可以提高分类的准确率，又可以降低模型的过拟合，所以现在神经网络层数都非常的多，深度很深，但是从此次的训练中发现，过于复杂的模型也会带来‘梯度消失’等的问题，导致模型损失函数的提高，所以选择更深层次更复杂的模型的同时也要考虑带来的负面影响，要有相应的办法去处理、减少负面影响，在深层次复杂模型和‘梯度消失’等负面影响之间选择平衡，选择最优的模型。

InceptionV3、XceptionInceptionV3和InceptionResNetV2这三个模型进行组合迁移学习，效果比单个神经网络模型效果和4个神经网络模型更好，也在这次的实验中得到了体现，也充分说明了模型层次深、复杂度高和‘梯度消失’等负面影响之间的平衡问题。在尽量降低‘梯度消失’等负面影响的前提下，提高模型的复杂度，通过多个模型处理数据并进行组合，可以有效降低模型的方差，减少过拟合程度，提高分类准确率。

通过上传预测结果数据到kaggle上进行打分，结果已完全说明此模型的预测结果完全满足前面设定的基准模型的要求，确确实实解决了此次的问题。

V. 项目结论

结果可视化



Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 6144)	37754880
dropout_1 (Dropout)	(None, 6144)	0
dense_2 (Dense)	(None, 3072)	18877440
dropout_2 (Dropout)	(None, 3072)	0
dense_3 (Dense)	(None, 2)	6146

Total params: 56,638,466
Trainable params: 56,638,466
Non-trainable params: 0

图片上方的概率表示的是图片中是狗的概率，越接近1表示为狗，越接近0预测为猫。并且从kaggle上的得分结果，充分说明了此次训练出的CNN模型对于图片预测准确度是相当高的。从模型概况图也充分说明了合理性分析中的相关理论，在取得模型深层次复杂度和‘梯度消失’等负面影响平衡后的CNN模型，完全能达到项目的要求。

对项目的思考

深度学习卷积神经网络毫无疑问是处理图像问题最佳的机器模型，近年来各大赛的前几名均是通过深度学习获取好成绩。本项目首先对训练数据中的异常数据进行了清理，统一输入图片的大小和生成了随机的训练集和验证集，之后利用现有的优秀模型对数据进行了特征向量的提取，本次选用了Xception、InceptionV3、ResNet50和InceptionResNetV2四个模型，之后对三组特征向量和4组特征向量进行了组合，并对单独使用一组特征向量和组合后的特征向量的数据输入定制模型中进行训练，最终比较发现选择更复杂的模型的同时也要考虑带来的负面影响，要有相应的办法去处理、减少负面影响，在复杂的模型带来优势和劣势中找到平衡，选择最优的模型。最终发现此次项目选用Xception、InceptionV3 和InceptionResNetV2三组特征项目组成的模型，在最终预测中得分最高。

在此次的项目试验中，我也陷入了困境，先入为主的认为4个特征向量的组合会带来更好的预测结果，所以无数次的对4个特征向量组合的进行训练，调整了定制层的深度、激活函数等，调整了优化器、评判标准等，并不断的修正代码，但是最终也无法降低模型的损失函数，在这种情况下困惑了10几天，最终突然认识到可能是CNN网络过多的组合和层次带来了负面效应。最终通过实验的方式得到了最优结果（三个模型的组合）。通过此次的项目和困惑，也充分认识到深度学习这门计算机技术，应更注重于实践的结果，实践结果是验证模型最好的方式，而不像传统计算机编程技术，注重的是代码的逻辑性。在深度学习中，实验占第一位，代码逻辑在第二位。本次项目只是使用Xception、InceptionV3、InceptionResNetV2 和 ResNet50 四个模型进行实验，并且只是简单的拼接，忽略了特征之间的位置关系，除了这四个模型，还可以选择更多新的模型，或者使用stacking的方法进行模型融合，进一步降低方差，寻找网络复杂度与模型负面影响最低化的方案，提高分类的准确率。对于数据清理部分，此次项目我只是粗鲁的分析和删除了异常数据，其实异常数据中还有很多正常数据，所以下一步应更精确的删除异常图片，从而提高训练集的数据量，并加入图片增强的技术，进一步的提高模型的预测准确度。

最终模型和结果完全符合我对这个问题的期望，我认为它完全可以在通用的场景下解决这些类型。

需要作出的改进

本次的项目我使用的是Keras框架，其实Keras框架的基础是TensorFlow，为了提高运算的速度，下一次我应该学习TensorFlow语法，直接使用google公司提供的更加先进的方法，利用TensorFlow来训练模型。

由于本次项目Keras框架对NASNetLarge模型的支持存在部分问题，无法利用这个被现今誉为最好的CNN模型，在下一步中，我可能利用此模型进一步提高我的模型的准确度。

在此次的执行流程中，没有加入k-折叠交叉验证和图像增强的技术，在下一步的完善中，通过加入这两种技术也将进一步提高我的模型的准确度。

IV. 参考文献

- [1]手把手教你如何在Kaggle猫狗大战冲到Top2% (<https://zhuanlan.zhihu.com/p/25978105>)
- [2]python用matplotlib绘制散点图、直方图、箱形图 (<https://jingyan.baidu.com/article/0aa223757124e088cd0d6444.html>)
- [3]Keras 中文文档 (<https://keras.io/zh/>)
- [4]InceptionResNetV2 (<https://arxiv.org/abs/1602.07261>)
- [5]在AWS上配置深度学习主机 (<https://zhuanlan.zhihu.com/p/25066187>)
- [6]ImageNet: VGGNet, ResNet, Inception, and Xception with Keras (<https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/>)
- [7]Inception-Resnet-V2的网络模型 (<https://www.cnblogs.com/yuehouse/p/9742647.html>)
- [8]面向小数据集构建图像分类模型 (https://keras-cn.readthedocs.io/en/latest/legacy/blog/image_classification_using_very_little_data/#bottleneck90)
- [9]机器学习基础与实践（一）----数据清洗 (<http://www.cnblogs.com/charlotte77/p/5606926.html>)
- [10]python使用matplotlib:subplot绘制多个子图 (<https://blog.csdn.net/gatieme/article/details/61416645>)

- [11]Matplotlib绘图常见设置说明 (<https://www.cnblogs.com/nju2014/p/5707980.html>)
- [12]利用AWS学习深度学习 For Udacity P5 (<https://zhuanlan.zhihu.com/p/33176260>)
- [13]Documentation for individual models (<https://keras.io/applications/#documentation-for-individual-models>)
- [14]Dogs vs. Cats: Image Classification with Deep Learning using TensorFlow in Python
(<https://www.datasciencecentral.com/profiles/blogs/dogs-vs-cats-image-classification-with-deep-learning-using>)