

猫狗大战 (Dog vs.Cat)

机器学习工程师纳米学位毕业项目

李达

2018-4-6

目 录

1 问题的定义	3
1.1 项目概述	3
1.2 问题陈述	3
1.3 评价指标	3
2 分析	5
2.1 数据的探索	5
2.2 探索性可视化	5
2.3 算法和技术	6
2.3.1 深度学习的历史	6
2.3.2 卷积神经网络	7
2.3.3 相关技术和工具	8
2.4 基准模型	9
3 方法	10
3.1 网络模型的选择	10
3.2 数据预处理	10
3.3 执行过程	11
3.3.1 生成数据	11
3.3.2 构建、训练模型	12
3.3.3 预测	14
3.4 完善	14
4 结果	17
4.1 模型的评价与验证	17
4.2 合理性分析	18
5 项目结论	19
5.1 结果可视化	19
5.2 思考	20
5.3 改进	20
参考文献	21

1 问题的定义

1.1 项目概述

项目来自kaggle竞赛：Dogs vs. Cats，所属领域是计算机视觉（Computer Vision），所要解决的问题是图像分类（Image Classification）——对猫或狗的图片进行正确分类。

图像分类是计算机视觉研究中的基础方向，基于该方向的研究成果和方法可应用于诸如目标检测以及图像摘要生成等同领域的其他方向。一年一度的ImageNet挑战赛^[1]代表着这些方向的世界先进水平。2012年，以卷积神经网络^[2](Convolutional Neural Network, CNN)为代表的深度学习方法开始在挑战赛中独领风骚。之后几年，基于CNN的网络不断有新的研究成果，并且持续问鼎该挑战赛的冠军，可见CNN的强大之处。

作为一个典型的图像分类问题，本项目拟使用CNN网络来构建模型。通过本项目，可深入理解不同CNN网络的结构和特点，并系统学习图像分类问题分析方法和处理流程，同时能够综合运用课程中所学的机器学习知识来解决实际问题。

项目使用kaggle提供的数据，训练集包括25000张图片，其中12500张被标记为猫，12500张被标记为狗；测试集包含12500张未标记的图片。

1.2 问题陈述

项目属于机器学习的范畴，对应监督学习中的二分类问题(1对应狗，0对应猫)。具体来讲，就是选择适当的机器学习模型并使用带标签的数据集来训练该模型(求解参数)，完成训练后将模型泛化至未知数据，即对测试集中的每一张图片，预测其为狗的概率。

从数据的角度看，输入图片是一个包含若干像素值的张量或者向量，经过模型运算输出特定类别(狗)的概率。机器学习的目的是，当输入是一张狗的图片，使模型输出狗的概率尽可能接近1；反之，输入猫的图片，输出尽可能接近0。

如前文所述，CNN是一种专门用来处理具有类似网格结构的数据（图片）的神经网络，因此用CNN解决图像分类问题非常合适。CNN网络将输入数据进行若干卷积层、池化层、批量归一化层（Batch Normalization）以及全连接层处理（新的网络很少再出现全连接层），在输出层定义一个节点做sigmoid激活得出狗的概率；定义交叉熵（Cross Entropy）为模型的损失函数；基于梯度下降算法（Gradient Descent）并配合特定的优化（如Mini-Batch, Momentum, 正则化等）找到参数的最优解（或次优解）。

1.3 评价指标

分类问题有多种评价指标，包含正确率（Accuracy）、F1分数、ROC-AUC分数等。上述指标主要用于衡量模型对正例和负例判断的准确度，即好的模型既要尽可能的多预测真正正例（True Positive），也要尽量避免假正例（False Positive）的情况。

在猫狗识别项目中，除了要符合上述标准，还需要模型对分类结果更加确定，即，在真正例（狗）的情况下，模型输出的概率要尽可能接近1；在真负例的情况下，概率要尽可能接近0。为此，采用对数损失这一指标来对模型进行评价，这也kaggle评分所采用的方式。

对数损失（LogLoss）用如下公式表示：

$$\text{logloss} = -\frac{1}{n} \sum_{i=0}^n [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \quad (\text{公式1-1})$$

其中：

- n是图片的数量
- \hat{y}_i 是模型预测为狗的概率
- y_i 是类别标签，1对应狗，0对应猫
- \log 是自然对数

理想情况下，当标签为1且模型预测概率为1，或者标签为0且模型预测概率为0时，公式1-1的输出为0。可见，对数损失越小，模型的表现越好。

2 分析

2.1 数据的探索

输入数据全部来自于[kaggle](#)，可以从网站上进行下载，包含训练数据和测试数据两部分，基本都是各种猫或狗的图片。其中，训练数据共有25000张图片，每张图片都带有类别标签，猫和狗的标签各占一半。由于猫狗的比例为1:1，因此不需要考虑类别不平衡问题。测试数据共有12500张图片。

2.2 探索性可视化

随机选取样本图片进行观察，如图2-1。基本都是RGB 3 channel的彩色照片，包含不同品种的猫和狗，并且姿态、背景各异。其中，有猫狗单独的照片，有由人类牵引、拥抱的合照，也有多个猫或狗的合照。大部分图片都算清晰可辨，小部分相对难辨认一些，如第二排第一列，根据标签，应该是一只初生的小猫，但人眼其实很难认出是小猫还是小狗。

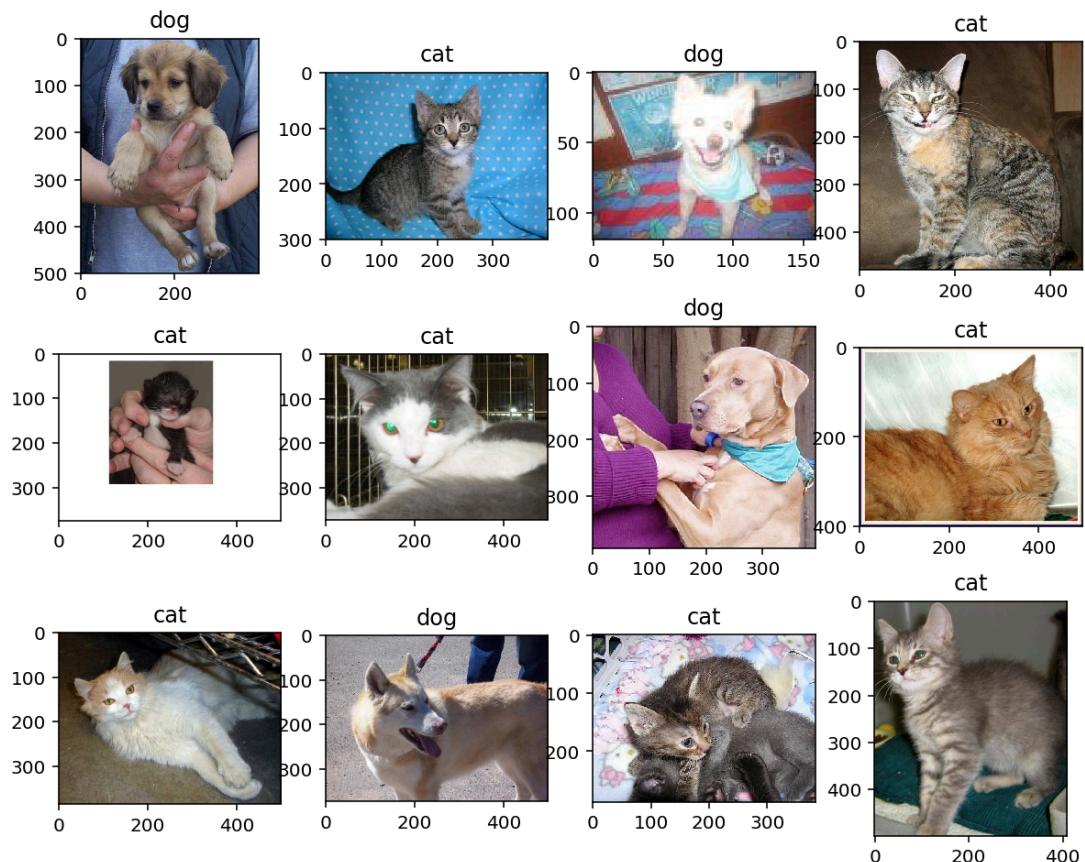


图2-1 训练样本可视化

图片分辨率大小（Height×Width）的分布如图2-2。可见，样本的分辨率各不相同，在输入模型进行处理之前，需要按照模型对分辨率的要求统一进行调整。

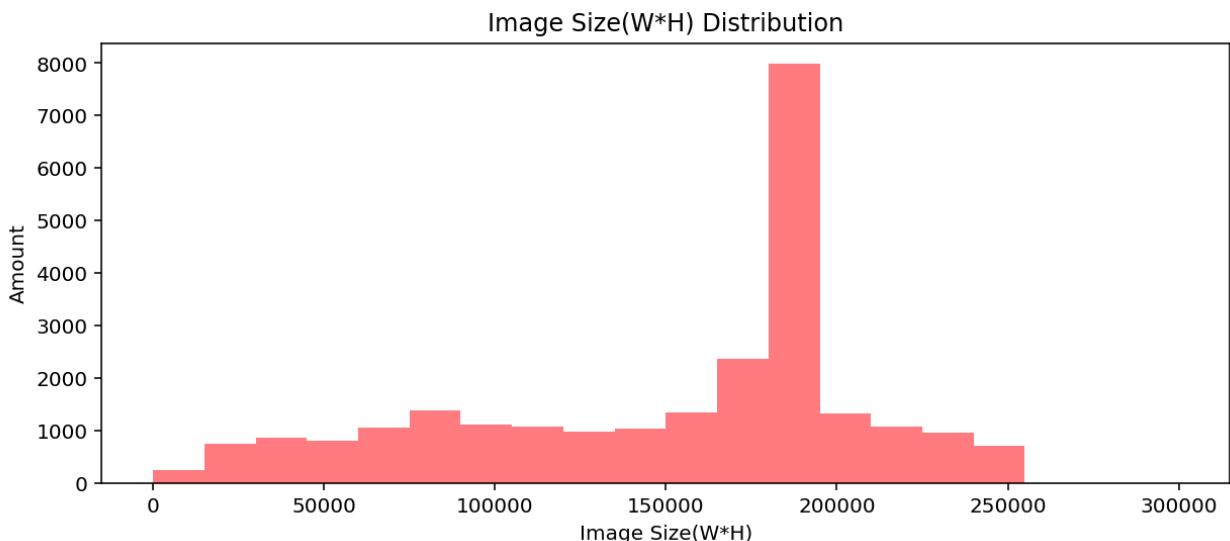


图2-2 图片分辨率大小分布

此外，约1/10左右的图片分辨率大小不足 200×200 ，甚至更小，猜测可能是使用早期的相机所拍摄，或是局部的截图，若这些图片质量不佳或异常，可能会影响模型的学习和预测。

2.3 算法和技术

2.3.1 深度学习的历史

深度学习并非最近才出现的技术，它的历史可以追溯到20世纪40年代，其代表是M-P神经元和感知机（Perceptron），这是神经网络发展的第一次高潮。典型的深度学习模型就是很深层的神经网络（远大于一层隐藏层），隐藏层变多，神经元连接权、阈值等参数会更多，相应的模型复杂度就会更高，这意味着它能完成更复杂的学习任务。但一般情况下，复杂模型的训练效率低，易陷入过拟合。

二十世纪80年代，联结主义潮流的一个重要成就是反向传播在训练深度神经网络中的成功使用以及反向传播算法（Back Propagation）的普及，截止到目前，这依然是训练深度模型的主导方法。BP算法的迅速走红，掀起了神经网络的第二次高潮。但是，二十世纪90年代中期，随着统计学习理论和支持向量机的兴起，神经网络学习的理论性质不够清楚、试错性强、在使用中充斥大量的“窍门”，神经网络的研究进入低谷。

而随着云计算、大数据时代的到来，计算能力的大幅提高可以缓解训练低效性，训练数据的大幅增加则可降低过拟合风险，因此，以“深度学习”为代表的复杂模型开始收到人们的关注，并且迎来了第三次高潮。

总之，深度学习作为机器学习的一种方法，在过去的几十年发展中，大量的借鉴了关于人脑、统计学和应用数学的知识。近年来，得益于更强大的计算机、更大的数据集和能够训练更深网络的技术，深度学习的普及性和实用性都有了极大的发展。

2.3.2 卷积神经网络

卷积网络提供了一种方法来特化神经网络，使其能够处理具有清楚的网格结构拓扑的数据，以及将这样的模型扩展到非常大的规模。这种方法在二维图像拓扑上是最成功的。

卷积网络的历史始于神经科学实验，远早于相关计算模型的发展。神经生理学家David Hubel和Torsten Wiesel把猫作为实验对象，试图弄清楚神经元在视觉皮层的工作原理。他们发现皮层空间映射的特点，即皮层中相邻的神经元只与视觉区域中某些相邻的部分相关联。并且他们发现这些神经元存在层级关系，其中包含有简单细胞（simple cells），会被不同方向的边缘（edges）所激活；进而连接至复杂细胞（complex cells），它们能够响应端点的移动；最终，皮层认出了边角、形状乃至完整的物体。

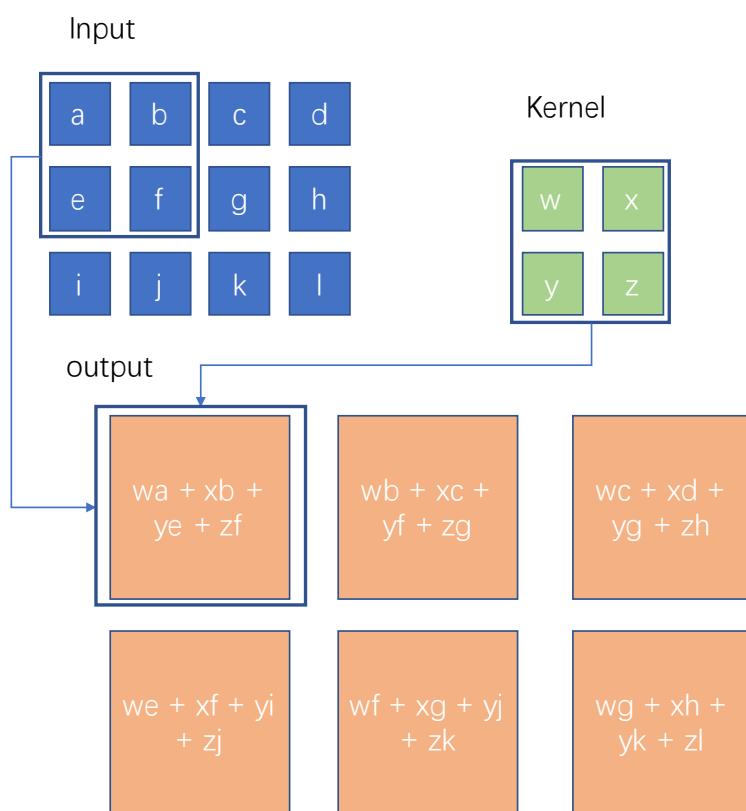


图 2-3 卷积运算原理

与常规的神经网络的结构基本一致，卷积网络的不同之处在于需要训练卷积层，而卷积层更能保留输入的空间结构。卷积层运算的原理如图2-3，左上蓝色部分为input，可以认为是一个二维张量；右上绿色部分为卷积核kernel或称为filter，其大小通常远小于input。卷积运算就是将kernel按照某个步长（step）依次滑过整个input张量，并与相应的input做矩阵

乘法，计算点积（dot product），得到图中橙色部分。一般地，上述结果还需要经过一个非线性激活单元（ex. ReLU），最终得到该卷积核对应的激活映射层（activation map）。

实际上，卷积运算通过三个重要的思想来改进机器学习系统：稀疏交互、参数共享以及等变表示。以稀疏交互和参数共享为例，当处理一张图像时，输入图像可能包含成千上万个像素点，可以通过只占用几十到上百个像素点的kernel来检测一些小的有意义的特征，如边缘（edges）。这意味着需要存储的参数更少，需要的计算量也更少。这正是卷积网络比全连接网络计算效率更高、更容易训练的原因所在。

图2-4视觉化了卷积神经网络（VGG-16）的处理过程。一般地，多个卷积层堆叠形成卷积网络，每个卷积层采用多个卷积核，每一个卷积核产生一个激活映射。前面几层的卷积核一般代表了一些低阶的图像特征，比如像一些边缘特征。对于中间层，可以得到一些更复杂的图像特征，其内容看起来更加丰富，比如边角和斑点等。而对于那些高阶的特征，可以反映出更加丰富的内容，如轮廓、形状乃至完整的物体。最后，通过一个简单的线性可分的分类器，即可完成对输入图像的分类。

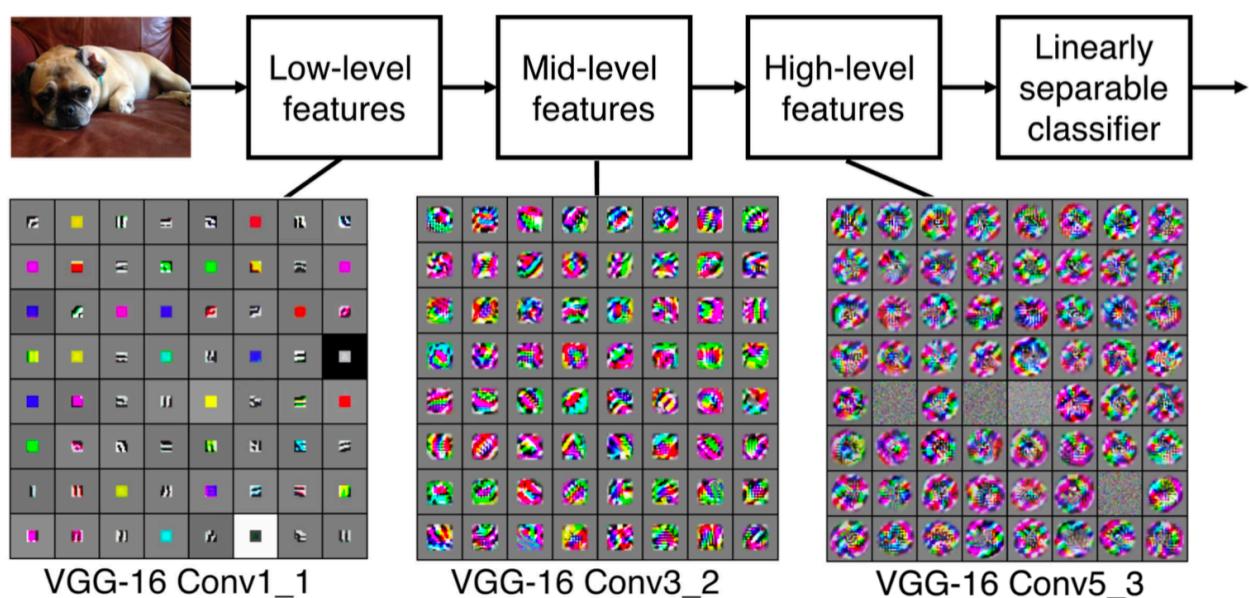


图 2-4 卷积网络可视化。图片来源：斯坦福大学CS231n

可以看到，卷积网络从简单的特征开始，通过堆叠在一起的卷积层，然后集成至更复杂的特征，这与Hubel和Wiesel在实验中对于大脑视觉皮层所观察到的现象是一致的。所以，即是没有明确地告诉模型去学习这些类型的特征，实际上当给出类似卷积网络的层次结构，并使用反向传播进行训练，模型最终也会学到这些特征。这便是卷积神经网络能够有效进行视觉处理之所在。

2.3.3 相关技术和工具

项目使用深度学习框架Keras + Tensorflow进行开发。

Tensorflow是由Google Brain开发的开源机器学习系统。它一个采用数据流图（data flow graphs），用于数值计算的开源软件库。在数据流图中，节点表示图中的数学运算，节点间的连线则表示输入输出的多维数组，即张量（Tensor）。理论上，只要将计算表示为一个数据流图，就可以使用Tensorflow，因此神经网络也不例外。Tensorflow可以在CPU和GPU上运行，比如台式机、服务器和手机移动设备等。同时，Tensorflow提供了一套基于python界面的API，其中包含有大量的神经网络的操作。对于项目中构建、训练卷积神经网络的需求可以很方便的满足。

Keras是一个高层神经网络API，可以基于Tensorflow、Theano以及CNTK作为后端，并可灵活的在这些后端之间切换，而不需要做任何的改动。Keras支持CNN和RNN，提供了高度模块化和极易使用的API，可以非常方便和快速的进行原型设计。此外，Keras还包含有大量预训练的CNN模型，如Inception V3、ResNet50等，极大简化了模型设计。因此，使用Keras搭配Tensorflow来完成项目是一个不错的选择。

深度学习需要消耗大量的计算资源，特别是训练样本较多、结构较为复杂的网络，单纯使用CPU来完成并不是明智的做法，而一般都会采用GPU进行加速。由于NVIDIA在深度学习领域布局较早，目前深度学习主要都是采用NVIDIA GPU进行开发。对于个人用户，通常选择GTX900或1000系列。

笔者在实做项目时，手边并没有上述GPU资源。为了使用GPU加速，采用了Amazon提供的云计算服务EC2（Elastic Compute Cloud），其中的P2/P3等实例适用于GPU加速计算。综合考虑性价比，最后选择p2.xlarge完成项目，该平台配备1个NVIDIA Tesla K80高性能GPU，Xeon E5-2686处理器，61GB CPU内存，以及12GB GPU内存。

2.4 基准模型

近年来，在ImageNet挑战中赛涌现了许多优秀的卷积神经网络架构，如Inception V3^[3]、ResNet^[4]、Xception^[5]、Inception V4^[6]等。前者通过Inception Module的堆叠提升整个网络的宽度和深度，并带来计算效率和性能的提升；后者通过残差块（Residual Block）的堆叠，可令网络达到非常深的深度（152-layer）同时具有非常好的性能。ResNet在2015年以Top-5错误率3.6%横扫其他对手拿到图像分类比赛第一名。

幸运的是，ImageNet猫和狗的分类，上述网络也有公开训练好的模型。因此，可直接将这些模型作为基准模型，可与本项目的方案做客观对比。

此外，项目还将定一个基准阈值：评估指标（下文将提及）要进入[kaggle排行榜](#)前10%，即LogLoss不高于第131位的0.06127。

3 方法

3.1 网络模型的选择

网络结构选择和设计是整个方法中的重要一环。一个尝试性的做法是，先自定义堆叠一些卷积层/池化层/BN（Batch Normalization）层，查看训练的效果并调整相应的网络结构和参数。不过这样做往往效果并不理想。

在基准模型一节中，已经提到ResNet、Inception V3等在实际应用中表现非常好的模型，且这些模型有公开模型结构和训练参数。采用迁移学习（Transfer Learning）方法，基于这些预训练的模型，然后进行微调（Fine Tune）是一个不错的选择。

目前在Keras中，开源的CNN模型包括VGG16、VGG19、ResNet50、Inception V3、Xception、InceptionResNetV2、MobileNet。

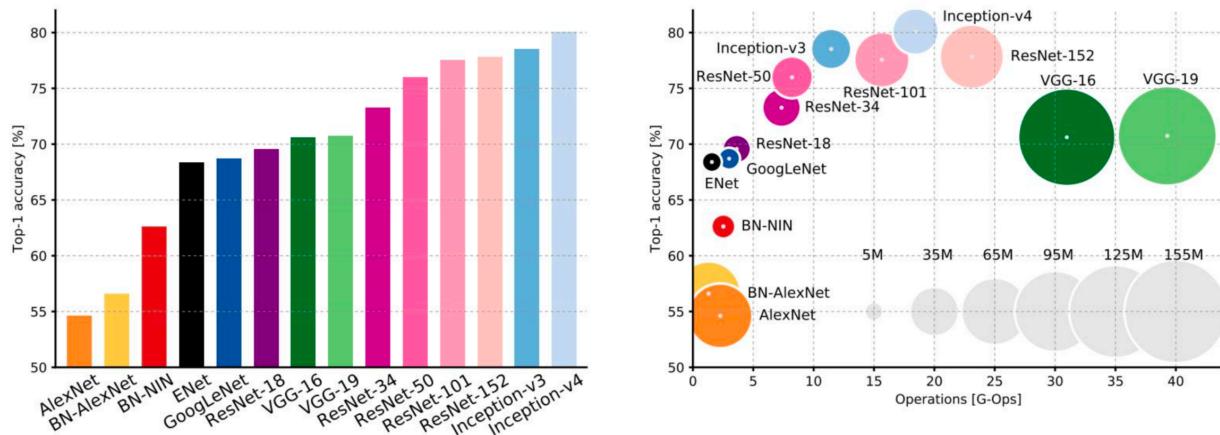


图3-1 不同CNN模型的比较。图片来源：斯坦福大学CS231n

图3-1列出了不同CNN模型之间性能（Top-1 Accuracy）、参数量以及运算量的对比。可以看到，早期的VGG-16/19网络，由于使用了较大的卷积核以及较多的全连接层，虽然网络深度并不是很深（23/26），但参数量和运算量都相对较大，并且Top-1准确率只有71%左右。后来发展出的ResNet和Inception等一系列网络，分别设计了残差块（Residual Block）和Inception Module的概念，不仅减少了网络的参数量，提升了运算效率，同时网络深度越来越深（ResNet50: 168, InceptionResNetV2: 572），并带来性能的提升（Top-1准确率达到80%左右）。

综合考虑各模型的表现，项目选取了ResNet50、Xception、InceptionResNetV2这三个模型作为迁移学习的基础模型。

3.2 数据预处理

由于图片的分辨率大小不一，因此要根据模型的输入需求统一调整图片的大小。此外，对数据的预处理还包括零中心化（Zero-Centered, feature-wise）、[-1, 1]区间的归一化（sample-wise）等等。

通过查看源代码，在上述基础模型中，都分别定义了相应的数据预处理。如ResNet50，要求输入是一个shape为(224, 224, 3)的张量，预处理需要先将channels从RGB转换为BGR，再按照ImageNet三个channel的均值[103.939, 116.779, 123.68]进行Zero-Centered处理；Xception和InceptionResNetV2类似，输入是一个shape为(299, 299, 3)的张量，预处理需要按sample-wise将输入归一化至[-1, 1]。

Keras的图片生成器ImageDataGenerator可方便的完成上述数据预处理，该类中的flow_from_directory方法返回了一个generator，每次生成一个batch的图像数据，并且可以无限生成，直到指定的epoch次数为止。使用该方法前，需要将图像分别分类单独放在不同的路径下面，如图3-2，train_gen目录用于生成训练数据，下面有两个子目录：cat和dog，分别存放10000张cat的图片和10000张dog的图片；相应地，val_gen目录用于生成验证数据，同样包含cat和dog两个目录，分别存放2500张cat的图片和2500张dog的图片。存放在train_gen和val_gen的图片是随机选取的，从数量上看，训练数据和验证数据的比例为4:1。test_gen用于生成测试数据，其中包含了所有的12500张图片。

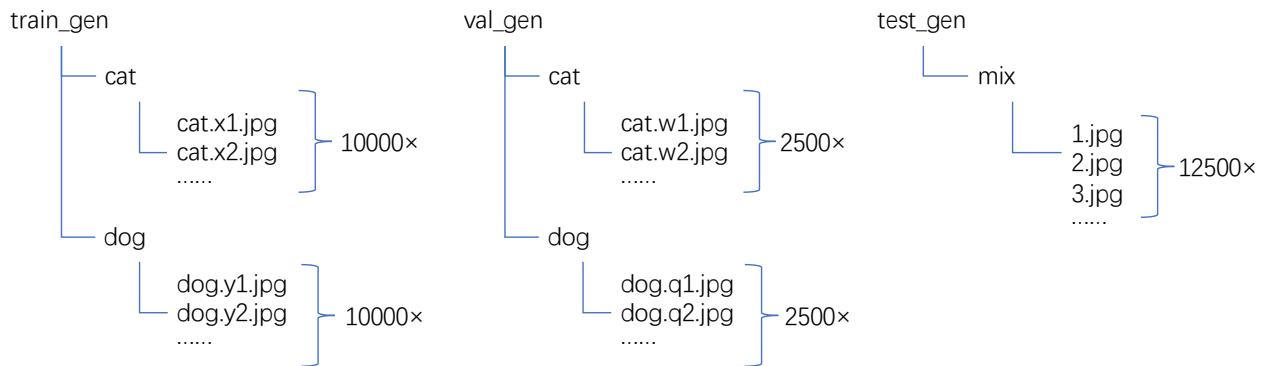


图3-2 与flow_from_directory方法适配的目录结构

此外，考虑到项目的样本量并不是特别多，使用ImageDataGenerator提供的数据增强（Data Augment）方法扩充数据集，具体用法将在下一小节描述。

3.3 执行过程

执行过程使用Keras搭建迁移学习模型并完成训练和预测。

3.3.1 生成数据

使用上一节提到的ImageDataGenerator类搭配flow_from_directory方法生成训练数据、验证数据以及测试数据。该方法返回一个generator，可与Keras的模型训练与预测API无缝结合，自动产生相应的数据。

在生成训练数据与验证数据时，使用了数据增强（Data Augment）。具体做法是在初始化ImageDataGenerator类时设置相应的参数，如rotation_range=10、zoom_range=0.1、width_shift_range=0.05、height_shift_range=0.05、channel_shift_range=10、horizontal_flip=True，分别表示在[0°，10°]范围内随机角度旋转、在0.1范围内缩放、在0.05范围内上下/左右平移、幅度为10的随机通道偏移、随机水平翻转。generator在生成训练或验证数据时，会根据上述参数对原始图片进行变换，以达到扩充数据集的目的。

3.3.2 构建、训练模型

如前文所述，项目使用迁移学习（Transfer Learning）方法，基于预训练模型进行微调（Fine Tune），其原理如图3-3。Xception和InceptionResNetV2使用同样的方法构建模型。

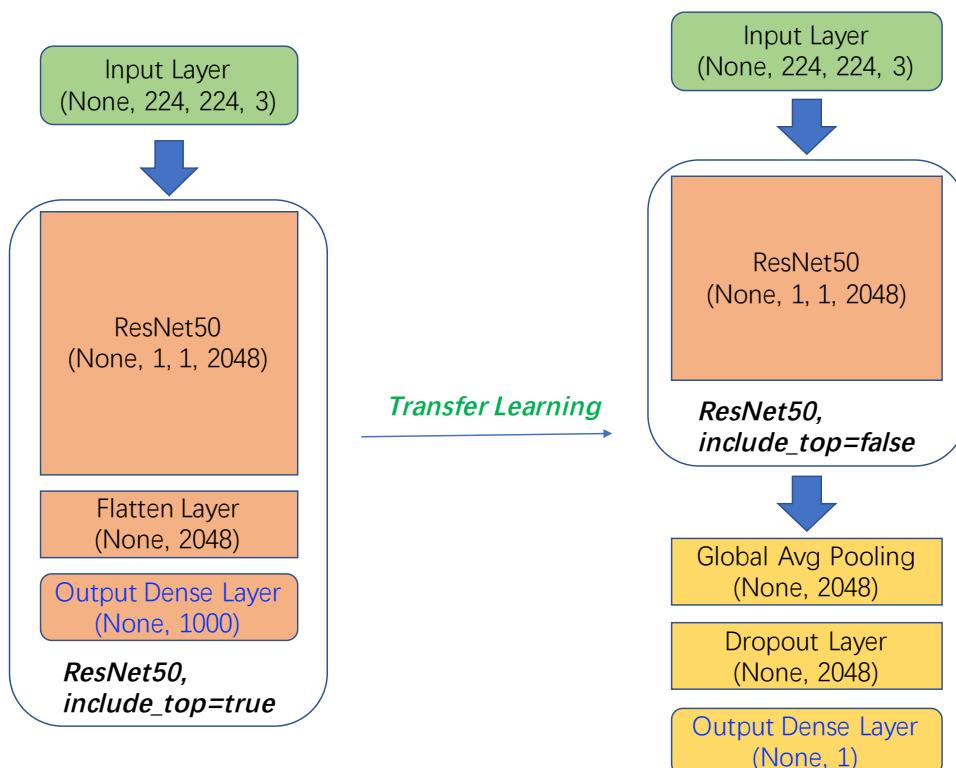


图3-3 ResNet50迁移学习示例

以ResNet50为例，Keras提供resnet50.ResNet50作为模型API，传递include_top=True时，对应图3-3左侧，为原始的预训练模型，输入为(224, 224, 3)张量，输出为ImageNet 1000类的softmax激活。迁移学习时，传递include_top=False，将原模型的最后两层（Flatten和Dense）去掉，新增Global Average Pooling、Dropout、Dense这三

层，输出二分类的sigmoid激活，优化器选择adam，初始学习率使用默认的0.001，loss为binary_crossentropy。这样就构建了一个适用于项目的新模型。

模型新增的部分使用GAP (Global Average Pooling) 层对原模型去除top之后的输出进行向量化，即由(1, 1, 2048)的张量转换为2048维的向量。与Flatten层相比，GAP可以减少后面输出层的参数，一定程度减少过拟合。

构建好模型之后，选择需要微调的参数，一般是选择输出端的参数。以ResNet50为例，选取最后20个layer的7,884,289个参数重新进行训练。

	Fine Tune Layers	Total Parameters	Trainable Params
ResNet50	[-20:]	23,589,761	7,884,289
Xception	[-32:]	20,863,529	9,480,393
InceptionResNetV2	[-80:]	54,338,273	12,971,201

表3-1 各模型微调layer及参数

训练使用fit_generator方法。该方法可直接使用训练和验证数据的generator，另外还可以通过设置了callback参数。callback传递了ModelCheckpoint和EarlyStopping对象，在每个epoch结束时都会执行。其中，ModelCheckpoint可以将每轮epoch训练得到的参数全部保存为hdf5文件，这可以确保获取到最优的训练参数；EarlyStopping可以设定监测参数（如val_loss），一旦该参数不再改善，便停止训练。以上callback可提高训练效率。

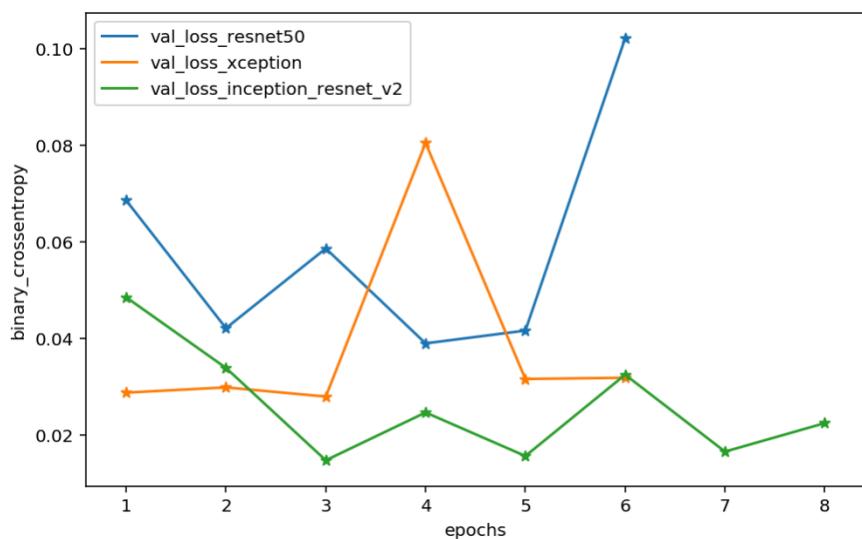


图3-4 val_loss训练曲线

三个模型的训练曲线如图3-4。总的来说，所有模型在4个epoch之内都开始出现过拟合，因此训练的epoch数都不多。从val_loss来看，模型的表现如预期，从好到差分别是：

InceptionResNetV2, Xception, ResNet50。最后，每个模型都选取val_loss最低那一轮epoch对应的参数。

3.3.3 预测

模型预测使用predict_generator方法。该方法可以直接使用测试数据的generator，并且返回一个numpy数组对应每一张测试图片为狗的概率。

需要注意的是，模型存在错判的情况，特别是在对错误判断非常确定的情况下。例如，假设图片为狗（1），模型输出概率为0.0001，即非常确定是猫，这一样本得到的LogLoss是非常大的，很容易影响整体的分数。

参考知乎的文章，采用一种截断的方法，对输出概率最小值截断为0.0045，最大值截断为0.9955。当发生上述错判情形，可以限制单个样本的LogLoss不至于太大。而在模型判断正确的情况下，因为Log0.9955与Log1差别不大，并不会对整体分数造成大的影响。

pred_inception_resnet_v2.csv a month ago by Jeff Li inception_resnet_v2	0.04073
pred_xception.csv a month ago by Jeff Li xception	0.04210
pred_resnet50.csv a month ago by Jeff Li add submission details	0.05784

图3-5 三个模型各自在kaggle的分数

将测试集的预测概率提交至kaggle，成绩如图3-5，InceptionResNetV2的成绩最好，为0.04073。

3.4 完善

单模型的结果基本达到预期，使用多模型进行优化将是下一步完善的方向。

简单集成

集成是多模型场景下的常用优化方法，而平均法是其中较为简单常用的结合策略。模型间的相关系数如图3-6，相关性最小的是ResNet50与Xception，为0.98；相关性最大的是InceptionResNetV2与Xception，为0.99。可见，三个模型之间的相关性总的来说都比较高，相互差异较小，因此预期简单求平均对结果改善应该会比较有限。

将三个模型的概率结果取平均后，成绩为0.03893，相对单模型提升4.4%。

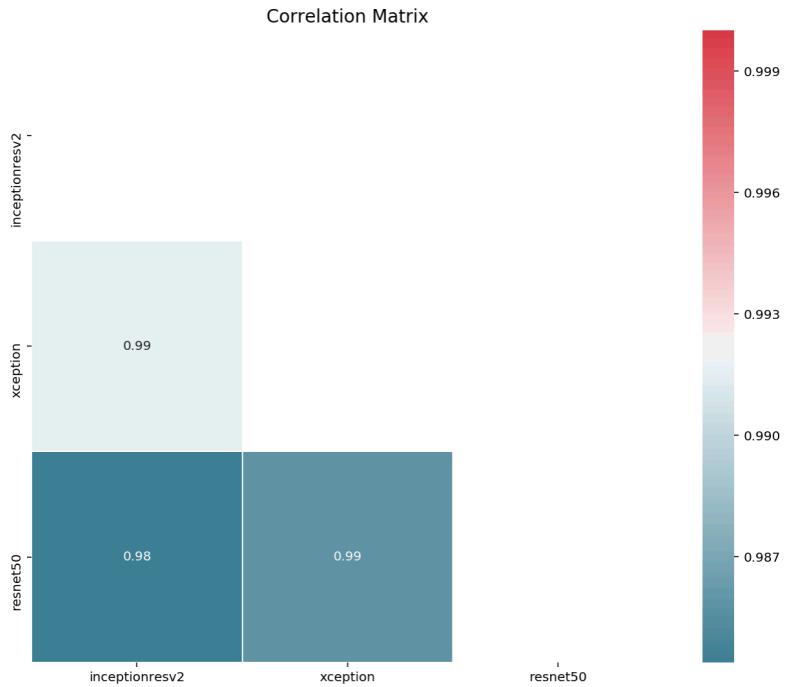


图3-6 模型间的相关系数

多模型特征提取+组合

参考知乎的文章，单个模型可看作是一个高阶特征的提取器，可将多个模型提取出的高阶特征组合到一起，再输入给分类器。无论是训练或预测，都需要先经过特征提取，训练集提取的特征送入分类器做训练，而测试集提取的特征送入分类器做预测。

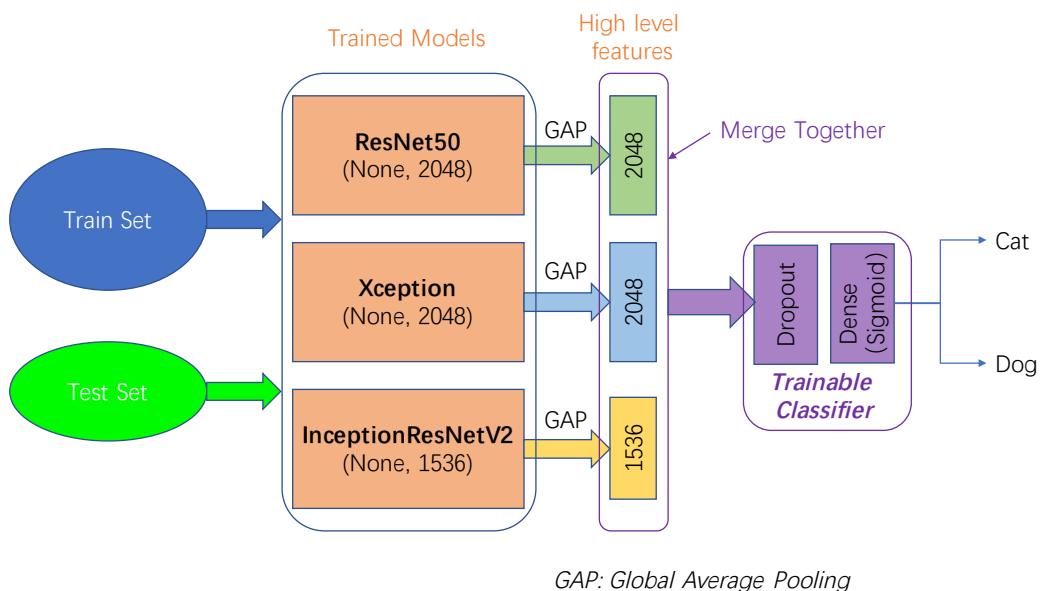


图3-7 特征提取+组合

具体做法如图3-7。训练集和测试集经过三个模型，分别在的Global Average Pooling 层输出2048维、2048维和1536维特征，将这些特征拼接在一起产生新的样本，每个新样本

包含5632维特征。分类器依然使用一个Dropout层加一个Sigmoid输出层，优化器选择adam，初始学习率使用默认的0.001。用训练集对应的新样本训练分类器，训练完成后输入测试集，输出预测结果。模型训练曲线如图3-8。

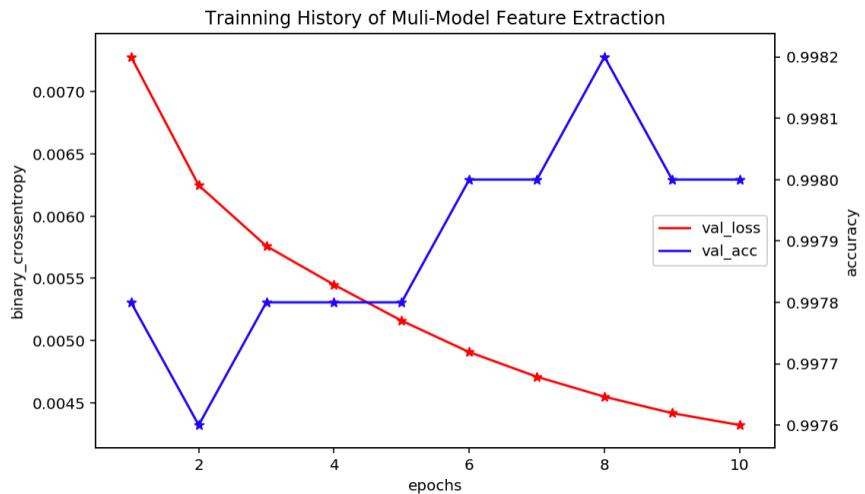


图3-8 多模型特征提取训练曲线

多模型特征提取方法的kaggle成绩为0.03452，相对单模型显著提升了15.2%。

剔除异常样本+数据增强

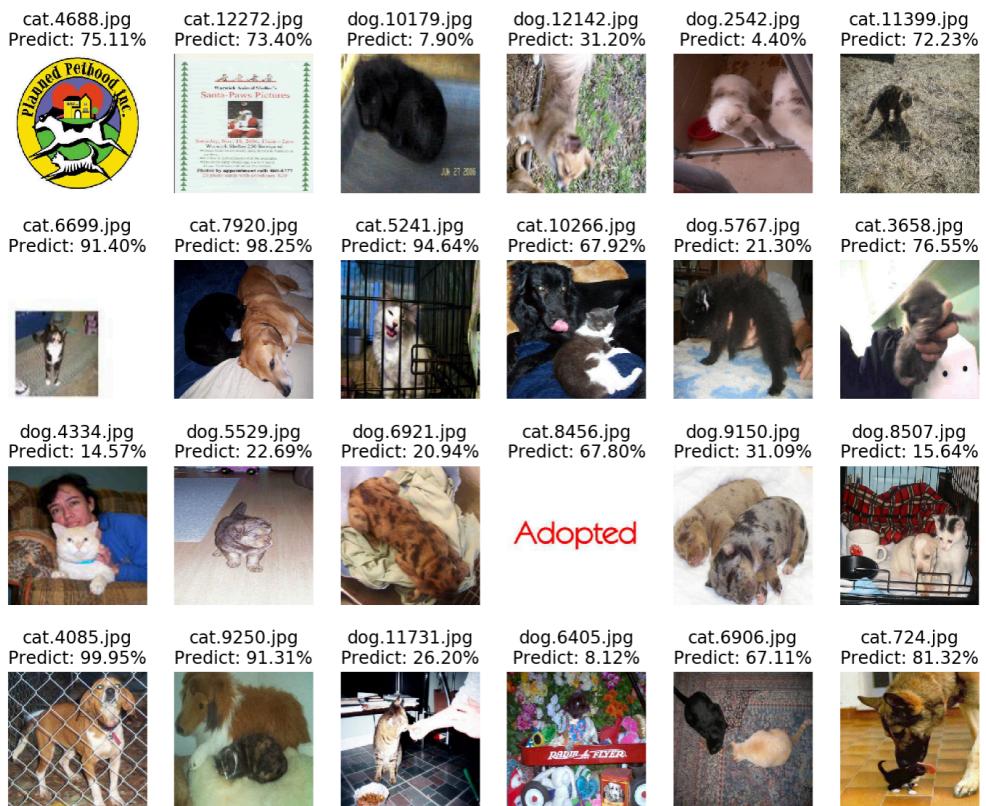


图3-9 训练图片中的异常样本

将训练好的模型对所有训练数据进行预测，并与实际标签对比，取出预测结果与标签不符图片，挑选一部分可视化如图3-9。

其中，cat.7920.jpg、dog.4334.jpg、cat.4085.jpg、dog.11731.jpg等样本的标签被写错，实际为狗，却标记为猫，或者反之；dog.8507.jpg、cat.9250.jpg、cat.724.jpg等样本的图像中既有猫也有狗；cat.12272.jpg、dog.8456.jpg等样本非猫非狗；还有诸如dog.10179.jpg、dog.2542.jpg、cat.6699.jpg、cat.5241.jpg等样本被模型以很确定的概率认错了。上述样本总共24个。

将这些样本剔除出训练集，重新训练分类器，并且尝试更大的迭代次数（18次），如图3-10。训练完成后，测试集的kaggle得分为0.03495，反而比剔除样本之前的表现更差了。可能包含两个原因，一方面迭代次数增加之后可能出现了过拟合；另一方面，剔除的样本当中包含默认以很高概率“错认”的图片，而测试集当中也一定存在被模型“错认”的图片，缺少这些样本的训练，可能会加剧增加模型“错认”的概率。

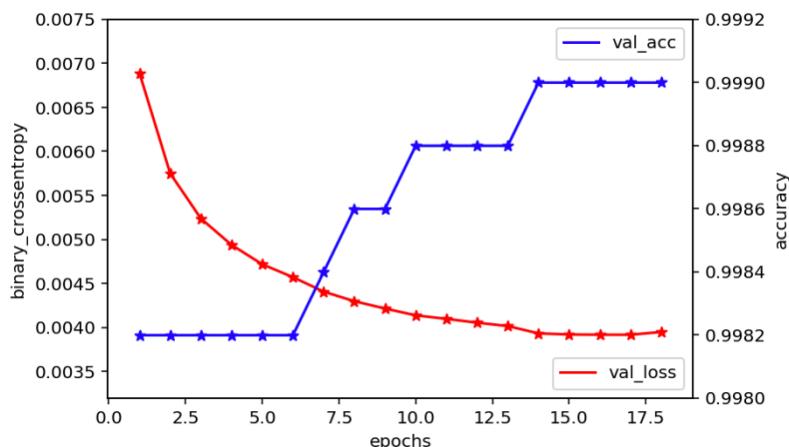


图3-10 异常样本剔除后的训练曲线

如果在训练好的模型之上，再用完整的训练集继续训练1~2次，也许能有所提高。使用原始训练数据再训练2个epoch，得分为0.03463，效果并不理想。

接下来尝试使用了数据增强。与图3-7的做法相同，将训练图片经过数据增强变换后分别通过三个单模型提取特征，同样再将特征组合到一起。使用新的训练集继续迭代2个epoch，得分为0.03402。

模型最终完善的结果为0.03402，较单模型提升16.5%。

4 结果

4.1 模型的评价与验证

从单个模型来看，项目使用的三个模型虽然都属于卷积神经网络，它们构建网络的思想却各不相同。ResNet50提出了残差连接（Residual Mapping）的概念，使网络用来训练残差而非传统的卷积输出，这样的好处是可以令网络具有非常深的深度同时不至于难以优化从而达到更好的性能。Xception和InceptionResNetV2都是最初由GoogleNet（Inception）发展而来，其中，Xception基于Inception V3网络，其特点是采用一种称为深度方向上的可分离卷积方法，即先对每个通道分别卷积，再使用 1×1 逐点进行卷积，同时也应用了残差的结构；InceptionResNetV2主要利用残差来改进Inception V3的结构，基本思想就是使用Inception module替代原本ResNet中的卷积单元。

从当前趋势来讲，卷积网络正朝着越来越深的方向发展。InceptionResNetV2不仅具有高效的网络结构，其深度相比其他两个网络也超出很多（572 vs. 168/159），并且从前文实现的结果来看，也具有最佳性能。

相比任何单一模型，从多个模型中提取特征并将其组合的方法可以全面利用不同网络提取的不同信息，因而有机会得到更好的泛化能力，如表4-1。因此选择多模型提取特征搭配简单的神经网络分类器作为最终模型。

	Validation Accuracy	Validation Loss	Test Loss
ResNet50	0.9866	0.0384	0.05784
Xception	0.9902	0.0279	0.04210
InceptionResNetV2	0.9952	0.0147	0.04073
Multi-Model Feature Extraction	0.9976	0.0067	0.03402

表4-1 各模型（包含多模型特征提取）的评估指标

最终模型在验证集的准确率达99.8%，在测试集的LogLoss也是几种模型当中最低的，因此可以说该模型得出的结果是足够可信的。此外，由于在训练过程中使用了数据增强，即对原始数据做特定的微小变换，从图3-4的训练曲线得知，输入的变化对模型结果并不会造成大的影响，因此最终的模型是足够稳健的。

4.2 合理性分析

项目的基准模型对应表4-1中的单个模型，从Validation Accuracy、Validation Loss以及Test Loss等评价指标看，最终模型都要优于基准模型。对于kaggle的评价指标Test Loss，最终模型相比基准模型提高至少16.5%，在kaggle排行榜排名第3位。从测试集的表现来看，最终结果很好的完成了猫狗图片的分类。

5 项目结论

5.1 结果可视化

从测试集随机挑选一些图片进行可视化，如图5-1。可以看到，对于大部分测试图片，模型不仅判断正确，而且对结果非常肯定。测试集里也混有“非猫非狗”图片，如12099.jpg，模型虽然判断为狗，但不太肯定（概率只有51%），这也算合理。



图5-1 测试结果可视化1

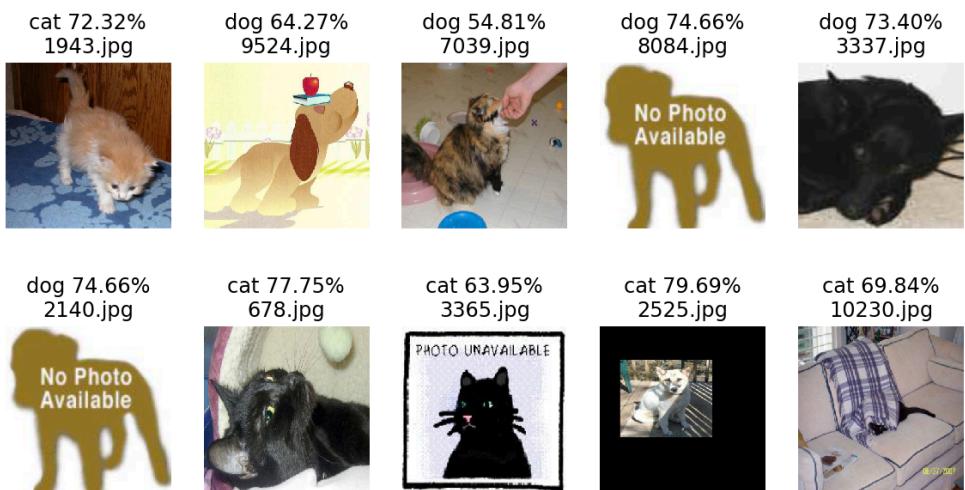


图5-2 测试结果可视化2

预测概率为0.2到0.8之间的图片总共56张，选取一部分进行可视化如图5-2。模型对这些图片虽然不是很确定，但大部分判断是正确的。少部分如2525.jpg、7039.jpg，模型判断错误。这一类图片存在图像不清晰、背景较复杂或遮挡物较多的情况，有些人眼也很难辨认。

总的来说，模型的表现已经达到了预期。

5.2 思考

根据项目数据集的特点，使用迁移学习方法，基于ImageNet表现靠前的三个预训练CNN模型，通过Keras完成数据预处理，构建、训练模型，完成预测，并基于结果进行优化。该项目所使用的方法和流程，对于通用场景下解决图片分类问题也是适用的。

使用迁移学习，是项目的亮点，相对于自建模型，站在巨人的肩膀上可以走得更远。灵活运用Keras API是简单、快速、高效完成项目的关键。虽然不需要自己搭建预训练模型，但至少需要了解模型的输入、输出以及大致的网络结构，这关系到能否顺利实现模型的fine tune以及多模型的特征提取。

项目的困难之处在于，在模型已经取得一个期望结果之后，在此基础再做优化。显然，到了这个阶段，模型性能的提升已变得非常困难，常规的方法也都用过，此时需要不断尝试新的思路和方法。

5.3 改进

有一些值得尝试的算法和技术，由于时间和资源有限，在项目中并未实施。

如果单纯为了提高分数，可以考虑在剔除异常样本后重新fine tune单一模型，并尝试调整可训练的层数。

也可以尝试新的方法，例如在多模型特征提取再进行组合之后，考虑使用PCA对特征进行降维，因为组合之后可能会包含重复、冗余的特征，直接训练分类器可能会影响其分类性能。另外由于样本较大，维度较高，PCA变换涉及很大的计算开销，需要考虑使用Tensorflow等框架在GPU上完成计算。

或者在分类器上尝试一些改进，例如使用stacking方法集成多个分类模型（如SVM、XGB、Logistic、Random Forest、GaussianNB等），通过交叉验证的方式训练初级学习器和元学习器（Meta Learner）。

参考文献

- [1] O Russakovsky, J Deng, H Su and J Krause. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision, 2015.
- [2] Pierre Sermanet, Soumith Chintala and Yann LeCun. Convolutional Neural Networks Applied to House Numbers Digit Classification. The Courant Institute of Mathematical Sciences - New York Universit, 2013.
- [3] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens. Rethinking the Inception Architecture for Computer Vision. arXiv:1512.00567v3 [cs.CV] 11 Dec 2015.
- [4] Kaiming He Xiangyu Zhang Shaoqing Ren. Deep Residual Learning for Image Recognition. Microsoft Research, 2015
- [5] Franc,ois Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. arXiv:1610.02357v3 [cs.CV] 4 Apr 2017.
- [6] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. arXiv:1602.07261v2 [cs.CV] 23 Aug 2016.