

# **机器学习工程师纳米学位毕业项目**

## **猫狗大战**

成凯华

2018 年 3 月 6 日

# 目录

目录 .....	2
1. 问题的定义 .....	4
1.1 项目概述 .....	4
1.2 问题陈述 .....	4
1.3 评价指标 .....	4
2. 分析 .....	5
2.1 数据可视化 .....	5
2.2 算法和技术 .....	6
2.2.1 神经网络 .....	6
2.2.2 卷积神经网络(CNN) .....	7
2.2.4 CNN 模型选取 .....	8
2.2.4 迁移学习 .....	12
2.2.3 实现技术 .....	12
2.3 基准指标 .....	13
3. 方法 .....	13
3.1 数据预处理 .....	13
3.1.1 训练数据类别整理 .....	13
3.1.2 异常值分析 .....	13
3.1.2 图像缩放和归一化 .....	15
3.2 执行过程 .....	16
3.2.1 单模型训练 .....	16
3.2.2 生成结果 .....	16
3.3 完善 .....	18
4. 结果 .....	19
4.1 模型的评价和验证 .....	19

4.1.1 网络结构和性能 .....	20
4.1.2 训练时间 .....	20
4.1.3 模型大小 .....	21
4.2 合理性分析 .....	21
5. 项目结论 .....	22
5.1 结果可视化 .....	22
5.2 对项目的思考 .....	22
5.3 需要作出的改进 .....	23
参考文献 .....	24

# 1. 问题的定义

## 1.1 项目概述

Dogs vs. Cats Redux: Kernels Edition(猫狗大战)是 Kaggle 大数据竞赛 2016 的一道赛题, 是 2013 之后又 Kaggle 又重新推出的猫狗识别项目, 利用给定的数据集, 用算法实现猫和狗的识别。数据集可以从 Kaggle 官网下载:

<https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/data>

本项目中主要使用卷积神经网络(Convolutional Neural Networks)来进行图片的识别来达到猫狗识别的目的。

## 1.2 问题陈述

数据集分由训练数据和测试数据组成, 训练数据包含猫和狗各 12500 张图片, 测试数据包含 12500 张猫和狗的图片, 训练集中的图片以类别为命名前缀, 比如 cat.0.jpg, 而测试集中文件名只有序号, 没有类别信息。Kaggle 提供的数据都是从真实世界采集来的包含猫和狗的图像, 图像的分辨率差异大, 图像质量参差不齐, 背景多样, 姿态各异, 猫狗在图像中所占比例也各不相同, 这些增加了分类的难度。

项目的结果需要提交一个包含所有测试集的预测结果的 submission.csv 文件, 文件中记录每个序号的图片判别为 dog 的概率, 如下图所示

	A	B	C
1	id	label	
2	1	0.5	
3	2	0.5	
4	3	0.5	
5	4	0.5	
6	5	0.5	
7	6	0.5	
8	7	0.5	
9	8	0.5	
10	9	0.5	
11	10	0.5	
12	11	0.5	
13	12	0.5	
14	13	0.5	
15	14	0.5	
16	15	0.5	

图 1. submission.csv 文件内容

## 1.3 评价指标

Dogs vs. Cats Redux: Kernels Edition 中, 使用的评价指标为对数损失, 分数的计算方式如下

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

LogLoss 是一个连续值, 它不仅能够评价分类的准确度, 同时可以评价分类的置信度, 就是说模型不仅要能够正确分类, 同时要对分类的结果十分肯定才能够的高分。

## 2. 分析

### 2.1 数据可视化

正如上文中提到的, 竞赛的数据集中有两个子集, 一个训练数据集, 一个测试数据集, 训练数据集中图片的文件名的前缀为图片的标签, 说明了图片所属的类别。图 2 展示了训练集中的图片和名称, 图 3 展示了测试集中的部分图片和名称。



图 2. 训练集图片



图 3 测试集图片

这些图片的分辨率各异, 进行预处理的时候需要将图片统一缩放到同一尺寸再进入网络。图片都是三通道的彩色图, 像素值的范围[0,255]。

## 2.2 算法和技术

### 2.2.1 神经网络

人工神经网络 ( Artificial Neural Networks ) 是早期机器学习中的一个重要的算法，历经数十年风风雨雨。神经网络的原理是受我们大脑的生理结构——互相交叉相连的神经元启发。但与大脑中一个神经元可以连接一定距离内的任意神经元不同，**人工神经网络具有离散的层、连接和数据传播的方向。**

神经网络的出现最早可以追溯到 1958 年, 康奈尔大学的 Rosenblatt 提出感知机 (Perceptron) 的概念<sup>[1]</sup>, 并用它来识别印刷体英文字母, 如下图所示就是一个感知机的结构, 包括了输入特征  $X(x_1 \sim x_4)$ , 经过与权重  $W(w_1 \sim w_4)$  线性加乘  $\sum_{i=1}^4 [x_i * w_i]$  之后, 经过一个非线性激活单元 unit step function, 得到输出  $y$ 。感知机确立了神经网络的基本架构就是线性加乘和非线性激活。

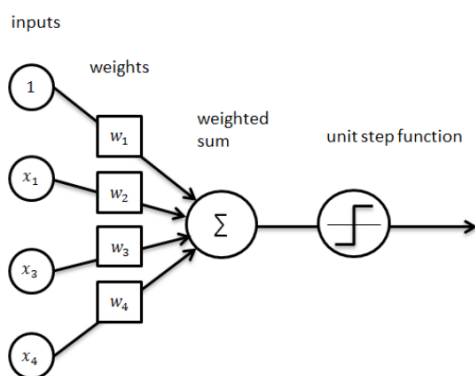


图 4 感知机结构

如果我们在输出  $X$  和  $y$  中间再加入新的层连接, 这个时候就出现了神经网络更普遍的一种结构——全连接层(如下图所示), 在输入和输出之间的层我们称为隐藏层, 有了隐藏层, 神经网络的学习能力变得更强。下图展示了一个有两个隐藏层的网络结构, 全连接发生在 input layer 与 hidden layer1 还有 hidden layer1 和 hidden layer2 之间, 全连接的意思就是当前层的结点和下一层每个结点都相关联。

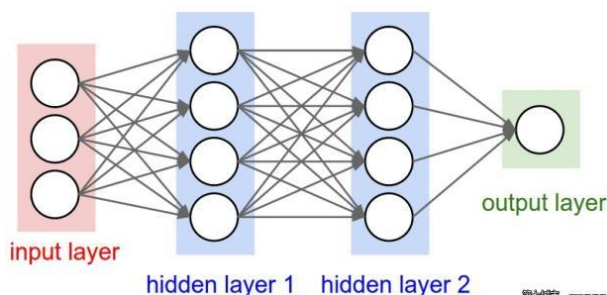


图 5 全连接层结构

虽然很早就被提出, 但事实上, 直到前不久, 神经网络也还是为人工智能圈所淡忘。其实在人工智能出现的早期, 神经网络就已经存在了, 但神经网络对于“智能”的贡献微乎其微。主要问题是, 即使是最基本的神经网络, 也需要大量的运算。神经网络算法的运算需求难以得到满足。

随着 GPU 带来的强大算力的支持, 神经网络又重新回到人们视线, 一个里程碑事件是 Alexnet<sup>[2]</sup>夺取了 2012 年 ILSVRC 比赛的冠军, 随之, 在计算机视觉(CV)领域, 一个强大的工具-卷积神经网络受到越来越多的关注并且相关的理论和技术迅速发展。

### 2.2.2 卷积神经网络(CNN)

卷积神经网络是一种多层神经网络, 擅长处理图像特别是大图像的相关机器学习问题。卷积神经网络与之前我们谈到的神经网络的一个重要差别就是引入了卷积层(Convolution Layer), 我们先看一下卷积是什么, 卷积运算的定义如下图所示:

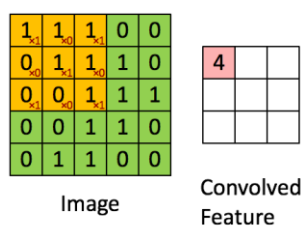


图 6. 卷积运算示例

如图所示, 我们有一个 5x5 的图像, 我们用一个 3x3 的卷积核 $[1,0,1;0,1,0;1,0,1]$ 来对图像进行卷积操作(可以理解为有一个滑动窗口, 把卷积核与对应的图像像素做乘积然后求和), 得到了 3x3 的卷积结果。

这个过程我们可以理解为我们使用一个过滤器(卷积核)来过滤图像的各个小区域, 从而得到这些小区域的特征值。在实际训练过程中, 卷积核的值是在学习过程中学到的。

卷积层就是利用一个或多个卷积核来提取当前输入(Image)的特征作为该层的输出(Convolved Feature), 卷积层和全连接层的差别主要在于不是每个结点都和下一层的每个结点都要关联, 关联只发生在局部, 如下图所示<sup>[3]</sup>, 并且在不同位置所用的卷积核共用一套参数

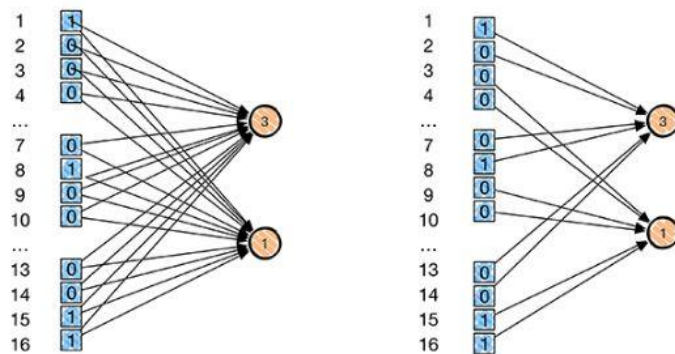


图 7. 卷积层和全连接层区别示意

这就引出了卷积层的两个特点: 局部连接和权值共享。

局部连接使输出的结点不再跟所有的输入结点有关系, 全值共享则大大降低了参数的数量, 以上面这个图为例, 输入结点有 16 个, 输出结点 2 个, 如果全连接层(左图), 不考虑偏置, 则一共有  $16 \times 2 = 32$  个权重参数, 而如果是卷积层(右图), 不考虑偏置, 一共有  $2 \times 2 = 4$  个参数, 因为卷积大小是  $1 \times 2$ , 在不同位置共享, 输出结点为 2 就只有 2 个卷积核, 卷积层相比全连接层参数个数减少到  $1/8$ , 这只是小网络的例子, 当输入图像变大, 网络更深的时候, 卷积层相比全连接层参数会更少, 参数减小使得训练网络的计算量变少, 并且降低了过拟合的风险。

所以, 对于猫狗分类的这个图像识别项目, 我们将用卷积神经网络来完成。

#### 2.2.4 CNN 模型选取

在模型的选取方面, 我们主要考虑的是 Google 推出的 Inception 系列最近几年的成果, 主要包括 InceptionV3<sup>[4]</sup>、Xception<sup>[5]</sup>和 InceptionResNet<sup>[6]</sup>, 这几个模型包括了从 2015 年获得 ILSVRC 冠军的 GoogLeNet<sup>[7]</sup>开始的一些重要的卷积神经网络模型的设计思路, 比如 group convolution, convolution factorization, depth-wise separable convolution 以及与 ResNet<sup>[8]</sup>的融合等。下面重点介绍一下这三个模型的结构特点和优势:

##### -1. GoogLeNet 和 InceptionV2

要说清楚 InceptionV3 和后面出现的 Xception<sup>[5]</sup>和 InceptionResNet<sup>[6]</sup>, 就不得不提到 Inception 系列第一个获得 ILSVRC 冠军的 GoogLeNet<sup>[7]</sup>模型, 下图是 GoogLeNet 模型的其中两个 inception module, 我们通过这个图来解释一下 GoogLeNet 的结构特征和带来的好处:



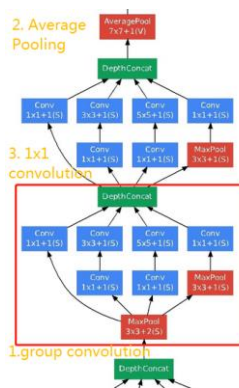


图 8. GoogLeNet Inception Module

### ① Group Convolution

就是在某一层不是用一个尺度的卷积来提取特征, 而是并行使用集中尺度的卷积, 在 GoogLeNet 中是同时使用 1\*1、3\*3、5\*5 的卷积, 这样可以提升网络对于尺度的适应性, 并行提取的特征最终会通过 DepthConcat 融合在一起, 这样同时增加了网络的宽度, 使提取出的特征更加丰富, 并行卷积层然后特征融合构成了一个 Inception Module, 而这种卷积层并行操作, 再合起来作为特征输出的方法称之为 Group Convolution。

### ② 1\*1 卷积

之前的卷积过程特征的降维升维是和空间卷积放在一起做的, 1\*1 卷积的引入可以单独来做特征的升维降维, 而与空间卷积过程无关, 在 GoogLeNet 中 1\*1 卷积用来对上一级传入的 feature map 降维, 可以显著减少 weights 大小和 feature map 的维度。

### ③ Average Pooling

之前我们已经讨论过卷积层相比全连接层 weights 会少很多, 在进入 softmax 分类之前, GoogLeNet 利用 7x7 的 average pooling 取代直接进行全连接, 显著减少了 weights 的个数, 相比于之前参加 ILSVRC 获得不俗成绩的 AlexNet<sup>[2]</sup>和 VGG<sup>[9]</sup>, weights 个数分别是它们的 1/12 和 1/27。

所以, GoogLeNet 通过这些网络架构的设计, 使得参数的总量更小, 网络更深更宽, 在显著减少模型 memory cost 和 compute cost 的同时, 提高模型的表现。

在 GoogLeNet 之后, Inception 系列出现的下一个重要的模型是 InceptionV2<sup>[10]</sup>, 在保留了 GoogLeNet 的结构优点的基础上, InceptionV2 又做出了两点改进, 如下图所示:

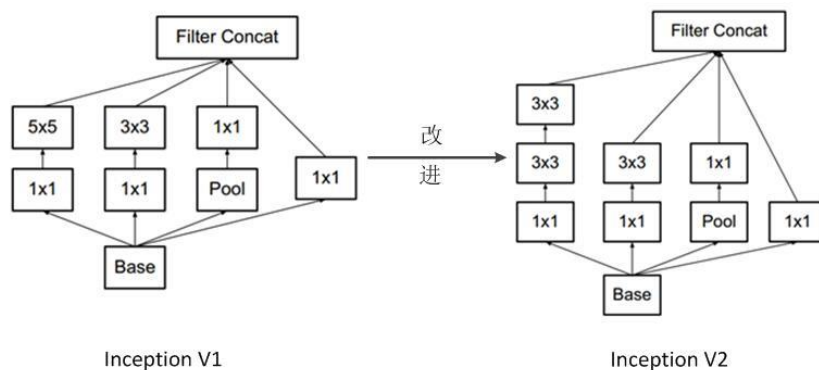


图 9. InceptionV2 对 GoogLeNet 做出的改进

### ① BN 层的引入

BN(Batch Normalization), 就是不光对输入  $X$  做 Normalization, 而且对每一层的 feature map 都做 Normalization, BN 层的引入, 减少了 Internal Covariate Shift, 使每一层的输入都规范到一个合理的分布, 增加了模型的鲁棒性。

② 用 2 个连续的  $3 \times 3$  卷积代替 GoogLeNet 中的  $5 \times 5$  卷积, 这样做的好处是网络深度更深, 性能更好, 缺点是增加了 25% 的 memory cost 和 30% 的 compute cost。

### -2. InceptionV3

在 InceptionV2 的基础上, InceptionV3 的重要改进是提出了卷积分解(Factorization), 将  $7 \times 7$  卷积分解成两个一维卷积( $1 \times 7$ ,  $7 \times 1$ ), 对于  $3 \times 3$  卷积也是一样, 如下图所示, 这样做, 可以明显减少 weights 的个数和计算量, 对  $3 \times 3$  卷积来说, 计算量只有之前的  $1/3$ , 同时, 一层卷积拆分成两层卷积, 又加深的网络的深度, 增加了网络的非线性, 提升了模型的表现。

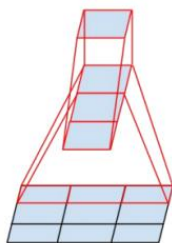


Figure 3. Mini-network replacing the  $3 \times 3$  convolutions. The lower layer of this network consists of a  $3 \times 1$  convolution with 3 output units.

图 10. convolution Factorization

### -3. Xception

在卷积分解的改进上, Xception 相比 InceptionV3 走得更远, 是把空间的卷积和通道的升维降维完全分开, 如下图所示, 这种方式叫 depth-wise separable convolution, 这种设计的假设是“跨通道的相关性和空间相关性是完全可分离的, 最好不要联合映射它们”, 在具体的实

现上, 就是  $1 \times 1$  卷积先进行通道降维, 然后在每个通道上分别做  $3 \times 3$  卷积, 每个卷积都只是空间卷积, 通道之间没有关联, 之后再进行 concat 把每个通道的特征融合在一起作为这一层卷积的输出特征。这样做当然又节省了一部分 memory cost, 模型需要训练和存储的 weights 更少了, 同时模型表现也得到提升, 在 ImageNet 数据集上, Xception 的表现稍优于 InceptionV3(Top-5 error 5.5% vs. 5.6%)

Figure 3. A strictly equivalent reformulation of the simplified Inception module.

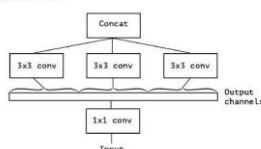


Figure 4. An “extreme” version of our Inception module, with one spatial convolution per output channel of the  $1 \times 1$  convolution.

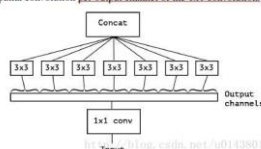


图 11. Depth-wise Separable Convolution

#### -4. InceptionResNet

如果说从 InceptionV3 到 Xception 卷积分解是改进的主要方向, 那 InceptionResNet 则是从整体架构上做了一次重大改动, 那就是引入了 ResNet[8]中的 residual connection, residual connection 也被称作跳远连接, 它是 ResNet[8]网络的首创, 简单说就是输出的 feature 不只是来自经过卷积后的结果, 还包括直接从输入连接过来的 feature。Inception-ResNet 其中一个 module 的结构如下图所示, 从图中我们可以看到, 最后的那个 “+” 就是 residual connection, 它将经过 group convolution 提取的特征和输入特征用相加的方式融合起来, residual connection 的引入可以使网络更深, 有效减缓深层网络带来的梯度消失和梯度爆炸问题, 有助于训练的稳定性, 在 ImageNet 数据集上, 因为 residual connection 的引入, 也确实提高了网络的表现, Top-5 Error 已经降低到 4.9%, 这已经要比通常认为的人眼能达到的错误率 5.1%还低了。

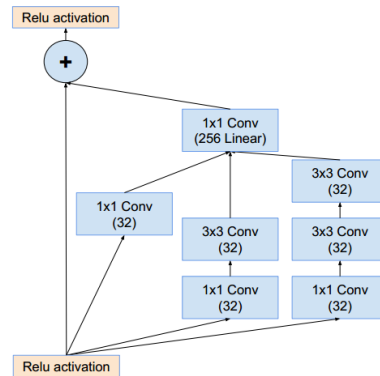


图 12. Inception-ResNet-A 网络结构

## 2.2.4 迁移学习

考虑到我们的训练数据有限, 并且从零开始训练一个卷积神经网络, 需要进行细致的网络结构设计, 并调整大量的参数, 可能很难获得理想的效果。我们采用迁移学习的方式, 利用经过 ImageNet 大量数据训练好的参数, 加上后端的重新分类来适应我们自己的分类任务。迁移学习就是把已学训练好的模型参数迁移到新的模型来帮助新模型训练。考虑到大部分数据或任务是存在相关性的, 所以通过迁移学习我们可以将已经学到的模型参数 (也可理解为模型学到的知识) 通过某种方式来分享给新模型从而加快并优化模型的学习效率不用像大多数网络那样从零学习。

## 2.2.3 实现技术

在本项目中, 我们利用 Keras 作为前端 API, Tensorflow 作为后端深度学习框架, 来搭建我们的网络实现图片分类。

### -1. Keras<sup>[11]</sup>

Keras 是一个前端神经网络 API, Keras 由纯 Python 编写而成并基 Tensorflow、Theano 以及 CNTK 后端。是一个高度模块化的神经网络库, 支持 GPU 和 CPU。Keras 为支持快速实验而生, 能够把你的 idea 迅速转换为结果。

### -2. Tensorflow

TensorFlow 是谷歌于 2015 年 11 月 9 日正式开源的计算框架。TensorFlow 计算框架可以很好地支持深度学习的各种算法, 但它的应用也不限于深度学习。

Tensorflow 程序通常被组织成一个**图的构建**和**图的执行**阶段:

我们搭建一个神经网络, 组织各个层及之间关系的过程称为图的构建, 然后通过不断反复的执行图中的训练 op 来逐渐优化参数。在图的构建阶段, 就是各种 op 的拼接组合, op 之间流通的 tensor 是由最初的一个 op 产生的, 它被称为源 op, 没有输入 tensor, 只有输出

tensor, 比如说常量(Constant)就是一个源 op。

而输入源给我们构建的图, 获取输出结果就是图的执行阶段, 在图的执行阶段, 需要指定会话模式 `tf.Session`, 然后 `run` 这个会话获取图的执行结果。

## 2.3 基准指标

Dogs vs. Cats Redux: Kernels Edition 比赛是 2017 年 3 月 2 日结束的, 在 Public Leadboard 已经公布了之前比赛排行榜和每名选手的分数, 我们的基准指标定在 Top-10 以内, 如下图所示, 第 10 名的分数是 0.03807, 所以我们的目标是要将最终的 score 在 0.038 以内











#	△1w	Team Name	Kernel	Team Members	Score 🏆	Entries	Last
1	—	Cocostarcu			0.03302	29	1y
2	—	guangsha			0.03305	34	1y
3	—	malr87			0.03483	89	1y
4	—	Bojan Tunguz			0.03506	435	1y
5	—	DeepBrain			0.03518	56	1y
6	—	lefant			0.03580	84	1y
7	—	matview			0.03778	40	1y
8	—	Bancroftway Systems (Andy...			0.03803	41	1y
9	—	Arvinder Chopra			0.03805	5	1y
10	—	Ranjeeta			0.03807	5	1y

图 13. Dogs vs. Cats Redux: Kernels Edition 比赛排行

# 3. 方法

## 3.1 数据预处理

### 3.1.1 训练数据类别整理

首先, Kaggle 提供的训练数据图片都放在一个文件夹中, 判断标签的时候比较麻烦, 我们先把 dog 和 cat 两类图片分别找出来放到不同的文件夹中。

### 3.1.2 异常值分析

根据用 ImageNet 预训练模型预测的结果, 我们可以看出 Kaggle 提供的训练数据中有一部分的图片不属于所标注的类别。比如下面图中这些是 dog 训练图像中的一部分标签与图像内容不符的“脏数据”, 也就是异常数据。这些异常数据有些是非猫非狗的图片(例如 dog.2614.jpg 和 dog.10237.jpg), 还有些是类别标错了(例如 dog.4334.jpg), 还有卡通图画(dog.8898), 还有背景非常复杂, 猫狗虽然有, 但是在图像中并不占主体的图片。这些异常数据的存在会影响最终的训练模型的质量, 降低模型提取特征的可靠性, 所以在训练之前, 我们需要对这些异常值进行清洗。

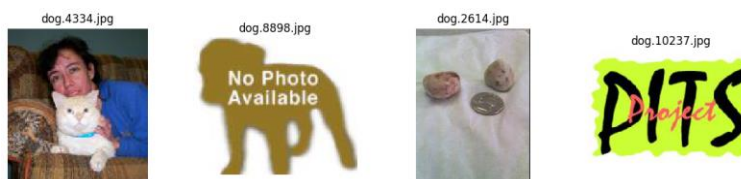


图 14. 训练集中存在的异常数据

之前在做预测房价, 生还率等作业时, 我们会通过四分位差等方法来选出明显离群的数据, 但与之前这些项目不同, 我们的训练数据是图片, 我们很难用直方图、图像均值、颜色平均值或者是其他某一种特征来发现问题图片。当然, 用肉眼筛查的办法把训练图片全部看一遍也不太现实。但是我们可以用 keras 提供的已经训练过 ImageNet 的模型来挑出分类不属于标记类别的图片, 然后从这些已经不多的图片中手动筛选标记正确的图片放回训练集中。

当然, 这个方法并不能保证完全把脏数据都挑出来, 但是至少可以保证经过异常值处理的训练集要比之前更加纯净, 更有利于网络对于类别特征的归纳。

首先, 我们先用 Xception 抽样预测一些图片(1/5 训练集, 2500 张图片), 将与标签不符的图像先挑出来, 预测的时候我们先使 top=20, 也就是预测前 20 类里包括特定的类别就认为是标签正确的, 图是在 cat 类别中 Xception 认为不属于 cat 的数据, 图是在 dog 类别中 Xception 认为不属于 dog 的数据。



图 15. Xception 预测结果不是 cat 的 cat 训练数据(top-20)



图 16. Xception 预测结果不是 dog 的 dog 训练数据(top-20)



从 Xception 挑出来的这些“脏数据”我们可以看到, cat 数据集中的脏数据很多是背景过于复杂但是包含猫的图片, 比如 cat.12227.jpg 还有 cat.6429.jpg; 有些是完全无关的图片, 比如 cat.8456.jpg; 还有卡通的 cat 图片, 比如 cat.11565.jpg; 还有分辨率过低很难判断是不是猫的图片, 比如 cat.2433.jpg; 当然, 还有些确实是 cat 但是被错误挑出来作为脏数据了, 比如 cat.2429.jpg, 我们会将 cat.8921.jpg、cat.6429.jpg、cat.252.jpg、cat.2429.jpg、cat.5492.jpg、cat.9596.jpg、cat.12476.jpg、cat3766.jpg、cat.2663.jpg、cat.3766.jpg 这些能够辨识主体是 cat 并且背景干扰不太严重的图片放回训练集, 其他的作为异常数据挑出。

对于挑选出来的 dog 的“脏数据”, 我们可以看出四张图中后面三张都是明显的异常数据, 但是挑出来的脏数据太少, 秉着挑选出的“脏数据”需要保持一定冗余, 然后再用人工筛查的思路, 我们减小 top 的值, 将更多的“脏数据”放进来, 经过试验, 当 top=5 的时候, “脏数据”中误差的比例在 1/3 左右, 比例基本合适, 如下图所示



图 17. Xception 预测结果不是 dog 的 dog 训练数据(top-5)

我们会将 dog.12155.jpg、dog.3035.jpg、dog.9160.jpg、dog.5170.jpg 再放回到训练集中。

基于以上的基本方法, 把数据规模扩大到全部的训练数据, 同时用 InceptionV3, InceptionResNetV2, Xception 三个预训练模型来分别挑出异常数据, 然后求这三个异常数据集的并集, 再手工把误判为异常数据的样本放回训练集中, 之所以要用三个模型预测的并集是保证尽量不漏掉异常数据, 因为每个模型预测的异常数据有可能是不同的。

### 3.1.2 图像缩放和归一化

InceptionV3, InceptionResNetV2, Xception 对于输入特征的维度要求都是(299,299,3), 所以训练之前应该把图片统一缩放到(299,299)尺寸, 同时还要对输入特征的值进行归一化处理, 将图像数据从[0, 255]缩放到[-1,+1]范围, 利用 keras 模型中\*.preprocess\_input 可以输入特征的归一化操作[\[11\]](#)。

## 3.2 执行过程

### 3.2.1 单模型训练

载入模型后, 我们需要给原始模型的后面加上 GlobalAveragePooling 层, Dropout 层, FC 层来构成完整的分类模型, 学习时, 原始模型的参数不变, 只训练后面加上的 FC 层的参数。

训练时, 我们从训练集中分出 10% 作为验证集, 每个 epoch 结束时计算验证集的 loss 和 accuracy, 验证模型的泛化性。

在单模型训练中, 我们采用同样的超参数来训练三个模型

```
batch_size = 16
epochs = 8
Optimizer = Adadelta
Learning_rate = 0.001
beta_1 = 0.9
beta_2 = 0.999
decay = 0
dropout_rate = 0.25
```

在设定学习率的时候, 我们并没有用 Learning rate decay 的方法来在训练过程中调整学习率的大小, 而是在后面用 Keras 的 Callback 模块中的 ReduceLROnPlateau() 函数来根据我们最终目标的 improving 来动态调整学习率, 也就是说如果我们 logloss 在一段时间内都没什么变化, 我们就会动态把学习率降低, 这么做可以提高训练的效率, 尽快使模型收敛。

### 3.2.2 生成结果

在训练过程中, 利用 Keras 的 Callback 模块中的 TensorBoard() 函数把训练过程中 loss 和 accuracy 的变化。InceptionV3, Xception, InceptionResNetV2 训练过程中的 accuracy 和 loss 变化如下三幅图所示。

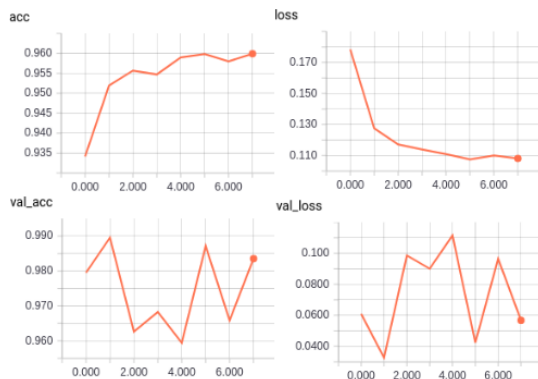


图 18. InceptionV3 训练 accuracy 和 loss 变化





图 19. Xception 训练 accuracy 和 loss 变化

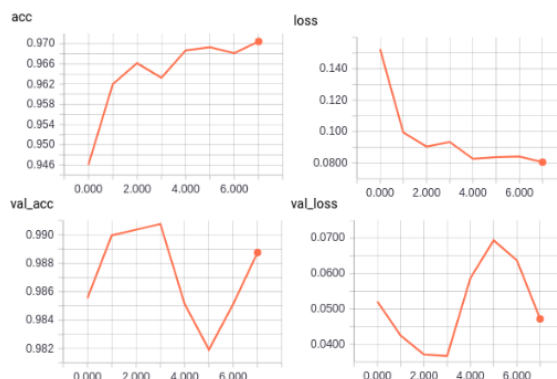


图 20. InceptionResNetV2 训练 accuracy 和 loss 变化

从训练过程中的 Accuracy 和 Loss 变化我们可以看出, 三个模型基本都收敛了, 而且 validation loss 和 train loss 差不多, 甚至比 train loss 还低, 证明过拟合情况不是很严重, 但是 val\_acc 和 val\_loss 都有不同程度的振荡, 这有可能与我们的训练 epoch 设置有关, 如果多训练几代, val 数据的 acc 和 loss 应该会更加收敛, 同时, 由于显存限制, batch\_size 没法设得太大, batch\_size 超过 16 时, 显存会超出, 如果 batch\_size 设得更大一些, 收敛的速度应该会更快, 这点可以作为下一步的改进点。

训练过程结束后, 我们会把最终的模型和参数用 model.save() 的方法固化下来, 然后再用固化的模型来预测测试集中的图片, 生成 predict.csv 在 Kaggle 上提交看每个模型的得分。结果如下图所示

<a href="#">InceptionResNetV2_pred.csv</a> 15 hours ago by <a href="#">conson</a> inception_resnet_v2 fix layer weights lock	0.06982	<input type="checkbox"/>
<a href="#">Xception_pred.csv</a> 15 hours ago by <a href="#">conson</a> Xception fix layer weights lock	0.07438	<input type="checkbox"/>
<a href="#">InceptionV3_pred.csv</a> 15 hours ago by <a href="#">conson</a> Inception V3 fix layer weights lock	0.07696	<input type="checkbox"/>

图 21. InceptionV3, Xception, InceptionResNet 三个模型的预测得分

从结果来看, InceptionResNetV2 的得分最高, 而 Inception\_V3 的得分最低, 这个结果和

在 ImageNet 的预测结果一致。但是即便是得分最高的 InceptionResNetV2, 得分也在 0.698, 与我们的目标(0.038 以下)还有差距, 当然, 如果在训练时使用数据扩增, 尝试调整优化器, 改善学习率, 得分应该还有可能提高, 但针对这个项目, 参数调优的改进空间不大, 下一步, 我们采用在 Kaggle 竞赛中特别常用的集成学习(Ensemble Learning)的方法来改善我们的模型。

### 3.3 完善

集成学习(Ensemble Learning)简单说, 就是集各种模型所长, 预测结果不依赖于某个模型, 而是多个模型综合的结果, 简单方式就是投票, 但是这个项目的评价指标使用的是对数损失, 不是简单分类, 所以不能简单做多模型平均或者是投票, 我们打算利用 Inception 模型的基本思路, 做 feature concat 将三个模型得到的特征组合, 然后再重新训练分类[\[12\]](#)。

在进行迁移学习的时候, 这三个模型的特征提取都是在 GlobalAveragePooling2D 层上完成的, Xception、InceptionResNetV2、InceptionV3 模型的特征数量分别为 2048、1536、2048, 针对这三个模型, 在训练时, 将训练集图像特征全部提取出来, 将特征 concat 成 1 维, 重新设计分类器, 重新进行迁移学习, 在测试时, 将测试时, 将这三个模型的测试集的图像特征全部提取出来, 按照和训练特征相同的方式, concat 成 1 维, 作为输入特征, 送入训练好的分类器, 进行预测。

将特征融合, 重新搭建的模型如下图所示:

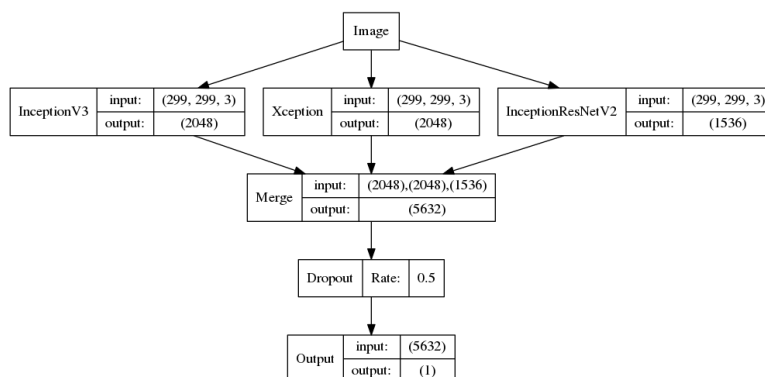


图 22. 特征融合后搭建的新模型架构

在模型融合中, 我们采用如下的超参数进行训练

```

batch_size = 128
epochs = 8
Optimizer = Adadelta
Learning_rate = 0.001
beta_1 = 0.9
beta_2 = 0.999
  
```

decay = 0

dropout\_rate = 0.5

训练过程中, loss 的收敛和 accuracy 的变化如下图所示



图 23. 模型融合训练 Accuracy 和 loss 变化

从训练过程中 loss 和 accuracy 的变化情况看, 相比单模型最好的结果 Xception, train\_accuracy 和 train\_loss 的最终结果差别不太大, 但是 validation\_accuracy 和 validation\_loss 明显模型融合更好, 而且从 accuracy 和 loss 的变化曲线来看, 融合模型的收敛明显更快, 这说明至少在 validation 数据集的泛化性上, 模型融合要更胜一筹。

将融合模型的预测结果上传 Kaggle, 我们得到了下面的 score:



图 24. 融合模型预测得分

相比于单模型最好的结果 Xception(score:0.06432), 模型融合的得分(score:0.03782)提高了 41%, 在已经公布的 Public Leaderboard 中, 可以排到第 8, 达到了预先的目标(Top-10)

## 4. 结果

### 4.1 模型的评价和验证

总结一下我们之前用过的模型和相应的得分:

模型名称	预测得分
InceptionV3	0.07696
Xception	0.07438
InceptionResNetV2	0.06982
Model Concat	0.03782

下面我们从网络结构和性能对比, 训练时间, 模型大小三个方面评价这几种模型

### 4.1.1 网络结构和性能

其中, InceptionV3 采用的是 group convolution 结构, 在每一个 block 里都有多个大小不同的卷积来分别提取图像特征然后在 block 的结点 stack 在一起, 保证了网络的宽度, 同时, 因为进行了大量的卷积分解, 将  $7 \times 7$  卷积分解成两个一维卷积( $1 \times 7$ ,  $7 \times 1$ ),  $3 \times 3$  分解成( $1 \times 3$ ,  $3 \times 1$ ), 计算的效率提高, 节省的算力用来加深网络。比起 Inception 系列最早的 googLeNet, 深度和宽度都提升了。

Xception 在 InceptionV3 的基础上又将空间卷积和升维降维的过程完全分开, 增加了网络深度, 同时因为每个卷积都只是空间卷积, 提高了模型解释时的计算速度。

InceptionResNetV2 是结合了 Incepiton 和 ResNet 两套架构的有点, 既有增加宽度的 group convolution, 又增加了跳远连接, 使得深度可以更深, 表现上好于 InceptionV3。

模型融合的方法本质上是利用集成学习的方法来结合各种模型的优势, 使每种模型提取出的特征都参与到最终的分类中, 比单个模型更鲁棒, 泛化性更好, 从结果上看, 得分的提高是数量集上的。

### 4.1.2 训练时间

三个单个模型训练的时间如下表所示

模型名称	训练时间(s)
InceptionV3	953
Xception	1286
InceptionResNetV2	1902

从上表可以看出, 最复杂的模型 InceptionResNetV2 训练的时间最长(0.5 hours), InceptionV3 所用的训练时间最短, 基本是训练 InceptionResNetV2 的  $1/2$ , 而 Xception 的训练时间是 InceptionResNetV2 的  $2/3$ , 训练时间和表现成正比, 需要训练越久的模型表现越好。但从表现和训练时间的平衡考虑, InceptionResNetV2 比 InceptionV3 多用了一倍的时间, 表现仅提高了 10%左右, 所以如果训练资源有限的话, InceptionResNetV2 可能不是最佳的选择, 而 Xception 用 1.3 倍于 InceptionV3 的训练时间, 获得了 3.3%的表现提高, 也没有在训练时间和表现的 balance 中更出色。从这个对比可以看出, 至少对这三个模型, 想要获得更好的表现, 就需要付出更大的训练代价。

而融合模型的训练时间要包括提取特征的时间和训练的时间, 提取三个模型特征的时间为 853s, 训练时间只有 5s, 也就是说训练融合模型的时间一共不到 860s, 这比训练 InceptionV3

的时间还低, 那么看来, 融合模型的训练时间成本要比单个模型还低。

### 4.1.3 模型大小

首先, 我们用列表的形式比较一下各个模型在训练结束后冻结保存下来的大小

模型名称	模型大小(MB)
InceptionV3	88.1
Xception	83.9
InceptionResNetV2	219.4

可以看出来, InceptionResNetV2 的模型最大, 而 Xception 的模型最小, Xception 用比 InceptionV3 更小的内存占用, 获得了比 InceptionV3 更好的表现, 相比于 InceptionResNetV2(模型大小是 Xception 的 2.6 倍), 表现也只降低了 6%左右, 所以从 memory cost 和 performance 的平衡来讲, Xception 的表现也要更好一些。

而融合模型的大小基本上就是三个模型大小的总和, 就是  $88.1\text{MB} + 83.9\text{MB} + 219.4\text{MB} = 391.4\text{MB}$ , 之前我们讨论了模型的训练时间, 融合模型的训练时间最短, 但是融合模型的大小最大, 这么大的模型基本就是比赛和研究的时候可以采用, 真正部署是很困难的。

## 4.2 合理性分析

从模型最终的表现看, 符合理论上的预期(和三个模型在 ImageNet 数据集上的表现的趋势一致), 当然单个模型可以改变的条件很多, 比如增加深度, 调整超参数, 在数据预处理的时候加入数据扩增等, 这些都会最终影响模型的表现, 但我们在实验过程中, 除了模型本身外, 保持其他的参数都相同, 所以最终的得分基本上可以体现模型的性能差别。从我们前面的分析能看出, 单论模型的性能, InceptionResNetV2 当然是这三个模型中最高的, 但是从训练时间成本和模型大小的角度上去考虑, 真正部署时, 我们可能会考虑用性价比更高的 Xception 模型, 虽然损失了一点性能, 但是成本却成倍降低了。

对于模型融合, 在 Kaggle 比赛中, 基本排名靠前的方法都或多或少用到集成学习的方法, 我们所用的模型融合, 也是集成学习的一种形式, 这也验证了多模型的融合确实可以显著提高模型的表现。融合模型的训练成本极低, 性能高出单个模型一个数量级, 但是缺点就是模型太大, Memory cost 太高, 使这种方法在实际部署时有很大困难。

## 5. 项目结论

### 5.1 结果可视化

以融合模型作为我们最终提交的模型, 从测试集中随机选取几张图做预测, 来直观看一下预测的准确性, 结果如下图所示:



图 25. 预测结果可视化

图中, 标题的命名前面的数字是测试集中的图片名称, 下划线之后的是对这张图像预测的类别, 可以看到对随机选取的这几张图像的预测都是准确的。

### 5.2 对项目的思考

通过这个项目, 我们提高了对 CNN 的理解和应用, 对于使用 Keras 作为前端 API, Tensorflow 作为后端框架来进行深度学习模型的搭建进行了实战操作。

同时, 比较了 Inception 系列最近的三个模型 InceptionV3, Xception, InceptionResNetV2 的表现。从这三个模型的比较来看, Xception 虽然不是表现最好, 而且模型更小, 训练时间也不是最长, 是综合表现最好的模型, 但是目前又有一些模型体量更小, 表现更好的模型出现, 比如 NasNet, DenseNet 等, 以后也会慢慢尝试。

为达到我们开始设定的 Cats vs. Dogs Top-10 的目标, 我们又采用集成学习的方法将这几个模型融合起来, 尝试了竞赛中常用的集成学习的手段来提高准确性。结果也证实多个模型融合确实要比单个模型的 Performance 高很多。

当然, 提高模型表现的方法绝不止有模型融合这一种, 之前也尝试过用单模型+XGBoost 的方法来提高预测准确性, 但是并没有获得 Performance 的提高, 在以后的学习过程中, 还会继续进行研究新的方法来提高准确性。

## 5.3 需要作出的改进

1. 单模型的 Performance 实际上并没有达到最优, 改善优化器, 数据扩增, 学习率调整都会使单模型的表现有所提升;
2. 单模型在 validation 数据集上 accuracy 和 loss 有不同程度的振荡, 可以增加训练 epoch 再试一试;
3. Xception 在得分上高于 InceptionResNet 这点, 因为与 ImageNet 数据集的结果不一样, 所以还是存疑的, 但是检查过训练数据, 训练参数, 前处理方式, 以及预测过程, 没有看到这两个模型在训练预测的过程中有纰漏, 所以只能把这点留到下一步的改进 ;
4. 已经谈到由于模型的 Memory cost 问题, 融合模型很难部署到实际项目中, 那有没有能够部署到项目中(比如手机应用), 并且 Performance 还可以接受的方法呢?这是我们下一步要研究的方向;
5. 单模型+XGBoost 的效果比较差, 我们采用 InceptionResNetV2 提取特征, XGBoost 拿这些特征来分类的思路, 想要获取至少高于 InceptionResNetV2 的 Performance, 但是实际得分要比 InceptionResNetV2 还差, 为什么会出现这种情况, 我们下面还要继续研究

xgb\_submission.csv  
13 days ago by conson  
xgb after grid search

0.22014



图 26. InceptionResNetV2+XGBoost 的预测得分



## 参考文献

- [1] Rosenblatt, Frank et al. (Cornell Aeronautical Laboratory), "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain". Psychological Review 1958
- [2] A. Krizhevsky et al. (Toronto), "ImageNet Classification with Deep Convolutional Neural Networks". NIPS 2012
- [3] "卷积神经网络 CNN 图解" <http://xilinx.eetrend.com/article/10827>
- [4] C. Szegedy et al. (Google), "Rethinking the Inception Architecture for Computer Vision". CVPR 2016
- [5] F. Chollet et al. (Google), "Xception: Deep Learning with Depthwise Separable Convolutions". CVPR 2017
- [6] C. Szegedy et al. (Google), "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning". AAAI 2017
- [7] C. Szegedy et al. (Google), "Going Deeper with Convolutions". CVPR 2015
- [8] KM He et al. (Microsoft), "Deep Residual Learning for Image Recognition". CVPR 2016(best paper)
- [9] K. Simonyan et al. (Oxford), "Very Deep Convolutional Networks for Large-Scale Image Recognition". ICLR 2015
- [10] S. Ioffe et al. (Google), "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". PMLR 2015
- [11] "Keras 第三方中文文档", <http://keras-cn.readthedocs.io/en/latest/other/application/>
- [12] "手把手教你如何在 Kaggle 猫狗大战冲到 Top2%", <https://zhuanlan.zhihu.com/p/25978105>