**机器学习纳米学位毕业论文**

**猫狗大战**

**刘珅珅**

**2019年4月14号**

1 问题的定义

* 1. 项目概述

计算机视觉是让计算机能够感知和理解图像的意义，在现实生活中有广泛的应用。图像分类是计算机视觉的一个重要的研究方向，通过对不同类别图像的进行特征提取并构建模型来预测未知图像的类别，在很多实际问题上收到良好的效果，例如udacity课程中讲解的皮肤癌检测的例子，通过对各种类别的皮肤病图片的分类识别来判断患者是否患有皮肤癌，其准确率甚至超过了一些经验丰富的医生。kaggle推出dogs-vs-cats比赛，就是一个典型的图像分类问题，通过机器学习的方式来识别出图片是猫还是狗。选择这个项目作为毕业项目的原因是本人对计算机图像处理感兴趣，并且项目本身会用到深度学习和卷积神经网络的知识，这也是目前人工智能的热点。

* 1. 问题陈述

该项目是一个监督学习的二分类问题，训练集通过图片名称已经打好了标签，区分是猫还是狗，通过机器学习构建模型并训练，然后预测测试集中的图片是猫还是狗。

* 1. 评价指标

在kaggle竞赛中，该项目采用的评价指标为LogLoss，即对数损失。本项目采用与之相同的评价指标。

对于一个二分类问题，LogLoss的计算公式如下所示：



其中为测试集样本数，代表图片的真实标签，如果是狗，否则。表示模型预测图片为狗的概率。

从LogLoss的计算公式可以看出，模型不仅需要预测正确，而且预测的概率要尽可能的大，即对预测结果非常肯定，否则LogLoss的值就会较大。

2 分析

2.1 数据可视化

本项目的数据集为kaggle提供的数据集，训练集为25000张图片，其中12500张为猫类图片，12500张为狗类图片。测试集为12500张猫狗图片，通过名称来区分类别。

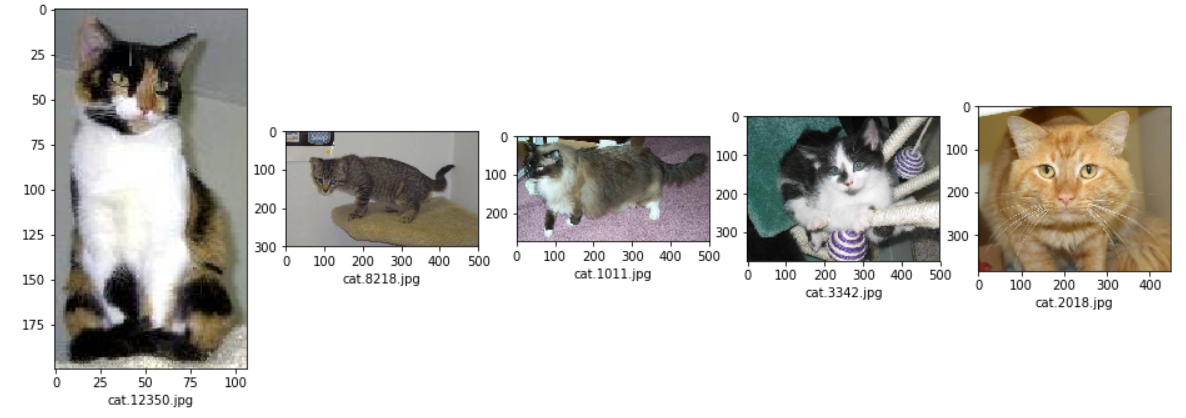


图2-1 训练集中猫类示例图

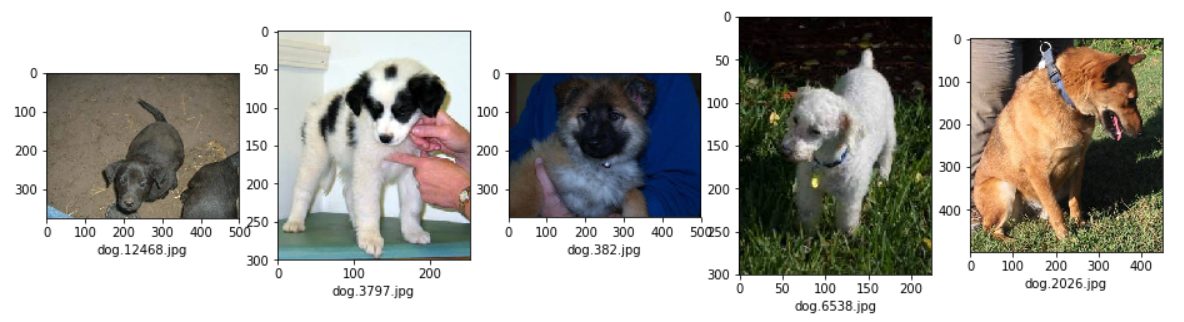


图2-2 训练集中狗类示例图

测试集总共有12500张图片，没有分类。

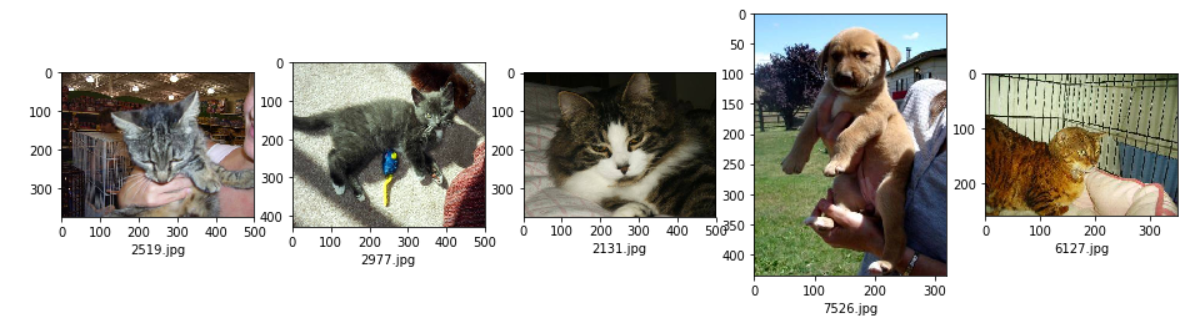


图2-3 测试集示例图

从上图可以看出，训练集中的大部分图片中猫和狗的特征都非常明显，清晰度和分辨率都良好，但有部分图片中存在标识错误的情况，图片与标识不匹配，另外还有些图片中猫狗的特征非常小，在后续第3章会在预处理中解决这些问题。

图2-4为训练集中图片的尺寸分布图，表2-1为图片尺寸的统计信息表。从中可以看出图片的宽高基本都小于500，绝大部分的图片的宽高超过了300，而且大部分宽高比都在1附近波动。

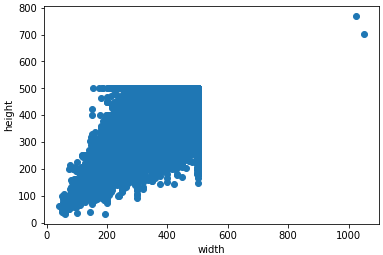


图2-4 训练集图片尺寸分布图

表2-1 训练集图片尺寸统计信息

|  |  |  |
| --- | --- | --- |
|  | width | height |
| mean | 404 | 360 |
| std | 109 | 97 |
| min | 42 | 32 |
| 25% | 323 | 301 |
| 50% | 447 | 374 |
| 75% | 499 | 421 |
| max | 1050 | 768 |

2.2 算法和技术

过去几年中，卷积神经网络(Convolutional Neural Network, CNN)在图像分类中有很好的效果，考虑到数据集较大并和ImageNet的图片集有很大的相似性以及Udacity课程中深度神经网络有关迁移学习的介绍，可以采用迁移学习[1]的方式来解决问题。将问题分为两部分：特征提取和模型分类。特征提取是采用预训练模型提取实际图片的边缘特征、轮廓特征以及更高级的能区分猫狗的特征等。模型分类是利用之前提取的特征，输入到一个模型中进行分类训练和预测，并根据训练过程中的参数如准确率、损失大小等来判断是否满足要求来调整模型参数，最终得到符合要求的分类结果。

CNN通常包含以下2层：

1. 卷积层（Convolutional Layer）
2. 池化层（Pooling Layer）

2.2.1卷积层(特征提取层)

传统神经网络由全连接层，即各个层之间的每个结点都相互连接，这样就会导致权重系数非常多，如果网络层较多，训练变得非常麻烦。

卷积的概念：



其中称为卷积窗，为输入矩阵。

CNN中的卷积层的每个结点只与部分输入结点相连，通过一个卷积窗对输入矩阵进行卷积得到输出，称为局部感知。假设输入为，卷积窗为，则经过卷积后得到的结点数为，并且由于是用相同的卷积窗进行的操作，所以可以认为这些结点的权重系数和偏移值都是相同的，提取的是图片的同一特征，称为权值共享，如果希望提取图片不同的特征，就需要多个卷积窗。相比之前的全连接层，卷积层的参数就会减少很多。

2.2.2 池化层

池化层的主要作用就是减少卷积层输入的参数，一般位于一个或多个卷积层后面。池化层也使用一个窗口，但不进行卷积操作，而是通过扫描之前卷积层输入并对窗口中的参数取最大值、平均值等操作得到一个输出参数。通过池化层后，参数会进一步减少。

2.2.3 技术实现

本项目采用基于Tensorflow的开源机器学习库Keras[2]来实现模型的构建、训练和预测。数据集本身包含较多的图片，从头开始构建一个卷积神经网络并进行参数优化非常复杂耗时，在ImageNet中，包含通过猫狗图片进行训练并将参数优化的预训练CNN模型，所以可以采用迁移学习(Tranfer Learning)的方式来进行模型构建。

迁移学习就是将已经预训练CNN模型的参数迁移到新的模型上已提高模型的训练速度和效果。在本项目中使用的训练集与之前ImageNet训练模型的数据集有很高的相似度，所以可以用预训练模型的卷积层参数来进行特征提取，才添加自己的全连接输出层进行分类识别。

在Keras中有5种著名的CNN模型，分别为VGG16[3]，VGG19，ResNet50[4]，InceptionV3[5]，Xception[6]。本项目将分别使用这5种模型进行特征提取，然后对比它们的训练效果，选取最优的模型。在Keras中，上述5种模型都使用一个布尔型参数include\_top来表明是否需要最后的全连接层，采用迁移学习提取特征时，一般令include\_top=False，这样只使用模型的卷积部分。

1. VGG16, VGG19

VGG16和VGG19由牛津大学提出，其中，VGG16有13个卷积层和3个全连接层，VGG19有16个卷积层和3个全连接层。查看Keras中VGG16和VGG19的源代码，可以看出这两种模型都是比较传统的卷积神经网络。VGG16的结构如图2-5所示，VGG19的结构如图2-6所示。从图中可以看出，二者的结构非常相似，所有卷积层都分为5块，每块之间插入一个最大池化层。5块卷积层的卷积窗都是窗口，采用小窗口可以减少计算量，并且能提取图片局部细微的特征。两个模型的第1,2两块分别有两个卷积层，VGG16的第3,4,5块有3个卷积层，VGG19的第3,4,5块有4个卷积层，二者前1,2,3,4块的卷积层结点都是递增的，浅层的卷积层提取图片的边缘、轮廓等信息，深层的卷积层提取图片高级特征，例如猫狗图片的眼睛、耳朵等特征。

图2-5 VGG16结构图 图2-6 VGG19结构图

1. InceptionV3

InceptionV3由Google研究院提出。与之前的VGG模型每个卷积层只有一种类型的卷积窗不同，InceptionV3使用Inception模块，这个模块会对同一个输入采用不同的卷积窗执行多次卷积，并将所有输出结果拼接为输出。相比较于VGG模型，InceptionV3的参数更少，准确率更高。

查看Keras中InceptionV3的源代码，InceptionV3的卷积层操作为Conv+BN，即卷积+批归一化，执行完卷积后，批归一化将输出的均值接近0，标准差接近1.



图2-7 InceptionV3的卷积模块Conv\_BN

InceptionV3总共有11个Inception模块，分为5种不同类型。类型1如图2-8所示，总共有3个Inception模块（部分模块的卷积窗个数略有不同）。类型2如图2-9所示，有1个Inception模块。类型3如图2-10所示，有4个Inception模块（部分模块的卷积窗个数略有不同）。类型4如图2-11所示，有1个Inception模块。类型5如图2-12所示，有2个Inception模块。



图2-8 Inception模块类型1



图2-9 Inception模块类型2



图2-10 Inception模块类型3



图2-11 Inception模型类型4



图2-12 Inception模块类型5

InceptionV3模型的结构如图2-13所示。



图2-13 InceptionV3结构图

1. ResNet50

ResNet50是由微软提出，其核心是所谓的残差模块。对于某一块神经网络，假设输入为X，期望输出为H(X)，ResNet50的作者假设神经网络可以逼近复杂函数H(X)，则认为其也可以逼近H(X)-X（即残差F(X)），而且认为网络逼近H(X)-X要更容易[4]，这样网络块的输出就变成了F(X)+X。查看Keras中的源代码，残差模块主要有两类，分别为Conv\_Block(kernel\_size, [filters1,filters2,filters3])和Identity\_Block(kernel\_size, [filters1,filters2,filters3])，如图2-14，图2-15所示，二者的区别是Identity\_Block的输入直接与残差相加，Conv\_Block的输入进过一个卷积+批归一化后与残差相加。

图2-14 Conv\_Block 图2-15 Identity\_Block

ResNet50的结构如图2-16所示。



图2-16 ResNet50结构图

1. Xception

Xception由Google提出，在InceptionV3的基础上进行了改进，采用深度可分离卷积[7]替代了传统的卷积，并且还使用ResNet中的残差结构。Xception的SeparableConv2D是先用卷积窗对输入进行卷积，然后用卷积窗对输入的每个通道进行卷积，这样做是假设[6]输入数据的空间相关性和通道之间的相关性完全无关，可以分开学习，这样Xception相比之前的模型参数要更少。Xception结构如图2-17所示。

图2-17 Xception结构图

2.3 基准指标

毕业项目要求模型的LogLoss必须在kaggle pubic learnboard中排入前10%，即LogLoss<0.06127。

3 方法

3.1 数据预处理

1) kaggle提供的数据有部分是异常数据，标记不正确，因此清理异常数据。参考<https://zhuanlan.zhihu.com/p/34068451>提供的异常值校验方法，本项目采用VGG16，VGG19，ResNet50，InceptionV3和Xception这5种预训练模型分别对训练集的图片进行预测，然后根据其返回的分类结果判断是否在ImageNet中的猫狗种类中，如果不在，则认为图片针对该模型是异常值，取5种模型异常值判断结果的交集再结合kaggle上讨论的异常图片，最终得到需要删除的异常图片。代码实例图如图3-1所示。

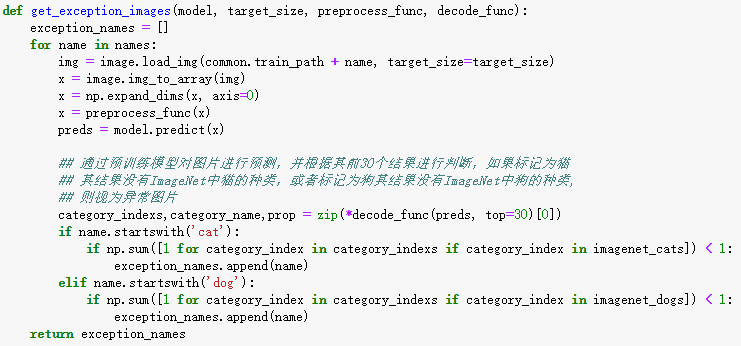


图3-1 预训练模型获取异常图片

最终删除的异常图片为cat.10029.jpg，cat.10365.jpg，cat.10636.jpg，cat.10712.jpg，cat.11184.jpg，cat.11565.jpg，cat.12227.jpg，cat.12272.jpg，cat.12424.jpg，cat.12493.jpg，cat.2337.jpg，cat.252.jpg，cat.2520.jpg，cat.2893.jpg，cat.2939.jpg，cat.3216.jpg，cat.3731.jpg，cat.3868.jpg，cat.4085.jpg，cat.4338.jpg，cat.4688.jpg，cat.4833.jpg，cat.4852.jpg，cat.503.jpg，cat.5351.jpg，cat.5418.jpg，cat.6429.jpg，cat.6442.jpg，cat.6699.jpg，cat.7377.jpg，cat.7564.jpg，cat.7920.jpg，cat.7968.jpg，cat.8383.jpg，cat.8456.jpg，cat.8470.jpg，cat.9090.jpg，cat.9171.jpg，cat.9444.jpg，dog.10161.jpg，dog.10190.jpg，dog.10237.jpg，dog.1043.jpg，dog.10801.jpg，dog.12376.jpg，dog.1773.jpg，dog.1895.jpg，dog.2422.jpg，dog.4367.jpg，dog.5604.jpg，dog.6475.jpg，dog.8736.jpg，dog.8898.jpg，dog.9517.jpg，共54张，图3-2为随机抽取的异常图，可以看出图片要么根本没有猫狗，要么就是猫狗在图片中的区域太小，其它特征占的空间太大，这些都会对训练效果产生影响。

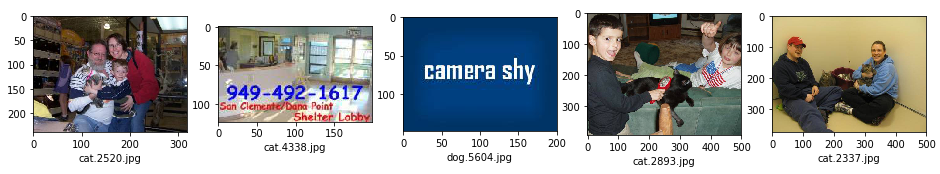


图3-2 异常图片

2) kaggle中提供的训练集图片名称以cat和dog开头，统一放在train文件夹下，所以先将其分类放在cat和dog文件夹。另外需要分出一部分数据作为验证集用于交叉验证，本项目将训练集数据的20%作为验证集，存放在valid文件夹的cat和dog文件夹下。之前讨论过要删除异常数据，删除后训练集猫类图片有12461张，狗类图片有12485张，总共有24946张图片，最终train文件夹中的cat和dog文件夹总共有图片19957张，valid文件夹下的cat和dog文件夹总共有图片4989张，测试集test有图片12500张。

3) 本项目采用预训练模型VGG16，VGG19，ResNet50，InceptionV3，Xception进行特征提取，这几种模型对图片的预处理要求有所不同。根据keras.applications的源代码可以看出，VGG16，VGG19，ResNet50的预处理方式是相同的，需要图片每个通道都减去ImageNet的mean值[103.939, 116.779, 123.68]，并将图片从RGB转换为BGR；InceptionV3，Xception的预处理方式是相同的，将图片数据缩放到[-1, 1]之间。

4) 本项目使用keras中的ImageDataGenerator类通过实时数据增强批量将图片转换为张量图像数据，其中VGG16，VGG19，ResNet50默认的图像尺寸为(224, 224)，InceptionV3，Xception默认的图像尺寸为(299, 299)。

3.2 特征提取

分别创建上述5种预训练模型用于特征提取，创建时需要将模型的include\_top=False，这样就不包含预训练模型最后的全连接层，只通过卷积层来提取特征。在使用小批量数据集（训练集1600张，验证集400张）进行测试时，发现输出的特征数据文件非常大，超过了1G，因此统一将模型的pooling=’avg’，即在卷积层的最后添加一个GlobalAveragePooling2D层，用于减少特征量，防止过拟合。

3.3 模型分类构建与训练

特征提取完成后，使用特征数据作为输入数据，仿照预训练模型构建3个全连接层的模型进行训练，模型信息如下图所示。

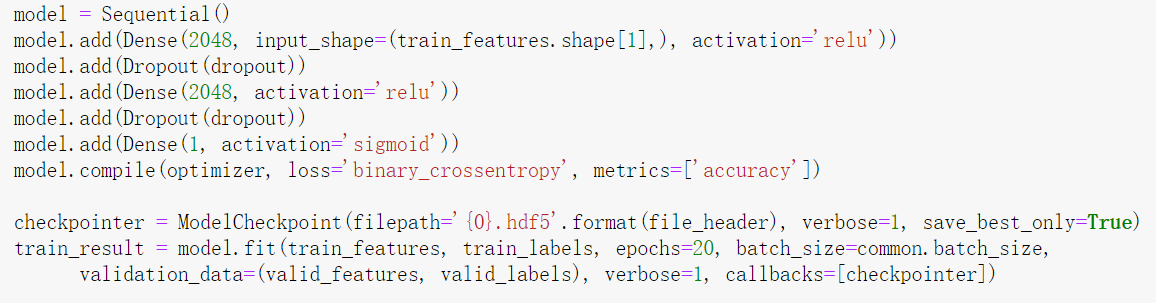


图3-3分类模型源代码

针对5种特征提取模型，分类模型架构几乎完全一致，各个模型的输入参数input\_shape不同，VGG16和VGG19的input\_shape为512，ResNet50，InceptionV3和Xception的input\_shape为2048.

卷积层输出的特征向量维度为2048，分类模型添加两层全连接层结点数也设置为2048，激活函数使用ReLU函数，并添加Dropout层用于防止过拟合。由于是二分类问题，输出层结点为1，激活函数使用Sigmoid函数，损失函数为二分类的交叉熵，训练期间评估指标设置为准确率。

3.4 遇到的问题及模型参数优化

1) 预处理函数的修改。之前根据Udacity课程讲解及参考网上资料，自己写预处理函数，在ImageDataGenerator函数中添加了图片的平移、旋转和随机缩放等步骤，在小批量的数据集（训练集1600张，测试集400张）发现效果并不理想，模型的训练损失和验证损失都比较高，模型不收敛，后来之间使用VGG16等5种模型自带的预处理函数，发现效果比较理想，推测是由于数据集与ImageNet有较大的相似性，使用自带的预处理函数，效果反而更好。

2) dropout参数和优化器的选择。在图像分类中，常用的优化器有RMSprop，Adam和SGD等。最初优化器选择的是RMSprop，在小批量数据集训练时，模型很难收敛，增加训练次数训练损失和验证损失几乎没有改进，因此不采用这种优化器。之后尝试了SGD和Adam作为优化器，并选择不同的dropout参数，进行对比进行参数优化。

VGG16：

表3-1 VGG16 LogLoss对比表

|  |  |  |
| --- | --- | --- |
| optimizer  dropout | Adam | SGD |
| 0.3 | 0.14984 | 0.06094 |
| 0.5 | 0.13477 | 0.06269 |
| 0.7 | 0.13246 | 0.06202 |

从表格3-1对比图中，VGG16作为预训练模型，选择dropout=0.3，optimizer=SGD(lr=0.001)

VGG19：

表3-2 VGG19 LogLoss对比表

|  |  |  |
| --- | --- | --- |
| optimizer  dropout | Adam | SGD |
| 0.3 | 0.13807 | 0.05623 |
| 0.5 | 0.11114 | 0.05524 |
| 0.7 | 0.11886 | 0.05842 |

从表格3-2对比图中，VGG19作为预训练模型，选择dropout=0.5，optimizer=SGD(lr=0.001).

ResNet50：

表3-3 ResNet50 LogLoss对比表

|  |  |  |
| --- | --- | --- |
| optimizer  dropout | Adam | SGD |
| 0.3 | 0.09590 | 0.05364 |
| 0.5 | 0.09680 | 0.05463 |
| 0.7 | 0.09386 | 0.05516 |

从表格3-3对比图中，ResNet50作为预训练模型，选择dropout=0.3，optimizer=SGD(lr=0.001).

InceptionV3：

表3-4 InceptionV3 LogLoss对比表

|  |  |  |
| --- | --- | --- |
| optimizer  dropout | Adam | SGD |
| 0.3 | 0.04353 | 0.04326 |
| 0.5 | 0.05137 | 0.04347 |
| 0.7 | 0.05453 | 0.04342 |

从表格3-4对比图中，InceptionV3作为预训练模型，选择dropout=0.3，optimizer=SGD(lr=0.001).

Xception：

表3-5 Xception LogLoss对比表

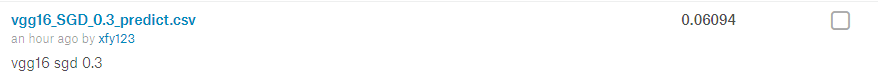
|  |  |  |
| --- | --- | --- |
| optimizer  dropout | Adam | SGD |
| 0.3 | 0.05575 | 0.04373 |
| 0.5 | 0.05653 | 0.04317 |
| 0.7 | 0.05683 | 0.04379 |

从表格3-5对比图中，Xception作为预训练模型，选择dropout=0.5，optimizer=SGD(lr=0.001).

4 结果

4.1 模型评价与验证

使用VGG16，VGG19，ResNet50，InceptionV3和Xception这5种模型进行特征提取，然后在自建分类模型上训练并对测试进行预测，得到的结果如下：



1. VGG16



1. VGG19



1. ResNet50



1. InceptionV3



1. Xception

图4-1 kaggle评分结果图

4.2 合理性分析

从图4-1可以看出，5种模型均满足LogLoss<0.06127，符合项目设置的要求，其中Xception的效果最好。

5 项目结论

5.1 结果可视化

随机抽取部分测试集图片，使用Xception模型进行预测，结果如图5-1所示。

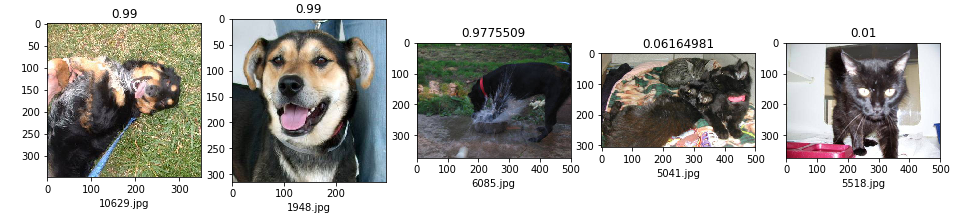


图5-1 Xception模型的预测结果图

可以看出，模型预测结果正确，而且对结果比较肯定。

5.2 对项目的思考

本次项目，从数据预处理到模型选择，再到参数的调整都完整的经历了一遍，我有以下收获：

1) 从头在阿里云上搭建了整个训练环境，中间经历了安装Tensorflow失败，Keras版本不匹配的错误，代码也爆出了各种与环境配置相关的错误，最终都成功解决。

2) 在迁移学习中，对于不同的数据集，要多选择几种模型进行比较操作，这样可以选择出最优的模型，没有那个模型可以适合所有类型的数据集。

3) 对优化器的选择一定要多尝试几种不同的情况，之前根据Udacity中项目的经验选择的RMSprop由于结果不收敛，一度让我怀疑迁移学习是否能成功，在以后的机器学习项目中，针对优化器要多选几种进行比较，也没有那一种优化器可以适合所有的项目。

5.3 改进措施

本项目模型的最终得分满足了项目要求，但仍然有提高的空间。

1. 可以尝试多种预训练模型的融合操作，参考网上的资料，ResNet50，InceptionV3和Xception的融合得到的LogLoss可以减小到0.04.
2. 可以尝试预训练模型的微调来提高准确率。

参考文献

[1] 庄福振，罗平，何清，等迁移学习研究进展[J]. 软件学报，2015,26(1):26-39

[2] Keras Documentation[Z] <https://keras.io/>

[3] Karen Simonyan, Andrew Zisserman Very Deep Convolutional Networks for Large-Scale Image Recognition[EB/OL] <https://arxiv.org/abs/1409.1556>, April 10, 2015

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun Deep Residual Learning for Image Recognition[EB/OL] <https://arxiv.org/abs/1512.03385>, December 10, 2015

[5] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna Rethinking the Inception Architecture for Computer Vision[EB/OL] <https://arxiv.org/abs/1512.00567>, December 2, 2015

[6] François Chollet Xception: Deep Learning with Depthwise Separable Convolutions[EB/OL] <https://arxiv.org/abs/1610.02357>, October 7, 2016

[7] Min Wang, Baoyuan Liu, Hassan Foroosh Design of Efficient Convolutional Layers using Single Intra-channel Convolution, Topological Subdivisioning and Spatial “Bottleneck” Structrue[EB/OL] <https://arxiv.org/abs/1608.04337,August> 15, 2016