

机器学习工程师纳米学位毕业项目  
猫狗大战

蔡炜

2019 年 8 月 7 日

# 1 定义

## 1.1 项目概述

本项目出自于 kaggle 的一个娱乐型竞赛项目——Dogs vs Cats Redux: Kernels Edition，是一个计算机视觉领域中典型的图像分类问题，训练集包含了 25000 张猫狗图片，其中各有一半被标记为猫或狗，项目要求竞赛者训练一个模型来对测试集中的 12500 张猫狗图片进行分类识别。

项目模型的构建主要基于卷积神经网络（Convolutional Neural Networks, CNN），这是一类包含卷积数学计算且具有深度结构的神经网络<sup>[1]</sup>，是深度学习的代表性算法之一，广泛应用于计算机视觉领域。

经过学习 udacity 的机器学习课程，本人深刻体会到计算机视觉领域的乐趣，该领域也是人工智能的热点，在现实生活中有广泛应用。

## 1.2 问题陈述

训练集图片已打好标签，标签类别仅有两种，因此该项目属于监督学习（Supervised Learning）中的二分类问题。

不难发现，训练集中的图像画质参差不齐，猫狗种类繁多，图像背景复杂，这些因素都直接影响到模型的分类预测。此外，仔细观察后还能发现，其中有些图片根本不属于猫狗，这会误导我们的模型，因此我们首先要对训练集中的图片进行“清洗”，筛选掉这些“不友好”因素。

接着，我们可以使用一些知名的预训练模型如 Xception、ResNet、

Inception 等对图像进行特征提取，从而抽象出图像相邻像素点之间不同级别的组合信息，这会大大有助于模型的学习与训练。

最后，我们将提取出的图像特征输入到经过训练的模型中，经过分类函数得到分类预测结果。由于是二分类，使用 sigmoid 分类函数映射到(0, 1)区间即可，接近 0 代表分类结果是猫，接近 1 代表分类结果是狗。

### 1.3 评价指标

本项目的评价指标采用对数损失 (LogLoss)，也称交叉熵 (Cross Entropy) 损失，它起源于信息论，主要用于度量两个概率分布间的差异性信息。使用交叉熵作为损失函数有一个好处就是使用如 sigmoid 激活函数在梯度下降时能避免像使用均方误差等其它损失函数时存在学习速降低的问题，因为学习速率可以被输出的误差所控制<sup>[2]</sup>。对于二分类问题，LogLoss 的计算公式如下：

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

其中  $N$  为测试集样本数， $y_i$  代表第  $i$  个样本的真实标签，如果是狗则  $y_i = 1$ ，否则  $y_i = 0$ ， $\hat{y}_i$  表示模型预测第  $i$  个样本为狗的概率。

由 LogLoss 的计算公式可以看出，模型预测样本对应其真实标签的概率要尽可能大，即对于真实标签为猫样本，模型输出的概率值要尽可能接近 0，而对于真实标签为狗的样本，模型输出的概率值要尽可能接近 1。

## 2 分析

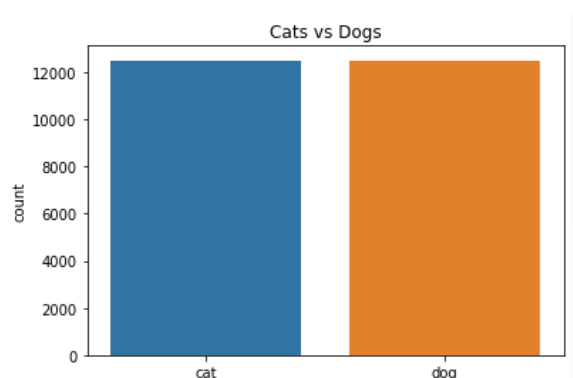
### 2.1 数据研究与可视化

本项目数据集由 kaggle 提供，分为训练集和测试集，分别包含 25000 张和 12500 张猫狗图片，其中，训练集的图片通过文件名称来标记其所属类别是猫还是狗。

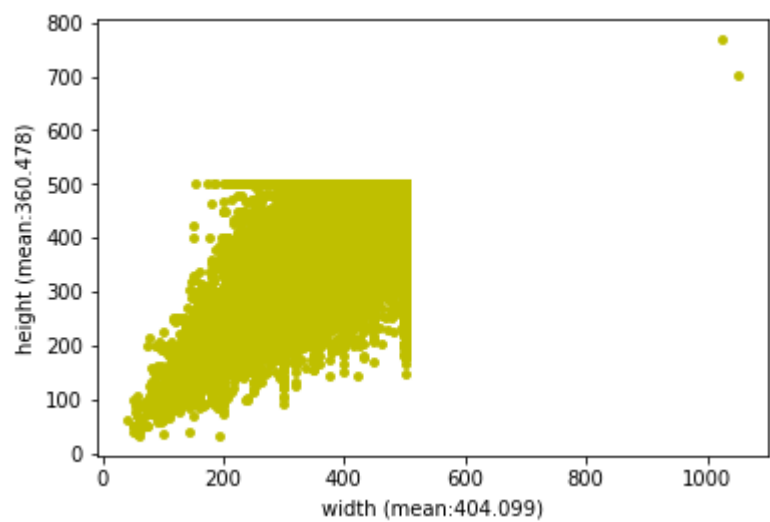


训练集图片示例

可以看到，图片中猫狗姿态各异，背景复杂多样，这为我们模型的识别增加了一定难度。另外，根据图片文件名称来统计，训练集中猫狗图片分布各占一半，如下所示：



再来研究下训练集中图片的尺寸分布，可以知道图片宽度范围在 42 至 1050 之间，平均值在 404 左右，高度范围在 32 至 768 之间，平均值在 360 左右。根据四分位值，得知大部分图片的宽高在 300 以上、500 以下。



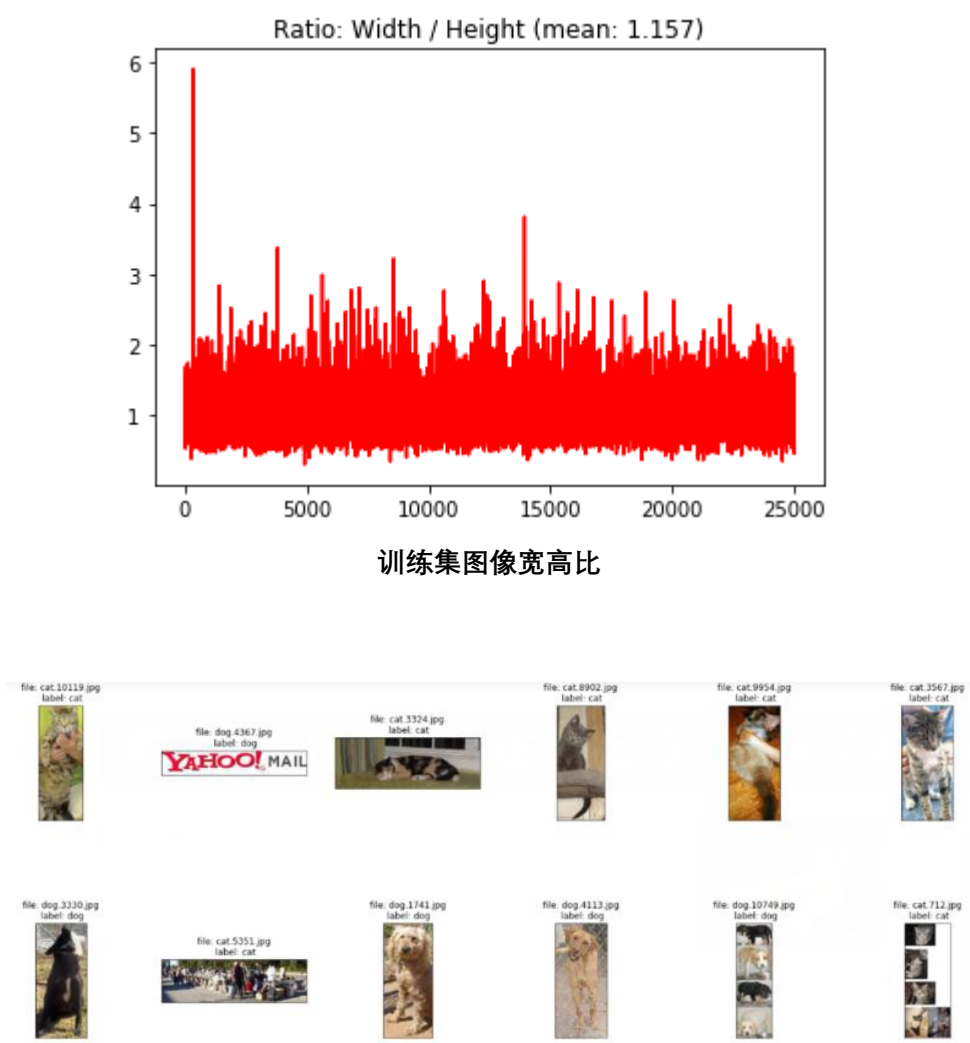
训练集图像尺寸分布散点图

	width	height
count	25000.00000	25000.00000
mean	404.09904	360.47808
std	109.03793	97.019959
min	42.00000	32.00000
25%	323.00000	301.00000
50%	447.00000	374.00000
75%	499.00000	421.00000
max	1050.0000	768.00000

训练集图像尺寸统计特征

接着，我们可以计算图片的宽高比，发现平均值在 1.157 左右，大部分图片的宽高比超过 0.930，小于 1.337，图像宽高比太小或太大在一定程度上都会影响模型的学习与训练，若某张图像的宽高比与其它图

像偏离得太远，则可以认为它是异常数据，需要对其进行清理。



训练集中宽高比异常的部分图像

进一步探索，我们会发现，部分图片的标注错误，即猫的图片被标注为狗（或者相反），还有部分图片既不是猫也不是狗，或者在某些图片中，同时存在猫和狗，又或者图片中有人或者其它复杂背景，而猫狗在图片中所占比例太小，这样会导致模型在提取特征时主要学习到了非猫狗的其它因素的特征，因此我们需要对这些异常进行处理。



dog.4334.jpg (猫的图片被标注为狗)



cat.2159.jpg (图片中同时存在猫狗)



dog.2614.jpg (非猫狗图片)



dog.1194.jpg (非猫狗图片)



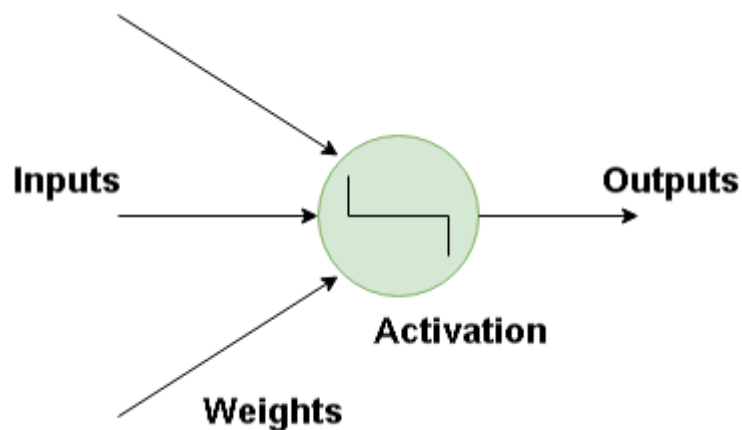
dog.2877.jpg (一只卡通猫被标注为狗)

## 2.2 算法和技术

### 2.2.1 深度学习与神经网络

深度学习(DL, Deep Learning)是机器学习(ML, Machine Learning)领域中一个研究方向，它使用包含复杂结构的多个处理层对数据进行高层抽象的算法，这些处理层被称为神经网络(Neural Networks) <sup>[3]</sup>。

神经网络是一组模仿生物大脑神经元结构而设计的算法，它有许多隐藏层共同组成，每层由许多计算单元构成，这些计算单元类似于大脑神经元，它们会对输入信号进行加权计算并且相互交互，而后被激活最终产生输出信号 <sup>[4]</sup>。



神经元处理信号



在深度学习中，神经网络主要分为 3 层：

1) 输入层

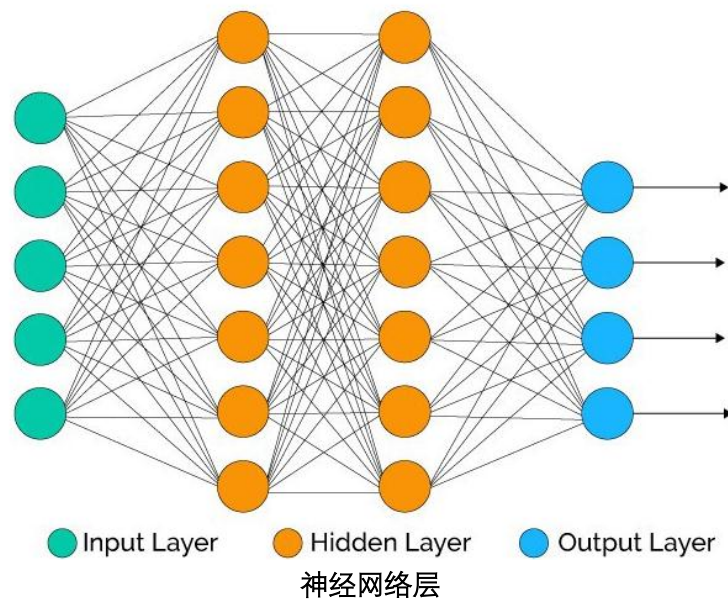
接收输入数据，神经元的数量通常对应于数据的特征数。

2) 隐层

负责对数据进行线性加权计算并激活（非线性计算），一张神经网络可以有多个隐层，每个隐层有由多个神经元组成。

3) 输出层

基于隐层的计算结果产生输出，作为预测结果。



可以看出，深度学习是学习数据的内在规律和表示层次，通过多层处理，将初始特征表示转化为高度抽象特征后利用模型完成学习任务，因此，通常将深度学习纳入表征学习（Representation Learning）<sup>[5]</sup> 范畴。

## 2.2.2 卷积神经网络

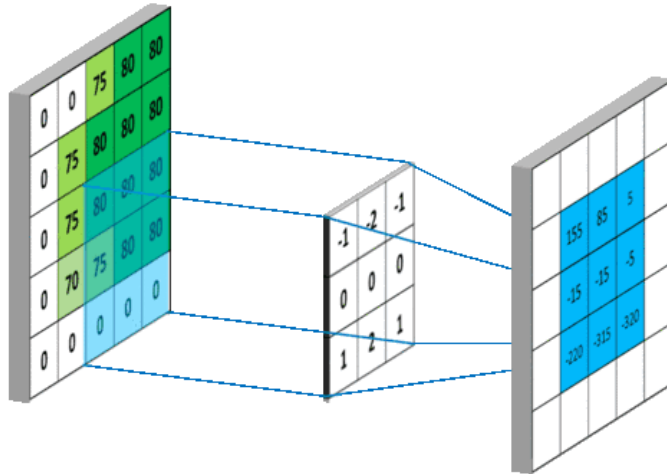
卷积神经网络 (Convolutional Neural Networks, CNN) 是一类包含卷积数学计算且具有深度结构的前馈神经网络 (Feedforward Neural Networks), 是深度学习的代表性算法之一。

在传统神经网络中, 每层的每个神经元都与下层的每个神经元相连, 这种连接关系称为全连接 (Full Connected)。以处理图像数据为例, 那么输入就是每个单位像素, 假如一张黑白图像尺寸是  $10^3 \times 10^3$ , 则输入层需要  $10^6$  个结点, 假如后面隐层有  $10^4$  个结点, 那么需要的权重系数就达到了  $10^{10}$  个。若隐层的结点数量更多, 那么计算量就会更大, 而且参数过多还容易导致过拟合。卷积神经网络则是通过局部感知 (local awareness) 也称稀疏交互 (sparse interactions) 来解决这一问题, 也就是每个神经元只与上一层的部分神经元相连交互, 获得局部信息, 最终在高层将所有局部信息综合起来得到全局信息, 这是受启发于生物视觉系统, 对外界的认知从局部到全局<sup>[6]</sup>。

卷积神经网络的结构与普通神经网络结构相似, 不同的是, 其隐层通常包含卷积层 (convolutional layer) 和池化层 (pooling layer) :

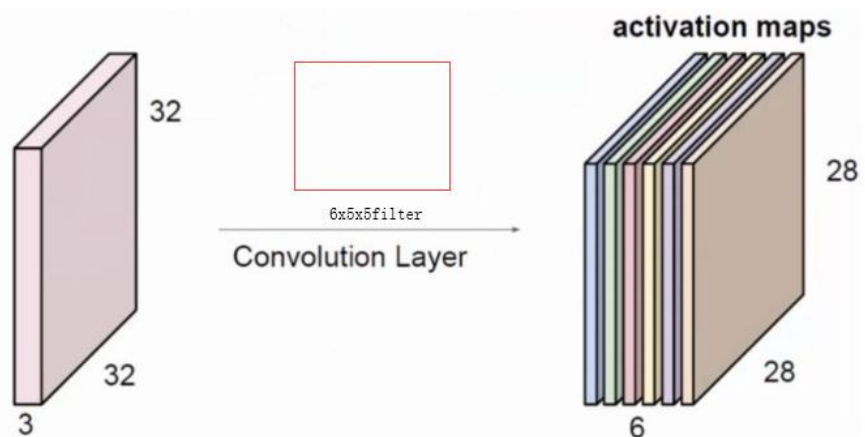
#### 1) 卷积层

卷积层采用一个或多个滤波器 (filter) 也称卷积核 (kernel) 对图像的像素矩阵进行卷积运算, 给一个卷积核设定大小尺寸和步长后, 通过卷积核在像素矩阵上不断扫描移动来完成整个运算过程, 每移动完一步便完成一次计算, 对应到隐层的一个结点, 从而将原始图像尺寸映射到另一个尺寸, 而卷积核的数量则将图像通道数映射到另一个通道数。



5x5 原始图像尺寸被 3x3 尺寸卷积核映射到 3x3 尺寸

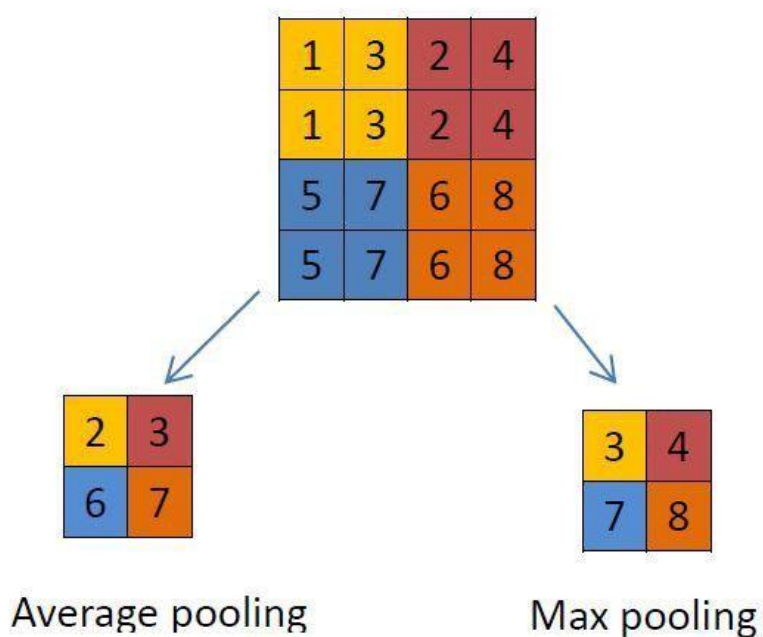
可以看出，经过相同卷积核局部感知的隐层结点的权重系数是相同的，这被称为权值共享 (weight sharing) 或参数共享 (parameter sharing)，这样设计是因为图片的底层特征与特征在图片中的位置无关，经过一个卷积核局部感知后获得的是一组特征映射，因此通常需要多个卷积核来获得多组特征，最终在上层将它们组合起来获得全局视野<sup>[7]</sup>。



32x32 尺寸的 3 通道图像被 6 个 5x5 尺寸的卷积核映射为 28x28x6 的特征图

## 2) 池化层

池化层主要起到降维作用，通常有最大池化（max pooling）和平均池化（average pooling）两种方式，最大池化是直接去 filter 区域内最大值作为映射输出，平均池化则是取平均值。



4x4 矩阵经过 2x2 filter 的两种池化方式得到的结果

### 2.2.3 技术实现

本项目实现采用开源机器学习库 Keras，其作为对 Tensorflow 框架的封装，使得开发过程更加便捷。此外，由于数据量和计算量较大，因此借助于亚马逊的 EC2（Amazon Elastic Compute Cloud，Amazon EC2）服务平台进行 GPU 加速计算。

从头开始构建一个 CNN 网络来训练并进行调参优化非常复杂

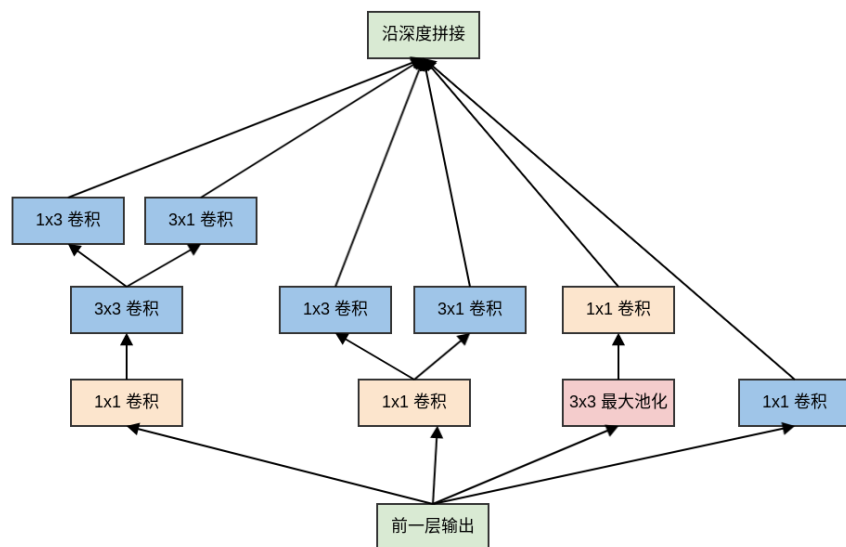
耗时，而在 ImageNet 中，已有不少优秀的开源预训练模型，因此项目采用迁移学习(Transfer Learning)的方式来进行模型构建与训练，从而简化训练过程和提高效果。预训练模型主要负责对图像进行特征提取，而后添加自定义的少数基层全连接层来对特征进行分类识别。

本项目使用了 4 种开源 CNN 模型，分别是 InceptionV3、Xception、InceptionResNetV2 以及 NASNetLarge：

#### 1) InceptionV3

Inception 网络目前有 4 个版本，本项目用到的是 V3，这里有必要先简单介绍下 Inception 网络。在 Inception 出现之前，大部分 CNN 结构为了提取到更上层的特征，偏向于把卷积层堆叠得越来越多，这便造成网络越来越深，使得网络越来越复杂，参数越来越多，进而导致网络容易出现过拟合，同时增加计算量。

Inception 网络则考虑通过多种卷积核的并行计算来扩展网络宽度，同时，其参考生物神经连接的稀疏性，构造了稀疏连接，然后将这些输出沿深度进行拼接，这种结构被称为“Inception”的模块，也是 Inception 网络的最大特点<sup>[8]</sup>。



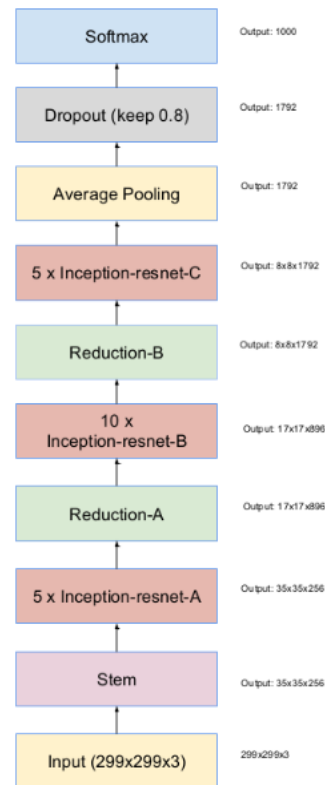
InceptionV3 中的 Inception 模块

相比较于之前的 Inception 网络版本，V3 版本强调的是网络结构的设计，其最终总结了以下设计原则：

- 在缩小特征映射（feature map）尺寸的同时，应该增加特征映射的通道数
- 通常在  $n \times n$  卷积之前，先利用  $1 \times 1$  卷积来降低输入维度
- 增加网络的宽度或者深度都可以提高网络的泛化能力，因此计算资源需要在网络的深度和宽度之间取得平衡

InceptionV3 共有 23,851,784 个参数，在 ImageNet 上的 Top-1 准确率为 77.9%，Top-5 准确率为 93.7%。

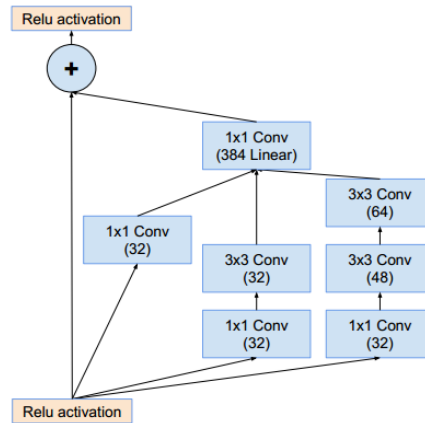
## 2) InceptionResNetV2



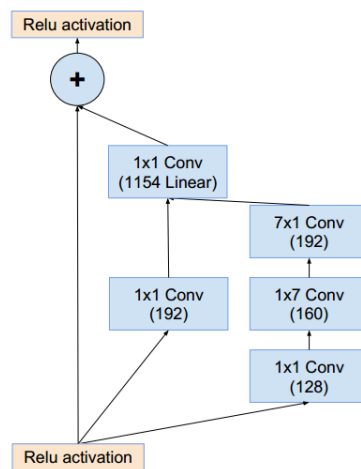
**InceptionResNetV2 整体结构**

InceptionResNet 目前有 2 个版本，本项目使用的是 V2 版本，其特点是在 Inception 网络的基础上结合了 ResNet 残差连接的特性，结果就是在 Inception 模块中，利用残差连接替代了池化运算。

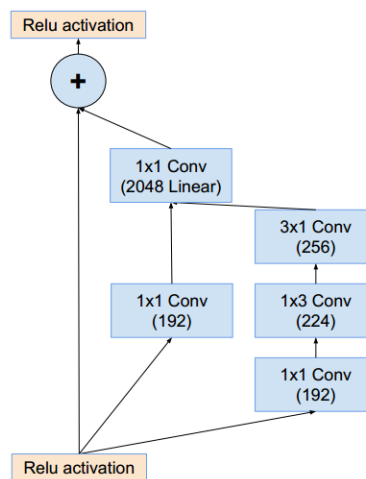
InceptionResNet 和 InceptionV4 在同一篇论文中被提出，引入了 Inception-A、Inception-B、Inception-C 三个模块，同时还引入了用于缩减 feature map 的宽、高专用的“缩减块”(Reduction Block)。



**Inception-ResNet-A 模块**



**Inception-ResNet-B 模块**

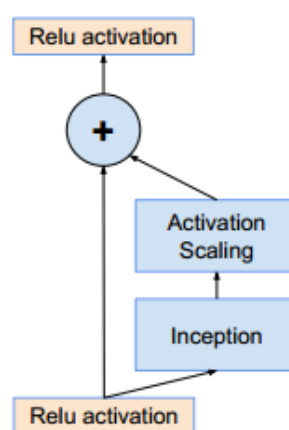


**Inception-ResNet-C 模块**

研究人员发现一个问题，如果滤波器数量超过 1000，则残差网络开始出现不稳定的情况，同时网络会在训练过程早期出现“死亡”，即经过成千上万次迭代之后，在平均池



化之前的层开始只生成 0。于是 Inception-ResNet 的解决方案是将残差模块添加到激活层（Activation）之前，对其进行缩放从而稳定训练，而降低学习率或者增加额外的 BN 都无法避免这种状况，这就是 InceptionResNet 中的 Inception-A、Inception-B 和 Inception-C 为何如此设计的原因<sup>[8]</sup>。



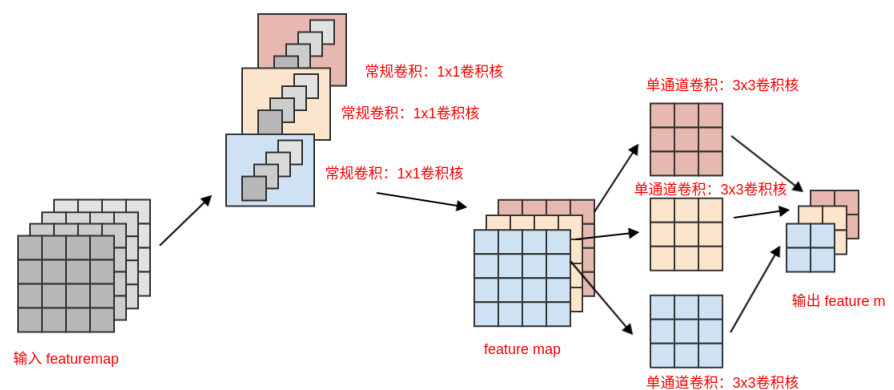
Inception 模块置于前后两个 Relu 激活层之间，并使用不带激活函数的线性 1x1 卷积层（Conv）来对残差进行线性缩放

InceptionResNetV2 在基于单类图像的 ILSVRC 2012 图像分类基准上的 Top-1 准确率为 80.4%，Top-5 准确率为 95.3%。

### 3) Xception

在 Inception 网络中，通常的做法是首先通过一组 1x1 卷积来查看跨通道的相关性，将输入数据映射到比原始输入空间小的三个或者四个独立空间；接着通过常规的 3x3 或者 5x5 卷积将所有的相关性（即包含跨通道相关性和空间相关性）映射到这些较小的三维空间中。

Xception 则将这一思想发挥到极致——其在使用  $1 \times 1$  卷积来映射跨通道相关性后，只分别映射每个输出通道的空间相关性，而非所有输出通道的空间相关性，从而将跨通道相关性和空间相关性解耦，因此该网络被称作 Xception，即 Extreme Inception，这便是其名字的由来，其中的 Inception 模块被称为 Xception 块。



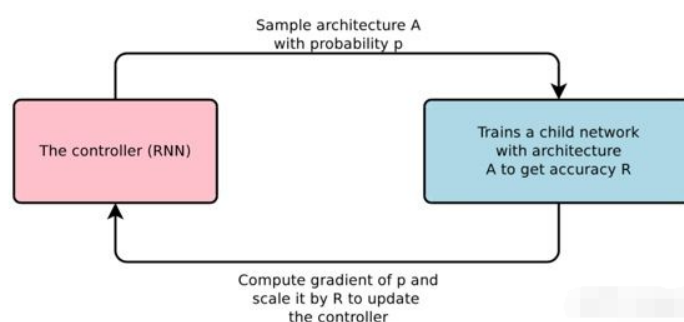
#### **Xception 将跨通道相关性和空间相关性解耦**

Xception 参数共有 22855952 个，在 ImageNet 上 Top-1 准确率为 79.0%，Top-5 准确率为 94.5%。

#### **4) NASNetLarge**

在 NASNet 出现之前，其它 CNN 的所有结构均由人手动设计，这就完全将网络结构依赖在人为经验上了。而 NASNet 的特点就是使用强化学习（Reinforcement Learning）来学习一个 CNN，其通过最大化验证集上的精度期望来优化网络，NAS 是 Neural Architecture Search 的缩写，说明其是使用强化学习来寻找最优网络。现在的神经网络一般采用堆叠 block 的方式搭建而成，这种堆叠的超参数可以通过一个序列来表示，而这种序列的表示方式正是 RNN 所擅长

的工作，于是 NAS 会使用一个 RNN 构成的控制器以一定概率随机采样一个网络结构，接着训练这个网络并得到其在验证集上的精度，然后再使用这个精度更新控制器的参数，如此循环执行直到模型收敛。



NAS 算法流程











NASNetLarge 在 ImageNet 上的 Top-1 准确率为 82.7%，Top-5 准确率为 96.2%，是当前最好的开源模型之一。

使用以上预训练模型的通常做法是提取瓶颈特征或微调，而本项目的做法是把去掉尾部的 Xception 和 InceptionV3 沿最后一个轴进行拼接生成一个模型，同时也把去掉尾部的 InceptionResNetV2 和 NASNetLarge 沿最后一个轴进行拼接生成另一个模型，达到融合它们提取特征的效果。由于 Xception 和 InceptionV3 的默认输入图像尺寸都是 299x299，因此将两者进行拼接；而对于另一个模型，我的想法是，NASNetLarge 是随机初始化权重然后通过强化学习来构建网络结构的，该方式始终存在一定的不可预估性，即没有手工构建网络的确定性，InceptionResNetV2 中

既存在了残差网络，也弥补了残差网络的不稳定性，相对来说较为稳定，于是将其与 NASNetLarge 进行拼接。详见以下 3.3 “模型构建”。

## 2.3 基准指标

本项目要求模型的 LogLoss 在 kaggle pubic leaderboard 中排入前 10%，该竞赛共有 1314 位参与者，因此需要排位在 131 名以前，即  $\text{LogLoss} < 0.06127$ 。

124	Chase		0.06026	6	3y
125	John Vial		0.06037	18	2y
126	tmuzap		0.06054	21	3y
127	icodingc		0.06068	12	3y
128	Wilson Sun		0.06082	23	2y
129	Jeremy Howard		0.06086	11	3y
130	RaviKiranK		0.06114	35	2y
131	Reziproke		0.06127	4	3y
132	mathieuzaradzki		0.06149	32	2y
133	Mouatez		0.06240	39	3y

kaggle public leaderboard

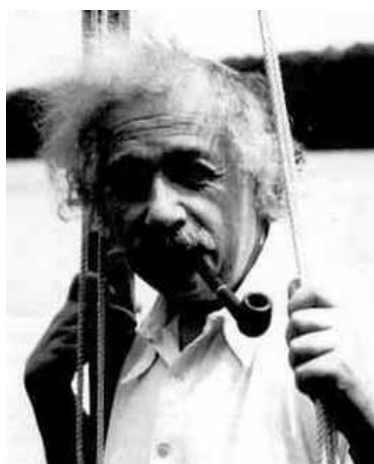
## 3 方法

### 3.1 数据预处理

#### 3.1.1. 异常数据清理

在上述数据探索部分，我们探索到部分图像的宽高比例异常，本项目将宽高比小于上四分位值一半或者大于下四分位值两倍的图片标记为宽高比异常，并且记录下来，作为需要清除的脏数据。

另外，我们也剖析了 kaggle 提供的数据集中存在标注错误等异常，因此，在模型利用这些数据进行训练之前，我们需要清理掉这些异常数据。



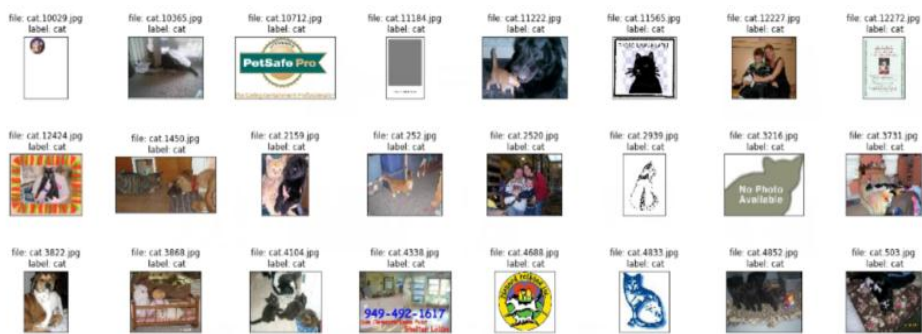
异常数据示例 dog.1773.jpg



异常数据示例 dog.9517.jpg

关键问题来了，训练集总共包含 25000 张图片，我们几乎不可能用肉眼一一分辨出这些异常数据，那么该怎么办呢？

众所周知，ImageNet 项目是一个用于视觉对象识别软件研究的大型可视化数据库，其中超过 1400 万张图像被标注，其中恰恰包含了猫和狗，官方每年举办的大规模视觉识别挑战赛（ILSVRC）诞生了许多优秀的视觉识别模型，因此本项目采用的方法是使用已在 ImageNet 中预训练的模型对训练集中的图片进行分类识别，最终根据模型预测结果是否在 ImageNet 数据集的猫狗类别中来判断是否是异常数据。具体地，本项目使用了 5 种预训练模型，分别是 ResNet50、Xception、InceptionV3、InceptionResNetV2 以及 NASNetLarge，将它们的预测结果取并集，并且加上 kaggle 讨论区中人们已被发现的异常数据，最终作为需要清理的异常数据。



部分异常数据展示

### 3.1.2. 数据集划分

本项目将部分训练集中的图片划分出来用作验证集，用于训练过程中交叉验证，从而评估每个周期训练结果的好坏。划分方法是使用 sklearn 的 train\_test\_split 方法，最终训练集和验证集的比例是 3:1，并

且通过设置参数 stratify，使得在最终的训练集和验证集中，标签分布和原训练集中的分布相同。

### 3.1.3. 常规预处理与数据增强 (Data Augmentation)

项目将图像尺寸分别处理为 299x299 以及 331x331，分别用于构建的两个模型。另外还使用了 4 个预训练模型的预处理方法，从源码可以知道，InceptionV3 和 Xception 的预处理方式都是将图片数据缩放到 $[-1, 1]$ 之间。

接着，项目使用了数据增强，具体包括随机旋转的度数范围设置为 25、随机缩放范围设置为 0.2、剪切强度设置为 0.1、宽度和高度转换范围设置为 0.1 以及使用随机水平翻转，这增加数据样本的同时通常能够提高模型的泛化能力，考虑到本项目使用的 4 种预训练模型对图像处理的默认尺寸不完全同，因此生成了 299x299 和 331x331 两种尺寸的生成器实例，并且批次大小分别为 16 和 32。

## 3.2 特征提取

通常，特征提取是解决图像分类与识别问题的必要步骤，CNN 的卷积核恰是在其中发挥了重要作用，卷积层的各卷积核通过设置不同的权重能够提取出图像的边缘、纹理等甚至其它高层次的特征。

本项目使用了 4 种在 ImageNet 上预训练的模型，分别是 Xception、InceptionV3、NASNetLarge、InceptionResNetV2，将它们最后的全连接层去掉，权重设置为在 ImageNet 上训练的结果。先让图像分别经它们各自的预处理方法进行处理，然后分别对应输入到这些模型中，达到提取图像特征的效果。此外，还为它们都设置了平均池化，用于

压缩提取的特征，防止过拟合。

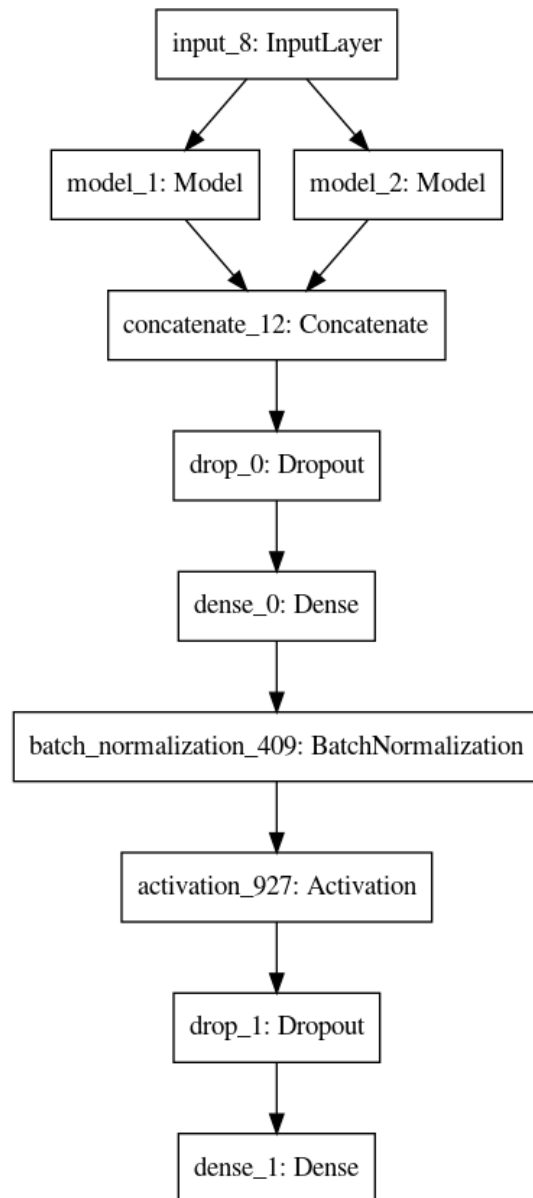
### 3.3 模型构建

将几种预训练模型提取出的特征进行融合，具体方法是使用 Keras 提供的 Concatenate 层，它可以将多个张量按照指定的维度进行拼接，默认是最后一个维度。

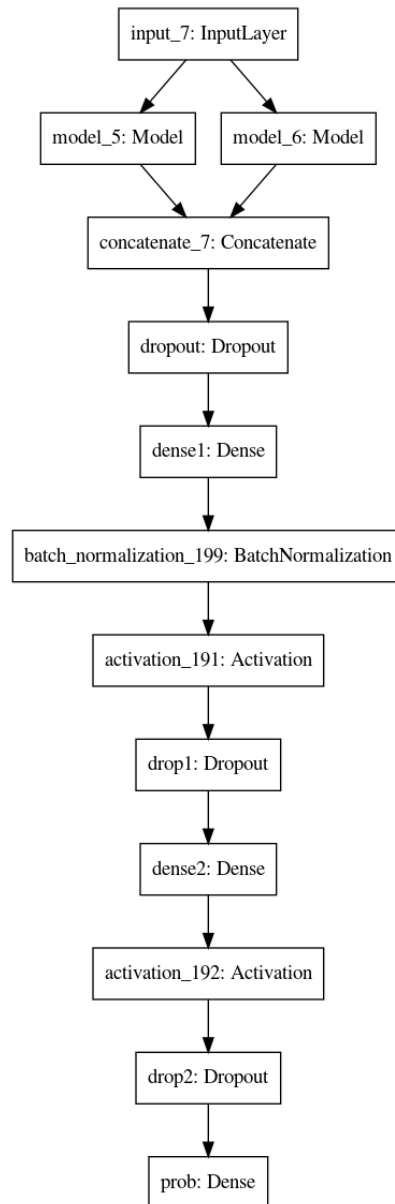
特征融合后，只需再使用少数几层全连接层和丢弃层，激活函数使用 ReLu，在反向传播求误差梯度时，其相对于 sigmoid 等其它激活函数计算量较小，在深层网络中也不容易出现梯度消失。此外，它会将部分输出映射为 0，为网络增加了稀疏因素，减少了参数相互依存的关系，一定程度上缓解了过拟合问题。最后一个全连接层使用 sigmoid 激活函数，它可以将输出映射到(0,1)的区间，从而作为二分类的结果。

由于本项目使用的 4 种预训练模型对图像的预处理方式以及默认输入尺寸不完全相同，其中 NASNetLarge 的默认输入尺寸为 331x331，而 Xception、InceptionV3 和 InceptionResNetV2 为 299x299，本项目将 Xception 和 InceptionV3 这 2 种模型提取的特征进行融合，而将 NASNetLarge 和 InceptionResNetV2 另外 2 种模型提取的特征进行融合，接着分别对两个融合的特征添加少数几层网络，最终构建出 2 种模型。





基于 InceptionResNetV2 和 NASNetLarge 构建的模型

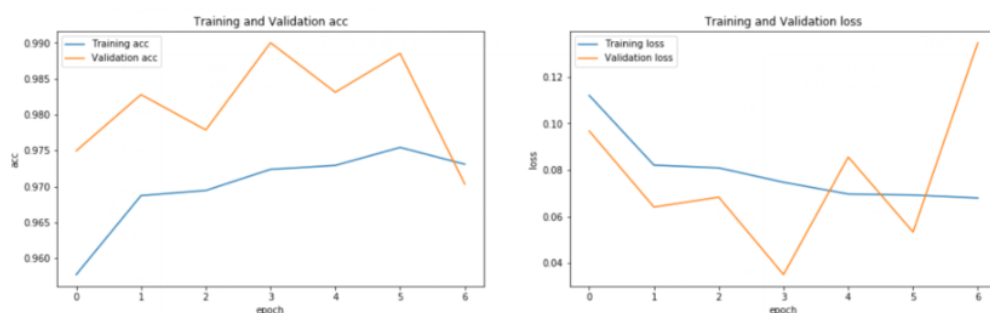


基于 Xception 和 InceptionV3 构建的模型

### 3.4 模型训练与调参改进

- i. 前期，本项目尝试过将图片尺寸处理为 299x259 和 331x287，使得其与训练集图像的平均宽高比 1.157 接近，之所以将宽设置为 299 和 331 是因为使用的预训练模型的默认输入图像尺寸为 299x299 和 331x331，但是发现效果并不理想，训练过程中震荡比较严重，模型在训练集和验证集的 loss 都较大，猜测可能是因为

使用了预训练模型自带的预处理方法，因此处理为它们的默认输入尺寸才比较合适。



模型训练过程中精度和损失抖动

- ii. 模型训练使用的损失函数为二分类交叉熵 `binary_crossentropy`，与该二分类问题匹配，度量为精度 `accuracy`，批次大小 `batch size` 通常设置为 2 的幂指数，这里选择了 16 和 32，太小容易震荡并且出现过拟合或者陷入局部最优，太大则每一批次所需的时间更长。模型总共训练 10 个周期，根据 `val_loss`，训练过程会自动保存最优权重，此外，还设置了早期停止 `EarlyStopping`，在连续 3 个周期模型表现没有改进时便会停止训练。
- iii. 批次正则化的使用（Batch Normalization）  
起初，在网络的最后几层仅使用了 Dropout，后来考虑到既然输入数据已经过预训练模型那么多层的映射，那么之后的分布多少会出现偏离甚至偏离严重，于是在模型拼接后以及第一个 Dense 层之后都加入了 Batch Normalization，其能将偏离的分布拉回到均值为 0 方差为 1 的标准正太分布，这样能避免梯度消失，而且加快收敛，最终发现效果相比仅使用 Dropout 更佳。
- iv. 优化器优先使用 Adam，它是一种自适应学习率的方法，在

RMSprop 的基础上加入了偏差校正和动量，收敛速度较快。但是，后来尝试了其它优化器如 Nadam、RMSprop 以及 SGD，其中 SGD 是使用带动量的模式，这样能加快收敛，同时还设置了学习率衰减，发现也可能达到相当甚至更好的效果，于是在这 4 种优化器中分别设置不同参数进行了比较，具体结果详见以下 4.2“结果分析”。

## 4. 结果

### 4.1. 模型评价与验证

项目将 Xception 和 InceptionV3 进行融合，构建出输入图像尺寸是 299x299 的一个模型，以及将 InceptionResNetV2 和 NASNetLarge 进行融合，构建出输入图像尺寸是 331x331 的另一个模型，两个模型分别进行训练后对测试集进行预测，LogLoss 分别为 0.04313 和 0.04170。

submission\_299.csv  
16 days ago by cwcai

0.04313

Xception&InceptionV3(SGD optimizer)

#### 基于 Xception 和 InceptionV3 的模型在测试集上的表现

submission\_331.csv  
10 days ago by cwcai

0.04170

NASNetLarge&InceptionResNetV2(SGD optimizer)

#### InceptionResNetV2 和 NASNetLarge 的模型在测试集上的表现

以上结果均满足  $\text{LogLoss} < 0.06127$ ，符合项目通关要求。

### 4.2. 结果分析

本项目前期做了几组基础实验进行比较，用于确定后续训练过程使用的图片尺寸，批次大小。

针对 Xception&Inception 模型，图片分别尺寸处理为 299x259（与训

训练集图像的平均宽高比相当) 和 299x299, 批次大小分别为 16 和 32。根据 LogLoss, 结果发现图像尺寸为 299x299 (与预训练模型的默认输入尺寸相当) 时效果较好, 而批次大小的变化对于最终效果似乎没有显著变化。

针对 NASNetLarge&InceptionResNetV2 模型, 图片分别尺寸处理为 331x287 (与训练集图像的平均宽高比相当) 和 331x331, 批次大小分别为 16 和 32。根据 LogLoss, 结果发现图像尺寸为 331x331 和批次大小为 16 时效果较好。

**Xception&IncpetionV3 模型**

<div>Size Batch</div>	299x259	299x299
16	0.06962	0.06883
32	0.06992	0.06411

**NASNetLarge&InceptionResNetV2**

<div>Size Batch</div>	331x287	331x331
16	0.05236	0.04924
32	0.05615	0.05102

根据以上结果, 本项目决定在后续训练过程中对

Xception&InceptionV3 模型使用的图像尺寸为 299x299, 而对

NASNetLarge&InceptionResNetV2 模型使用的图像尺寸为 331x331, 批次大小为 16。

确定了以上的“基本”参数后, 项目接下来对优化器 (包括其中的学习率等参数) 进行比较, 使用的优化器包括 SGD、RMSprop、Adam 以及 Ndam。结果发现, 两个模型均选择带动量以及学习率衰减的 SGD

作为优化器效果最佳，其中 momentum=.9, decay=1e-3, 并且使用 nesterov 动量，即设置参数 nesterov=True。另外，Xception&InceptionV3 模型表现最好的时候训练批次为 32。

**NASNetLarge&InceptionResNetV2 模型**

<b>Batch Optimizer</b>	16
RMSprop(lr=0.001, rho=0.9, epsilon=None, decay=0.0)	0.05022
Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)	0.05018
Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=1e-6, amsgrad=False)	0.04322
Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=1e-8, amsgrad=False)	0.04605
Nadam(lr=0.002, beta_1=0.9, beta_2=0.999, epsilon=None, schedule_decay=0.004)	0.04588
SGD(lr=1e-2, momentum=.9, decay=1e-3, nesterov=True)	<b>0.04170</b>
SGD(lr=9e-3,	0.04192

momentum=.9, decay=1e-4, nesterov=True)	
SGD(lr=1e-2, momentum=.9, decay=1e-3, nesterov=False)	0.04307
SGD(lr=9e-3, momentum=.9, decay=1e-4, nesterov=False)	0.04406

Xception&Inception 模型调参效果

Batch Optimizer	32	16
RMSprop(lr=0.001, rho=0.9, epsilon=None, decay=0.0)	0.04439	0.04597
Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)	0.04684	0.05153
Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=1e-6, amsgrad=False)	0.04360	0.04382
Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=1e-8, amsgrad=False)	0.04435	0.04413
Nadam(lr=0.002, beta_1=0.9, beta_2=0.999, epsilon=None, schedule_decay=0.004)	0.04708	0.04350

SGD(lr=1e-2, momentum=.9, decay=1e-3, nesterov=True)	<b>0.04313</b>	0.04947
SGD(lr=9e-3, momentum=.9, decay=1e-4, nesterov=True)	0.04804	0.04580
SGD(lr=1e-2, momentum=.9, decay=1e-3, nesterov=False)	0.04900	0.05022
SGD(lr=9e-3, momentum=.9, decay=1e-4, nesterov=False)	0.05330	0.04314

从 LogLoss 来看，两个模型表现相当，利用 NASNetLarge 和 InceptionResNetV2 进行融合模型表现稍微优秀，其中 Xception 和 InceptionV3 融合模型在 kaggle 的 leaderboard 上排到第 24 位，而 NASNetLarge 和 InceptionResNetV2 融合模型可以排到第 20 位，均位于 leaderboard 前 10%，即 131 名以前。



## 5. 结论

### 5.1. 结果可视化

抽取部分测试集图片进行展示，并且附上模型的预测结果，可以看到

展示出来的情况还是比较乐观的。



NASNetLarge&InceptionResNetV2 模型预测结果示例



Xception&InceptionV3 模型预测示例

## 5.2. 项目思考

猫狗大战是图像分类识别问题中的经典，虽然“入手”难度不大，但是想要取得优异结果还是不容易的，在 kaggle 竞赛排名榜上，LogLoss 在 0.4 以内的仅有 15 名选手。

整个项目的完成涉及到数据的预处理、特征提取、模型选择与构建、模型训练、参数调优、预测结果分析等，其中需要仔细思考以及多次尝试实验的步骤还是不少的，经历了这次项目，简述下我的以下收获：

- 1) 通常计算机视觉问题对 PC 的显卡和内存要求较高，可以尝试在亚马逊等云服务平台上进行开发，根据需求申请和配置云服务器，这会大大有助于项目开发；
- 2) 从头至尾构建一个完成的 CNN 既耗时又耗资源，在没有充分掌握 CNN 之前，建议利用迁移学习的方式去解决问题，多尝试几种预训练模型，再结合自己构建的少数几层网络，而后多进行几次测试，通过测试结果反馈来调整参数甚至改变模型的构造方式，直至到达满意的效果；
- 3) 不要小看数据预处理，我觉得这是一门博大精深的学问！现在的人工智能可以说十分依赖数据，没有数据一切都是空谈，只有将数据处理好，包括异常数据的清洗、归一化、图像的实时增强等，才不至于误导你的模型，从而模型才能学习到正确的方向，最终获得好的结果。

## 5.3. 突破

在本项目中，我始终没有成功将 LogLoss 突破至 0.4 以内，有可能异常数据清理做得不够科学，我使用了几种模型预测结果的并集，这有可能导致部分正常的猫狗图片也被清理掉了，从而使得模型没有学习到部分特征；另外，还可以尝试选择其它预训练模型、改变自定义的几层网络层或者改变下优化器和其它参数等，又或者干脆不做模型融合，选定单一预训练模型，然后对其进行微调，或许能获得惊喜的结果。

## 参考文献

- [1] Goodfellow, I., Bengio, Y., Courville, A. . Deep learning (Vol. 1) . Cambridge : MIT press, 2016 : 326-366
- [2] 薛景浩, 章毓晋, 林行刚. 图像分割中的交叉熵和模糊散度算法 . 《 电子学报 》, 1999
- [3] 基于卷积神经网络的深度学习算法与应用研究 . 知网 [引用日期 2019-07-17]
- [4] 深度学习是什么?它的工作原理是什么?百度文献 [发布日期: 2018-11-30]
- [5] 周志华 . 机器学习 . 北京 : 清华大学出版社, 2015 : 114-115
- [6] CSDN 卷积神经网络超详细介绍 [发布日期: 2018-09-19]
- [7] 基于图像底层特征的图像聚类与检索研究[D]. 昆明理工大学, 2018.
- [8] 《AI 算法工程师手册 作者 华校专》