

# 机器学习工程师纳米学位项目报告

## 猫狗大战

刘佳业

2018 年 05 月 15 日

### 1、问题定义

#### 1.1、项目概述

这是一个图像识别的项目，识别图像中的物体是猫或者是狗（图片中会有许多其他各种物品，但我们只关心其中的猫或者狗），因此，这是一个二分类问题。

该项目源自于 Kaggle 平台的一个娱乐竞赛项目，因此训练集图片，测试集图片，皆由 Kaggle 提供，分类结果的评判标准也是以 Kaggle 平台为准。

本项目将使用卷积神经网络（CNN）作为基础模型，搭建一个具有学习能力的深度学习网络，对输入的训练集图片进行学习及调参，并最终得到一个分类算法模型，可以对训练集的图片进行分类。

CNN 是一种类似于滤波器一样的东西，只是普通的滤波器是一维的，而 CNN 可以是多维的。

项目的输入数据包括 12500 张标记好的猫的照片和 12500 张标记好的狗的照片，这 25000 张图片构成训练图片集，另外有 12500 张未分类的猫和狗的图片，是测试集，用于最终评估分类算法的性能。

#### 1.2、问题陈述

Kaggle 竞赛项目中所用到的图片，都是从现实生活中采集的包含猫或者狗的图片，图片的分辨率，图片中的场景、光线、背景各不相同。猫或狗的颜色，姿态，数量也各不相同。这些都有利于提高算法模型的适用范围。

项目过程中，使用了 2 个步骤进行模型的搭建和模型的学习。其一是使用现成的算法模型 Xception 提取图片特征；其二是搭建神经网络对提取的特征进行分类，并训练该部分神经网络。

同时，分类算法需要给出分类的自信度，取值为-0.1：1 代表 100%是狗；0 代表 0%是狗，也就是 100%是猫（这是个二分类问题，不存在其他第三种状态）。

#### 1.3、评价指标

Kaggle 本次猫狗大战竞赛，使用的评价指标为对数损失函数（越低越好），计算方法如下：

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

可以看出，该函数不但与正确率相关，也与分类的自信度相关，在当前算法正确率越来越接近 100%，正确率对算法的优劣的区分度越来越弱的情况下，自信度对于算法优劣就更加重要了。

## 2、分析

### 2.1、数据的探索

输入数据为 Kaggle 上下载的图片数据，包括：25000 张标记好的图片作为训练数据集（训练过程中需要将其拆分为 2 部分，作为训练数据集和验证数据集），这 25000 张图片又分为 12500 张标记为猫的照片和 12500 张标记为狗的照片。另外，还有 12500 张图片是未标记的，将作为测试数据集，用于最终评估算法模型的优劣。

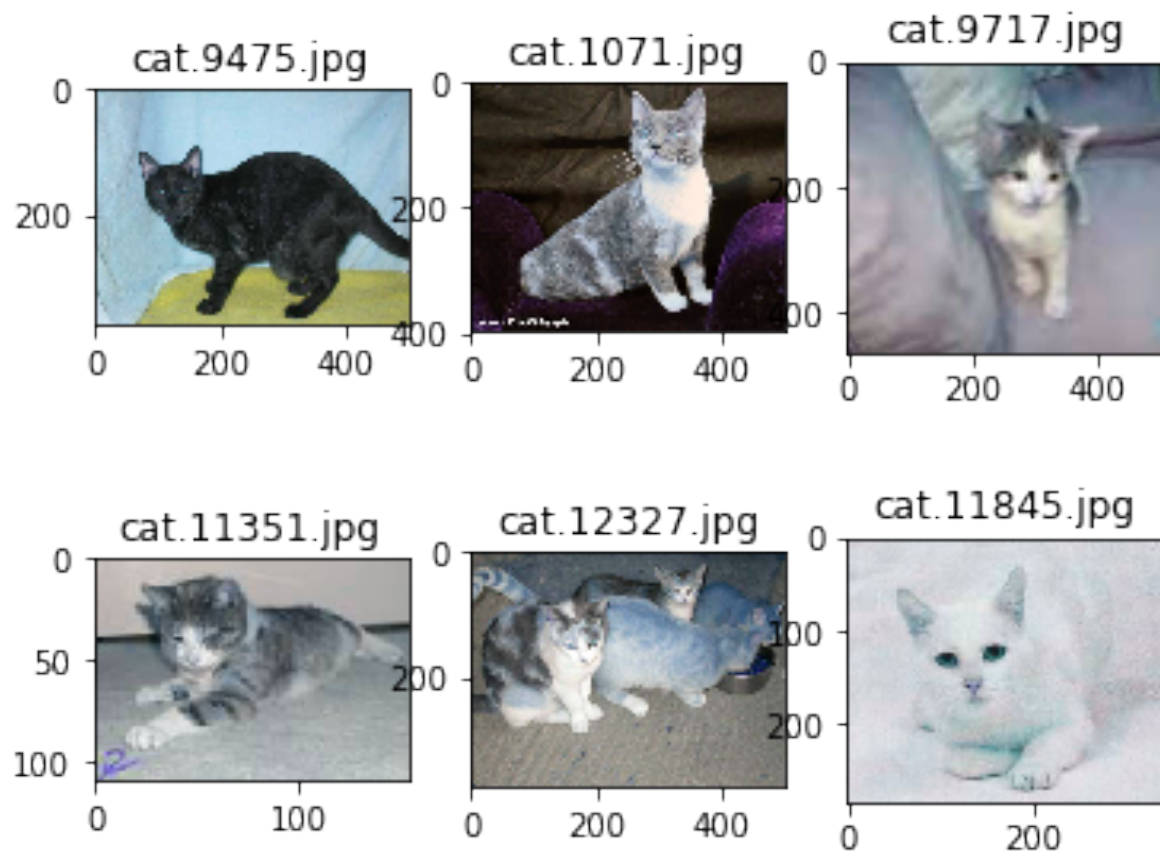
随机浏览这些图片可以罚下，在这总共 37500 张图片里面，图片的分辨率，图片中的场景、光线、背景各不相同。猫或狗的颜色，姿态，数量也各不相同。

另外，由于猫狗所占比重一样，构建训练模型时不再调整类别权重。

### 2.2、探索性可视化

每次随机抽取 18 张图片，其中 6 张是标记为猫的，6 张标记为狗的，以及 6 张未标记的测试集图片：

图 1、训练集图片



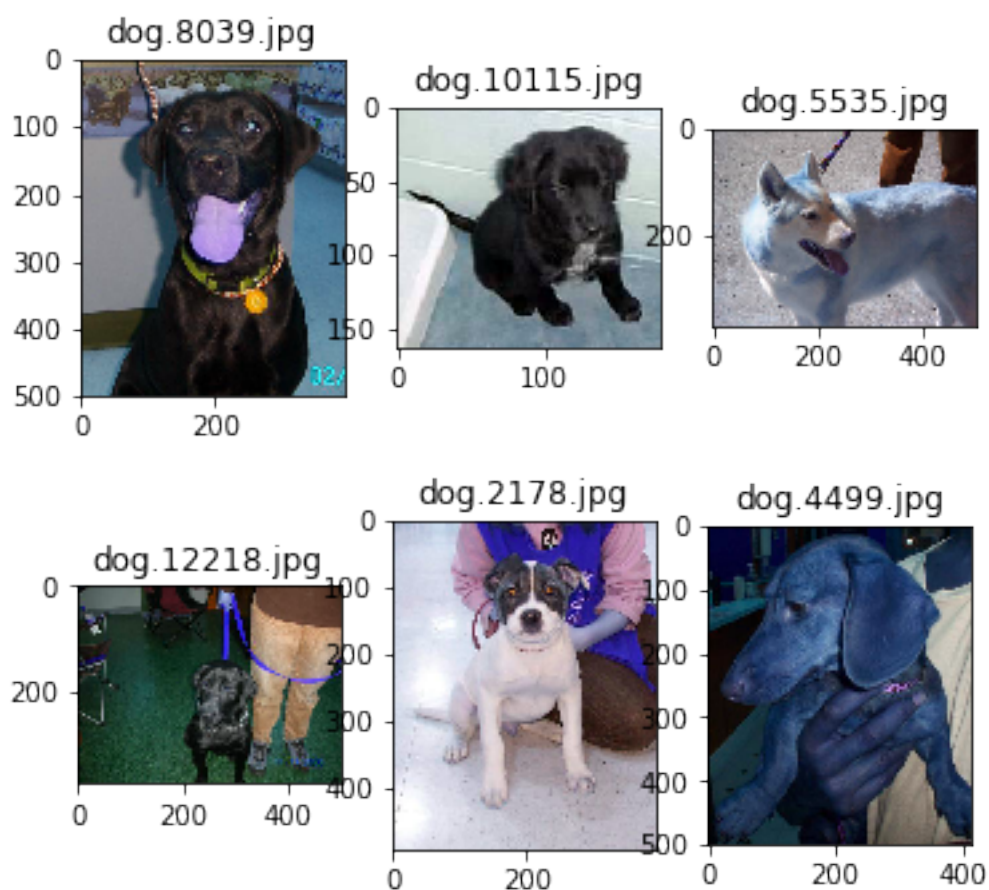
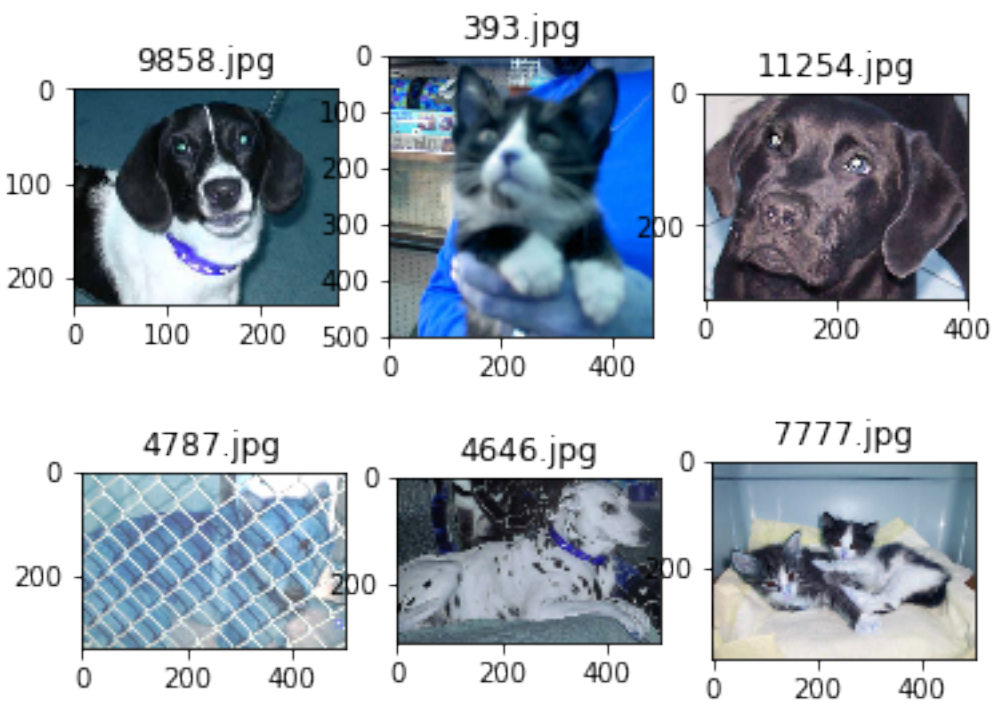


图 2、测试集图片



可以看到，从动物形态，图片背景，照片光线等角度看，数据的多样性还是比较完备的。

## 2.3、算法和技术

### 2.3.1 基础算法

这是一个图片识别问题，也是一个图片分类的问题，在这类问题中，卷积神经网络有较好的优势。

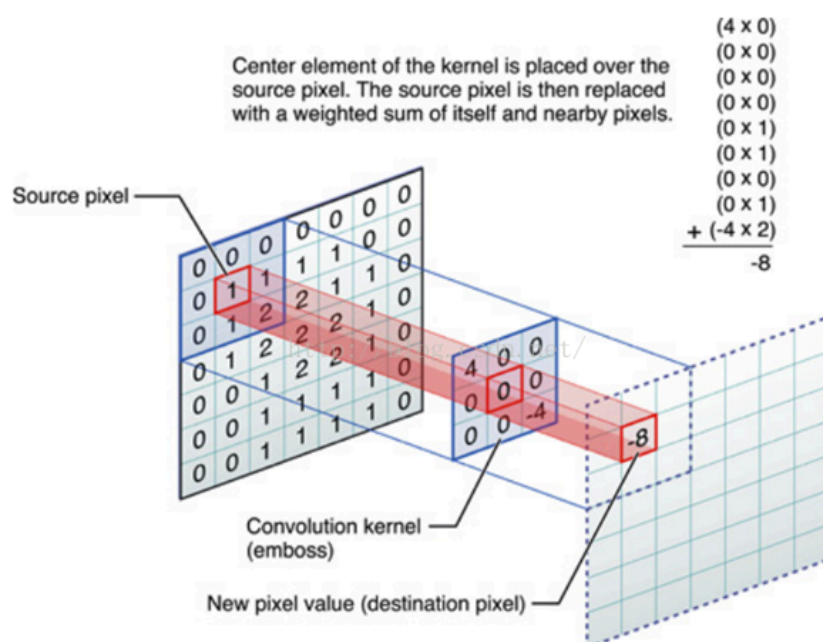
卷积神经网络 (Convolutional Neural Network, CNN) 是一种前馈人工神经网络，一般情况下，卷积神经网络由一个或者多个卷积层，池化层，激活函数以及顶层全连接层组成。它通过较小的卷积单元（也代表较少的参数），在图片上提取特征，然后又可以使用多个这种较小的卷积单元，以及多层卷积单元，实现对图片不同特征以及非常抽象的特征的提取。

卷积层 (Convolutional layer) 是卷积神经网络的核心模块，在卷积神经网络中，一般第 1、2 层卷积层可以提取简单的图片特征，如边缘、线条、角等，更多层、更深层的网络，提取的特征更复杂，更加抽象。

尽管全连接前馈神经网络也可以实现对图片特征的提取以及分类，但这样做是非常耗费资源的，因为全连接的网络会有太多参数。如一输入图片维度为  $300 \times 300$ （这里没有考虑 RGB，只有一层灰度），那么第二层（全连接层）的单个神经元就有 9 万个参数，当每层的神经元变多，且层数增加时，需要训练的参数数量将会变得极大，难以训练。






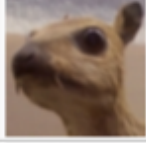
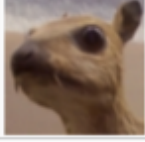
而卷积神经网络就不一样，卷积核 (Convolution kernel) 一般都较小，如  $3 \times 3$ ，如图 3，卷积核先对输入图片的一小块区域进行特征提取，然后移动卷积核（按照确定的移动步长，如步长为 1 代表卷积核每次移动 1 小格），使其作用在输入图片的另一小块区域，最终在整个输入图片上进行特征的提取。

图 3、卷积



卷积核虽小，但是对于提取图片特征却也能起到较好的作用，如图 4 边缘检测，图片锐化，模糊化处理等。

图 4、卷积核作用

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

在图片上使用多组不同的卷积神经元，以及多层神经网络结果，再应用反向传播算法进行训练，更新这些神经元（包括卷积层）的权重，使得神经单元的参数能高效和准确地提取图片特征，可实现更准确的分类。

池化层（Pooling Layer），是用于降低采样的方法，它在尽量保持原有图片特征的基础上，降低特征的维度，通常池化方法有两种：最大池化和平均池化，假设池化的核维度为  $2 \times 2$ ，那么最大池化就是在  $2 \times 2$  的空间中获取最大的那个参数并提取、输出，如果是平均池化，就是在  $2 \times 2$  空间中的 4 个参数取平均值并提取、输出。池化层和卷积层一样，有移动步长参数。

全连接层（Fully connected layer），将输入层的每一个神经单元连接到输出层的每一个神经单元。这也是传统意义上的多层感知机神经网络。

另外，还有其他的算法，如 flatten 层，用于将输入的输出展平成一维的，方便后续处理。

在深度学习快速发展的今天，卷积神经网络越来越复杂，层数也越来越多，达到了几十层，在这些网络里面应用的技术也有长足的发展。

相较于全连接网络，

### 2.3.2 Xception

在 ResNet 使用残差连接网络[1]的基础上以及 Inception[2]的基础上，又发展出了 Xception 这种连接网络。

简而言之，Xception 是一种在深度上分开堆叠的卷积网络，且应用了残差连接方式（a linear stack of depthwise separable convolution layers with residual connections）[3][4]，这里所说的深度，也是指通道，见图 5（通道分开卷积）、图 6（Xception block 结构）、图 7：

图 5、depthwise separable convolution layers

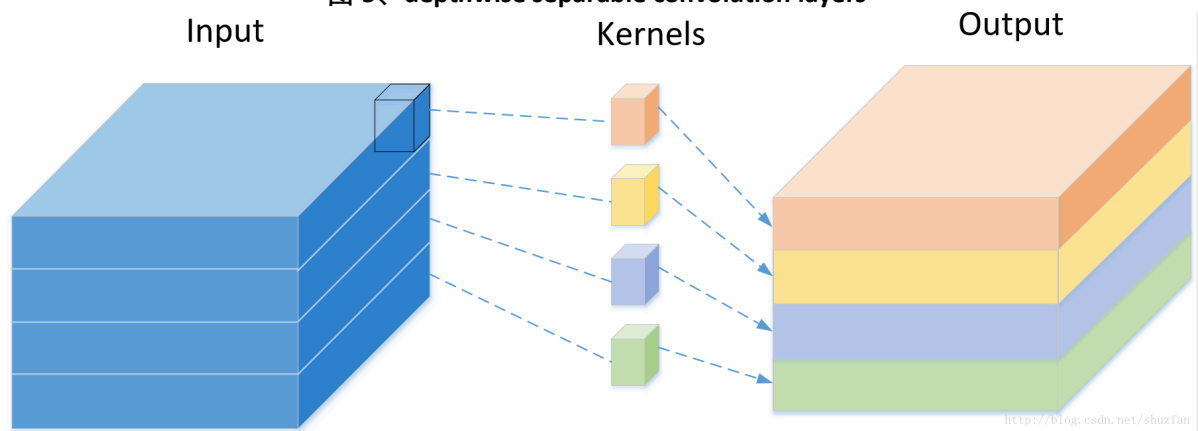


图 6、Xception 的 block7 图示



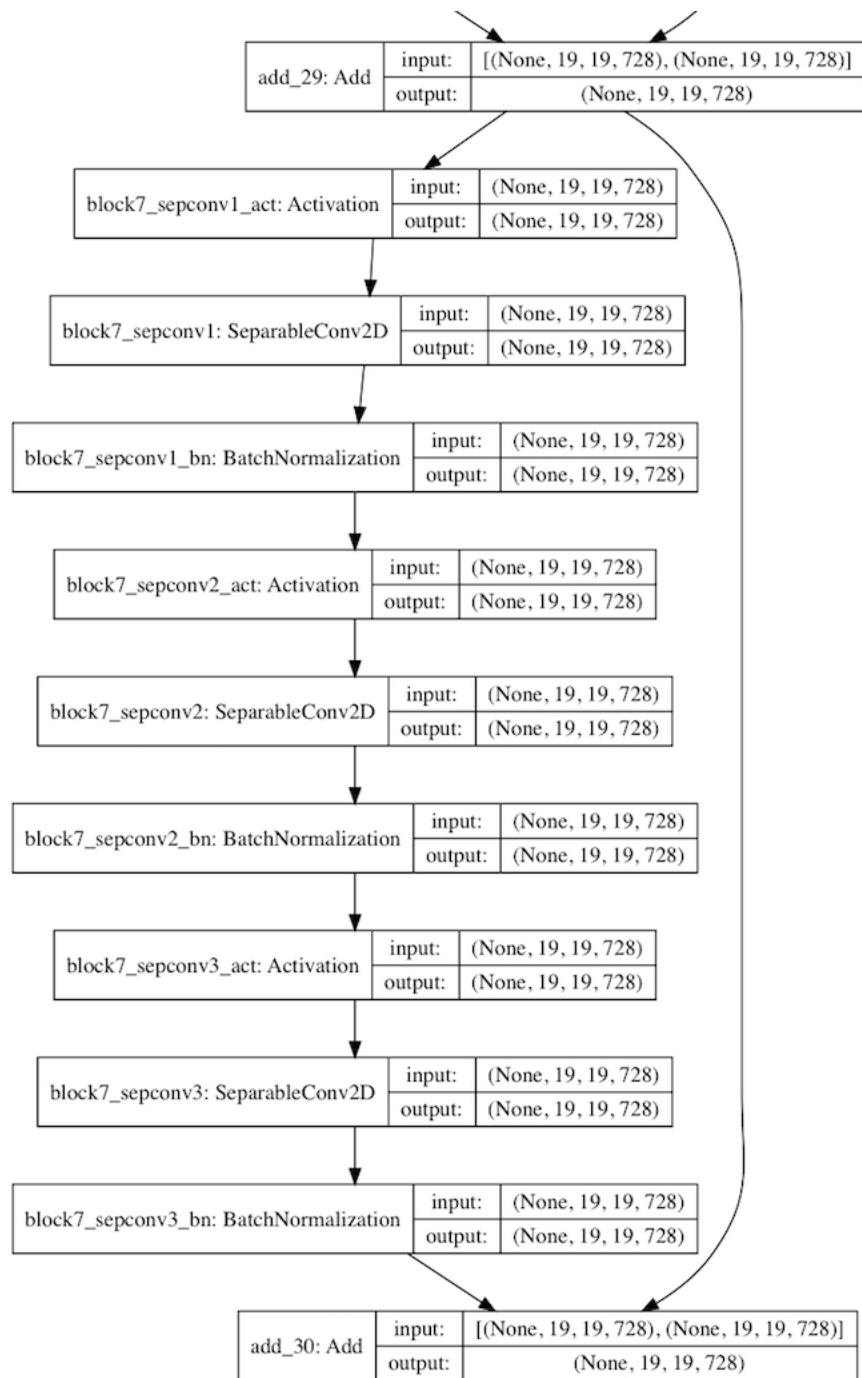
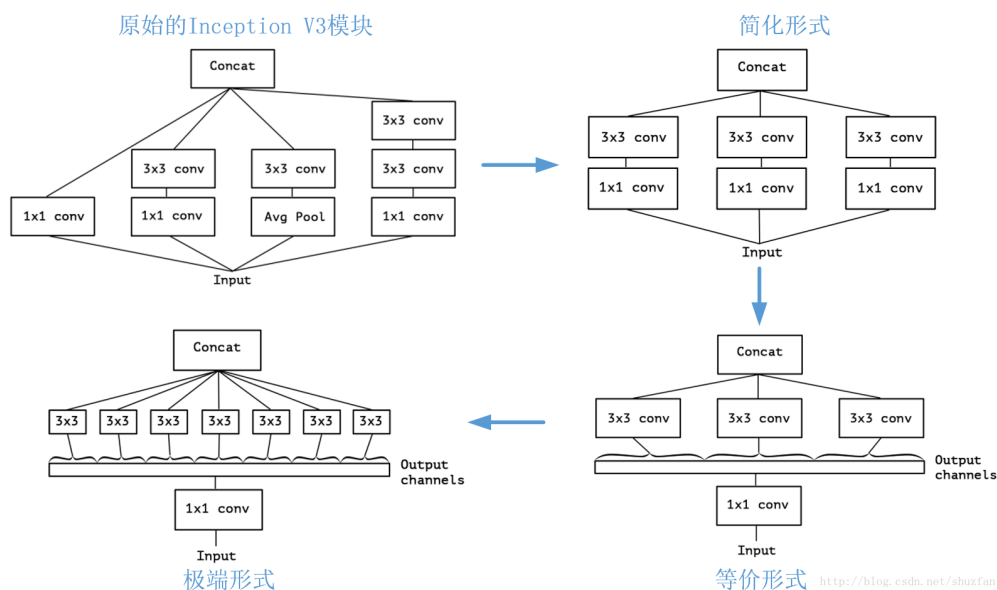


图 7、极端 Inception 发展 (Xception)



同时，这种极端的架构方案使得 Xception 更容易理解和搭建。

## 2.4、基准模型

以 Kaggle 的 Public LeaderBoard 排名 10%作为基准，也就是在测试集上的  $\text{logloss}=0.06127$  (这是当前 Public LeaderBoard 上排名为 131/1314 的分数)。

我在此将会沿用该  $\text{logloss}$  算法评估我的分类器性能，期望是使算法模型能在 kaggle 的 Public LeaderBoard 排名 10%以内，即  $\text{logloss}<0.06127$

## 3、方法

### 3.1、数据预处理

#### 3.1.1 异常数据处理

使用预训练网络 ResNet50 (在 imagenet [5]数据集上训练过)，以及已有的权重，仅对训练数据集中已经标记的猫或狗的图片进行预测，而不对测试数据集进行预测。

在 imagenet 中，猫的种类有 7 种 (分别标记为：['n02123045', 'n02123159', 'n02123394', 'n02123597', 'n02124075', 'n02125311', 'n02127052'])，而狗的标记有 118 种 (此处不予列举)，那么我们预测出的结果的 Top\_x (x 可以是 3, 5, 10, 或者 100 等)，如果没有出现对应的猫或者狗，我们就将图片打印出来，再进行人工辨别。

最终，对于猫的异常图片识别，使用了 Top100，在 Top100 里，仍有 49 张图片预测没有出现猫：

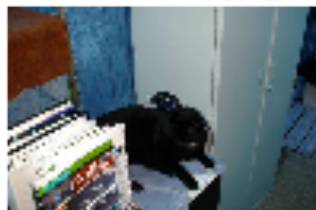
图 8、错误标记的猫 (举例)



cat.10700.jpg



cat.2150.jpg



cat.10636.jpg



cat.5351.jpg



cat.9090.jpg



cat.7968.jpg



cat.7564.jpg



cat.9290.jpg



cat.6345.jpg



cat.2939.jpg



cat.3672.jpg



cat.11399.jpg



cat.10536.jpg



cat.10365.jpg



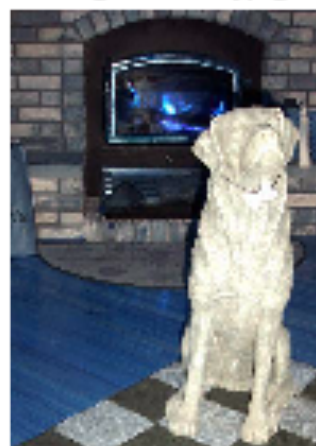
cat.9622.jpg



对于狗的异常图片识别，使用了 Top20，在 Top20 里，仍有 20 张图片预测没有出现：

图 9、错误标记的狗（举例）

dog.7706.jpg



dog.9681.jpg



dog.11299.jpg



dog.8736.jpg

Adopted

dog.5336.jpg



dog.1043.jpg



dog.12155.jpg



dog.11437.jpg



dog.4367.jpg

YAHOO! MAIL

dog.1259.jpg



dog.5604.jpg



dog.10225.jpg



这些判定异常的 49 张猫的图片中，再根据肉眼判断，最终有 15 张得以保留，其余 34 张删除。

这些判定异常的 40 张狗的图片中，再根据肉眼判断，最终有 7 张得以保留，其余 33 张删除。

总共删除训练数据 67 张图片。

### 3.1.2 图片输入处理

由于在项目中是用 Xception 及其在 imagenet 的预训练权重提取图片特征，而 Xception 输入的图片尺寸需要统一为 (299,299,3)，将数据从(0, 255)缩放到 (-1, 1)，这里使用 keras 模块中的自带的 `keras.applications.xception.preprocess_input` 进行输入数据的预处理。

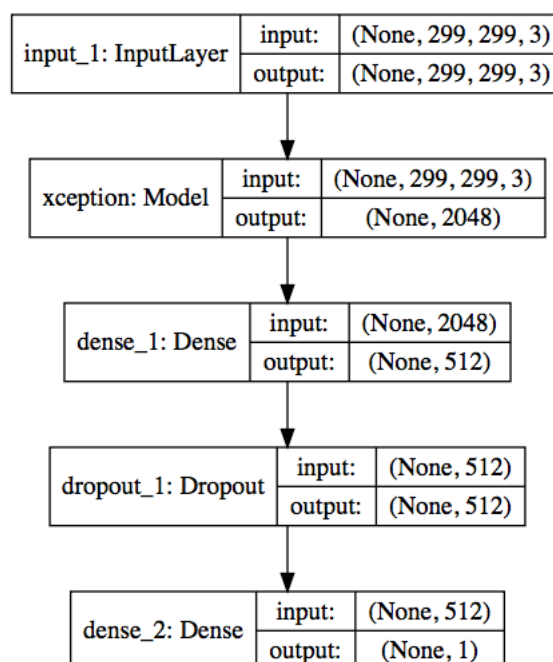
## 3.2、执行过程

### 3.2.1 整体方案

整体模型搭建如图 10，具体如下：

- A) 先将图片预处理为维度：(299,299,3)；
- B) 然后使用一个预训练的 Xception 模型（也使用其在 imagenet 上的预训练权重），对图片进行特征的提取；
- C) 使用 GlobalAveragePooling2D 进行池化，是的输出特征的维度变为 (nb\_samples, 2048)；
- D) 全连接层，激活函数为'relu'，加快收敛速度，输出维度为 (nb\_samples, 512)；
- E) Doupout 层，防止过拟合；
- F) 全连接层，激活函数为'sigmoid'，输出维度为 (nb\_samples, 1)，也就是每个图片的输出都是 (0,1) 区间的一个数，就是进行分类。

图 10、图片分类算法模型



### 3.2.2 特征提取

因为 Xception 的权重在项目中不进行调整，而且 Xception 的参数多，层数也多。那么在训练过程中，就使用 Xception 模型将图片的特征提取出来，再将这些特征提供给后面的神经元进行训练。

又因为可能会多次使用这些特征用于训练，因此将提取的特征保存为 h5 文件，然后就可以随时获取这些特征并进行训练了。

另外，对于 Xception 模型的输出，如果使用 Flatten 输出，维度是：(samples,204800)，这个比较大，此处不太必要。最终选择池化层 GlobalAveragePooling2D 进行输出并保存为 h5 文件。

### 3.2.3 分类模型及训练

读取由 Xception 提取的特征后，在上面应用：Doupout 层，Dense 层，Doupout 层，Dense 层最后一层使用激活函数'sigmoid'，输出 (0, 1) 区间的单个值，正好用于分类及分类自信度的表达。

使用随机梯度下降法进行训练，损失函数就是交叉商损失函数。

同时，对于整个训练集，切割其中的 20%作为验证数据集，剩下的 80 才是真正的训练集。

## 3.3、完善

对训练数据集图片进行数据增强：



调整亮度，对 RGB 像素先归一化（除以 255，使其处于[0, 1]区间）后，使用指数型放大公式，具体如下：

$$A' = (A/255)^{1.0/\gamma} * 255$$

其中，A 为原图像素的 RGB 值，A' 为变更后的 RGB 值。 $\gamma$  为放大指数，取值 0.5-2 之间。

当  $\gamma$  取值在(0.5, 1) 区间时，图片的亮度降低，但对比度增加；

当  $\gamma$  取值在(1,2) 区间时，图片的亮度增加，但对比度降低；

图 11、图片亮度处理( $\gamma$  分别取值 0.5 和 2)



最终在增强的过程中，随机不重复抽取 40% 图片改变亮度（随便变量或者变暗）， $\gamma$  取值在 (0.75, 1) 和 (1, 1.5) 两个区间。

所有抽取到的图片会保留原图，但是会复制一个副本对其进行改动。

## 4、结果

### 4.1、模型的评价与验证

对于卷积网络模型，可以发现，ResNet 的出现使得神经网络的深度可以做得更多层了，可以提取更抽象的特征，而 Inception 是为了更宽的神经网络提取更多特征而设计的。Xception 融合了两者的优点，而且，算法及参数也不比 Inception 复杂。

删除 67 张异常标记的图片后，进行训练（训练集取 20% 作为验证集），最终在测试集上得到的损失函数为：

$$\text{logloss} = 0.04119$$

在此基础上，对 40% 图片进行数据增强（复制副本进行亮度调节），则损失函数可以做到：

$$\text{logloss} = 0.04100$$

这两个损失函数的训练条件不一致，因为数据增加后，收敛速度变快，也更容易过拟合，因此减少了训练次数（epoch）：原始数据训练时，epoch=25；数据增前后，epoch=15。其余参数未变。

由此可见，不论是在训练集还是在测试集上，模型都取得比较好的成绩。

## 4.2、合理性分析

项目之初设定的基准要求是  $\logloss < 0.06127$ ，现在使用单个 Xception 模型就能以较大的幅度超越这个基准，可以说 Xception 对于此类图片特征的提取是高效的，而全连接的分类算法模型也是有效的。

## 5、项目结论

### 5.1、结果可视化

如图 12，图 13，该模型能以较快的速度收敛，而且在训练集和验证集上实现较好的成绩（精度大于 99%，损失函数小于 0.02）

图 12、训练集精度

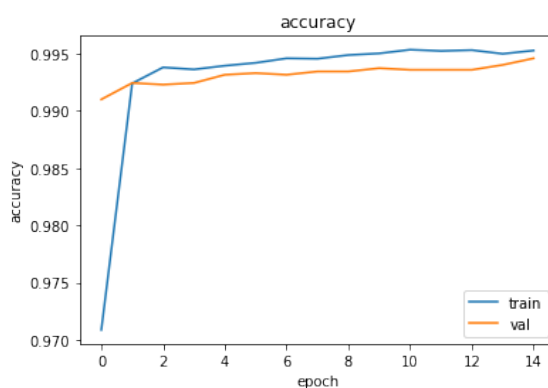
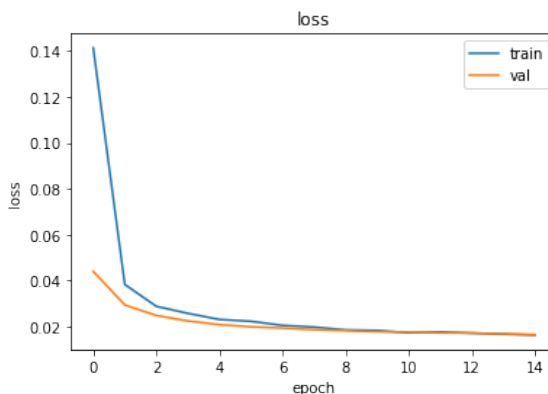
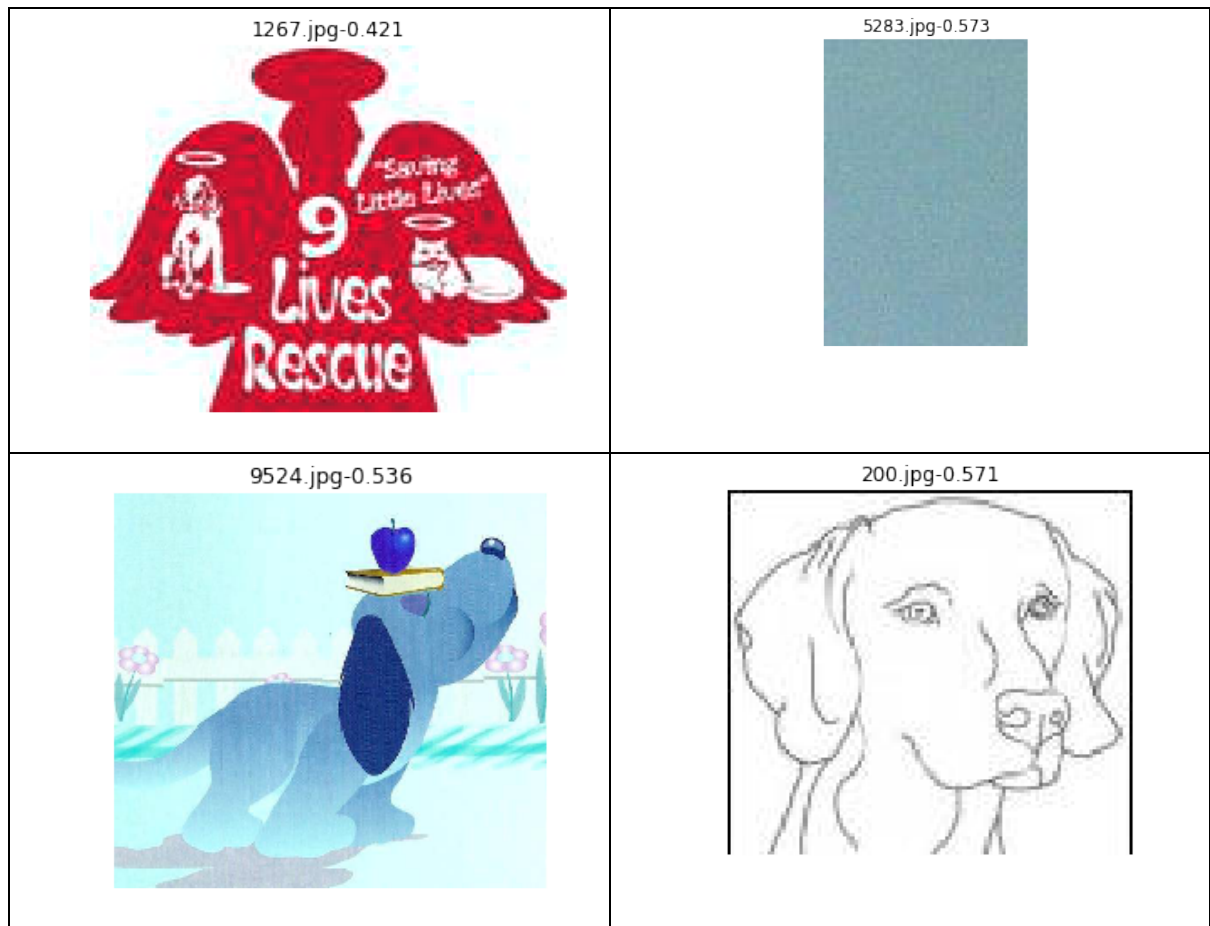


图 13、训练集损失函数



在测试集中，获取所有预测值在区间 (0.4, 0.6) 的图片，也就是算法难以判断分类的：

图 14、测试集难辨图片（举例）



由此可见，测试集也存在异常图片。若有必要，也应该对这些异常图片进行处理。

## 5.2、对项目的思考

A) 如果是在竞赛中，使用预训练模型提取特征，也就是使用 Xception 模型及其在 Imagenet 上训练好的权重，其实是一种舞弊行为，因为这算是将整个 Imagenet 图集也都当作是训练集了，也就是引用了外部的训练集，而且因为所用的训练集不一致，这样就没办法评估哪个模型更优了；

B) 但如果是在应用层面的开发，使用预训练的权重，可以降低收集训练数据集的数量、难度与时间，也能极大的减少训练时间；

C) 图片增强可以用较低的成本（降低收集数据的成本）实现对模型的有效训练，但可以有更优的数据增强办法，比如用一种人工智能网络，生成数据，给另一种模型训练，以此验证模型的可靠性。

## 5.3、需要作出的改进

可以使用不同模型，如 ResNet50 或 Inception V3 提取的特征并于 Xception 提取的特征进行合并，然后再训练。

引用：

[1] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385

[2] C.Szegedy,V.Vanhoudke,S.Ioffe,J.Shlens,andZ.Wojna. Rethinking the inception architecture for computer vision.arXiv preprint arXiv:1512.00567.

[3] Francois Chollet. Xception: Deep Learning with Depthwise Separable Convolutions.arXiv: 1610.02357



[4] C. Szegedy, S. Ioffe, and V. Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. arXiv preprint arXiv:1602.07261.

[5] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. ImageNet large scale visual recognition challenge. 2014.