

# 机器学习工程师纳米学位毕业项目

## 图片识别：猫狗大战

杜启蒙

2018 年 04 月 05 日

问题定义	1
项目概述	1
问题陈述	2
评价指标	3
分析	4
数据的探索	4
探索性可视化	6
算法和技术	7
基准模型	10
方法	10
执行过程	11
完善	12
结果	14
模型的评价与验证	14
合理性分析	15
项目结论	15
结果可视化	15
对项目的思考	16
需要作出的改进	17
参考文献	17

### 问题定义

#### 项目概述

计算机视觉在我们的日常生活中随处可见且变得越来越重要，如自动驾驶、OCR、图片分类等，深度学习在这些领域的突破起着关键性的作用。

项目使用的模型是卷积神经网络(Convolutional Neural Network)，卷积神经网络早在 1997 年便被用来解决字符识别问题，然而它最近地广泛应用源于 2012 年 ImageNet 图片分类竞赛中，CNN 以最高的分类准确率夺得头筹[6]。

CNN 算法模型从最早的 LeNet(1998)[1]、AlexNet(2012)[2] 模型再到 VGGNet(2014)[3]、ResNet(2015) [4]图像的识别准确率在不断的提升，最近的 Xception(2016)[5]模型更是在 ImageNet 的 Top-5 accuracy 中达到了 0.945。

本项目旨在依托前人创建的优秀模型，来构建一个可以对图片中猫狗分类的程序，项目中使用的数据集来自于 [kaggle dogs vs cats 比赛](#)。

## 问题陈述

从图片中侦测出猫狗并对其分类，可知此问题属于计算机视觉领域，且是识别出两类物体，因此又属于分类问题中的二分类问题。在选择的数据集中包含猫和够两种图片，每种图片都是被标记过得，图片要么是狗、要么是猫。

我们的目标就是从给定的标记图片数据集中构建一个模型，对于模型需要对输入的图片返回图片中含有猫或者狗的概率，如下图：



为了实现此目标，我们可以借助卷积神经网络(Convolutional Neural Network) 来构建此模型，通过开源的框架 Keras、TensorFlow、Opencv 快速的完成模型的搭建。

为了完成本项目，可以分为以下几个步骤：

1. 下载 kaggle 比赛使用数据，并对数据进行初步梳理
2. 数据处理成框架需要的格式、切分数据集、生成数据集
3. 通过模型导出向量特征以便在普通记笔记本上做训练
4. 载入向量特征并进行模型构建
5. 模型训练并在测试集上进行预测

## 评价指标

此项目虽然是一个二分类问题，但并不适合使用准确性 ( accuracy ) 作为评价指标，而是使用 kaggle 的 LogLoss 作为评价标准。因为假设有两个模型，它们的准确率虽然一样，但是第一个模型会对一张猫的图片做出 99% 是狗的判断，而第二个模型可能对一张猫的图片只做出 60% 是狗的判断，这样虽然不会影响准确率，但是第一个模型得到的 LogLoss 将会很大，而第二个模型的 LogLoss 就会小很多，这时候虽然无法

用准确度区分模型的好坏，但是可以用 LogLoss 来判断。另外测试集是没有 label 的，因此使用准确率是不可行的 LogLoss 的值越小说明模型表现的越好。

LogLoss 的评价公式如下：

$$\text{LogLoss} = -\frac{1}{n} \sum_{i=1}^n [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \quad [7]$$

其中：

n 表示测试数据集图片的数量，

$\hat{y}$  表示预测图片为狗的概率，

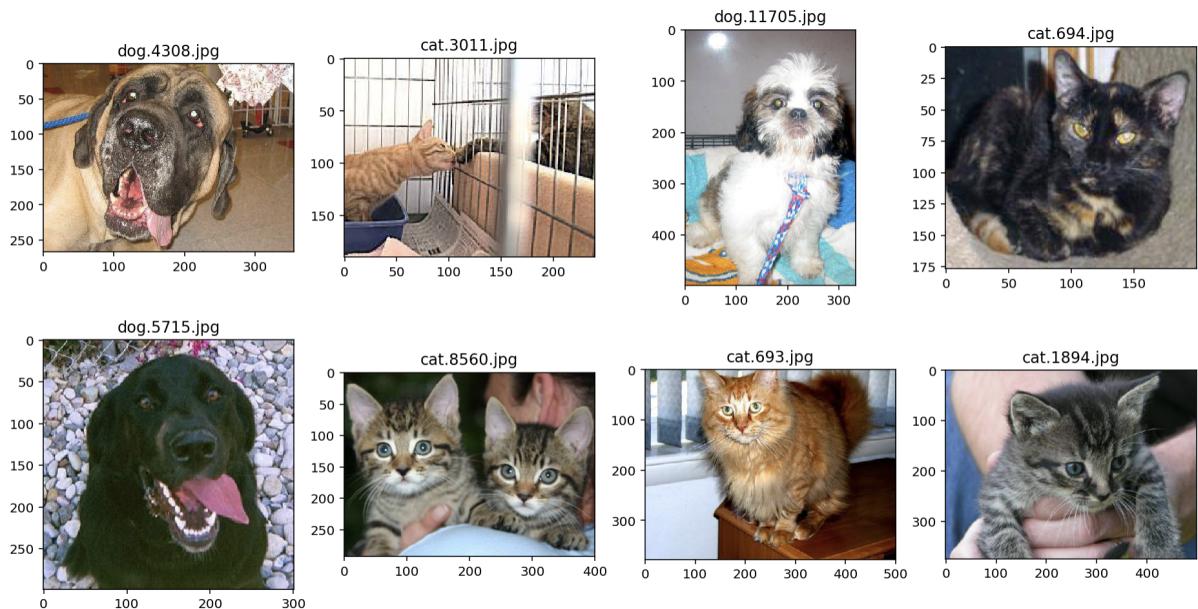
$y_i$  表示如何图片是狗则为 1 是猫则为 0，

$\log()$  表示以 e 为底数的自然对数

## 分析

### 数据的探索

本项目使用的数据集来自于 kaggle Dogs vs Cats 比赛，Kaggle 提供的数据集分为两部分：训练数据集与测试数据集。其中训练数据集是通过文件名字来标记图片，即如果名字为“cat.694.jpg”就表明这是一张猫的图片且编号为 694，标记为狗的图片与此类似，示意图如下：



从上图我们可以观察到：

- 图片具有不同的尺寸
- 图片具有复杂的背景
- 图片中的猫和狗大部分都处在中间位置
- 有些图片含有多个相同的动物（如第二行第二列）
- 图中的动物具有不同的姿势(站、坐、卧)和表情
- 图片中的猫和狗都有很多不同的品种

测试数据集是只有编号没有名称的无标记图片。

通过对使用在 ImageNet 上预训练的模型对训练数据的异常值进行检测，通过抽取 2500 张图片进行识别，将非猫狗的图片识别出来，结果如下图：

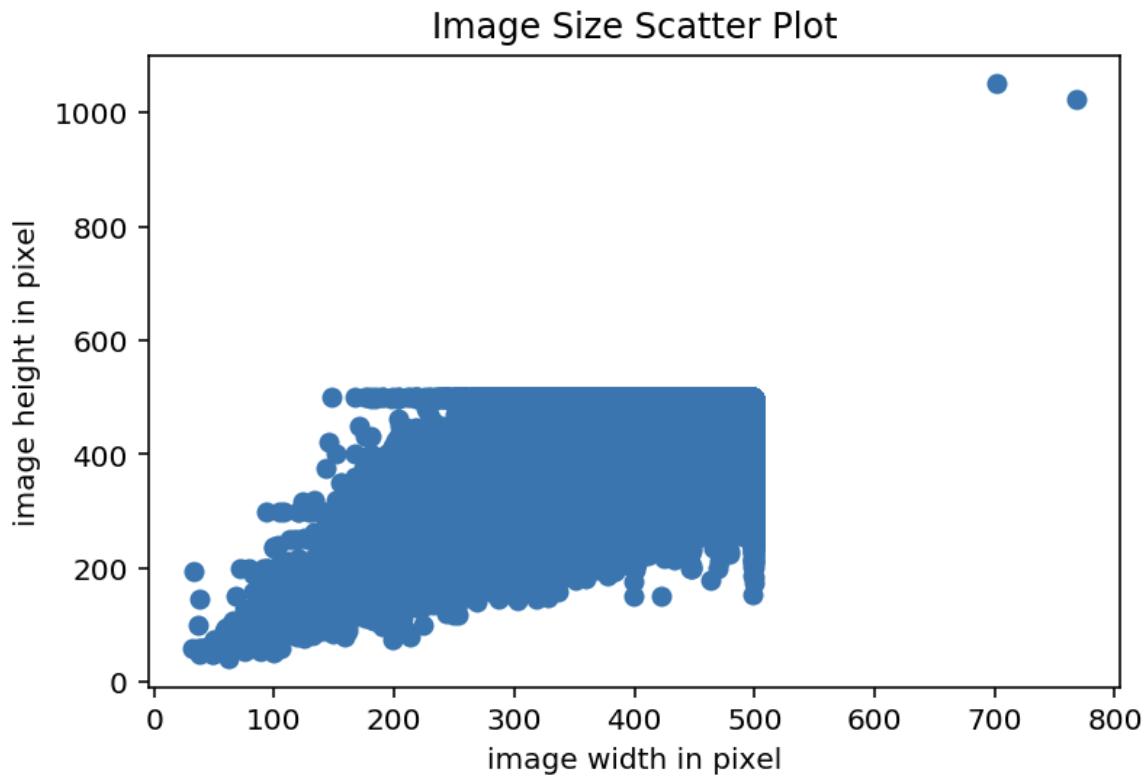


通过观察可以看到图片中确实存在异常的值，如“dog.10237.jpg”，但也有正常的值被当做了异常，通过 Xception 模型验证图片可以看到“cat.1004.jpg”被识别成了“床、被子”，可以看到该图大部分画面都是床和被子。具体比例估算非猫非狗的图片在 20 张左右，占数据的总比例很小，因此此处不选择进行处理。

训练数据一种含有 25000 张图片，其中猫和狗的图片个 12500 张，总大小为 569.9MB；测试数据集含有 12500 张图片，总大小为 284.5MB。

## 探索性可视化

图片大小的散点分布图如下：



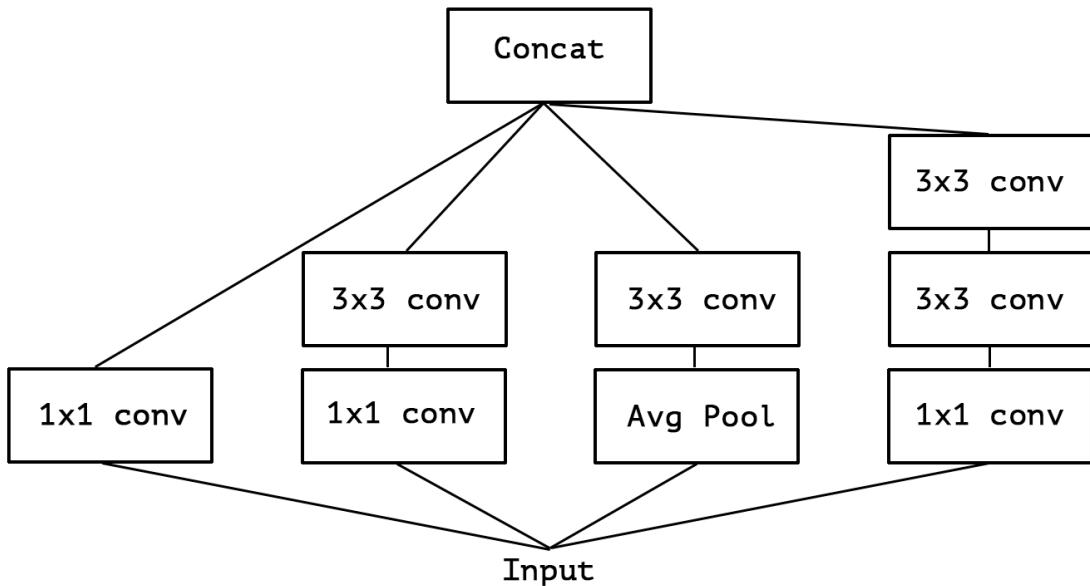
可以看到图片在尺寸分布上具有很大的方差，宽的平均值与中位数为 360、374，高的平均值与中位数为 404、447，从上图也可以观察到有两个异常值。然而 CNN 的输入层需要图片具有相同的大小，因此在输入到输入层之前需要将图片处理成模型需要的尺寸（Xception 需要的尺寸为  $299 \times 299$ ）。

## 算法和技术

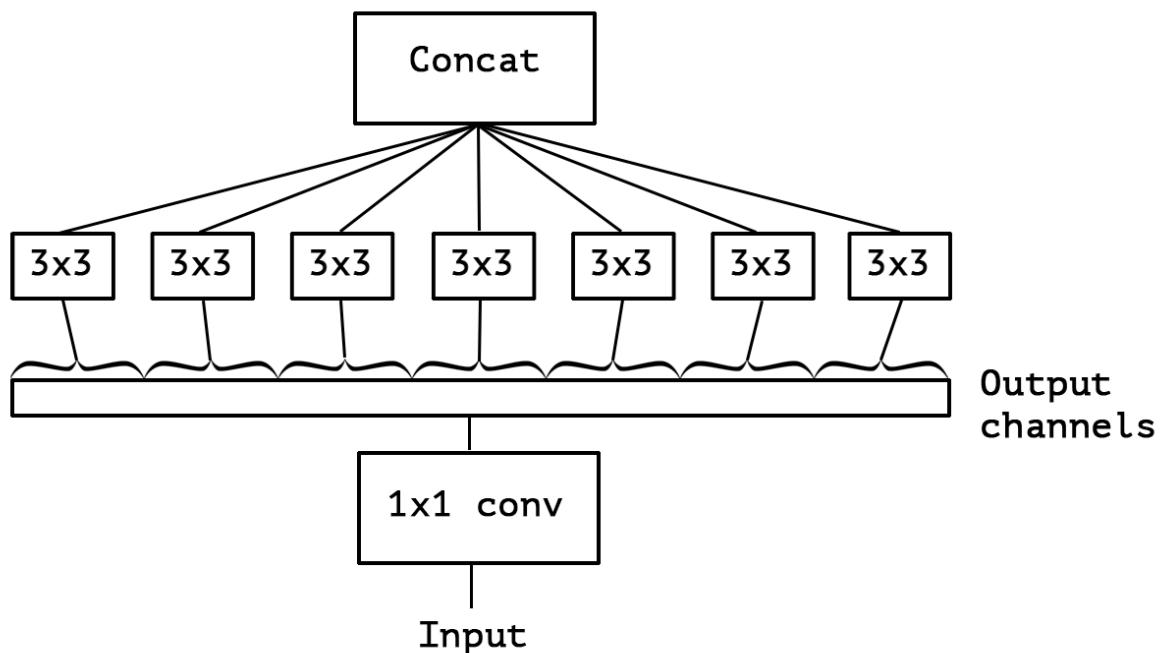
如上的分析可知，此处的项目属于二分类问题。用于分类的算法有很多，比如感知机、逻辑回归、支持向量机、决策树和人工神经网络等。这些算法统称为机器学习算法。算法的输入是由图片每一个像素所组成的特征向量，输出是数字的类别。输入图片是一个特征空间，像素的个数是空间的维度。然而感知机、逻辑回归、支持向量机、决策树在处理图片数据的时候并不是特别理想，而卷积神经网络(Convolutional Neural Network)在图片识别方面具有天然的优势，无论图片中的动物在何种位置，都可以识别出图片的特征。

此项目主要采用 Xception 模型来进行特征提取，将特征保存下来以便后续的训练。

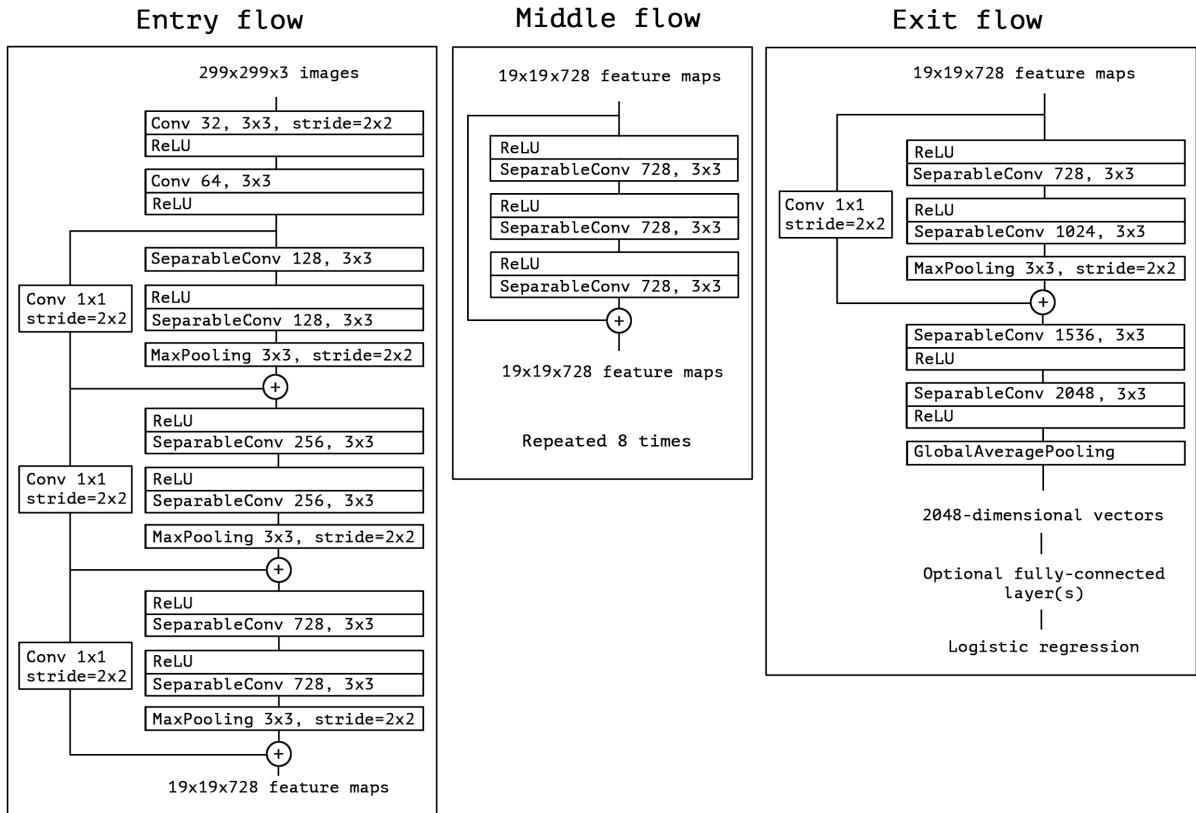
Xception 是 google 继 Inception 后提出的对 Inception v3 的另一种改进，主要是采用 depthwise separable convolution 来替换原来 Inception v3 中的卷积操作。Inception v3 是通过使用  $1 \times 1$  卷积、 $3 \times 3$  卷积、pooling 来完成特征的提取，然后再进行连接，如下图:[5]



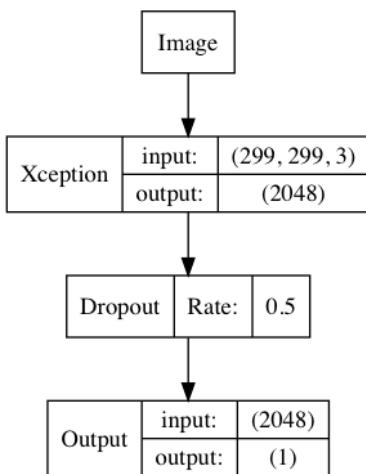
通过 Inception 模块简化，仅保留包含  $3 \times 3$  的卷积的分支，再将所有  $1 \times 1$  的卷积进行拼接，进一步增多  $3 \times 3$  的卷积的分支的数量，使它与  $1 \times 1$  的卷积的输出通道数相等，将会得到如下的结构图：[5]



此时每个  $3 \times 3$  的卷积即作用于仅包含一个通道的特征图上，这就是 Xception 的基本模块。使用这个基本单元结合 ReLU、Maxpooling 就可以搭建出 Xception 的框架：[5]



通过使用 Xception 提取特征之后，载入保存好的特征，构建如下的网络然后进行训练，训练完成就可以对测试数据进行预测。



网络中使用 sigmoid 作为激活函数，使用 Adadelta 作为模型的优化选项，损失函数使用二分类交叉熵(binary\_crossentropy)，度量使用 Accuracy。

项目中会使用到 Google 开源深度学习框架 TensorFlow , TensorFlow 会将神经网络的计算流程表示成为一个计算图 , 同时它还能够利用多核 CPU 和 GPU 的优势。项目中还会使用到 Keras 框架构建的 Xception 模型 ; 为了更快的完成特征提取 , 还会使用亚马逊的 AWS ; 图片数据的特征探索则使用 OpenCV。

## 基准模型

此处以 kaggle dogs vs cats 比赛中 leaderboard 成绩为标准 , 要求名次在前 10% , 也就是以评价指标中公式计算的得分必须在 0.06 以内。

## 方法

因为使用了 keras 来对图片进行处理 , 而使用 Keras 的 ImageDataGenerator 需要将不同种类的图片分在不同的文件夹中去。为了对模型进行验证 , 还需要将训练数据集分出来一部分作为验证使用 , 切分的比例为 5:1 , 即训练数据集 20000 张 , 验证集 5000 张 ( 使用 sclearn 切分 ) 。在训练集与验证集上猫和狗的图片都应该分别存放在不同的文件夹下 , 这样通过 ImageDataGenerator 的 flow\_from\_directory 就可以依据文件夹来判断图片有多少个类别。对于测试数据则只需要将全部图片处理到一个文件夹下即可。处理好的数据文件夹结构如下 :

```
data
└── test
    └── test [12500 entries exceeds filelimit, not opening dir]
└── train
    ├── cats [10000 entries exceeds filelimit, not opening dir]
    └── dogs [10000 entries exceeds filelimit, not opening dir]
└── valid
    ├── cats [2500 entries exceeds filelimit, not opening dir]
    └── dogs [2500 entries exceeds filelimit, not opening dir]
```

使用 `ImageDataGenerator` 从文件夹里读取数据时，将图片的大小重新设置为 `Xception` 需要的  $299 \times 299$ ，同时设置批次的大小，将混洗选项设置为 `False`。在特征提取的过程中使用 `keras` 的 `process_input` 对图片进行归一化等处理。

## 执行过程

首先使用 `Xception` 模型对数据的特征进行提取，因为 `Xception` 需要将数据限定在  $(-1, 1)$  的范围内，然后我们利用 `GlobalAveragePooling2D` 将卷积层输出的每个激活图直接求平均值，不然输出的文件会非常大，且容易过拟合，最后利用 `model.predict_generator` 函数来导出特征向量。

最后导出的 `h5` 文件包含五个数组：

- `train(20000, 2048)`
- `label(20000,)`
- `validation(5000, 2048)`
- `v_label (5000,)`
- `test(12500, 2048)`

经常上面处理之后，我们会获得一个 `xception.h5` 的特征向量文件，为了进行接下来的训练需要载入这个特征向量文件。载入特征文件之后提取里面的特征，然后构建模型。模型构建很简单，通过提取的训练数据的特征都构建一个 `Keras` 的 `tensor` 输入，之后直接 `dropout`，再添加一个 `dense` 层，在 `dense` 层使用 `sigmoid` 作为激活函数，模型就算构建完成。

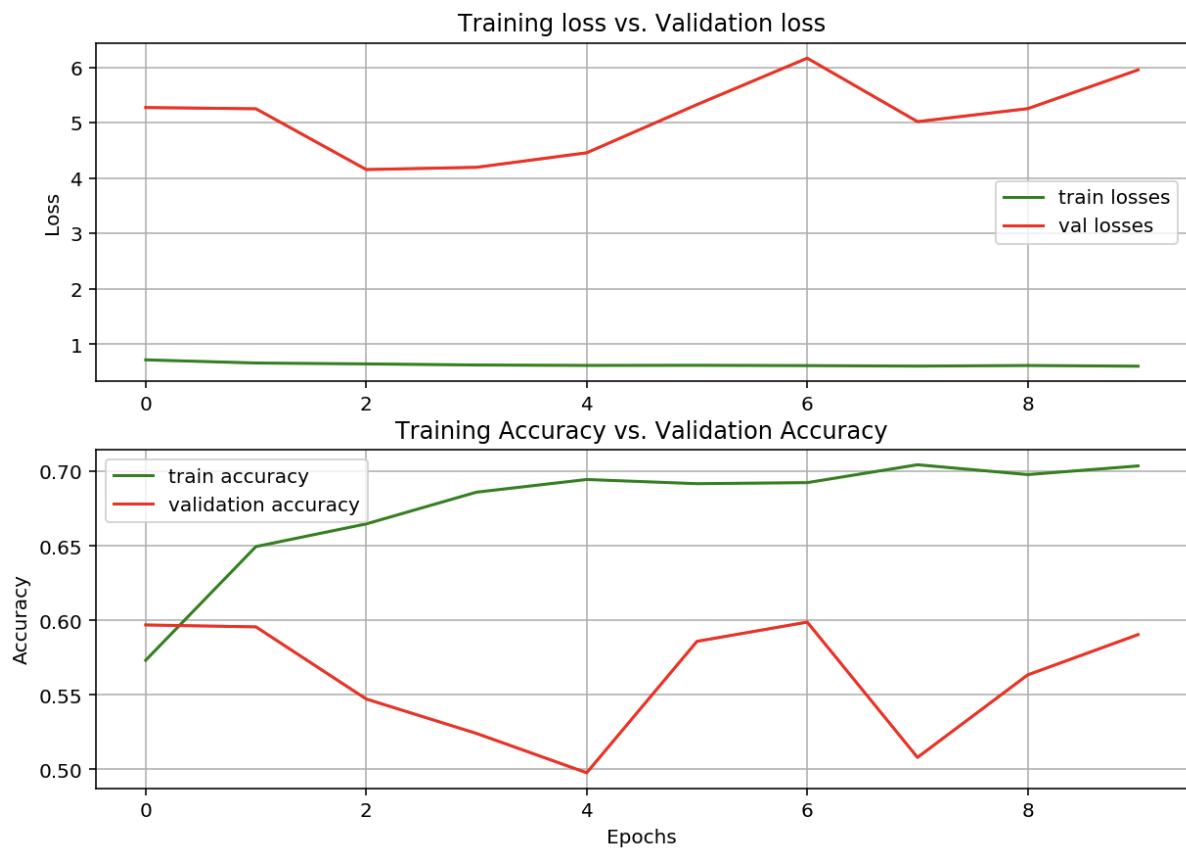
模型构建时对参数调节进行调节，如调整 `Dropout` 成的丢弃率，或者调整全连接层的激活函数类别。

模型构建完成之后，接下来就可以进行训练，经过特征预提取之后训练的速度会比较快，且在训练集上的准确率很高，在验证集上的准确率也达到了 99.3%。

训练完成之后就可以在测试集上进行测试，然后提交到 Kaggle 看到 LogLoss 的评分。在进行预测时，我们将每个预测值限制到了  $[0.005, 0.995]$  个区间内，这是因为，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。

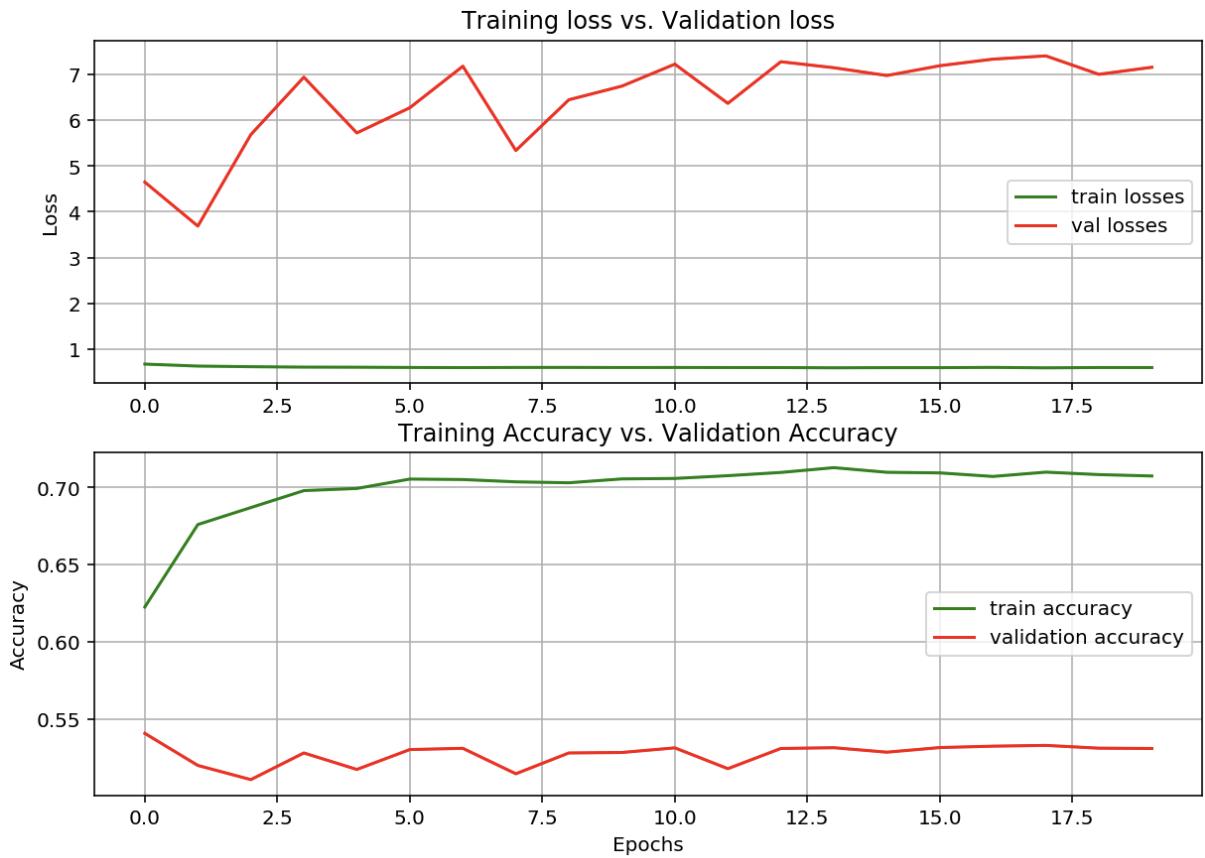
## 完善

技术的改善主要在模型的构建处理方式上，最开始构建模型使用的是直接进行模型的构建，在原有的 Xception 模型之后添加池化层、Dropout 层、Dense 层，编译模型并建立检查点，通过 `fit_generator` 在数据上训练模型，最开始设置的 `epochs` 值为 10、`steps_per_epoch` 值为 1024、`validation_steps` 值为 512，得到的训练结果如下图：



将预测结果提交到 Kaggle，得分为 0.73389，分数并不是太高，并没有满足基准模型的要求。

之后将 `epochs` 值为 20、`steps_per_epoch` 值为 2048、`validation_steps` 值为 1024，得到如下的结果图：



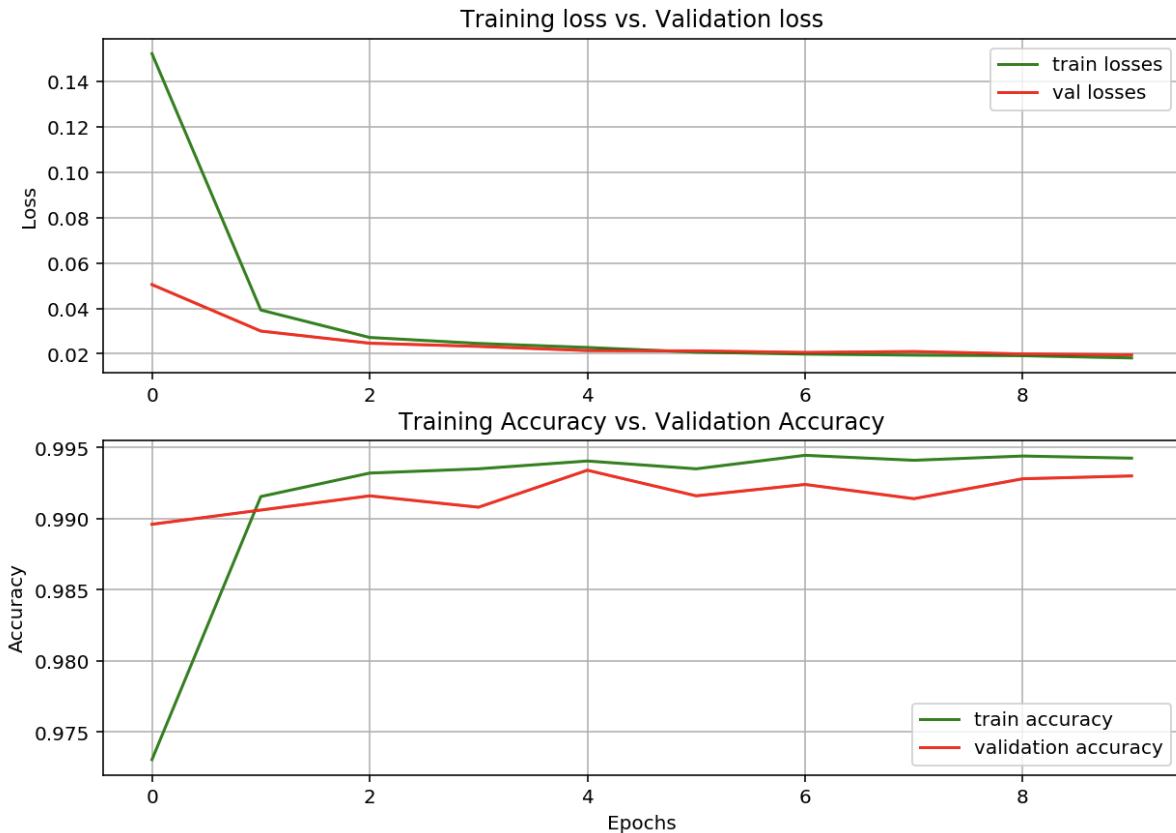
将预测结果提交到 Kaggle 得到的分为 0.35137，虽然分数所有提到但离基准模型的差距依然较远。

之后通过建立冻结卷基层的方式进行训练，得到的结果依然在 0.3 左右徘徊。最后改用先通过 Xception 在 ImageNet 上预训练的权重进行特征的提取，将提取的特征保存到 h5 文件中，在按照执行过程中描述的方式进行模型构建，让模型通过导出的特征向量进行训练，在构建模型的时候对 Dropout 的丢弃率与 Dense 层的激活参数进行了调节，得到的如下的表格：

Dropout	Activation	val_loss	val_acc
0.01	sigmoid	0.0199	0.9924
0.3	sigmoid	0.0210	0.9912
0.5	sigmoid	0.0195	0.9930
0.7	sigmoid	0.0206	0.9918
0.99	sigmoid	0.0760	0.9886
0.3	relu	0.0434	0.5520

0.5	relu	0.0567	0.5096
0.7	relu	0.0595	0.5030

可以看到在 Dropout 丢弃率为 0.5，且在激活函数使用 sigmoid 时模型的损失较低精度较高。在此参数下对模型训练，得到的结果如下：



从图中可以看到在训练集与验证集上 loss 都比较低，且在训练集与验证集上 accuracy 都达到了 0.99。在此模型下将预测的结果提交到 Kaggle，分数也达到了 0.04101，已经满足了基本模型的要求。

## 结果

### 模型的评价与验证

通过前面的改善环节可以看到，在最中的模型上训练集与验证集都有较好的表现，且在测试集上的预测结果提交到 Kaggle 也有较好的表现，控制在了 0.06 以内，因此最终的模型跟预期的结果是一致的。

在最终的模型中 Dropout 层选择的丢弃因子为 0.5，选择这个参数可以让模型不会出现过拟合的现象。

使用的 Xception 模型是在 ImageNet 上进行过预训练，可以说是经过了严格的验证，通过这个模型导出的特征向量可以很好的概括一张图片所具有的内容。因此通过 Xception 构建的模型是足够稳定的，即使输入数据有微小的改变也不会对结果造成极大的影响。

在预训练的 Xception 模型基础之上构建的模型所得出的结果还是比较可信的。

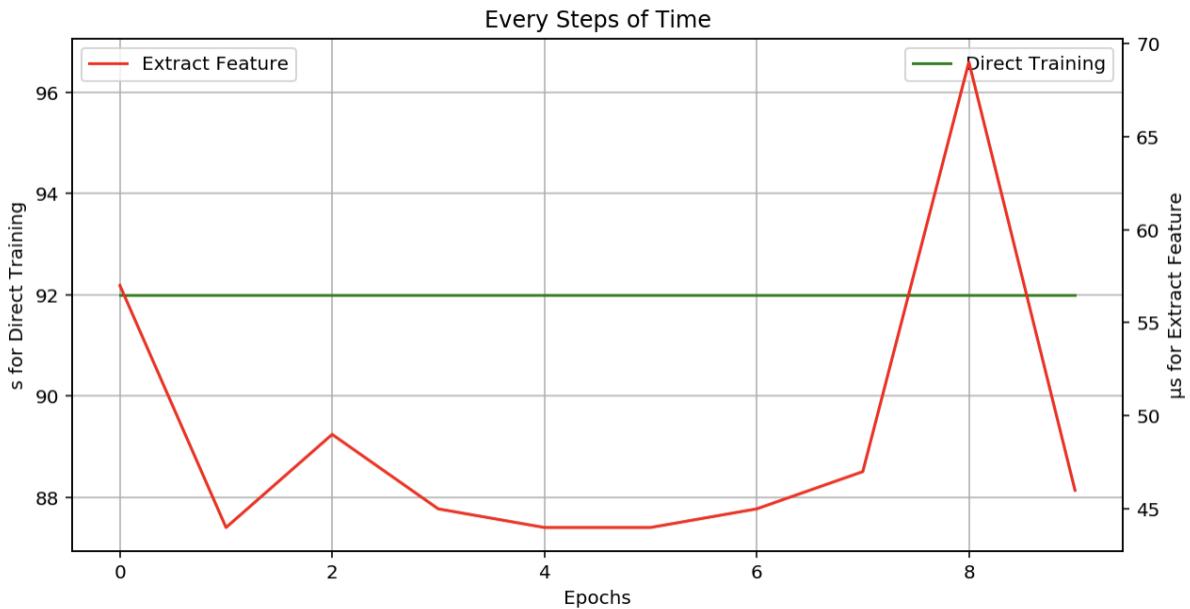
## 合理性分析

最终提交到 Kaggle 的得分为 0.04101，这个得分已经在以 0.06 为基准分数之内，可以说已经是表现的很好了。

## 项目结论

### 结果可视化

此项目中最重要的一点是使用 Xception 对特征进行了提取，在 AWS 上使用 p3.2xlarge 将其他的特征保存到 h5 中之后，将保存的特征下载到本地计算机上就可以进行后续的训练，而且可以在很快的时间之内就训练完成。而如果使用 Xception 和后续的层直接进行训练速度会非常的慢，两种方式的每一步训练时间对比图如下：



可以看到记性特征提取的方式平均每步时间为  $49\mu\text{s}$ ，而使用直接训练方式的每步时间为 92ms，可以看到通过特征提取的方式比之训练快了 2000 倍。

## 对项目的思考

在此项目中我们并没有选择从头搭建一个模型，而是选择了 Google 的 Xception 模型，此模型在 Keras 中已经有了很好的实现，以 Xception 为基础再添加 Dropout 与 Dense 层就完成了模型的搭建。在模型训练的过程中也不是选择自己去完成的去训练 Xception 模型，而是使用在 ImageNet 已经预训练好的，通过她完成了特征向量的提取，之后在本地计算机上完成模型搭建与训练。

刚开始在使用直接训练的方式时，无论怎么调整参数都无法将训练集与验证集上的准确率，而且训练的效率也特别低。后参考了《猫狗大战 99.6% 思路》这篇文章之后，改用预训练的方式，使得训练的效率与准确率都有了大幅度的提升，在 Kaggle 上的得分也提升了一个数量级。

在进行 Dropout 与 Dense 层参数调节是可以看到，Dense 层的激活函数的选择对模型的损失与精确度都有较大的影响，从 sigmoid 切换到 relu 精确度几乎减少了一半。而 Dropout 参数值对于损失与精确度的影响并不是特别大。

从最终的结果

来看，最终的模型还是符合了预期。

## 需要作出的改进

从上面的结果可以看到最终的模型已经满足了预期的要求，我们还可以通过多模型联合特征提取的方式来进一步提高精确度和再 Kaggle 上的得分。

## 参考文献

[1] [Yann LeCun, Leon Bottou , Yoshua Bengio, And Patrick Haffner \(1998\). Gradient-Based Learning Applied to Document Recognition](#)

[2] [Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton \(2012\). ImageNet Classification with Deep Convolutional Neural Networks](#)

[3 ][Karen Simonyan , Andrew Zisserman \(2014\). Very Deep Convolutional Networks for Large-Scale Image Recognition](#)

[4] [Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Microsoft Research \(2015\). Deep Residual Learning for Image Recognition](#)

[5] [Francois Chollet\(2016\). Xception: Deep Learning with Depthwise Separable Convolutions](#)

[6] [Dumoulin, V. and Visin F. \(2016\). A guide to convolution arithmetic for deep learning.](#)

[7] [Kaggle Dogs vs. Cats Redux: Kernels Edition](#)

[8] [Boqiang Hu\(2017\). My Cat Vs Dog solution using deep learning with transfer learning strategy](#)

[9] [Yang Peiwen\(2017\). 猫狗大战 99.6% 思路](#)