

# **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

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 .....	5
2. 分析 .....	5
2.1 数据可视化 .....	5
2.2 算法和技术 .....	6
2.2.1 人工智能(AI), 机器学习(Machine Learning)和深度学习(Deep Learning) .....	6
2.2.2 卷积神经网络 .....	8
2.2.3 实现技术 .....	10
2.3 基准指标 .....	10
3. 方法 .....	11
3.1 数据预处理 .....	11
3.1.1 模型选取和迁移学习 .....	11
3.1.2 数据预处理 .....	13
3.2 执行过程 .....	13
3.2.1 单模型训练 .....	13
3.2.2 生成结果 .....	13
3.3 完善 .....	15
4. 结果 .....	17
4.1 模型的评价和验证 .....	17
4.1.1 网络结构和性能 .....	18
4.1.2 训练时间 .....	18
4.1.3 模型大小 .....	19
4.2 合理性分析 .....	19
5. 项目结论 .....	20
5.1 结果可视化 .....	20

5.2 对项目的思考 .....	20
5.3 需要作出的改进 .....	21

# 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

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

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。

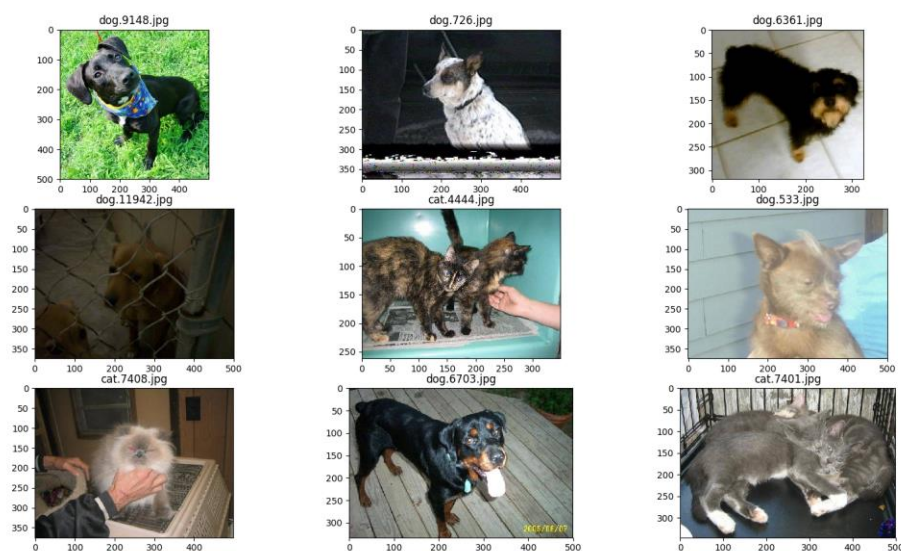


图 2. 训练集图片



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 人工智能, 机器学习, 和深度学习

因为本项目是要利用深度学习来解决图片分类问题, 所以, 我们首先来搞清楚什么是深度学习。

在 2016 年, Google DeepMind 的 AlphaGo 打败了韩国的围棋大师李世乭九段, 2017 年, AlphaGo 又击败了柯洁九段, 人工智能(AI)的风头一时无两, 引来了媒体的争相报道和越来越多圈外人的关注。但是在媒体描述 DeepMind 的胜利的时候, 将人工智能(AI)、机器学习(Machine Learning)和深度学习(Deep Learning)这些词都用上了。那这些概念之间究竟是什么关系, 深度学习又是怎样一种存在呢?

首先, 我们用一张图来表示它们三者的关系和应用



图 4. 人工智能、机器学习和深度学习关系

正如在图 4 中我们看到的, 人工智能是最早出现的, 也是最大、最外侧的同心圆; 其次是机器学习, 稍晚一点; 最内侧, 是深度学习, 当今人工智能大爆炸的核心驱动。五十年代, 人工智能曾一度被极为看好。之后, 人工智能的一些较小的子集发展了起来。先是机器学习, 然后是深度学习。深度学习又是机器学习的子集。深度学习造成了前所未有的巨大的影响。

### ①. 人工智能(AI) - 为机器赋予人的智能

1956 年, 几个计算机科学家相聚在达特茅斯会议(Dartmouth Conferences), 提出了“人工智能”的概念。人工智能的先驱们当时就梦想着用当时刚刚出现的计算机来构造复杂的、拥有与人类智慧同样本质特性的机器。这就是我们现在所说的“**强人工智能**”(General AI)。这个无所不能的机器, 它有着我们所有的感知(甚至比人更多), 我们所有的理性, 可以像我们一样思考。

人们在电影里也总是看到这样的机器: 友好的, 像星球大战中的 C-3PO; 邪恶的, 如终结者。强人工智能现在还只存在于电影和科幻小说中, 原因不难理解, 我们还没法实现它们, 至少目前还不行。

我们目前能实现的, 一般被称为“**弱人工智能**”(Narrow AI)。**弱人工智能是能够与人一样, 甚至比人更好地执行特定任务的技术。**例如, Pinterest 上的图像分类; 或者 Facebook 的人脸识别。

这些是弱人工智能在实践中的例子。这些技术实现的是人类智能的一些具体的局部。但它们是如何实现的? 这种智能是从何而来? 这就带我们来到同心圆的里面一层, 机器学习。

### ②. 机器学习(Machine Learning) - 一种实现人工智能的方法

**机器学习最基本的做法, 是使用算法来解析数据、从中学习, 然后对真实世界中的事件做出决策和预测。**与传统的为解决特定任务、硬编码的软件程序不同, 机器学习是用大量的数据来“训练”, 通过各种算法从数据中学习如何完成任务。

机器学习直接来源于早期的人工智能领域。传统算法包括决策树学习、推导逻辑规划、聚类、强化学习和贝叶斯网络等等。众所周知，我们还没有实现强人工智能。早期机器学习方法甚至都无法实现弱人工智能。

### ③. 深度学习(Deep Learning) - 一种实现机器学习的技术

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。**

例如，我们可以把一幅图像切分成图像块，输入到神经网络的第一层。在第一层的每一个神经元都把数据传递到第二层。第二层的神经元也是完成类似的工作，把数据传递到第三层，以此类推，直到最后一层，然后生成结果。

每一个神经元都为它的输入分配权重，这个权重的正确与否与其执行的任务直接相关。最终的输出由这些权重加总来决定。

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 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

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。

卷积网络通过一系列方法，成功将数据量庞大的图像识别问题不断降维，最终使其能够被训练。CNN 最早由 Yann LeCun 提出并应用在手写字体识别上 (MINST)。LeCun 提出的网络称为 LeNet，其网络结构如下：

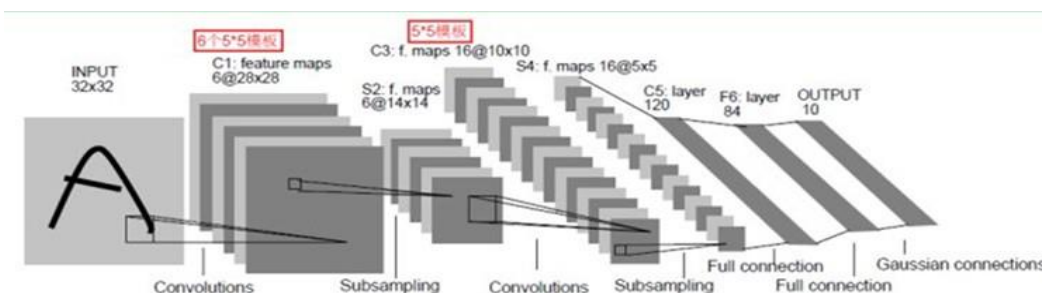


图 5 LeNet 网络架构



这是一个最典型的卷积网络，由卷积层、池化层、全连接层组成。其中卷积层与池化层配合，组成多个卷积组，逐层提取特征，最终通过若干个全连接层完成分类。

卷积层完成的操作，可以认为是受局部感受野概念的启发，而池化层，主要是为了降低数据维度。

综合起来说，CNN 通过卷积来模拟特征区分，并且通过卷积的权值共享及池化，来降低网络参数的数量级，最后通过传统神经网络完成分类等任务。

下面, 介绍一下 CNN 的两个基本结构卷积(Convolution)和池化(Pooling)。

### ①. 卷积(Convolution)

卷积运算的定义如下图所示：

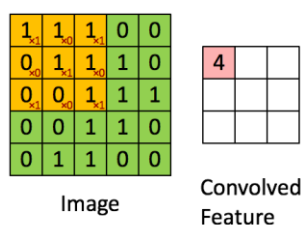


图 6. 卷积运算示例

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。

### ②. 池化(Pooling)

池化听起来很高深，其实简单的说就是下采样。池化的过程如下图所示：

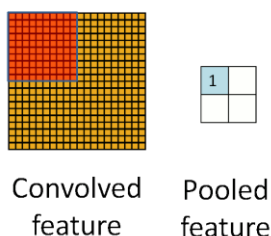


图 7. 池化运算示例

上图中，我们可以看到，原始图片是 20x20 的，我们对其进行下采样，采样窗口为 10x10，最终将其下采样成为一个 2x2 大小的特征图。

之所以这么做的原因，是因为即使做完了卷积，图像仍然很大（因为卷积核比较小），所以为了降低数据维度，就进行下采样。之所以能这么做，是因为即使减少了许多数据，特征的

统计属性仍能够描述图像，而且由于降低了数据维度，有效地避免了过拟合。

在实际应用中，池化根据下采样的方法，分为最大值下采样（Max-Pooling）与平均值下采样（Mean-Pooling）。

### 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

Keras 是一个高层神经网络 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 8. Dogs vs. Cats Redux: Kernels Edition 比赛排行

## 3. 方法

### 3.1 数据预处理

#### 3.1.1 模型选取和迁移学习

在 CNN 模型的选取上, 目前 Google 的 Inception 系列和 Microsoft 的 ResNet 系列是当前比较流行两种架构, 都获取过 ILSVRC 的冠军。

我们选用了 Keras 的库中提供的 InceptionV3, ResNet50 和结合两种架构 InceptionResNetV2 模型, 可以比较 Inception 和 ResNet 在分类任务上的优劣, 同时可以研究将这两者结合带来的实际好处。

以下三张图分别是 InceptionV3(图 9), ResNet50(图 10), InceptionResNetV2 结构中一个 block(图 11)。Inception 的特征结构在 Group Convolution, 就是在一个 block 中并行几种不同的卷积层, 然后将这些卷积层得到的特征 comcat 在一起再传到下一层; 而 ResNet 的特点在于出现了跳远连接结构, 可以有效抑制网络深度加深带来的梯度弥散和梯度爆炸问题; InceptionResNet 则是结合了这两种结构的有点, 既有 Group convolution 也保留了跳远连接结构。

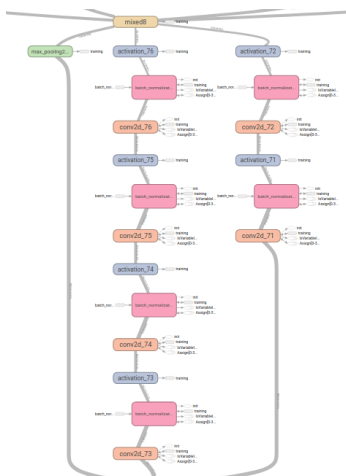


图 9. InceptionV3 block 结构

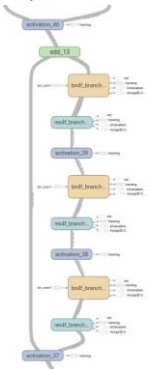


图 10. ResNet50 block 结构

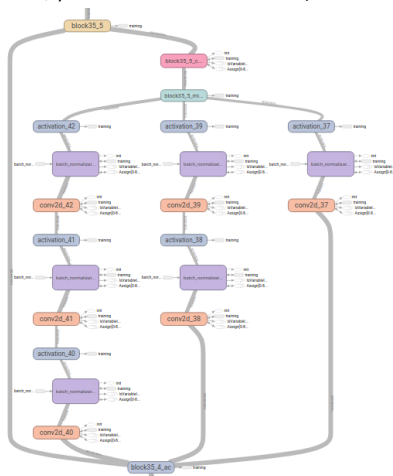


图 11. InceptionResNetV2 block 结构

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。

### 3.1.2 数据预处理

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。

其次, 因为 InceptionV3 和 InceptionResNetV2 两种模型的输入要求是(299,299,3)图像, 而 ResNet50 要求(224,224,3)图像, 所以还要在输入模型前将图片 resize 到指定大小。

## 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 = 10
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, ResNet50, InceptionResNetV2 训练过程中的 accuracy 和 loss 变化如下三幅图所示。

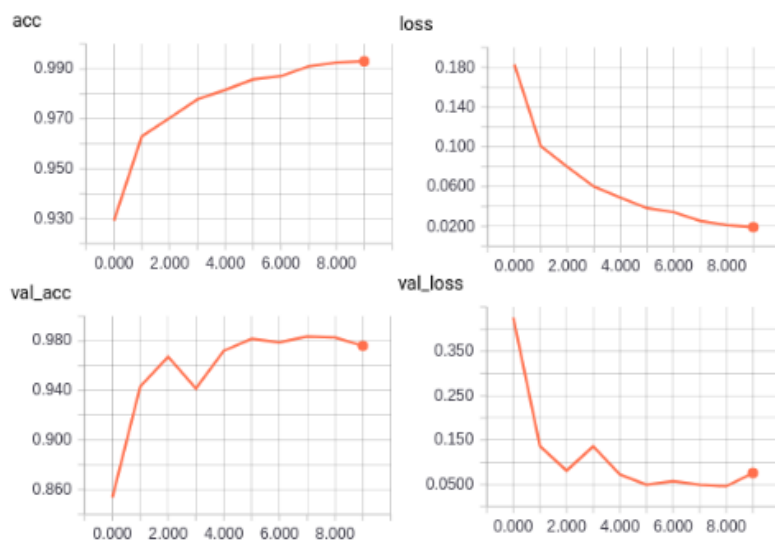


图 12. InceptionV3 训练 accuracy 和 loss 变化

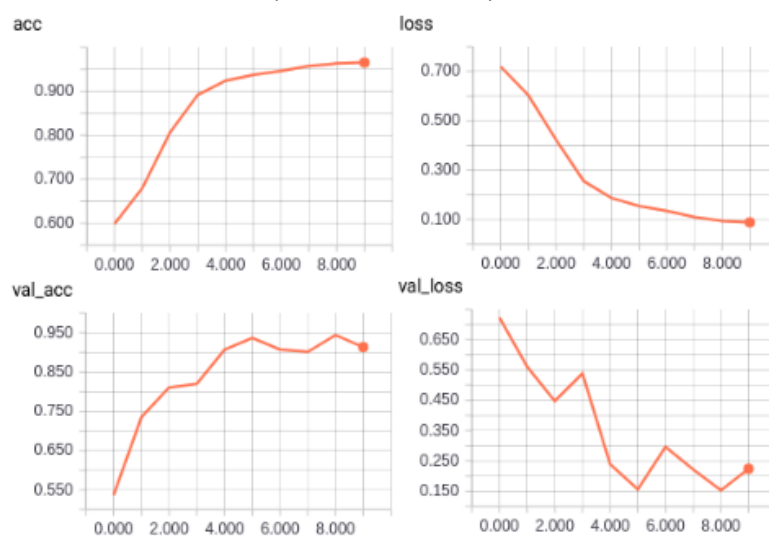


图 13. ResNet50 训练 accuracy 和 loss 变化

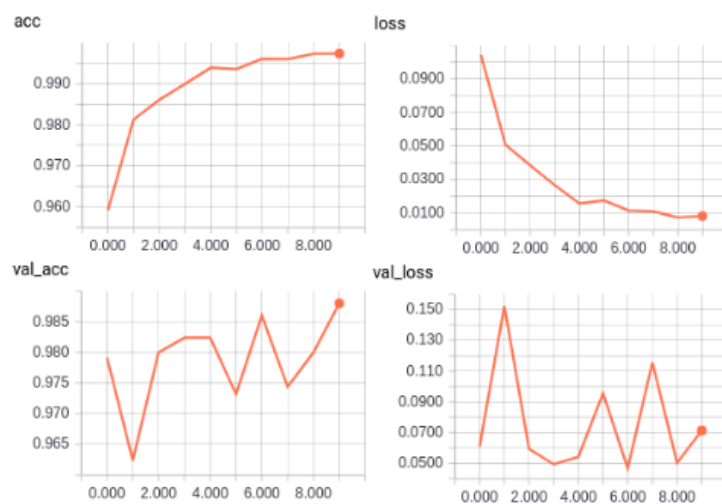


图 14. InceptionResNetV2 训练 accuracy 和 loss 变化

从训练过程中的 Accuracy 和 Loss 变化我们可以看出, InceptionResNetV2 的效果都最好, 而 ResNet50 的效果相对最差, 但是 InceptionResNetV2 的 validation accuracy 和 loss 的在每个 epoch 振荡的比较厉害, 这可能是由于我们的 batch\_size 太小导致的, 受限于显存大小, batch\_size 超过 16 时, 显存会超出, 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> 5 minutes ago by <a href="#">conson</a> InceptionResNetV2 keras final	0.12116	<input type="checkbox"/>
<a href="#">ResNet50_pred.csv</a> 5 minutes ago by <a href="#">conson</a> ResNet50 Keras final	0.19244	<input type="checkbox"/>
<a href="#">InceptionV3_pred.csv</a> 13 hours ago by <a href="#">conson</a> InceptionV3 Keras final	0.12516	<input type="checkbox"/>

图 15. InceptionV3, ResNet50, InceptionResNet 三个模型的预测得分

从结果来看, InceptionResNetV2 的得分最高, 而 ResNet50 的得分最低, 这也和我们从训练时的 loss 和 accuracy 的趋势一致, 但是即便是得分最高的 InceptionResNetV2, 得分也在 0.12, 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, 但是经过尝试后, 最多也就能到 0.07 左右, 所以, 我们改用在 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。

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, ResNet50、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:

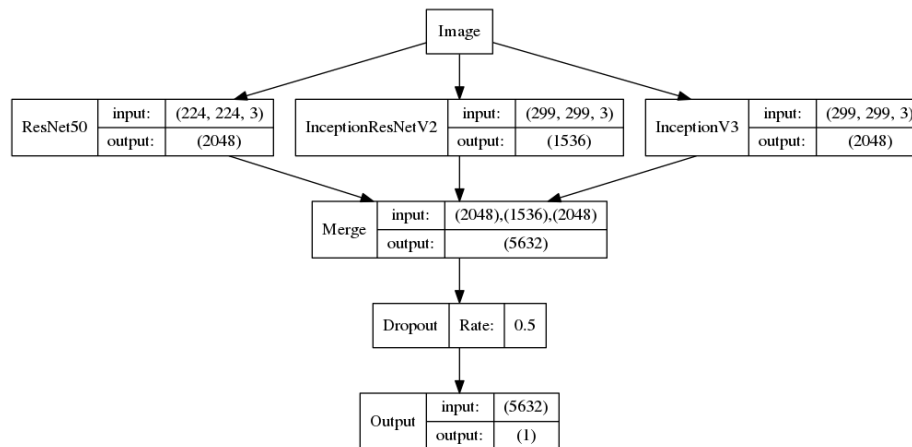


图 16. 特征融合后搭建的新模型架构

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



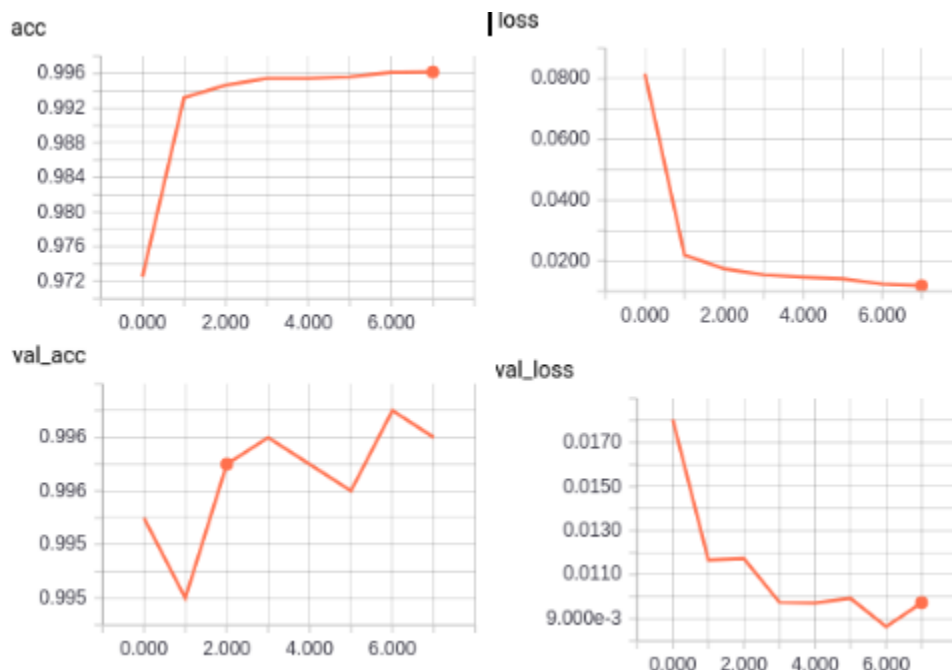


图 17 模型融合训练 Accuracy 和 loss 变化

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 InceptionResNetV2, train\_accuracy 和 train\_loss 的差别不太大, 但是 validation\_accuracy 和 validation\_loss 明显模型融合更好, 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 18 融合模型预测得分

相比于单模型最好的结果 InceptionResNetV2(score:0.12116), 模型融合的得分 (score:0.03689)提高了近 70%, 在已经公布的 Public Leaderboard 中, 可以排到第 7, 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	0.12516
ResNet50	0.19244
InceptionResNetV2	0.12116
Model Concat	0.03689

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, 我们用的这个版本一共有 16 层卷积(后面的 group convolution 单元一个 block 算一层)。比起 Inception 系列最早的 googLeNet, 深度和宽度都提升了。

ResNet50 是在传统的 CNN 网络基础上增加了跳远连接, 深度更深, 一共是 50 层卷积, 分为 16 个 Residual block, 跳远连接的加入使得训练更深的网络结构成为可能, 但是相比于 InceptionV3, 我们看到 ResNet50 的预测准确率明显更低。所以, 深度不是决定网络性能的唯一指标, 更深不一定更有效。

InceptionResNetV2 是结合了 Inception 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, 当然性能上也要明显好于 InceptionV3 和 ResNet。

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	3231
ResNet50	2411
InceptionResNetV2	6863

从上表可以看出, 最复杂的模型 InceptionResNetV2 训练的时间最长, 接近 2 个小时, ResNet 所用的训练时间最短, 几乎是训练 InceptionResNetV2 的 1/3, 而 InceptionV3 则将近是 InceptionResNetV2 的 1/2。当要获取最好的性能时, 我们需要更大的训练代价, 而如果要平衡 Train Cost 和 Performance, 那就单个模型而言显然 InceptionV3 是一个更好的选择, 因为预测的份上 InceptionV3 比 InceptionResNetV2 只高 0.004, 但是训练的时间只有它的一半。

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, 这比单模型训练用时最少的 ResNet50 都还低一个数量级, 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	262.4
ResNet50	283.3
InceptionResNetV2	654.0

可以看出来, 得分最高的 InceptionResNetV2 的模型最大, InceptionV3 的预测得分比 InceptionResNetV2 只高 0.004, 模型大小却是其 1/2 不到, 而 InceptionV3 性能比 ResNet 强, 模型大小还比 ResNet50 稍小, 所以 InceptionV3 应该是这三个模型中 Performance-Memory Cost 的平衡中性价比最高的。

而融合模型的大小基本上就是三个模型大小的总和, 后面加上的全连接层几乎可以不计, 所以就是  $262.4\text{MB} + 283.3\text{MB} + 654.0\text{MB} = 1199.7\text{MB}$ , 超过 1 个 G, 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, 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 InceptionV3 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 19. 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, 比较了两种流行的网络模型框架 Inception 系列和 ResNet 系列的代表 InceptionV3 和 ResNet50 的优劣, 以及结合了两种框架优点的 InceptionResNetV2 网络模型的优势(Performance 高)和带来的问题(Train cost 和 Memory cost 高)。从这三个模型的比较来看, InceptionV3 是更经济, Performance 也有保障的单模型, 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. 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;

3. 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 20. InceptionResNetV2+XGBoost 的预测得分