

猫狗大战

机器学习纳米学位

叶剑

2018-05-26

目录.....	2
1. 项目定义	2
1.1 项目概述	2
1.2 问题陈述	2
1.3 评价指标	3
2. 分析	3
2.1 数据描述	3
2.2 算法与方法	8
2.3 基准指标	16
3. 方法	16
3.1 数据预处理	16
3.2 实施	17
3.3 改进	17
4. 结果	20
4.1 模型评估与验证	20
4.2 方案是否足以解决问题的理由	21
5. 结论	21
5.1 自由形态的可视化	21
5.2 思考	22
5.3 改进	23
参考文献	23

目录

1. 项目定义

1.1 项目概述

猫狗大战，之前是 2013 年 Kaggle^[4]的一道竞赛题目，要求写一个算法区分图片里面是猫还是狗，是图像识别问题中典型的一种。图像识别问题（CAPTCHA 或 HIPS）可以用来区分人类和机器，以达到减少垃圾邮件，网站密码攻击等目的。对于图片里面是猫还是狗的问题，人很容易识别，而机器识别起来却比较困难。究其原因，人类从出生之日起，就有无数次机会看到猫或狗，从而区分它们；但是机器没有这种多次识别猫或狗的经验，所以无从识别。但是如果机器能够像人类一样，能够找到海量的猫或狗的照片加上一个智能的算法训练，并一一识别区分它们，经过训练后的机器理论上应该也和人类一样，能够识别图片里面的猫和狗。早在 2007 年就已经有论文指出机器识别图片中的猫或狗正确率超过 80%。而 kaggle2013 年的竞赛 kaggle Public Leaderboard 上面，榜首机器识别图片中的猫或狗正确率更是达到了 98.533%。所以从理论和实践都说明机器学习来区分图片里面是猫还是狗的问题，应该并且能够得到解决。利用图像识别来区分人和机器已经变得不再安全了。

1.2 问题陈述

该项目要求给出一张彩色图片，机器通过使用深度学习方法识别这张图片是猫还是狗，是一个监督学习的二分类问题。输入图片像素值向量 x ，输出样本 x 表示狗的概率 y ，那么样本表示猫的概率就是 $1-y$ 。

图片数据向量表示猫或狗的特征并不容易提取，而且不容易让人理解，人类本身也无法准确的描述分辨猫或狗的核心特征。但是如果不提取特征，以整个图片向量作为特征向量，以图片像素向量的高维度性，使用传统的机器学习方法将带来维度灾难，表现在对样本要求的指数级别增长和对模型规模和计算量的指数级别增长上。

针对维度灾难，卷积神经网络是一种空间上共享参数的神经网络，本质上也是一种深度神经网络，他使用共享权重的卷积层代替了一般的全连接层。卷积神经网络总的想法是形成一个金字塔结构，金字塔部是一张大而浅的图片，仅仅包括红绿蓝 3 色通道的深度，通过

卷积操作逐渐压缩空间的维度，同时不断增加深度信息，使深度信息大体可以表示出复杂语义，在金字塔顶端，定义一个分类器，所有空间信息被压损成一个表示，仅映射到图片内容的参数被保留。基于卷积操作的卷积神经网络可以使用较少的输入参数和计算量得到较高的准确率。

图像分类问题的代表性数据集位 ImageNet，从超过百万张图片的训练数据中学习识别 1000 中图像分类，包含猫和狗等动物。通常情况下，在 ImageNet 上表现优异的模型，对其他图像分类问题具有良好的泛化性能。由于猫狗大战是 ImageNet 的子问题，适合使用迁移学习方法，使用与训练模型在 ImageNet 中学到的经验参数表示猫狗图片的低维特征向量，并对这些特征向量进行分类。

1.3 评价指标

二分模型输出单元通常使用 Sigmoid 激活函数，损失函数使用对数损失函数，计算 log loss 方法如下：

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

n 是测试集中的图片数， \hat{y}_i 是第 i 张图片预测为一条狗的可能性， y_i 是测试集的正确结果， $y_i=1$ 代表图片是一条狗， $y_i=0$ 代表图片是一只猫。 $\log()$ 是自然对数（底数为 e 的对数）。对模型的评价使用 log loss，log loss 越低，方案越好，模型在测试数据集的分类结果可以上传到 Kaggle 进行评价，排名。Kaggle 会给出分类结果的 score 也就是 log loss，我们依据 Kaggle 给出的 score 评价模型的好坏。

2. 分析

2.1 数据描述

该项目使用了 Kaggle 竞赛提供的数据集，训练数据包括 25000 张图片，猫狗各占一半，在文件名中标注了该图片是猫还是狗，以 cat.编号.jpg 或 dog.编号.jpg 命名，例如 cat.100.jpg。测试集包括 12500 张图片，没有标注类别，图片以图片数作为 ID 命名，例如 1.jpg，2.jpg 等。

具体图像分布如下图：

训练集总量： 25000
训练集猫的数量： 12500
训练集狗的数量： 12500
测试集总量： 12500

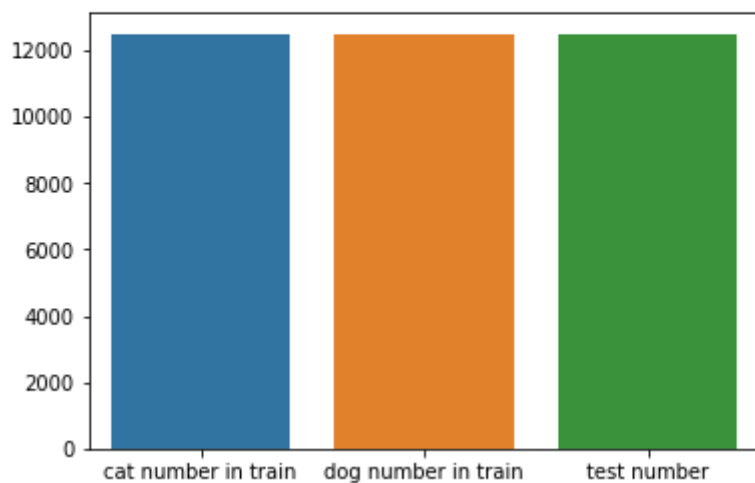


图 2-1-1 样本分布图

由图 2-1-1 样本分布图可以知道，训练数据中猫和狗的样本数一样，比例均衡，不必做均衡采样。

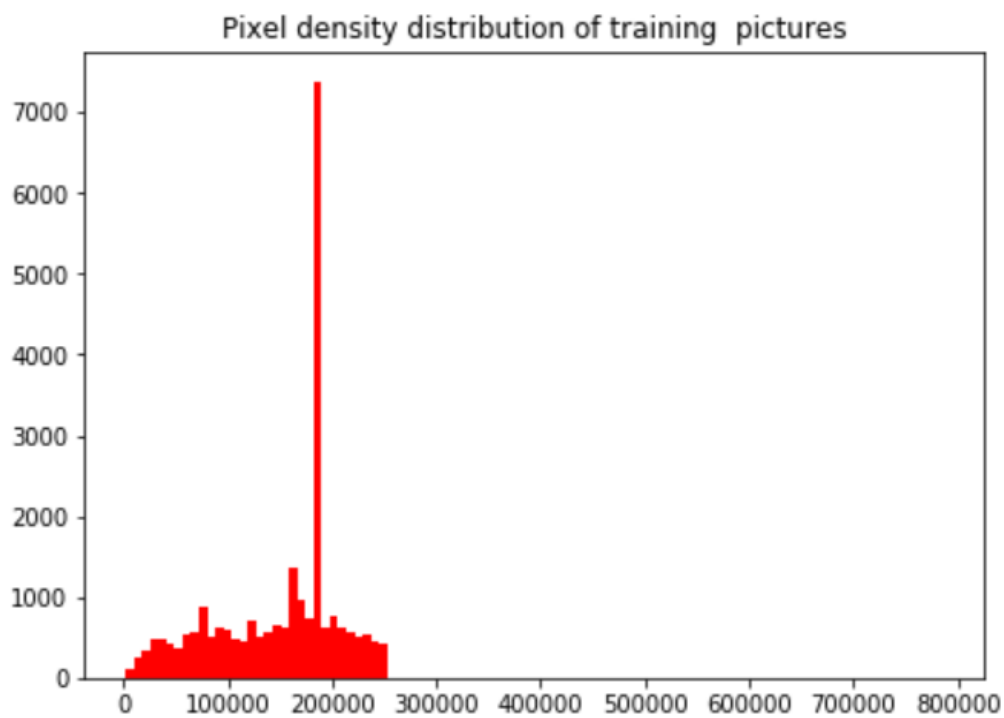


图 2-1-2 训练集像素分布图

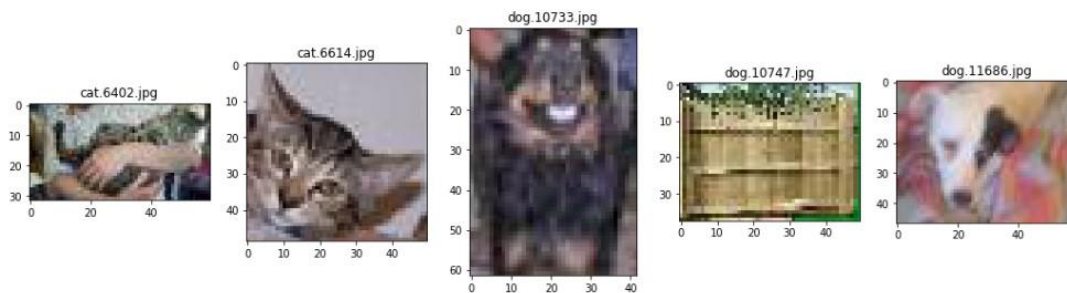


图 2-1-3 image size 小于 3000 样本抽样

由图 2-1-2 训练集像素分布图分析，绝大部分的图片像素较低，而且图片的像素大小不一，在分析像素分布图过程中打印出 image size 小于 3000 的图片名称，一共 17 张，然后用 pyplot 显示到页面，如图 2-1-3 从 17 张选出的 5 张，发现这些极小像素的图片显示非常模糊，即使人眼分辨也是比较困难的，其中 2 张模糊的猫，2 张模糊的狗，1 张非猫非狗的异常样本，打上了狗的标签。那种标签正确的数据，虽然显示模糊，像素及低，加大了训练的难度，但不应该删除，这些模糊图片可以考虑在数据增强中进行模糊增强，对于非猫非狗的异常数据，是需要删除的样本。

`['cat.0.jpg', 'cat.1.jpg', 'cat.10.jpg', 'cat.100.jpg', 'cat.1000.jpg']`

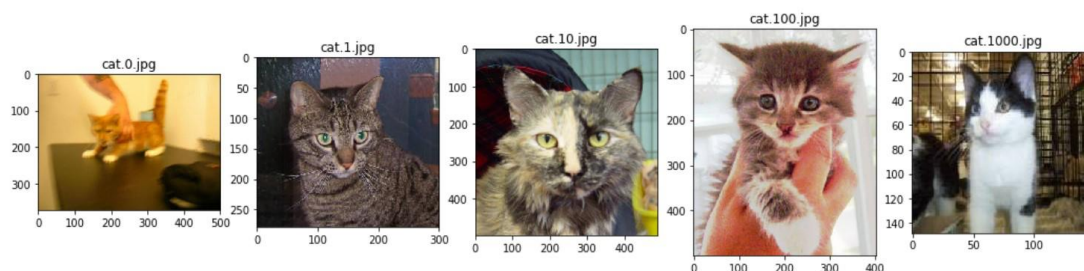


图 2-1-4 训练集中部分猫的图片

训练集中部分狗的图片：

`['dog.0.jpg', 'dog.1.jpg', 'dog.10.jpg', 'dog.100.jpg', 'dog.1000.jpg']`

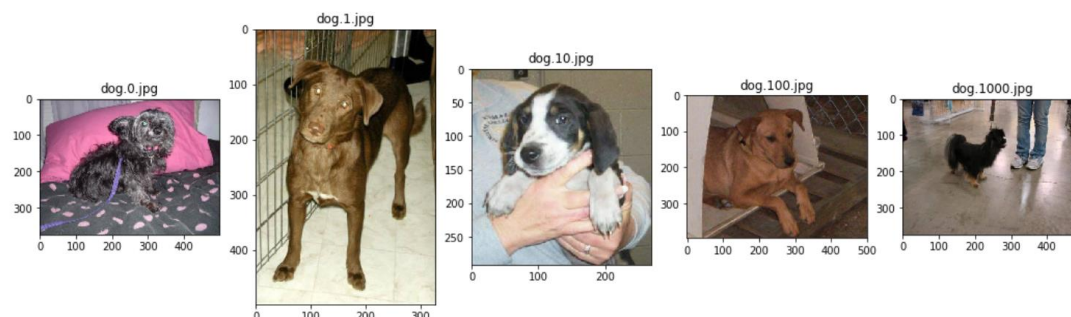


图 2-1-5 训练集中部分狗的图片

测试集中部分猫或狗的图片：

['1.jpg', '10.jpg', '100.jpg', '1000.jpg', '10000.jpg']

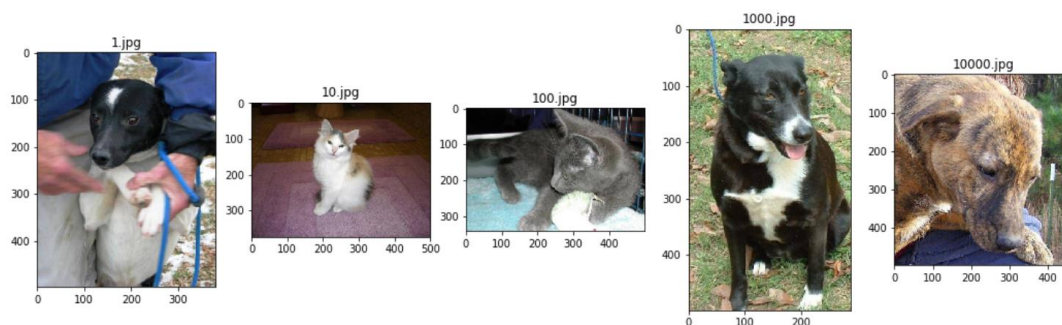


图 2-1-6 测试集中的图片

上面 3 个图分别抽取的训练集中的 5 张猫和狗的图片，测试集中 5 张图片。这些图片大小尺寸不一，有的比较模糊，角度变换多样，测试集图片不做任何预处理，训练集则需要考虑做模糊增强，image size 调整，异常样本删除等预处理操作。对于模糊增强，我们使用卷积核

$$blur = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$
 滑动窗口做卷积运算可以达到模糊增强的效果，因为低像素图片

较多，模糊增强比较耗时，且深度学习对于这些图片的鲁棒性，实际操作时没有使用模糊增强处理。因为 InceptionV3 需要的图片默认是大小是 299*299，所以，测试集和训练集图片的大小会统一调整为 299*299。至于异常样本的删除，使用了预训练模型 Inceptionv3 和 Xception 对训练集进行预测，测试集不予以处理，都取 ImageNet 的评价指标 top-60，然后取并集得到 67 张异常样本。如图 2-1-7 所示。Top-60 的意思是对于一个图片，如果概率前 60 中包含正确答案，就认为是正确的，不是异常值，否则认为是异常值。我们刷选出异常值，就是找出模型最不能识别的猫狗图片，训练集中的图片在模型给出的 60 个预测中，仍然没有包含正确的预测值，所以认为这张图片是异常值。

上面提到使用预训练模型的分类型对训练集进行初步分类，识别非猫非狗的异常数据，然后删除异常数据，在提出异常数据的基础上进行模型训练。这里只会对训练集进行异常数据清理，测试集数据不在清理范围内。清理异常数据后数据分布如下：

训练集总量: 24933
 训练集猫的数量: 12444
 训练集狗的数量: 12489
 测试集总量: 12500

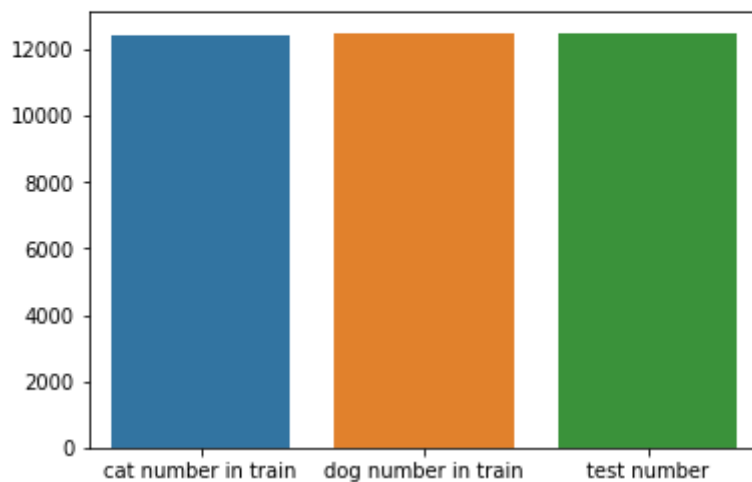


图 2-1-6 去掉异常样本后的数据分布图

部分异常数据如下图, 异常图片里面可能是非猫非狗图片, 也可能是有猫又有狗的图片, 对于这些错误标记的数据, 会给模型学习带来一定干扰, 清除异常图片有利于模型更好的训练学习。

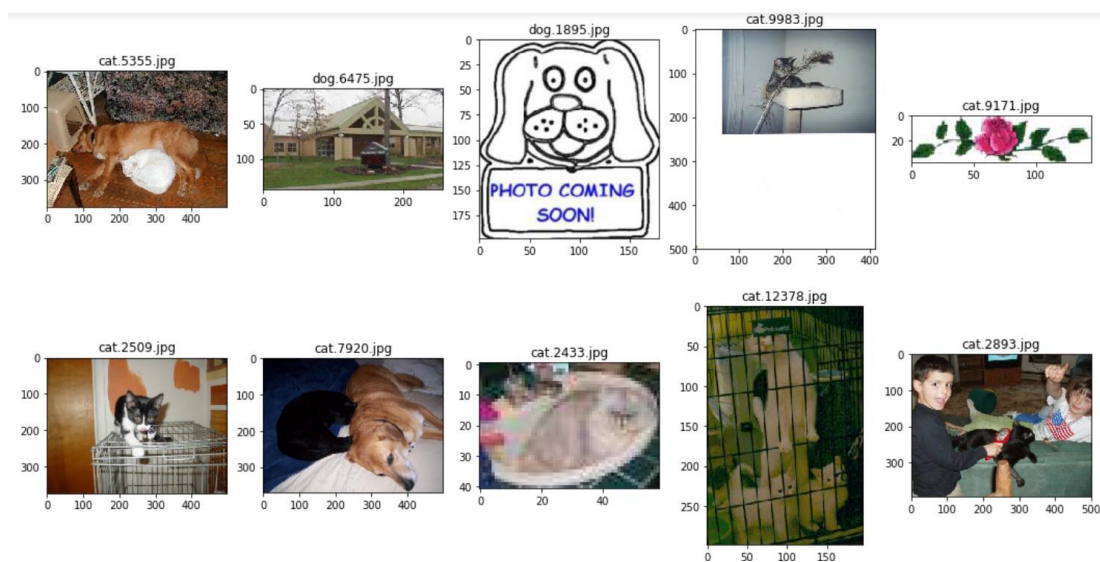


图 2-1-7 部分异常数据

2.2 算法与方法

2.2.1 机器学习

机器学习有许多的分支，深度学习是其中较新的一个分支，深度学习用图构成的网络对数据做高层次的抽象，专注于学习数据的表征。可以根据高层次的抽象结果做智能回应。

区别于深度学习，监督学习依照已经标志过的数据，推出新数据的标签。非监督学习可以把没有标签的数据打上标签，分成不同的组，挖掘数据的隐藏结构。增强学习可以建立一个模型，使得该模型能够在环境中不断学习来最大化它的奖励。

2.2.2 卷积神经网络结构

本项目使用的方法是深度学习中的卷积神经网络模型，简称 CNN。CNN 是一种理解视觉和听觉生物反馈过程的神经网络，基本结构为输入层，卷积层（带激活函数），池化层，全连接层。

输入层是图像等数据，计算机理解为输入若干个矩阵。

卷积层+池化层在 CNN 的隐藏层出现多次，次数是根据模型的需要来的。卷积层和池化层可以根据需要随意组合，模型构建设有限制，最为常见的 CNN 都是若干个卷积层和池化层的组合。

在若干个卷积层和池化层组合后面是全连接层，其实全连接层是 CNN 的输出层，添加 Softmax 激活函数来做图像分类识别。

微积分中的卷积表达式为：

$$S(t) = \int x(t-a)w(a)da$$

用矩阵可以表示为(其中*表示卷积)：

$$s(t) = (X * W)(t)$$

如果是二维卷积，表示为：

$$s(i,j) = (X * W)(i,j) = \sum_m \sum_n x(i-m, j-n)w(m,n)$$

CNN 中的二维卷积的定义为：

$$s(i,j) = (X * W)(i,j) = \sum_m \sum_n x(i+m, j+n)w(m,n)$$

其中，W 称为卷积核，而 X 为输入 W 的维度和 X 的维度一致。

2.2.3 卷积

根据 CNN 卷积公式中的定义，对图像的卷积，就是对输入的图像的不同局部的矩阵和卷积核矩阵各个位置的元素相乘然后相加得到。举个例子，我们由一个 5x5 的输入矩阵，卷积核是一个 3x3 的矩阵，卷积的步幅是 1。则卷积过程如下图所示：卷积结果是一个 3x3 矩阵

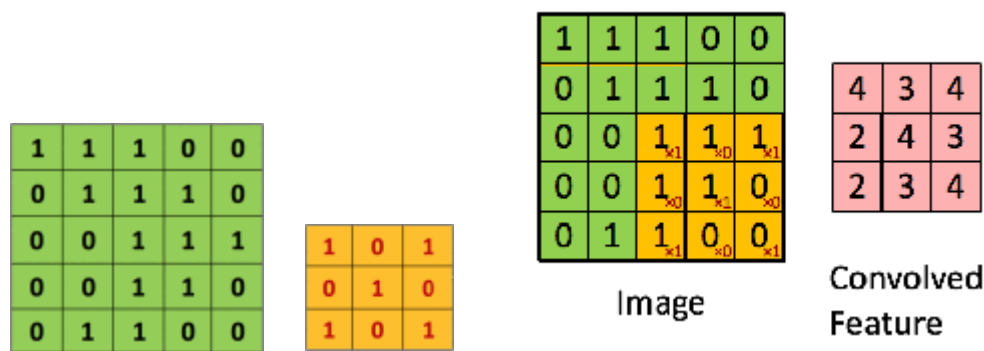


图 2-2-3-1 卷积操作图

卷积操作是 CNN 特有的，我们使用 5*5 大小的卷积核处理图像，原始图像中的像素点 x 在卷积核滑动的时候有 25 个可能位置，如果对图像从上往下看，只有 25 个输出受 x 的影响。设输出图像上的某点为 y，如果对图像从上往下看，则只有 25 个输入能够影响到有。如果连接不是稀疏矩阵，则所有输出都会收到 x 的影响，所有输入都能够影响到 y。

处于 CNN 网络更深层中的单元，它们接收域要比处于浅层的单元在接受域更大，这意味着在 CNN 中，即使直接连接都是很稀疏的，出在更深层中的单元仍然可以间接第连接到全部或者大部分输入图像。

该特征称为稀疏交互，使得 CNN 具有一定的平移不变形，举例来说，假如原始图像是一条狗，那么平行移动狗在图像的位置，卷积核滑动的时候，仍然可以检测到平移后的狗（当然狗得在这张图片内），稀疏交互使得狗的特征在同一副图的任何位置都可以被检测出来。也就是说不论狗的特征在图像中的任何位置都是可以被滤波器检测到的。

参数共享是指在一个模型中的多个函数中使用相同的参数，在 CNN 中，卷积核的每一个元素都作用在输入的每一个位置上。卷积运算中的参数共享保证了只需要学习一个参数集合，而不是对于每一个位置都需要学习一个单独的参数集合。

由于图片的底层特征是与特征在图像中的位置无关的，比如边缘，无论在图片什么位置出现的边缘，都可以用相同的卷积核进行特征提取。假设使用 5×5 大小卷积核处理图像，输出层的每一个像素都是由输入层对应位置的 5×5 大小的局部图像，与相同的一组 5×5 大小的卷积核参数做内积，再经过非线性单元计算而来的。这样无论图片原始大小如何，值需要使用 5×5 个参数就可以了。一组参数得到一个特征映射，实际使用中，一般会有多组参数，同时提取多组不同的特征。

参数共享使得卷积神经网络的计算量大大减少，解决了传统机器学习方法中带来维度灾难（表现在对样本要求的指数级别增长和对模型规模和计算量的指数级别增长上），使得这种机器学习能够运用到现实世界。CNN 的参数共享的特殊形式也使得网络层具有对平移等变的性质。

2.2.4 池化

CNN 中的池化层相对与卷积层简单，池化层是对输入数据各个子矩阵的进行压缩，把一个指定大小（比如 3×3 ）大小的子矩阵根据池化标准压缩成一个元素。池化标准一般有 Max 和 Average，即取子矩阵的最大值或平均值作为池化后元素的值。

例如：采用最大值池化方法

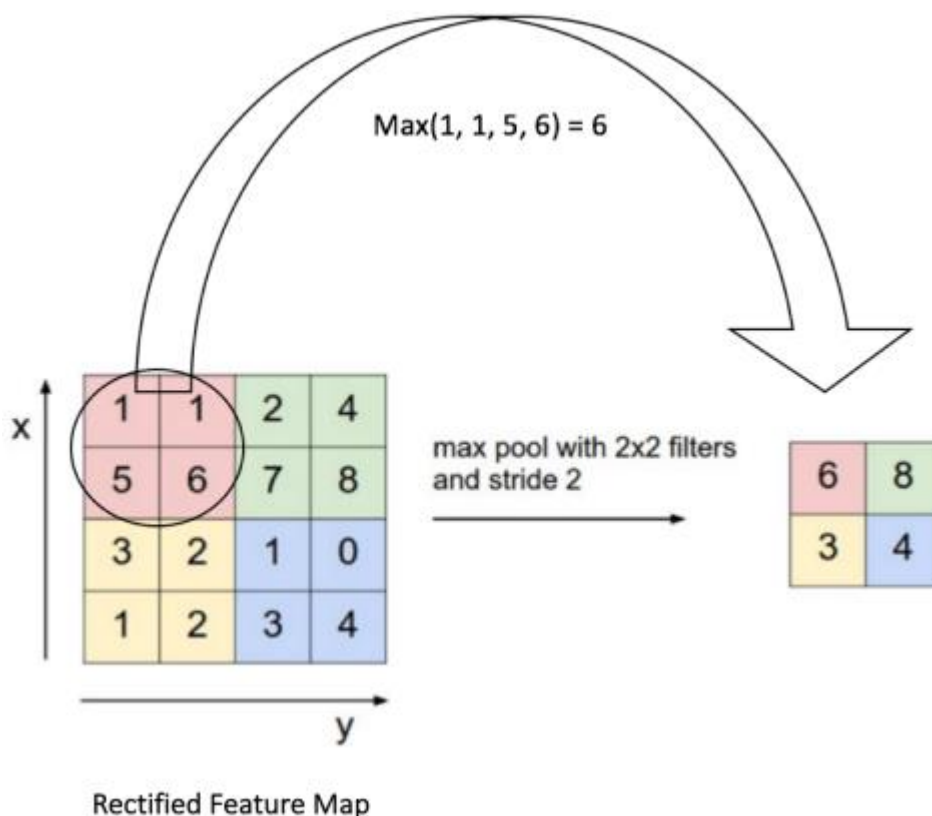


图 2-2-4-1 最大池化图

池化操作会提取相同的值，而不管图片中的狗是否有一定程度的平移或旋转和缩放。所以池化层的存在也使得 CNN 具有一定程度的平移或旋转和缩放不变性。例如，原始图像是一条狗，把图像中的狗旋转像 5 度，平移 5 个像素，或缩小 5%，池化操作还是会在狗相应的地方提取相同的值，二不管你把它平移、旋转、缩放到什么地方去了，之所以说一定的平移、旋转和缩放不变性，就是说旋转角度不能太大，太大了可能也会失效。一般来说，池化窗口越大，可以旋转保持抽取特征不变性的角度会越大。

2.2.5 dropout

Dropout 是一种正则化的方法，指的是在用前向传播算法和反向传播算法训练 CNN 模型时，一批数据迭代时，随机的从全连接网络中去掉一部分隐藏层的神经元，然后用这个去掉隐藏层的神经网络来进行一轮迭代，更新多有的权重。 ‘

当然，dropout 并不意味着这些神经元永远消失了，在下一批数据迭代前，会把 CNN 模型恢复陈最初的全连接模型，然后在用随机方法去掉部分隐藏的神经元，接着有去迭代更新权重。而这次用随机方法去掉的隐藏层和上次去掉的不同。

Dropout 每轮梯度下降迭代，他需要将数据分成若干批，然后分批进行迭代，每批数

据迭代时，需要将原来的 CNN 模型随机去掉部分隐藏层的神经元，用残缺的 CNN 模型来迭代更新权重。每批迭代更新完成后，要将残缺的 CNN 模型恢复成原始的 CNN 模型。Dropout 会将原始数据分批迭代，因此原始数据即最好较大，否则模型可能会欠拟合。

Dropout 可以防止过拟合并且提高效果，dropout 强迫一个神经单元，和随机挑选出来的其他神经单元共同工作，达到好的效果。消除减弱了神经元节点间的联合适应性，增强了泛化能力。经过交叉验证，隐含节点 dropout 率等于 0.5 的时候效果最好，原因是 0.5 的时候 dropout 随机生成的网络结构最多。

2.2.6 迁移学习

迁移学习利用原域的数据将知识迁移到目标域中，完成模型建立。在该项目，我把 ImageNet 数据集中学到的猫狗特征迁移到猫狗大战中的数据模型中，这就相当于我已经训练了百万张图片数据，然后再来训练猫狗大战中的训练数据，从头开始训练百万张样本耗时费力，通过迁移学习，可以达到这一目标，加快学习效率。举个例子：比如利用上千万的图象来训练好一个图象识别的系统（ImageNet），当我们遇到一个新的图象领域问题（猫狗大战）的时候，就不用再去找几千万个图象来训练了，只需把原来训练好的模型迁移到新的猫狗识别领域，在新的猫狗识别领域往往只需几万张图片就够，同样可以得到很高的精度。

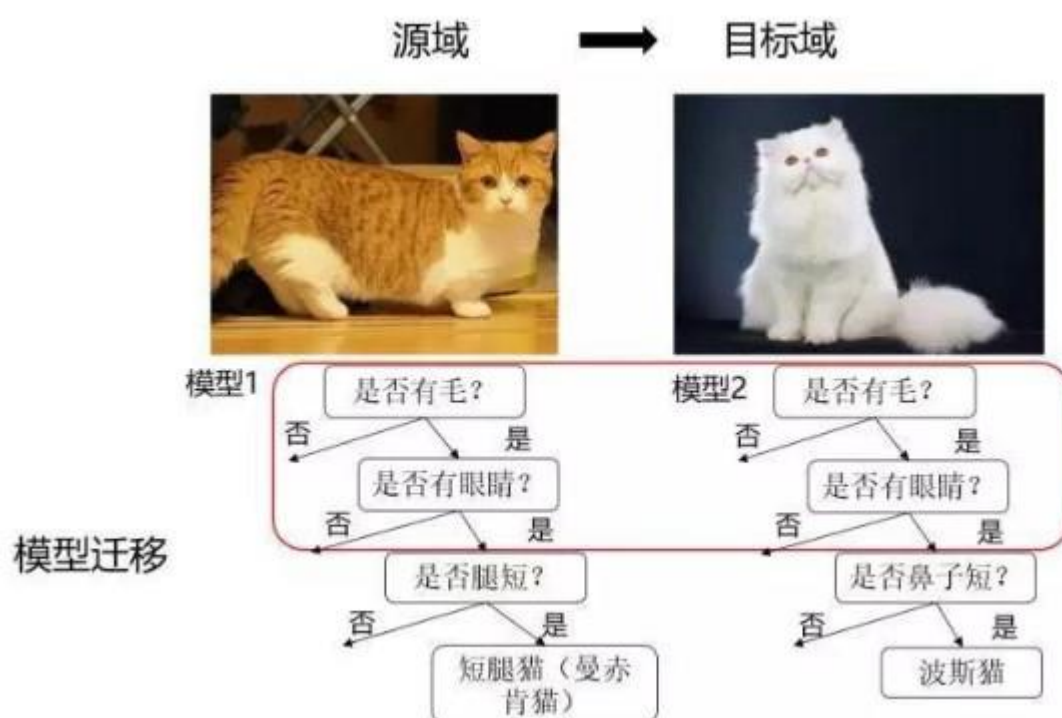


图 2-2-6-1 模型迁移图

2.2.7 InceptionV3

Inception_v1^[5]的网络，主要提出了 Inception module 的结构，使用 1x1,3x3 和 5x5 的卷积核，3x3 和最大值池化层组合一起，增加了网络的宽度和网络对尺度的适应性。

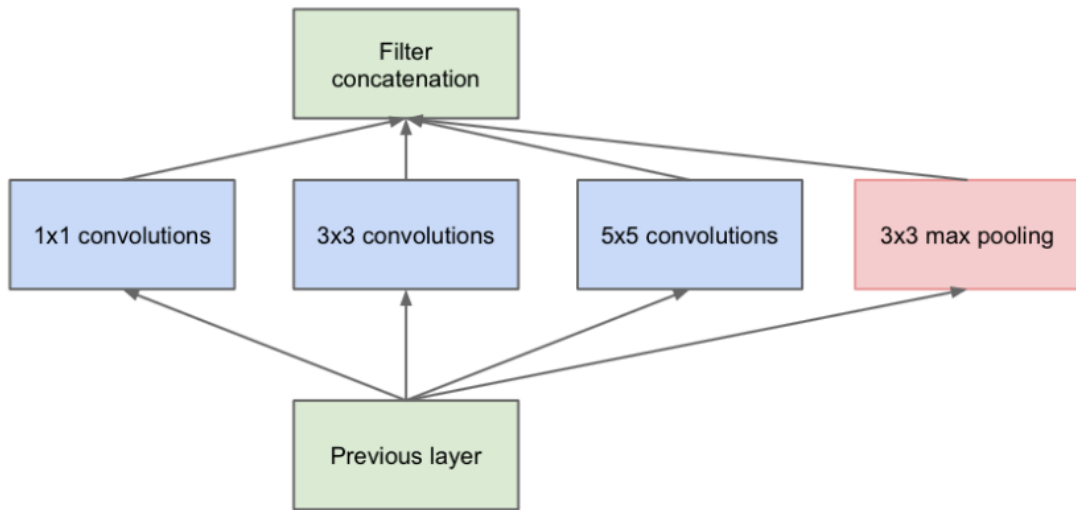


图 2-2-7-1 Inception module 结构图

另外因为上图所有卷积核都在上一层的所有输出上来做，5*5 卷积核计算量很大，使特征图厚度大。为了避免这一现象，从网络中引入了 1*1 conv，代表是 GoogLeNet。1*1 conv 一方面减少了权重参数，另一方面降低了维度。

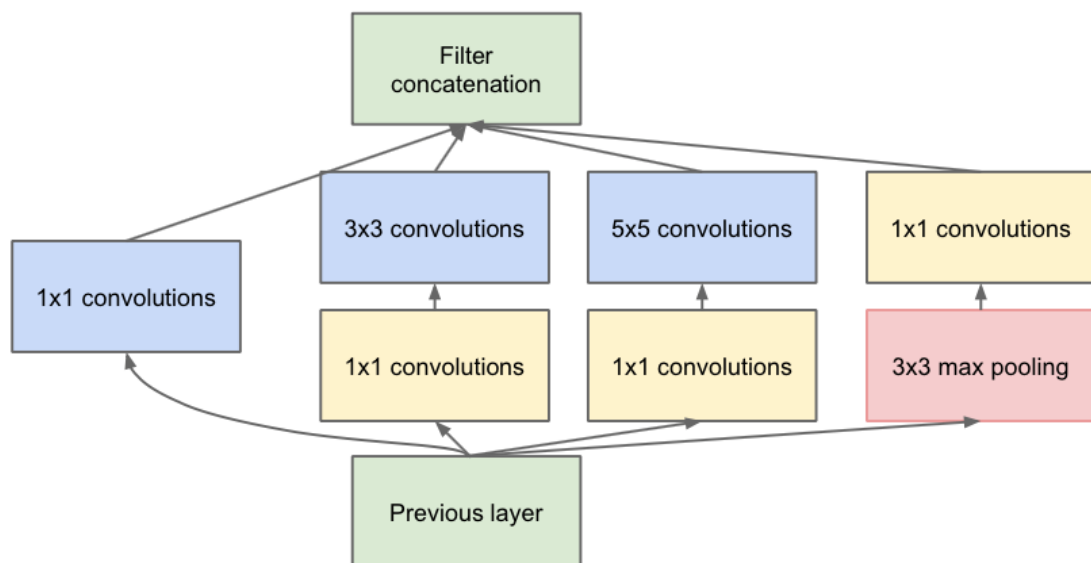


图 2-2-7-2 引入 1*1 卷积图

Inception_v2^[6]的网络，加入了 BN（Batch Normalization）层，并且使用了 2 个 3*3 替代 1 个 5*5 卷积作为改进。加入 BN 层的作用，减少了 Internal Covariate Shift（内部 neuron 的数据分布发生变化），使每一层的输出都规范化到一个 $N(0,1)$ 的高斯分布，从而增加了模型的鲁棒性，可以用更大的学习速率训练，可以使模型收敛更快，初始化操作更加随意，同时作为一种正则化技术，可以减少 dropout 层的使用。另外，2 个连续 3*3 卷积替代 inception 模块中的 5*5，，既降低了参数数量，也加速计算。

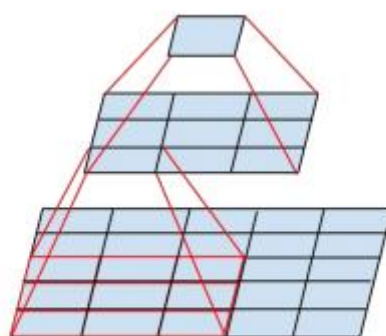


Figure 1. Mini-network replacing the 5×5 convolutions.

图 2-2-7-3 mini-network 替换图

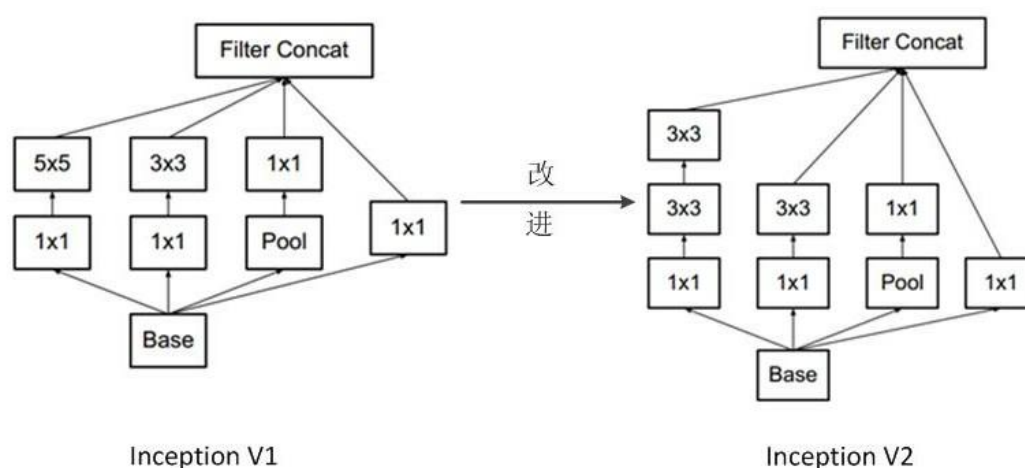


图 2-2-7-4 Inception v2 改进图

关于 Batch Normalization，是论文 Inception_v1^[5] 提出了 Internal Covariate Shift 这个问题，文中说在训练神经网络的过程中，因为前一层的参数变化而导致每层的输入分布都在不断变化，这使得我们需要更低的学习率和更小心地进行参数初始化，导致我们难以充

分构建一个具有饱满的非线性结构的模型，这种线像称为 Internal Covariate Shift。为了解决这个问题，google 提出了 batch normalization（批规范化），即在每次 SGD 的时候，通过 mini-batch 来对相应的 activation 做归一化操作，使得结果（输出信号各个维度）的均值为 0，方差为 1。

总结起来，inception_v2 是在 inception_v1 中的 inception 进行了改进，他的 batch normalization 是对 inception_v1 的一个补充。

Inception_v3^[1]网络，这要在 v2 的基础上，提出了卷积分解（Factorization），inceptionV3 将 7×7 卷积分解成 2 个一维的卷积（ $1 \times 7, 7 \times 1$ ）， 3×3 卷积分解成 2 个一维卷积（ $1 \times 3, 3 \times 1$ ），这样既可以加速计算（多余的计算能力可以用来加深网络），又可以将 1 个卷积拆成 2 个卷积，使得网络深度进一步增加，增加了网络的非线性，更加精细设计了 $35 \times 35, 17 \times 17, 8 \times 8$ 的模块。另外，增加了网络的宽度，网络输入从 224×224 变为 299×299 。

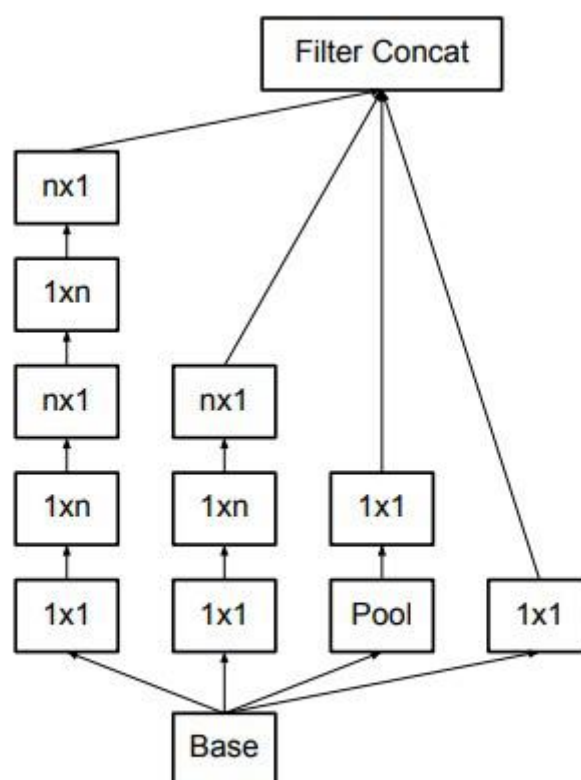


Figure 6. Inception modules after the factorization of the $n \times n$ convolutions. In our proposed architecture, we chose $n = 7$ for the 17×17 grid. (The filter sizes are picked using principle 3)

图 2-2-7-5 卷积分解图

2.2.8 小结

该项目基于 Python3.6，使用基于 Tensorflow 的 Keras API 进行模型实现与训练。TensorFlow 是 Google 提供的深度学习开源框架，主要完成了图的梯度计算和反向传播过程，使开发者可以专注与模型搭建与训练。支持 GPU 加速计算。GPU 架构计算性能远超 CPU，使用 NVIDIA GPU 加速计算，还需要安装 CUDA 以及 cnDNN。Keras 对 TensorFlow 进行了进一步封装，实现了常用的层次结构，调用接口易于使用，并提供了一些搭建好的预训练模型，以提高效率，避免重复造轮子。由于基于 GPU 的计算环境本地搭起来费时费钱，该项目使用 AWS 提供的云服务器 p2.xlarge 实例，支持 GPU 但是也不太贵的那种。该项目使用 keras 提供的 ImageNet 数据集预训练模型 InceptionV3^[1]进行迁移学习，通过添加全局平均池化层、分类器等构建新的模型，通过调整模型参数不断优化模型。

2.3 基准指标

该项目定义基准指标为 Kaggle 给出的 score (log loss) 低于 0.06，在 kaggle public leaderboard 上位列前 10%。该项目不定义基准模型，评价只与基准指标对比。

3. 方法

3.1 数据预处理

由于训练集数据里面有一些非猫非狗的异常数据，或者一些很模糊或背景复杂的猫或狗的图片，这些数据测存在对模型的训练还是有一定的影响的，去掉这些异常数据有利于提高模型的预测准确性。

另外由于我构建的模型是使用迁移学习，基于预训练模型 InceptionV3，InceptionV3 默认输入的分辨率是 299*299，本项目会把所有的训练集和测试集中图片尺寸都设为 299*299。

最后，由于 InceptionV3 的权值是由 TensorFlow 训练得来的，所以使用 inception_v3.preprocess_input 把图片数据缩放到 (-1, 1) 的范围内。

3.2 实施

在进行模型训练之前，会把训练集中的数据混洗切分，把 train 文件夹中的数据随机分成训练集和验证集 2 部分，比例为 8:2。切分方法是 sklearn.model_selection.train_test_split。设置 random.state 保证每次切分效果一样。

然后运用 keras 上的预训练模型 InceptionV3 构建一个新的模型用于训练和预测，新的模型结构如下图：

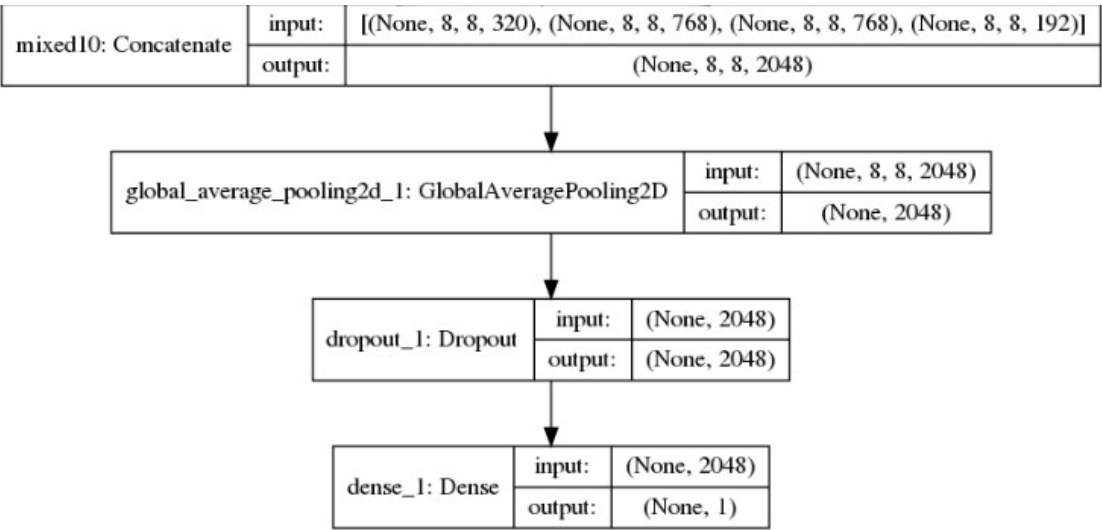


图 3-2-1 基于 InceptionV 的新模型结构图

该模型在不含池化层的 InceptionV3 基础上，添加一个全局平均池化层，然后添加 Dropout 层，最后添加 sigmoid 分类器。

模型锁住了 InceptionV3 前 280 层，单是对基础模型 InceptionV3 剩余的层进行训练。模型使用的优化函数是 keras.optimizers.Adam

3.3 改进

开始打算使用 ResNet50 预训练模型进行迁移学习，训练所得学习曲线并不理想，如下：

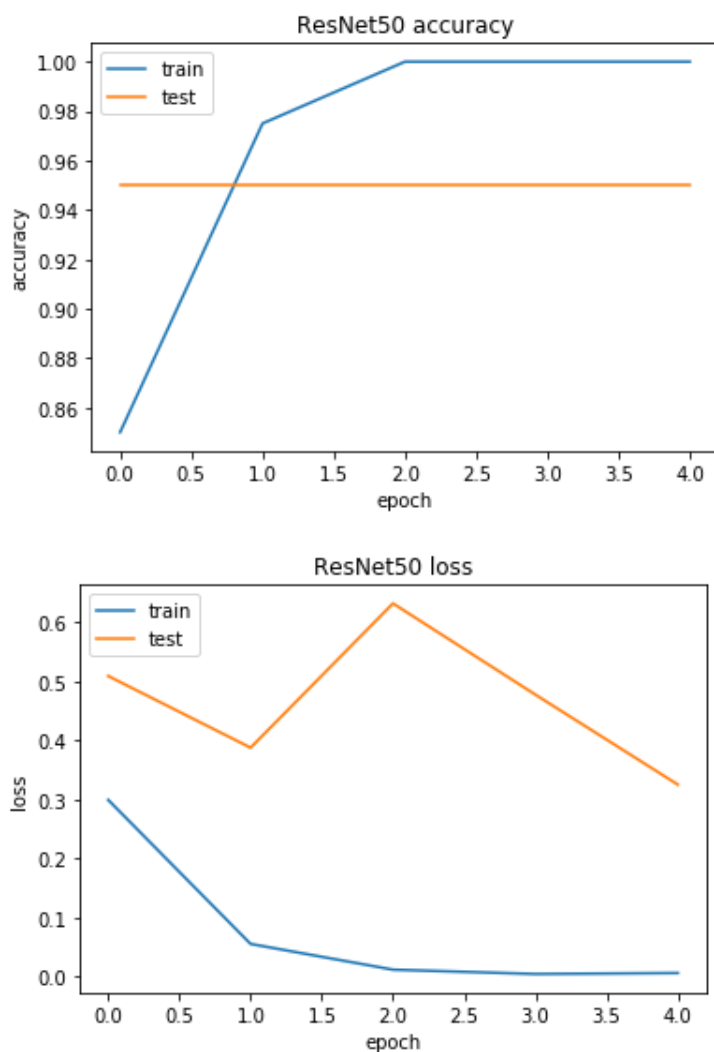


图 3-3-1 基于 ResNet50 的迁移模型训练预测图

后来使用了 InceptionV3, 发现曲线收敛效果更好, 就换过来了, 为了使模型的 log loss 尽可能的低, 对该模型进行了多次尝试, 模型的优化器学习率最终调整为 $lr=1e-5$, 每次更新后的学习率衰减值为 $decay=1e-6$, 模型训练使用 keras callbacks ModelCheckPoint 保存模型权重数据, 并使用 EarlyStopping 在验证集的 log loss 不再下降时自动终止训练。每次训练 128 各样本, 总共训练 8 个周期。

该项目 dropout=0.5, 之所以取 0.5 是借鉴了他人的经验, 本人应为时间和精力有限, 没有去试验小于 0.5 会怎样, 大于 0.5 会怎样。根据前面对 dropout 的介绍, dropout=0.5 时, 每轮训练会随机舍弃一半的隐藏层进行训练。

对于对训练模型, 有尝试过锁住前 291 层, 锁住更多的层并没有带来性能提升, 体现在收敛速度更慢, log loss 更高。所以最终选择锁住前 280 层, 至于更多的锁层尝试, 鉴于时间有限, 不多做尝试了。

锁住前 219 层的训练模型训练预测图如下：

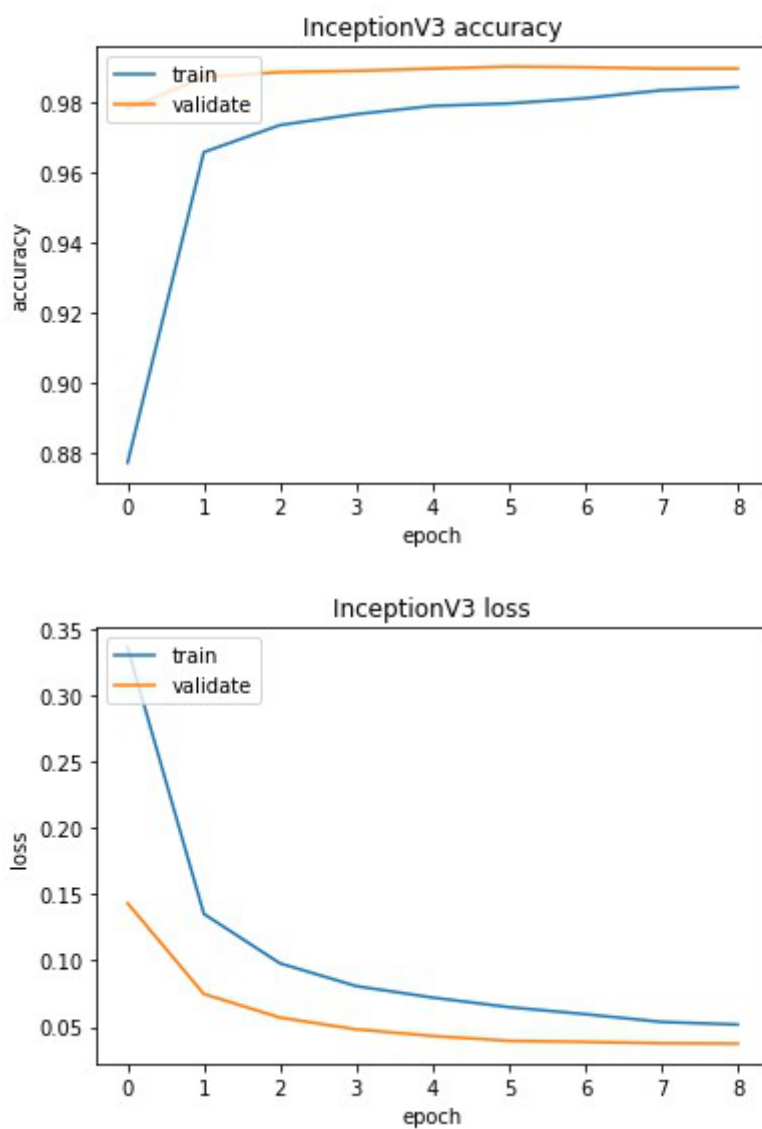


图 3-3-2 基于 InceptionV3 锁住前 219 层的迁移模型训练预测图

提交到 kaggle 的得分 0.05818 高于锁住 280 层的 0.05142：

Your most recent submission				
Name	Submitted	Wait time	Execution time	Score
predict004_InceptionV3.csv	12 minutes ago	1 seconds	4 seconds	0.05818
Complete				
Jump to your position on the leaderboard ▾				

图 3-3-3 基于 InceptionV3 锁住前 219 层的迁移模型 kaggle 得分

4. 结果

4.1 模型评估与验证

根据之前定义的基准阈值，模型在测试集上面的预测结构要在 kaggle 上的 Public leaderboard 上进 10%，也就是说 log loss 要低于 0.06. 通过训练过程中的模型预测图可见，模型的稳定性很好，模型的训练集和验证集准确率都平稳收敛于 0.99 左右，log loss 收敛于 0.035 左右。模型并没由出现过拟合问题。模型训练预测图如下：

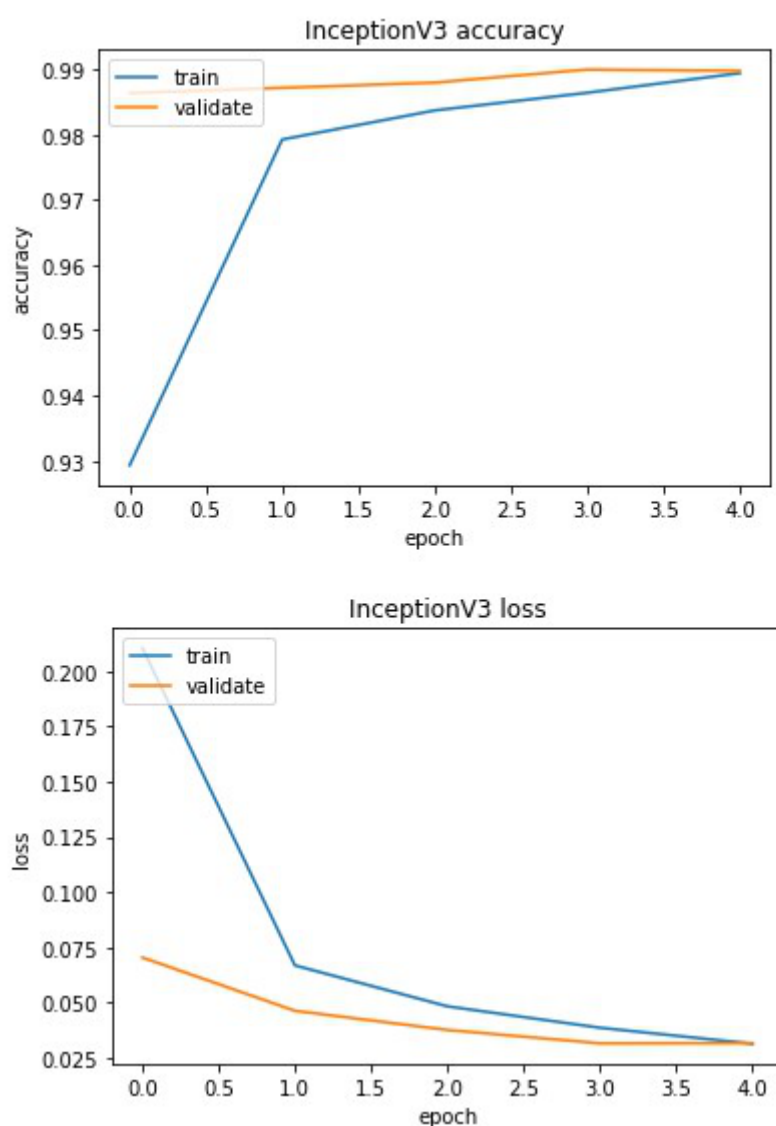
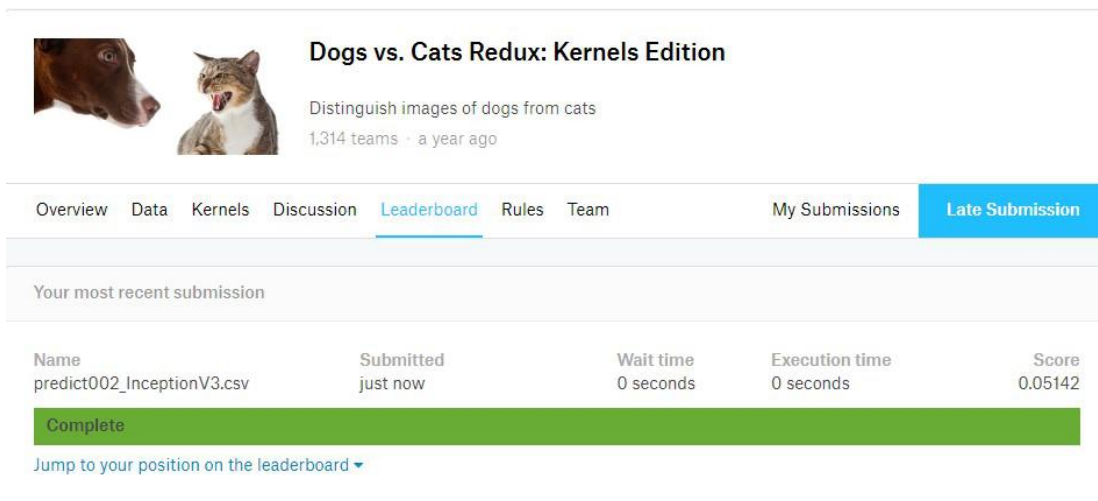


图 4-1-1 基于 InceptionV3 的迁移模型训练预测图

4.2 方案是否足以解决问题的理由

把模型对测试集的预测结果上传 kaggle，得到分数为 0.05142，低于我们之前定义的基准阈值 0.06，也就是说模型达到并超过了之前定义的基准。Kaggle 排名进入前 10%。这个结果证明，该模型能够完全胜任猫狗图像识别问题，准确率很高。



Dogs vs. Cats Redux: Kernels Edition				
Distinguish images of dogs from cats				
1,314 teams · a year ago				
Overview	Data	Kernels	Discussion	Leaderboard
				Rules
				Team
				My Submissions
				Late Submission
Your most recent submission				
Name	Submitted	Wait time	Execution time	Score
predict002_InceptionV3.csv	just now	0 seconds	0 seconds	0.05142
Complete				
Jump to your position on the leaderboard				

图 4-1-2 基于 InceptionV3 的迁移模型 kaggle 得分

5. 结论

5.1 自由形态的可视化

从测试集中随机挑选 16 张图片，然后用该模型进行预测，查看对图片的类别预测和图片预测概率，所有图片类别预测准确，而且每一张图片预测概率都接近 100%。

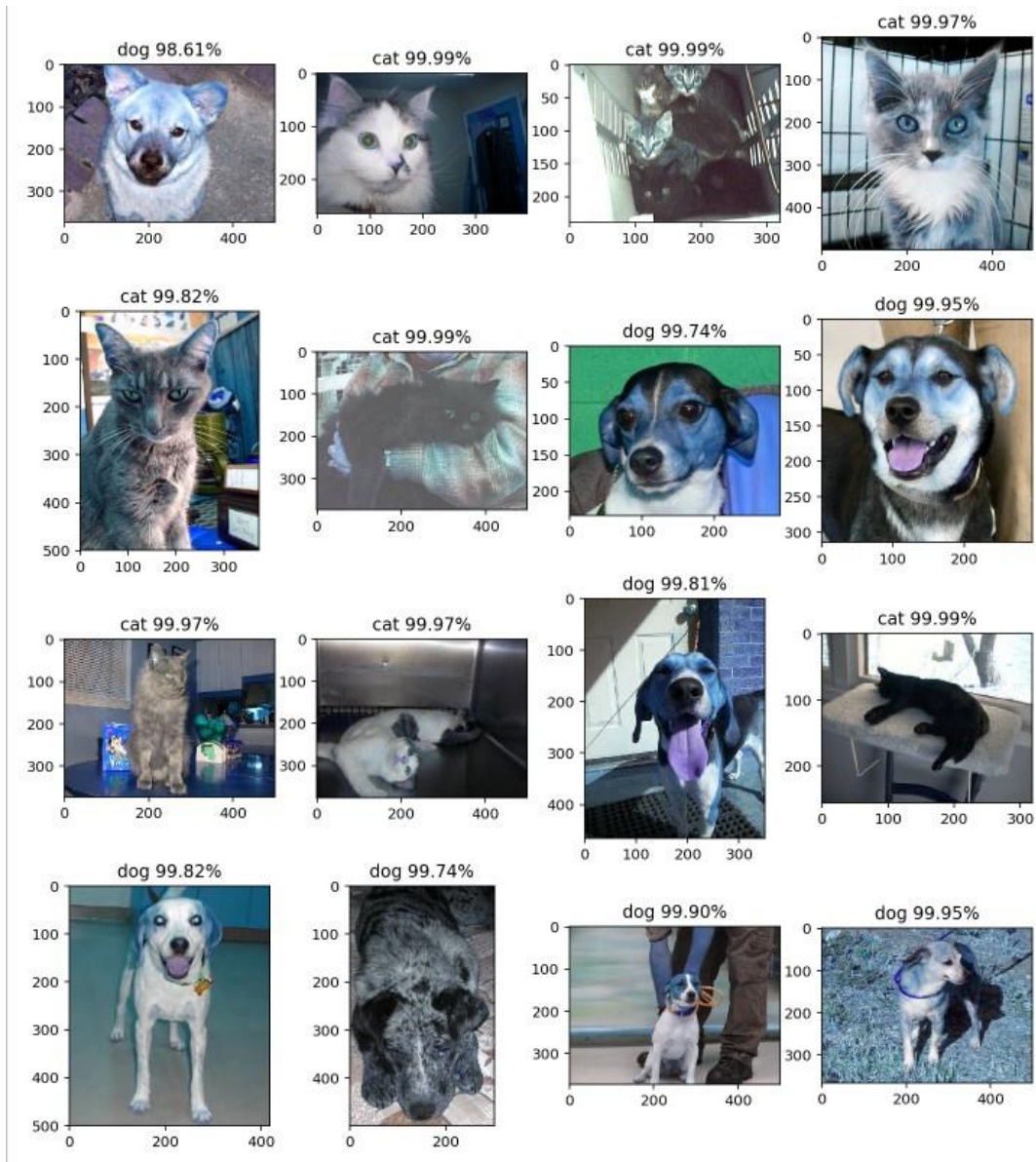


图 5-1-1 随机预测结果图

5.2 思考

该项目要求识别图片是猫还是狗，说实话，图片的参数多而且非常不易于理解，好在 keras、sklearn 等提供了强大的 API 支持，使这一任务变得简单许多。

该项目完成经历了数据获取、异常数据删除、训练数据混洗切分、图片的大小调整，入参范围调整，引入预训练模型 InceptionV3，基于 notop 的预训练模型构建一个新的模型，微调模型参数，在新模型上进行训练，预测测试数据等一系列过程。

该项目最终需要借助 AWS 的云服务器 GPU 去跑，所以在环境配置设置上花了至少由

一周的时间，项目的实际开发时间在 2 周左右吧，实际上该项目由于工作原因我本来已经中断 2 个多月没有弄了，眼看要过了毕业 **deadline**，后来被延期一个月，这才有时间坚持到现在，看来坚持到底，确实是达到目标的不变真理。

5.3 改进

该模型虽然达到了 **kaggle** 的前 10%，但是远远不是最好的成绩，所以在该猫狗分类问题上，该模型还有许多改进的地方。

可以使用 **K Fold** 验证法，消除部分混洗、分割造成的偶然性。

使用多模型融合，相当于不同的模型从不同的角度去预测，综合的预测结果一般会优于单模型预测

参考文献

- [1] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna Rethinking the Inception Architecture for Computer Vision [arXiv:1512.00567](#)
- [2] keras 中文文档 ([web](#))
- [3] Convolutional Neural Networks (CNNs / ConvNets). ([Web](#))
- [4] kaggle Public Leaderboard. ([Web](#))
- [5] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich Going Deeper with Convolutions [arXiv:1409.4842](#)
- [6] Sergey Ioffe, Christian Szegedy Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift [arXiv:1502.03167](#)