**机器学习（进阶）纳米学位毕业项目**

**基于深度学习的图像识别**

****

**董仁广**

2018.6.29

**目录**

一、问题定义 3

1.1 项目概览 3

1.2. 问题说明 3

1.3. 评价指标 3

二、项目分析 4

2.1 数据研究和可视化 4

2.2 算法与方法 6

2.3 基准测试 9

三、实现方法 9

3.1 数据预处理 9

3.2 实施 11

3.3 改进 12

四、结果 14

4.1 模型评估与验证 14

4.2 结果分析 16

五、结论 17

5.1 自由形态的可视化 17

5.2 思考 18

5.3 改进 19

六、引用文献 20

# 一、问题定义

### 1.1 项目概览

“猫狗大战”项目来源于一个Kaggle比赛。早在2013年，Kaggle就举办过一次猫与狗识别比赛：识别图片中的动物是猫还是狗。但是在那之后，机器学习领域——特别是深度学习与图像识别领域发生了一系列的突破性进展：理论上，诸多更加成熟高效的深度神经网络模型被提出，颠覆了传统的机器学习领域，并被大量应用于诸如：计算机视觉、自然语言处理、搜索引擎、医疗诊断等领域；实践上，诸如：Tensor Flow、Keras、Caffe等深度学习框架或平台被大量应用于工业生产环境，使得深度学习问题能够更加简单高效地得到解决。因此，现在重新将这个课题拿出来，应用更加”现代化”、成熟的理论和工具，来解决猫与狗识别的问题。Kaggle官方也提供了训练数据集和测试数据集：其中训练数据集中有25000张图片，猫和狗各12500张；测试数据集中有12500张图片，猫和狗都有。

### 1.2. 问题说明

本项目要解决的问题是在猫和狗的照片中将它们正确地识别出来。这是一个典型的二分类问题，因此我们的目标是实现一个分类器：向分类器输入猫或狗的图片，分类器会输出照片中的动物是狗的概率（值域为[0, 1]）有多大：1表示是狗，0表示是猫。因此便可以得出图片中的是猫还是狗。要实现这样的分类器，可以通过卷积神经网络，从已知的训练数据（知识）中学习猫和狗的特征，以识别出未知数据（测试数据集）中的是猫还是狗。

### 1.3. 评价指标

运用训练数据集训练好卷积神经网络后，用测试数据集来衡量模型的好坏。Kaggle提出了一个用于衡量模型好坏的标准：

LogLoss表示模型误差，值越小说明模型的误差越小，即模型越优。在2.3小节中将更详细地解释该公式。

# 二、项目分析

### 2.1 数据研究和可视化

Kaggle官方提供了训练数据集和测试数据集，其中训练集中有25000张图片，测试数据集中有12500张图片。除此之外，本项目中还是用了一份来自牛津大学的扩展数据集——*The Oxford-IIIT Pet Dataset，*用于扩展训练数据集，该扩展数据中包含了37个品种的宠物（猫和狗），共7349张图。

Kaggle官方提供的训练数据集中猫和狗各12500张。每张图片的文件名称中包含了”dog”、”cat”关键字以及编号，例如：cat.192.jpg、dog.90.jpg。因此可以通过文件名来标记图片中的是狗还是猫，即：标签（label）。训练集作为已知的”知识”，是分类器学习的对象，分类器将从这些标记好的图片中学习属于猫和狗各自的特征。图片中有各种场景下的、不同形态的猫或狗，以猫为例：有不同品种、颜色、花纹、姿势的猫，全身猫和半身猫，跟人的合影猫，两只合影的猫，不同光线条件下的猫。其中有一些异常图片，例如：cat.724.jpg，虽然训练集将它标志为”cat”，但是里面包含了狗和猫，而且狗占的面积比猫要大很多。所以训练之前需要考虑到对这些异常图片的处理。另外，训练数据集中的图片尺寸并非相同，根据不同的模型的需求，需要对图片预先进行裁剪，例如：下文中我们采用了Xception模型，需要将图片统一裁剪为299\*299尺寸的RGB图。

测试数据集中包含12500张图片，是按照数值ID来进行命名的，例如：1.jpg。测试数据集用于衡量我们的分类器的质量。分类器需要预测出测试集中每一张图片是狗的概率有多大，1表示是狗，0表示是猫。

此外，扩展训练数据中共有7349张图片。其中包含了12个品种共2371张猫的图片，和25个品种共4978张狗的图片。同上所述，扩展的训练数据集同样需要进行裁剪等预处理操作。

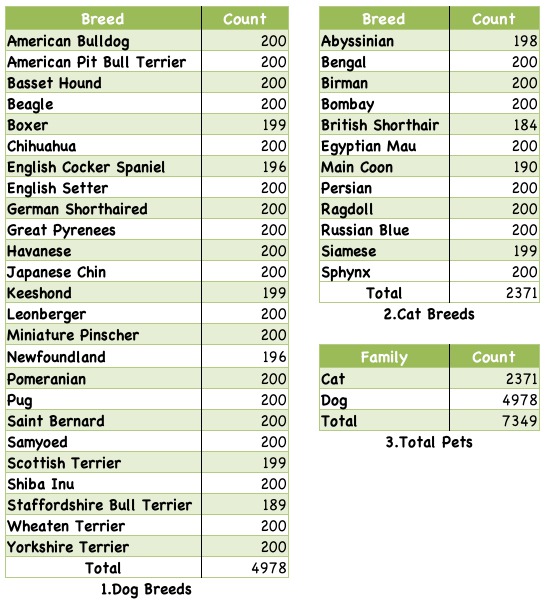


图 2.1 扩展训练数据的分布

综上，训练数据集和测试数据集的分布，以及训练数据集中猫和狗的分布如图2.2所示。训练数据中猫和狗的数量不一致，猫的数量比狗少了15%，这是由于我们的扩展训练数据中狗的图片数量比猫的大，但是由于训练数据基数较大，所以可以忽略这个数量不一致的问题。

图2.2 训练数据和测试数据分布（左），训练数据中猫和狗的分布（右）

### 2.2 算法与方法

机器学习典型的两个应用场景就是分类和回归。根据前面的叙述可知，本项目要解决的是一个分类问题，更具体地说是一个二分类问题。分类问题的解决方法有很多，比如传统的基于统计的机器学习方法有：支持向量机、决策树、逻辑回归和高斯朴素贝叶斯等，以及本文我们要使用的深度学习方法。之所以使用深度学习来进行本项目，而不采用传统的机器学习方法，是因为深度学习被证明可以更好地解决大数据量下图片分类或识别的问题。

#### 2.2.1 神经网络

神经网络是从人类脑神经元的研究中获得灵感，模拟其神经元的功能和网络结构，来完成认知任务的一类机器学习算法。早在1943年，神经网络就由McCulloc 和Pitts提出，后经由Frank Rosenblatt和Werbos等人的改进。但是受限与当时的数据量和计算条件，神经网络并不被看好，一度遭到冷落。

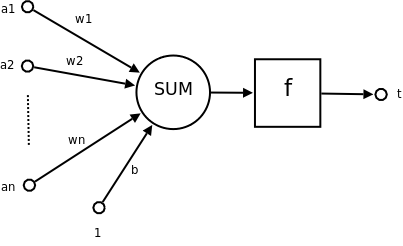
 机器学习中的神经元是以生物的神经元为原型，但是对其进行了抽象、简化。神经网络模型最基本的构成元素有：输入、激活函数和输出。神经网络由若干的神经元互相连接而成，每个神经元从其他神经元处获得输入信息，少部分神经元也从接收器获得信息；神经元处理这些输入信息，经过激活函数，一旦被激活，就会继续发送信号至其他相连接的神经元。

图 2.3 神经元模型

激活函数的作用是将输入数据的加权和进行一个映射，然后将结果传递给下一层网络或者是输出。如果这个映射函数是非线性的，那么整个神经网络就是非线性的。非线性是神经网络的一大重要的特性，这让神经网络具有了解决一些复杂的非线性问题的能力。

#### 2.2.2 深度学习

理论上说，神经网络模型越复杂、参数越多，意味着它能完成更复杂的任务。深度学习一般指深层神经网络，也就是加深神经网络的层数。如图2.4所示，相邻的两层之间有边相连，底层的数据输出作为其相邻的顶层的输入，然后一层一层往顶层传递，直至到达输出层。其中位于输入层和输出层之间的叫做隐藏层。

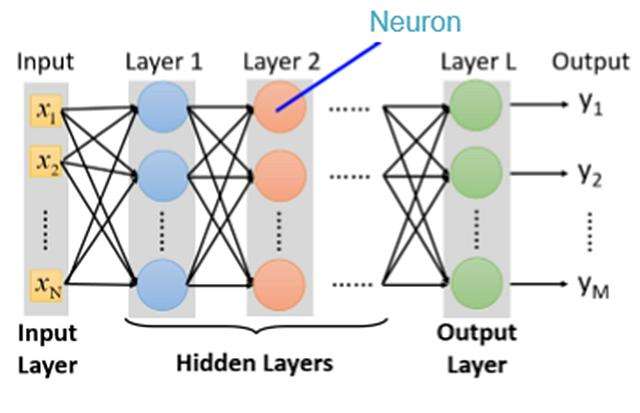


图2.4 一个典型的深度神经网络

除了激活函数外，多层结构是神经网络的另一个重要的特性。Marvin Minsky和Seymour Papert曾提出：感知机（可以简单理解为单层神经网络）无法模拟异或运算。而多层网络却可以，因为多层次的结构可以让网络逐层地学习到一些不同层面或者不同抽象程度的内容，所以可以解决更加复杂的问题。

#### 2.2.3 卷积神经网络

卷积神经网络是一种特殊的善于解决图像识别问题的深度神经网络。在图2.4中可以看到，相邻两层之间的每个结点间都有边相连，这叫做全连结网络。全连接网络最大的问题就是参数过多，导致训练效率低下，而且还可能造成过拟合。而卷积神经网络相邻两层之间只有部分结点相连。相比于全连接层，卷积神经网络的优点是极大地降低了参数的个数，并且可以很好地提高泛化能力，因为同一层共享参数。

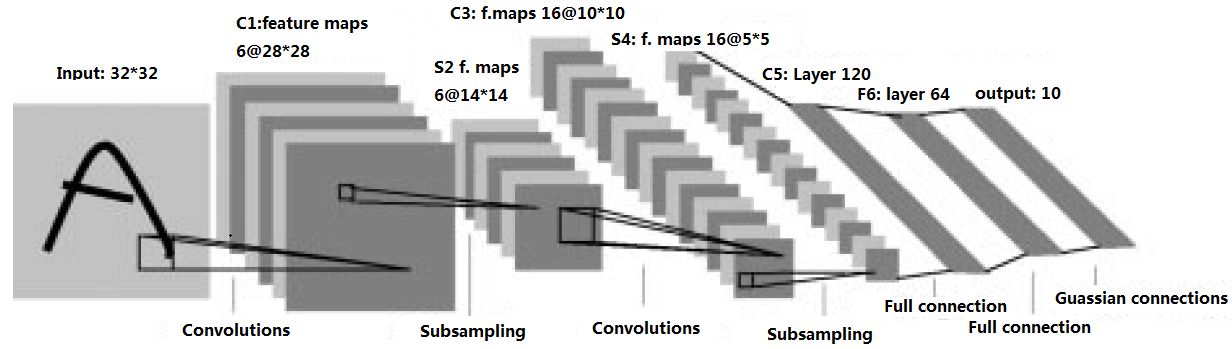


图 2.5 一个典型的卷积神经网络结构

除了结构类似外，卷积神经网络的输入输出以及训练流程与全连结神经网络也基本一致，卷积神经网络的输入层就是图像的原始像素，而输出层中的每一个节点代表了不同类别的可信度。它们唯一的不同就是相邻两层的连接方式。

#### 2.2.4 Xception模型

他山之石，可以攻玉。一般而言，从头开始训练一个复杂的模型是非常耗费精力的，而且由于数据量、枯燥的参数微调等原因，有可能得不到一个理想的结果。所以本项目中进行了迁移学习，也就是将别人训练好的模型直接引入，然后在此基础上进行调整。其中使用了Xception模型。Xception是Google继Inception后提出的对Inception v3的另一种改进，主要是采用depthwise separable convolution来替换原来Inception v3中的卷积操作。



图 2.6 Xception模型的结构图

### 2.3 基准测试

本项目使用了基于Xception模型的迁移学习。Xception模型的权重由ImageNet数据集训练而成。在ImageNet上，该模型能够达到验证集top1 0.790和top5 0.945的正确率。

在1.3 小节中提到， Kaggle官方提供了一套评价标准。对于12500条测试数据，参赛者需要提交一个相应具有12500条结果的csv文件。此csv文件中的每一条记录了测试图片的编号，以及模型预测的该图片是狗的概率——1表示是狗，0表示是猫。然后Kaggle通过下面的公式来计算模型表现的好坏：

LogLoss表示模型误差，值越小说明模型的误差越小，即模型越优。其中，n是测试集中的样本总量；是模型预测的图片是狗的概率；log为自然对数。

目前，该比赛已经结束，总有1314个团队参赛，本项目的目标是进入排名的top 10%，也就是至少进入前131名，LogLoss最多在0.06127。

# 三、实现方法

### 3.1 数据预处理

#### 3.1.1 异常数据

在2.1小节中我们阐述过，我们的训练数据中包含一些异常的图片。例如：cat.6.jpg中主题部分是人，猫只占一小部分，且不清晰；cat.724.jpg，虽然训练集将它标志为”cat”，但是里面包含了狗和猫等。但是，同时我们也发现测试数据集中也存在异常的图片，例如：7281.jpg中主题部分是两个人，11105.jpg中看不出有狗或者猫等。

所以我们没有清洗掉训练数据集中的异常数据，一方面是因为训练数据集中大部分数据是正常的，异常数据只是很少一部分，对模型提取猫和狗的特征影响不大；另一方面是因为最终的测试数据集中也是有异常数据的，如果我们去掉训练数据集中的异常数据，那么我们的模型就没法学习这些异常数据，也就没法针对测试数据集中的异常数据进行准确预测。

#### 3.1.2 预处理

由于本项目使用了基于Xception的迁移学习，所以应该按照Xception模型的要求来预处理图片。

首先是通过OpenCV库读取图片，然后进行裁剪，使其成为299\*299尺寸的RGB图，所以最终程序中获得的图片其实是299\*299\*3的数字矩阵。

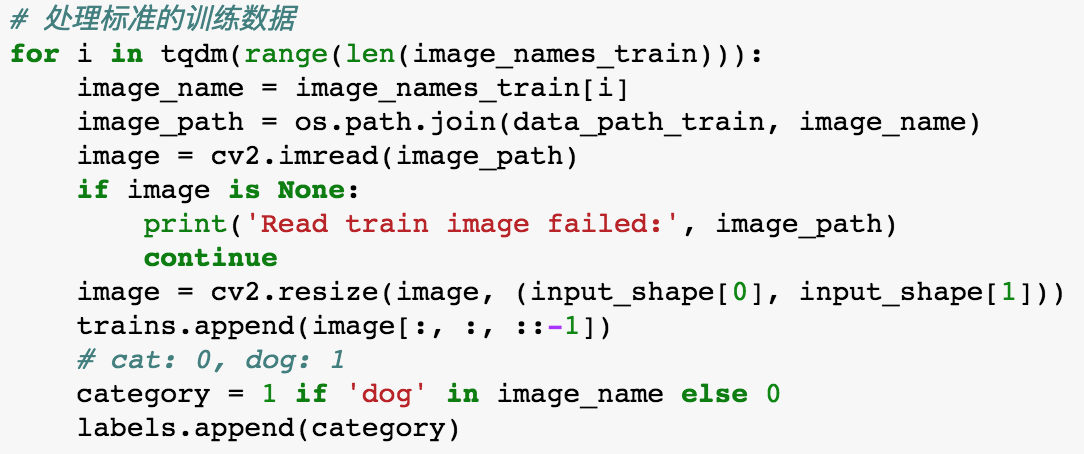


图 3.1 预处理测试数据

由于扩展的训练数据中的图片名中没有”dog”、”cat”关键字，所以只能通过种类来区分该图片属于哪个分类。这其中进行了防御性编程，因为扩展数据集集中除了.jpg图片外，还有.mat文件，以及会出现读取图片失败的情况，这种情况下需要将读取的数据丢弃（处理标准训练数据和测试数据时也是类似）。

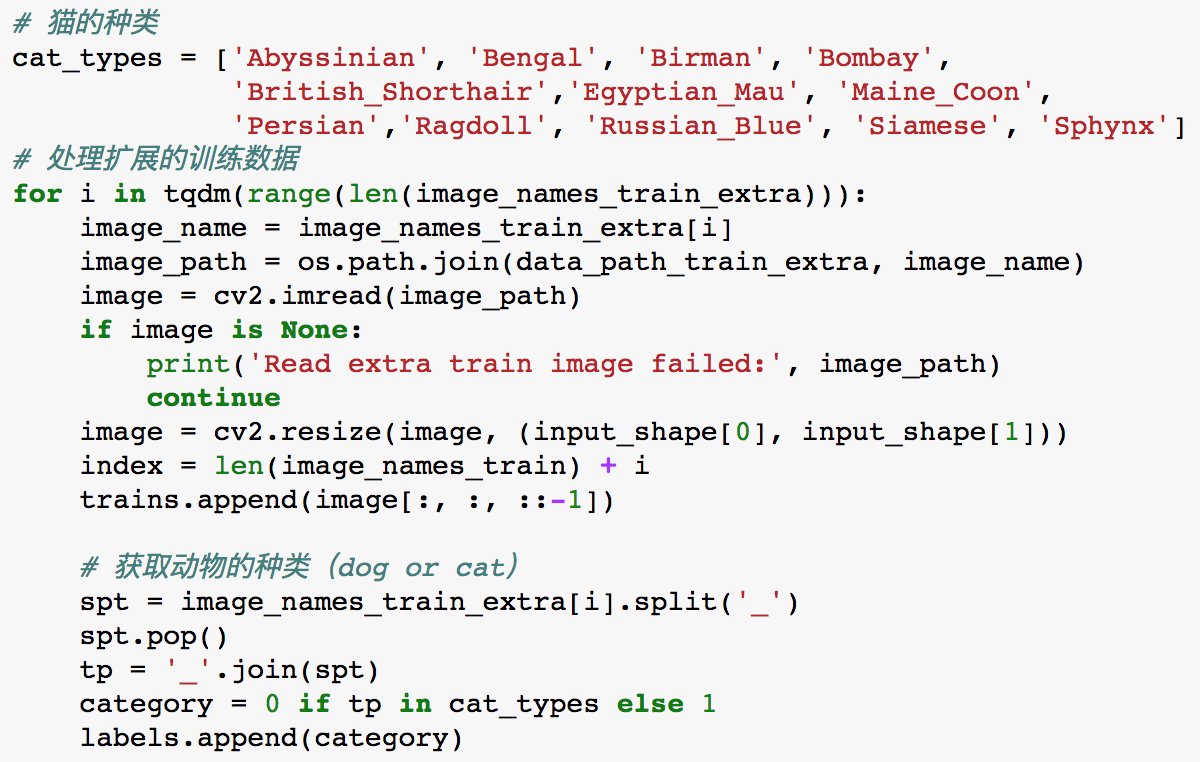


图 3.2 预处理扩展测试数据集

最终将训练数据集（trains）、标签集（labels）、测试集（tests）转化为numpy数组。

### 3.2 实施

本项目使用Xception模型，该网络已经在ImageNet数据集上进行了训练，因为ImageNet数据集包含多种“猫”类和多种“狗”类，这个模型已经能够学习与我们这个数据集相关的特征了。我们只利用Xception网络的卷积层部分，把全连接以上的部分去掉。然后在训练集和测试集上跑一遍模型，以得到训练集和测试集上图片的特征（即“bottleneck feature”，网络在全连接之前的最后一层激活的feature map）记录在两个numpy数组里。在特征提取之前，我们运用了预训练的Xception模型内置的数据预处理函数——xception.preprocess\_input来进行处理3.1中得到的输入数据，以使得输入符合Xception模型的要求。同时，对提取出来的bottleneck feature进行了全局平均池化，这样做一方面是为了减少后续训练参数，提高训练效率；另一方面也可以防止过拟合。另外，由于提取完一遍特征非常耗时，而且后续没必要重复提取，所以我们将得到的两组特征保存到本地。这样后续就可以读取本地数据而不需要再一次重复提取特征了，方便了后续的模型调整和迭代。

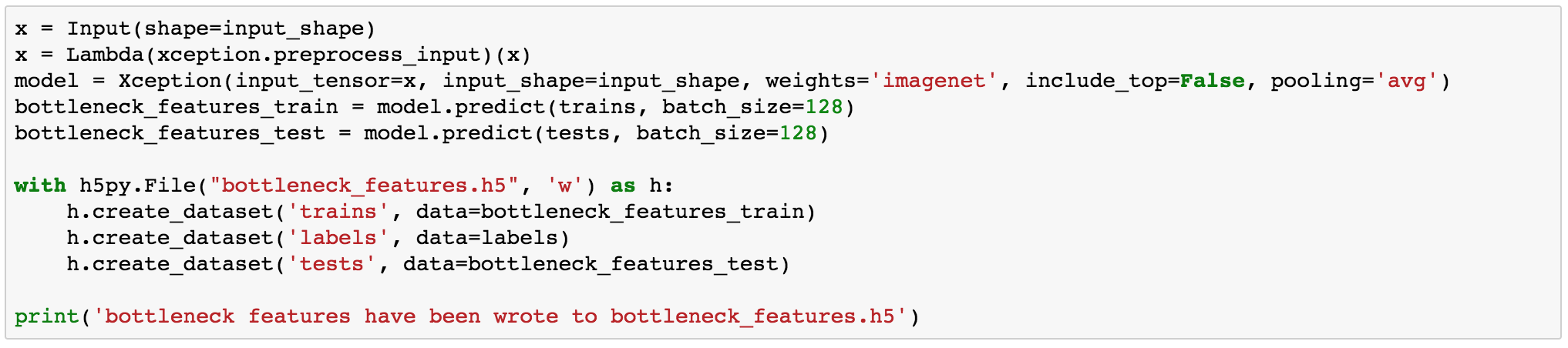


图 3.3 特征提取的实现

我们将训练数据集拆分为：训练集和验证集，其中验证集占比0.2，并且随机打乱数据的顺序。

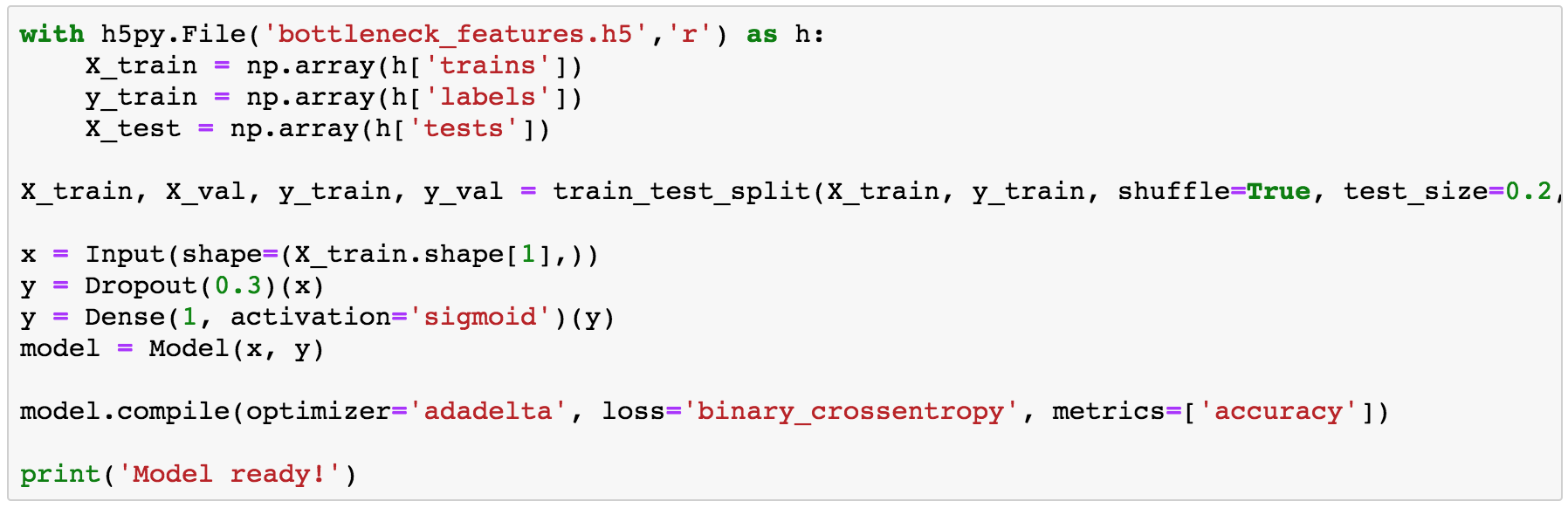


图 3.4 模型搭建

新模型的搭建较为简单，在得到特征向量后，直接经过一层Dropout，然后进行分类。由于是二分类问题，我们使用了sigmoid激活函数；损失函数选用binary\_crossentrop。具体的代码如图3.4所示。

然后，我们基于记录下来的训练集的特征向量来训练新模型。这个时候新模型的输入，也就是上面提取的训练数据集的特征，已经不是原始的图片，而是经过Xception模型（除了最后的全连接层，再加一个全局平均池化层）训练过得到的特征向量，该向量的第一个维度对应样本数量；第二个维度是经过全局平均池化之后的卷积核输出，对于Xception来说这个维度的长度是2048。

经过训练，我们的模型最终可以在验证集上得到0.995左右的准确率，损失值在0.02以下。再经过模型和参数调优后（将在3.3小节具体阐述），就可以用我们的模型来对测试数据进行预测了。

### 3.3 改进

因为使用了基于Xception模型的迁移学习方法，我们构建的新模型较为简单，可调的参数较少。我们选用了adadelta优化器，参数皆使用默认值（默认值来源于论文，已经具有较优的表现，所以我们没有再做调整）。

#### 3.3.1 扩展训练数据集

除了Kaggle官方提供的训练数据集——猫狗各12500张图片，我们还使用了扩展数据集。深度学习的流行，很大程度上也得益于大数据技术的发展，和计算机计算能力的提升。一般而言，深度学习的训练数据集越大，模型越可能学习到数据中存在的通用特征，模型准确率也就越高。因此除了Kaggle官方提供的训练数据外，我们还额外引入了牛津大学的扩展数据集——*The Oxford-IIIT Pet Dataset*用于训练，该扩展数据中包含了37个品种的猫和狗共7349张图片，且图片质量较好，可以很好地反映猫和狗的特征。

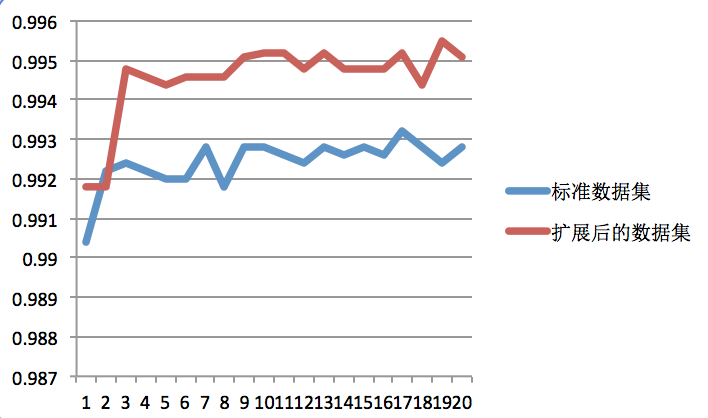


图 3.5 标准数据集和扩展数据集下验证数据集上的准确率

从图3.5可以看出，模型在扩展后的数据集上训练得到的准确率（这里使用了验证集的准确率）明显高于在标准数据集上得到的值。如未做特殊说明，下文中默认模型使用扩展后的训练数据集。

#### 3.3.2 Epochs

Epochs是训练的轮数，每一轮都会完整地训练完一遍训练集。如果Epochs太小，可能模型还没有充分学习好；如果Epochs太大，则有可能导致模型”过渡学习”了训练集，导致过拟合。所以选择一个合适的Epochs很重要。

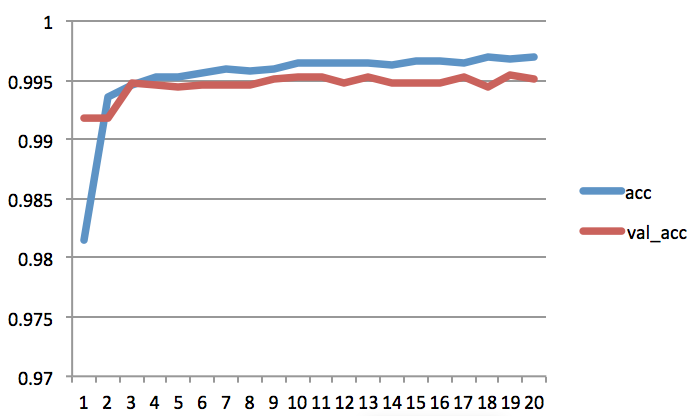


图 3.6 测试集和验证集上的准确率随Epochs的变化趋势

从图3.6可以看出在第4轮之后，验证集上的准确率开始逐渐趋于稳定；训练集上的准确率还在小幅提升。这里我们只参考验证集上的准确率，项目中最终选择的Epochs是10，因为在10的时候模型的准确性比较高，也较为稳定，且是局部最优值。

#### 3.3.3 Dropout

为了防止过拟合，我们还使用了Dropout层。Dropout将在训练过程中每次更新参数时按一定概率随机断开输入神经元，因此Dropout层可用于防止过拟合。丢弃的概率是模型的一个重要参数，下面比较了不同丢弃概率下模型在验证数据集上的准确率。

从图3.7可得，随着丢弃概率的增加，训练集和验证集上的准确率呈下降趋势，其中在验证集上的变化不是特别明显。我们的项目中选取的值为0.2，因为此时模型在验证集上的准确率较高，且是局部最优值。



图 3.7 测试集和验证集上的准确率随dropout的变化趋势

### 3.4 hack预测结果

在测试数据集上运行模型来预测结果的时候，我们采取了一种比较hacker的做法：将预测结果（也就是图片中是狗的概率）通过numpy的clip函数裁剪到[0.005, 0.995]的范围之内。也就是小于0.005的值会被重置为0.005；大于0.995的值会被重置为0.995。

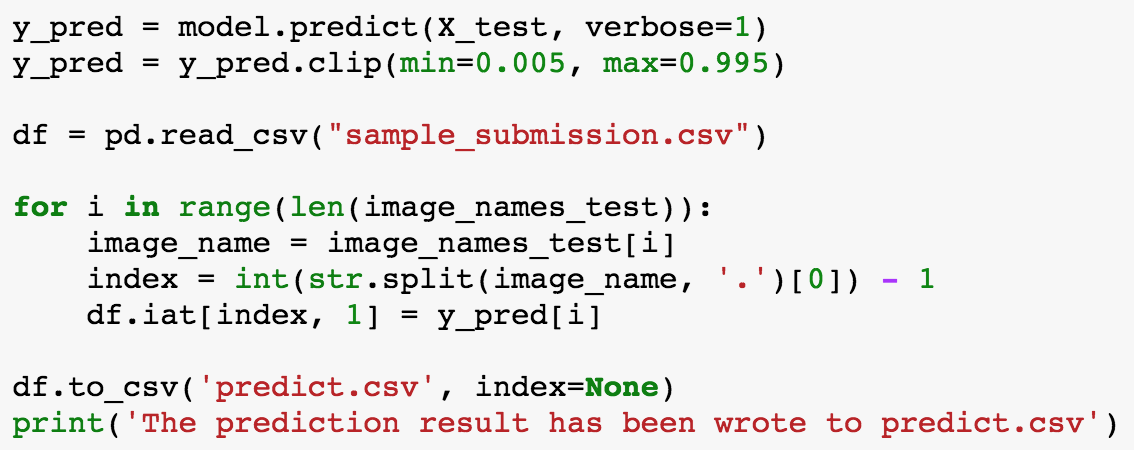


图 3.8 预测的实现

从f(x) = ln(x)函数的图像我们知道得知，x越趋近于0，f(x)的变化率越大（因为f ’(x) = 1 / x）。

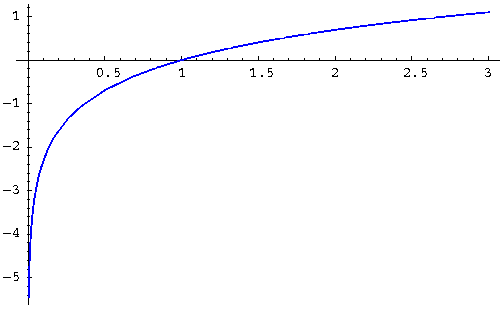


图 3.9 f(x) = ln(x)函数的图像

假设我们所预测的图片中实际上是一只狗（假设是猫的情况下是类似的）。根据2.3小节中我们介绍的LogLoss公式，此时每一条预测结果对于LogLoss的贡献是：

如果我们的模型预测是准确的，且预测结果大于0.995，会被clip到0.995，如此看来这种情况下clip会”拉低”模型的表现。

如果我们的模型预测是错误的，且预测结果小于0.005，会被clip到0.005，如此看来这种情况下clip会”提升”模型的表现。

所以在一部分情况下，clip会”拉低”模型的表现，而在另一部分情况下clip会”提升”模型的表现。但是如果仔细观察图3.9 f(x) = ln(x)函数的图像可知，f(x)在x为0.005附近时候的变化率是远大于x在0.995附近时候的变化率的，也就是说clip”提升”模型表现的时候其影响，是远大于其在”拉低”模型表现的时候的。另外，我们不知道测试数据集中猫狗各有多少个，只能假设这其中猫狗各自的数量是基本一致的，且我们的模型”拉低”和”提升”模型的次数基本上也是一致的。

综上所述：通过使用clip手段，模型的表现（至少Kaggle得分）会变得”更好”。

# 四、结果

### 4.1 模型评估与验证

经过上面的一系列优化方法和参数调优——使用扩展数据集、Epochs选择10、Dropout概率选择0.2，最终得出的模型在训练集和验证集上的准确率如图4.1所示，以及损失值如下图4.2所示。

可以看到，最终的模型在验证集上的损失值可以达到0.02以下；准确率可以达到0.995左右。

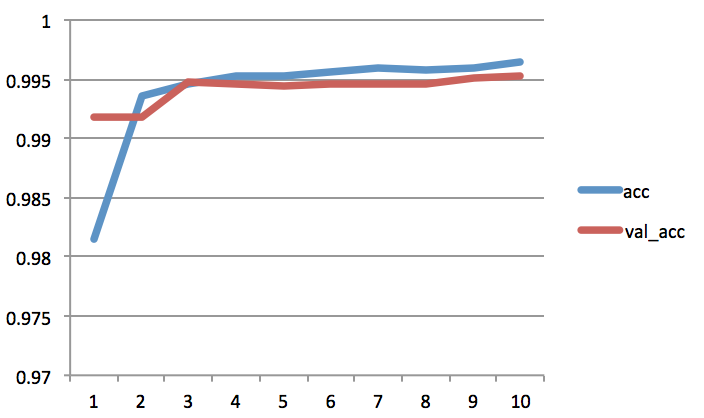


图 4.1 测试集和验证集上的准确率随Epochs的变化趋势

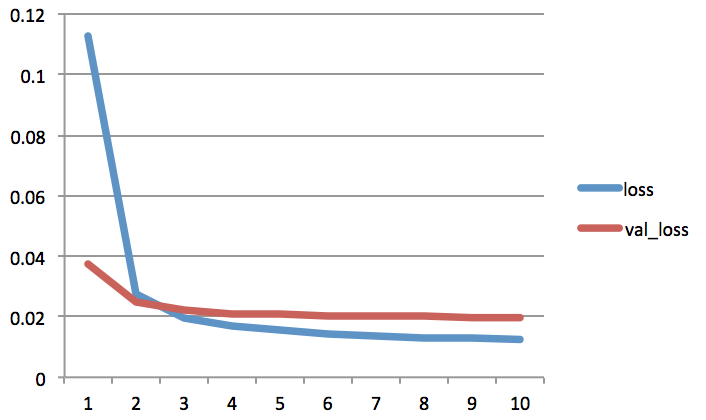


图 4.2 测试集和验证集上的损失值随Epochs的变化趋势

优化后的模型在测试数据集上也有很好地表现。在将预测结果提交Kaggle到后得分为：0.04112，可以排到第18名，达到了在2.3小节提出的基准要求。

综合以上数据可以说明，我们的模型的可以很好地解决猫狗识别的问题。

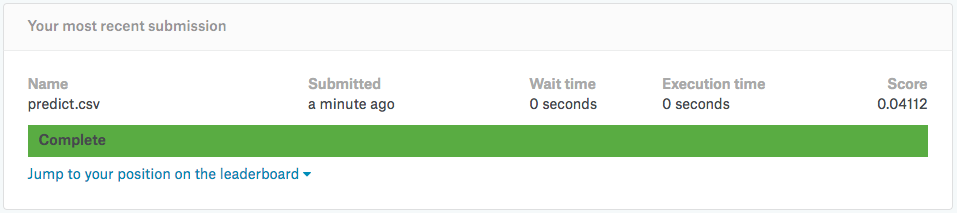


图 4.3 预测结果在Kaggle上的得分

### 4.2 结果分析

****

图 4.4 预测结果解释

我们随机抽取两张测试集中的图片，并通过查找最终生成的csv文件，图4.4中左侧的一张预测值是0.995，通过人眼很好识别这是一只狗；右侧图片的预测值是0.005，而且容易辨识这是一只猫，皆符合预期。因为我们前面的设定就是：预测值越接近1，说明是狗的概率越大；越接近0，则说明是猫的概率越大。

从上面结果可知，我们的模型识别准确率比较高，达到了99.5%以上，识别猫和狗的效果不错，已经达到了2.3小节中定下的最低目标。虽然我们的模型得出的结果达到了第18名的成绩，但是跟排行榜上的第一名（得分0.03302）还是有一定的差距。可能是由哪些原因造成的呢？下图展现了在12500个测试数据上，预测是狗的概率在0.4 ~ 0.6之间的图片（也就是说模型不能准确判断该图片是狗还是猫）。

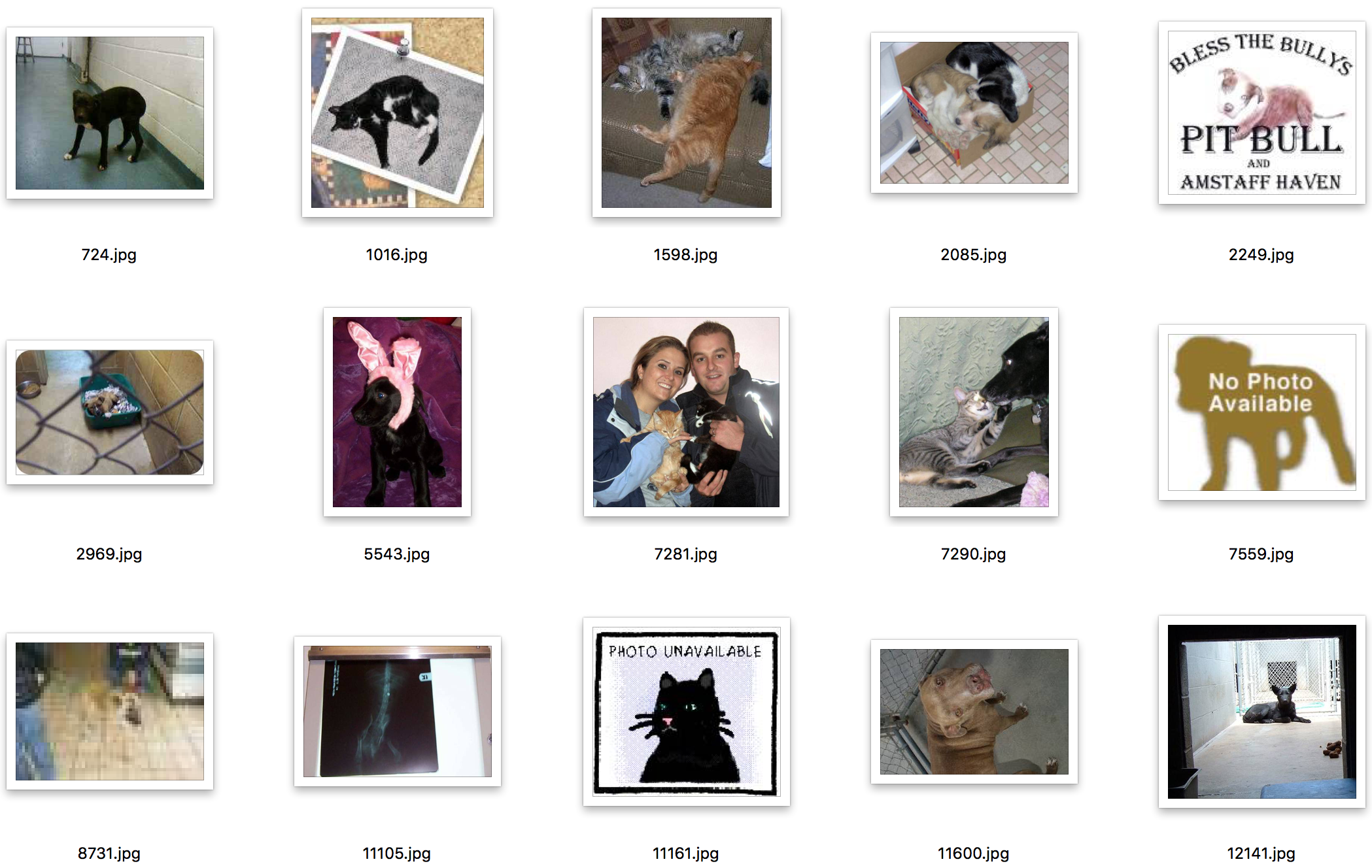


图 4.5 模型无法准确分类的图片示例

经过观察可知，模型无法分类这些图片可能有以下几种原因：

(1) 图片本身质量太差，例如：8731.jpg图片尺寸很小，画面很模糊，人眼也很难识别；其他的还有7559.jpg、11105.jpg等。

(2) 动漫类型的图片，图片风格跟真实的照片不一样，例如：2249.jpg、11161.jpg等。

(3) 图片中动物所占据的区域较小，而障碍物比较大，例如：7281.jpg、2969.jpg等。

(4) 图片中猫或狗的品种在训练数据中没有覆盖到，例如： 11600.jpg、12141.jpg。

所以模型仍然有提升的空间。

# 五、结论

### 5.1 自由形态的可视化

本项目基于卷积神经网络对图像进行识别取了不错的效果。使用了预训练的Xception模型进行迁移学习，只保留Xception的卷基层，先提取出训练数据集和测试数据集中的图片特征，为了减少训练的参数个数以及避免过拟合，还使用了全局平均池化方法。得到特征向量后，直接经过一层Dropout，然后进行分类。实验效果良好，提交到Kaggle后得分是0.04112。这其中很大一部分原因是Xception模型高效的网络结构，以及其预先在ImageNet数据集上训练过，已经能够非常好地概括图像的特征。所以我们基于Xception的迁移学习所得出的模型准确率比较高。

我们的模型对于测试数据集上的12500张图片进行了预测，并且给出了图片中是狗的概率。我们可以假设：预测值在0.9 ~ 1.0之间的便可以认为是狗的可能性非常大；数值在0.0 ~ 0.1之间的便可以认为是猫的可能性非常大；而在0.5左右，例如在0.4 ~ 0.6之间的便可以认为模型无法确切地判断是猫还是狗。我们以0.1为组间距，分析了落在各个区域的测试数据量，如图5.1所示。

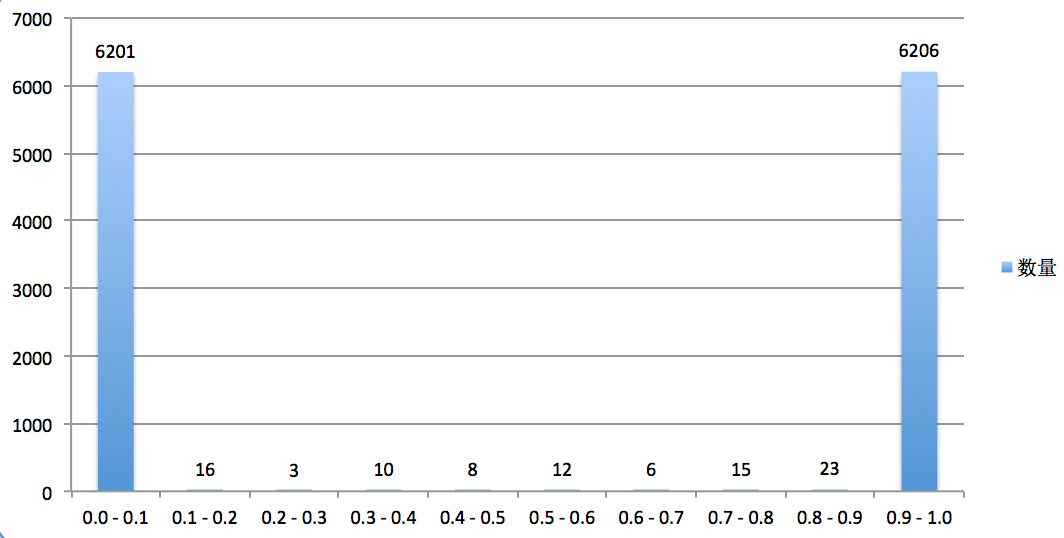


图 5.1 测试数据是狗的可能性分布情况

可以看出，绝大部分数据落在了0.0 ~ 0.1和0.9 ~ 1.0这两个区间，占比99.256%，对于这一部分数据，模型能够比较确切地判断是猫还是狗。这也从另一方面说明我们的模型能够较好的胜任猫狗识别的任务。

### 5.2 思考

经过本次项目，我们总结出解决此类图像分类问题的一个可行的步骤：

(1) 先明确问题域；

(2) 分析已有的数据，大致勾勒出一个解决方案；

(3) 具体实施过程中首先需要进行数据预处理；

(4) 接着构建模型（可能还需要先进行提取特征）；

(5) 然后进行训练，参数调优；

(6) 得到最优化的模型后，进行预测；

(7) 分析结果。

牛顿说：”如果说我比别人看得更远些，那是因为我站在了巨人的肩上”。这告诉我们要善于学习、利用前人的知识积累，并应用于解决自己的问题，或者在此基础上再提出新的想法。本项目使用的迁移学习方法就是一个例子：基于预训练好的Xception模型，来解决图像识别问题。一方面这样可以节省很多手动搭建卷积神经网络的时间，和繁琐的参数调整的时间；另一方面，Xception的参数由大数据量的ImageNet数据集训练而来，所以准确率比较高。所以这种方式较适合快速地解决一些问题，或是将想法付诸实践。当然，Xception模型和迁移学习并不是万能的，我们并不是只需要站在巨人的肩膀上发呆，我们也需要发挥主观能动性来使得我们的解决方案更优。

### 5.3 改进

可以看到，我们的模型最终预测的得分跟Kaggle上的最好成绩仍有一定差距。4.2节中也提出了一些造成这种结果的可能的原因，一个可行的解决方案是对原图片进行更多的预处理，例如随机地进行裁剪、旋转、翻转、调整饱和度等等预处理手段，还利用数据提升的方法；另一个优化方案就是对预训练的模型进行微调（fine-tune）。

另外，由于精力和能力的限制，我们并没有提供一个APP、HTML 5或者小程序一样的可视化、可交互的应用。后续可以考虑制作：用户通过我们的应用打开手机摄像头拍摄图像，实时传到我们的后端服务器，后端会调用我们的模型进行预测，然后将结果返回给用户。这样可以让我们的模型真正地发挥其用户价值。

# 六、引用文献

Simonyan K, Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition[J]. Computer Science, 2014.

Minsky M, Papert S. Perceptrons[J]. American Journal of Psychology, 1969, 84(3):449–452.

Zeiler M D. ADADELTA: An Adaptive Learning Rate Method[J]. Computer Science, 2012.

http://www.robots.ox.ac.uk/~vgg/data/pets/

Chollet F. Xception: Deep Learning with Depthwise Separable Convolutions[J]. 2016:1800-1807.

https://keras-cn-docs.readthedocs.io/zh\_CN/latest/blog/image\_classification\_using\_very\_little\_data/