机器学习纳米学位

基于深度学习的猫狗识别

王健欢

2016 年 1 月 6 日

目录

1 定义

1.1 项目概述

本项目将训练一个可以从拍摄了猫狗的照片中识别出图片中的动物是猫还是狗的深度学习模型。模型可以应用于畜牧行业、宠物行业，使机器能够实时区分猫狗，节省人力资源。

本项目涉及到图像识别领域，图像识别是指利用计算机对图像进行处理、分析和理解，以识别各种不同模式的目标和对像的技术。数字图像处理和识别的研究开始于 1965 年。数字图像与模拟图像相比具有存储，传输方便可压缩、传输过程中不易失真、处理方便等巨大优势，这些都为图像识别技术的发展提供了强大的动力。物体的识别主要指的是对三维世界的客体及环境的感知和认识，属于高级的计算机视觉范畴。它是以数字图像处理与识别为基础的结合人工智能、系统学等学科的研究方向，其研究成果被广泛应用在各种工业及探测机器人上[1]。

本项目选择的数据集来自于 kaggle 上的 dogs vs cat 项目，该项目提供的数据集包括训练数据集和测试数据集，其中训练数据集中所有的样本的名字里都包含标签；而测试数据集中不含有标签，其作用是用于 kaggle 网站评价用户的模型。本项目只使用训练集数据。

1.2 问题陈述

项目选择的数据集是从真是世界采集的猫狗照片。识别这些照片上的动物有以下难点。其一，拍摄的角度多样，动物姿势、大小，照片背景多种多样，这些多样性无疑增加了机器区分猫狗的难度。其二、照片质量参次不齐，因为照片并非都是专业人士拍摄，部分照片存光照过亮或过暗、失焦、像素太低等问题，这些问题加大了机器处理照片的难度。本项目解决以上问题，并期望建立一个能让机器区分平常人拍摄的非专业照片里的动物是猫还是狗，并有着较高的准确率的模型。

本项目将从两部分解决该问题，一是对照片进行预处理，除去一些可能影响

模型学习的照片，并将剩余图片调整至同一标准，尽可能减少因照片质量差异给模型学习造成的不利影响。二是模型的建立，本项目将打算使用卷积神经网络模型。卷积神经网络（Convolutional Neural Network,CNN）是一种前馈神经网络，它的人工神经元可以响应一部分覆盖范围内的周围单元，对于大型图像处理有出色表现。

1.3 评价指标

猫狗识别问题属于二分类模型，对于二分类模型常用的指标有准确率 LogLoss，F1-Measure。其中 LogLoss 即交叉熵损失函数，交叉熵可以用于度量两个概率分布（真实分布&预测分布）之间的差异性，可以用其衡量一个模型对真实值带来的额外噪音，其公式见图 1-1。F1-Measure 是精确率和召回率的调和平均数。因本项目打算使用神经网络解决该问题，并使用神经元数量为 1，激活函数为 sigmoid 的输出层对样本做最后的分类。样本的预测结果不是离散的，而是介于 0，1 之间的值。若使用 LogLoss 无需对预测值做进一步处理便能计算。若使用的 F1-Measure，则需要将预测的概率值转换成离散的标签，增加操作量。因此，在本项目中主要使用 LogLoss 对模型进行评价。

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

图 1-1，LogLoss 的计算公式，其中 n 为样本的数量，y^i 为第 i 个样本的预测值，yi 为第 i 个样本的真实标签（如果是狗为 1，猫为 0）

2 分析

2.1 数据可视化

本项目选择的数据集来自于 kaggle 上的 dogs vs cat 项目[2]，项目提供了带标签的训练集数据和不带标签的测试集数据，本项目只选用了训练集数据当作总数据集进行学习。数据集一共含有 25000 样本，其中有 12500 个样本为显示猫的照片，样品名字带有标签‘cat’；有 12500 个样本为显示狗的照片，样品名字带有标签‘dog’（图 2-1）。

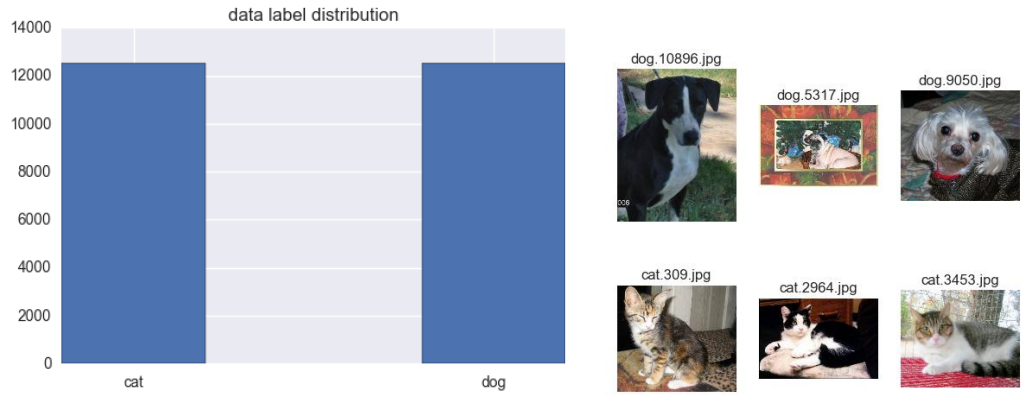

图 2-1 右图为数据的标签分布，横坐标为标签‘cat’和标签‘dog’，纵坐标为样本数量。标签为‘cat’和标签为‘dog’的样本数量都为 12500 个。左图为随机挑选的 3 个狗组和 3 个猫组的照片，图片上方是该样本包含标签的名字。

数据集中的样本皆为 3 通道图像，RGB 色彩模式是工业界的一种颜色标准，是通过对红(R)、绿(G)、蓝(B)三个颜色通道的变化以及它们相互之间的叠加来得到各式各样的颜色的，RGB 即是代表红、绿、蓝三个通道的颜色，这个标准几乎包括了人类视力所能感知的所有颜色，是目前运用最广的颜色系统之一[3]。每个通道都包含一个由像素组成的二维矩阵，矩阵的大小有两个维度，分别为宽和高。对样本的尺寸进行统计分析，发现分布比例最多的尺寸为 500,374（单位像素），一共有 2955 个样本使用该尺寸。其次是 499，375（单位像素），一共有 2912 个样本使用该尺寸。这两种尺寸的宽高比例十分接近 4/3，这个比例是主流的标准照片比例。进一步观察样本宽与高的散点图（图 2-2），图中可明显的看到斜率

为 3/4 和 4/3 的两条斜线，除了斜线之外还有几条很明显的水平线和垂直线，推测这些样本是由标准照片（比例为 3/4 或 4/3）裁剪或拼接得到的。

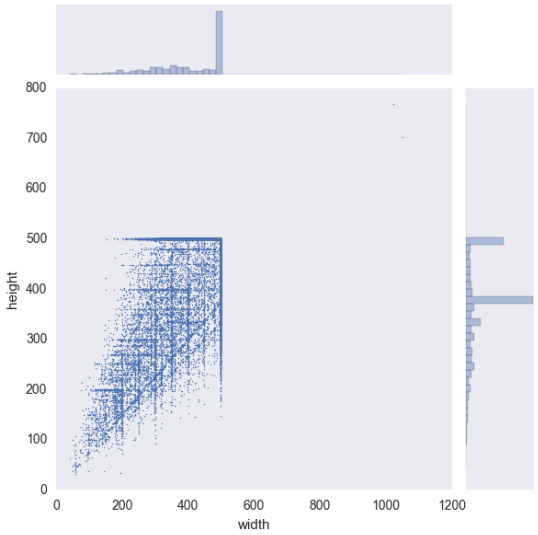

图 2-2 图片样本的宽与高的散点图(单位像素)，横坐标为图片样本的宽，纵坐标为图片样本的高。

图片的标准差即图片的均方根对比度，对比度对视觉效果的影响非常关键，一般来说对比度越大，图像越清晰醒目，色彩也越鲜明艳丽；而对比度小，则会让整个画面都灰蒙蒙的。高对比度对于图像的清晰度、细节表现、灰度层次表现都有很大帮助[4]。图片样本的标准差分布大致呈正态分布，样本的标准差大部分集中在 40-80 之间。

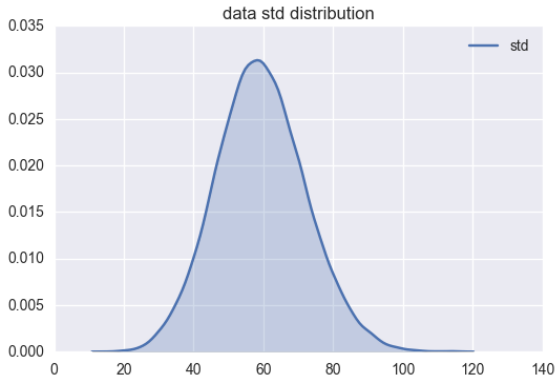

图 2-3 图片的标准差分布,横坐标为标准差的值，纵坐标为占样本总数的比例，分布大致呈正态分布。

2.2 算法和技术

2.2.1 分类算法

本项目属于二元分类问题，常用于分类的算法有朴素贝叶斯分类（normal_bayes_classifier）、支持向量机（support_vector_machines）、随机森林（random_trees）、神经网络（neural_networks）。

朴素贝叶斯分类器模型是基于贝叶斯定理与特征条件独立假设的分类方法，它是机器学习中的一个非常基础和简单的算法，常常用于解决分类问题。朴素贝叶斯分类器的分类原理是通过某对象的先验概率，利用贝叶斯公式计算出其后验概率，即该对象属于某一类的概率，选择具有最大后验概率的类作为该对象所属的类。其优点为算法逻辑简单，易于实现，算法实施的时间空间开销小，算法性能稳定，对于不同特点的数据其分类性能差别不大。在各属性条件独立的情况下表现最好。其缺点为模型假设属性之间相互独立，这个假设在实际应用中往往是不成立的。若属性之间相关性大，模型表现很差。

支持向量机（support_vector_machines，以下简称 SVM）是一个有监督的学习模型，通常用来进行模式识别、分类、以及回归分析。其原理是通过一个非线性映射 p，把样本空间映射到一个高维乃至无穷维的特征空间中（Hilbert 空间），使得在原来的样本空间中非线性可分的问题转化为在特征空间中的线性可分的问题。SVM 模型在解决小样本、非线性及高维模式识别中表现出许多特有的优势，可以避免神经网络结构选择和局部极小点问题。当数据非线性可分时用 SVM 模型效果很好。SVM 算法对大规模训练样本难以实施，当样本数目很大时该矩阵的存储和计算将耗费大量的机器内存和运算时间；经典的支持向量机算法只给出了二类分类的算法，不适合单独使用 SVM 模型解决多分类问题；对缺失数据敏感。当样本数量多或者并非二分类的时候使用 SVM 效果差。

随机森林是一种多功能的机器学习算法，能够执行回归和分类的任务。随机森林的原理是通过自助法（bootstrap）重采样技术，从原始训练样本集 N 中有放回地重复随机抽取 k 个样本生成新的训练样本集合，然后根据自助样本集生成 k

个分类树组成随机森林。当在基于某些属性对一个新的对象进行分类判别时，随机森林中的每一棵树都会给出自己的分类选择，并由此进行“投票”，森林整体的输出结果将会是票数最多的分类选项[5]。SVM 模型在解决小样本、非线性及高维模式识别中表现出许多特有的优势，可以避免神经网络结构选择和局部极小点问题。当数据非线性可分时用 SVM 模型效果很好。SVM 算法对大规模训练样本难以实施，当样本数目很大时该矩阵的存储和计算将耗费大量的机器内存和运算时间；经典的支持向量机算法只给出了二类分类的算法，不适合单独使用 SVM 模型解决多分类问题；对缺失数据敏感。当样本数量多或者并非二分类的时候使用 SVM 效果差。

2.2.2 卷积神经网络

卷积神经网络（Convolutional Neural Network,CNN）是一种前馈神经网络，它的人工神经元可以响应一部分覆盖范围内的周围单元，对于大型图像处理有出色表现。2012 年 Alex Krizhevsky 使用的卷积神经网络在 ILSVRC 2012 中赢得了第一名,Top5 错误率 15.3%遥遥领先第二名。这一举奠定了卷积神经网络在图像识别领域的地位。之后，研究人员不断改进卷积神经网络，先后创建了 GoogLeNet、VGG、ResNet[6]等神经网络模型。

卷积网络一般分为 4 种操作，分别为卷积、非线性处理（ReLU）、池化、分类（全连接层）

卷积操作是卷积神经网络的核心，其目的是为了从输入图像中提取特征，具体操作为用滤波器（filter）在输入的图像上从左上到右下滑动，每次滑动都向右移模型规定的像素（一般为 1），当滑动到最右边则从下一行第一列开始继续滑动。每次滑动都在该位置上都用滤波器矩阵与图像的相应位置进行点积操作。当滑动结束可得到一个有各乘积组成的矩阵。这个矩阵被称为“卷积特征”。对于同样的输入图像，不同值的滤波器将会生成不同的特征图（如图 2-4）。在每次的卷积操作后都使用了一个叫做 ReLU 的操作。ReLU 表示修正线性单元（Rectified Linear Unit），是一个非线性操作。其公式为 f(x) = max(0,x)。它是由神经科学家 Dayan、Abott 从生物学角度，模拟出了脑神经元接受信号更精确的激活模型。

这个模型对比 sigmoid 有三个变化：①单侧抑制 ②相对宽阔的兴奋边界 ③稀疏激活性。之后进行池化操作，池化操作的目的是为了降低各个特征图的维度。池化操作有下面几种方式：最大化、平均化、加和等等。本项目使用的是最大池化，具体操作为要定义一个空间领域（如 2*2），并将输入的特征图图象分割成若干个空间领域，对于每一个空间领域，从中提取最大的值并组成新的矩阵。经历了多次卷积池化操作之后，检测出的高级特征再输入至全连接层进行分类。

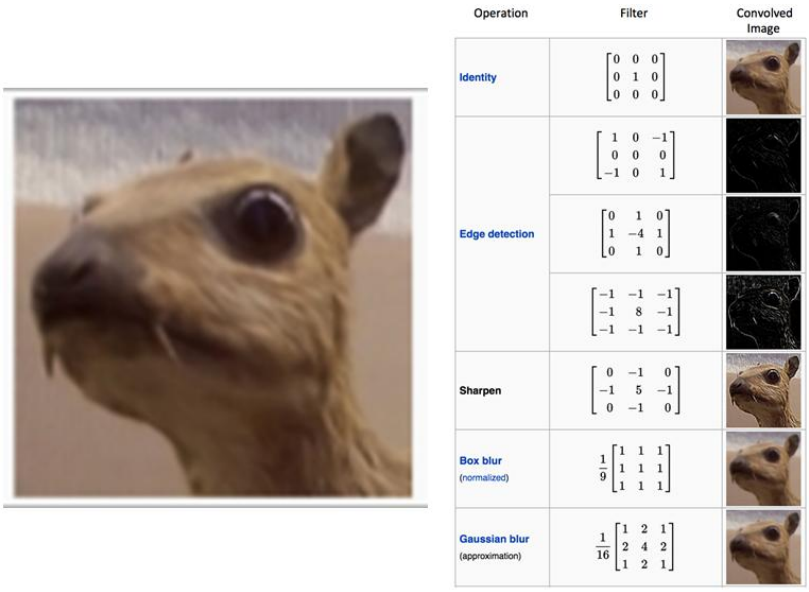

图 2-4 左图为原图，右图不同滤波器对右图卷积的效果（图片来源于[7]）

2.2.3 技术

本项目使用的是深度学习框架 keras（以 theano 为依赖库），Keras 是基于 python 语言的高度模块化的神经网络库，支持 GPU 和 CPU。其开发目的是为了更快的做神经网络实验。适合前期的网络原型设计、支持卷积网络和反复性网络以及两者的结果、支持人工设计的其他网络。

2.3 基准指标

在图像识别领域，resnet 模型是近年来表现最为优秀的模型之一。2015 年的 ImageNet 比赛中，Kaiming He 使用了 ResNet152 模型赢得了第一名,top5 error 仅为 3.57%[8]。

因为实验条件有限，本人难以承担训练神经网络的所需要较高配置的硬件和需要的大量时间。因此我选择在含权值数据的 ResNet50 模型的基础上进行训练，并期望准确率能达到甚至超越 ResNet152 模型在数据集 ImageNet 的准确率 96.43%

3 方法

3.1 数据预处理

经数据的可视化分析，发现样本的尺寸分布及标准差分布范围较广，可能并非所有样本都适合模型去学习。所以我希望对样本中的异常样本进行观察，判断其是否要被移除。对于异常样本的判断使用 Tukey 的定义异常值的方法: 一个异常阶（outlier step）被定义成 1.5 倍的四分位距（interquartile range，IQR）。一个数据点如果某个特征包含在该特征的 IQR 之外的特征，那么该数据点被认定为异常点。

在本项目中选择使用样本的宽高比例（经 log 处理）、标准差、像素大小作为分析异常值的特征。利用上述方法找出可能的异常值（对于像素大小，异常阶调整为 1.3 倍的四分位距），经观察决定除去以下样本（图 3-1），这些样本均不是猫狗在真实世界的照片。

图 3-1 以上 18 张图片均为被除去的样本

剩下的图片样本的尺寸太小不一，而训练网络要将数据的维度统一化，因此要将图片的尺寸统一调节成 128*128。调整图片尺寸大小有裁剪、填充、拉伸等方式，其中裁剪要识别图中动物的位置，难度较大；填充会对数据集额外添加信息，因此本项目选择使用拉伸的方式。图 3-2 为随机挑出的一个样本拉伸前和拉伸后的图片。最后打乱全部数据集的顺序，挑出 10%（2498）的样本作为测试集，

剩余的 90%（22483）的样本作为训练集。

图 3-2 左图为原图，尺寸为（500，374），右图为尺寸调整至（224，224）的图片

3.2 执行过程

3.2.1 提取瓶颈特征

之后将预处理完的 24982 个样本随机打乱顺序，记录样本的标签。随后搭建一个不包含全连接层的 ResNet50 模型，并载入其经 Imagenet 数据集训练过的权值数据。随后将预处理完的样本输入至该模型中，经模型的若干次卷积池化操作后，图像的特征越来越高级和稳定，最后输出一个强稳定性和语义性的高级特征，即瓶颈特征。

3.2.2 分类

建立分类模型处理瓶颈特征，并将样本分为两类，模型结构如图 3-3。首先先使用 Flatten 层将 3 维的瓶颈特征数据展开成一维的数据。之后将数据输入到一个包含 1024 个神经元的全连接层，全连接层的激活函数选择 ReLU。全连接层之后再连接一个包含 1 个神经元的输出层做最后的分类运算，输出层的激活函数选择的是 sigmoid，输出的范围为 0，1，输出层输出的结果表示该样本是正类（狗）的概率。

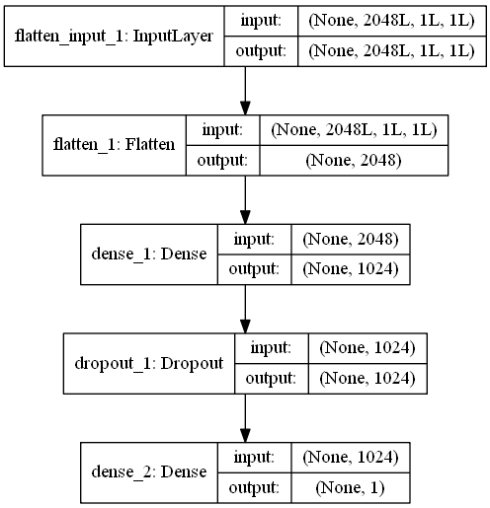

图 3-3 分类模型的结构

训练过程使用的是学习率为 1 的 Adadelta 优化器。Adadelta 是一种自适应学习优化器，相对于 Adagrad， Adadelta 只累加固定大小的项，简化了运算，并且也不直接存储这些项，仅仅是近似计算对应的平均值[9]。Adadelta 的特点是训练初中期加速效果，后期会反复在局部最小值附近抖动。

输出层选择的 sigmoid 作为激活函数，由于 sigmoid 的特性，当输出值非常接近 1 或 0 时，会导致权值更新速度非常慢。为了克服这一缺点，我选择了 binary_crossentropy 作为损失函数，binary_crossentropy 是针对二分类问题的交叉熵代价函数，其特点是误差大时权值更新速度快，误差小时权值更新速度慢。

初次训练每次 epoch 使用 10%的样本作为验证集，每次同时训练 64 个样本。在经历了 50 个 epoch 后训练集的 loss 为 1.8637e-04，准确率达到 100%；验证集 loss 为 0.1952，验证集准确率为 96.35%。测试集的 loss 为 0.200，准确率为 96.12%，

3.3 完善

测试集的准确率离期望目标有一定的差距，我希望调整参数以提升算法的性能。模型中可以调整的参数有优化器的‘learning rate’，‘batch_size’，全连接层神经元个数，dropout，最大权值约束。

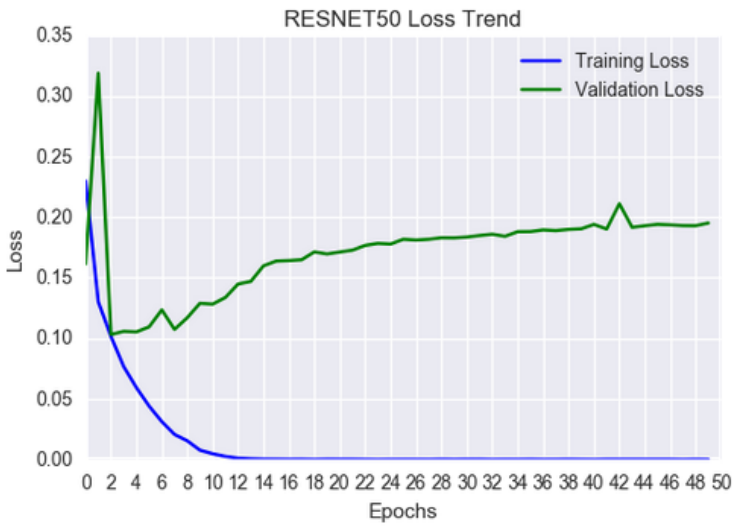

图 3-4 图为第一次训练的误差曲线，其中蓝色曲线为训练集的误差曲线，呈不断下降的趋势；绿色为验证集的误差曲线，当 epoch 小于等于 2 时，曲线急速下降，之后，曲线逐渐上升

观察第一次训练的误差曲线（图 3-4），在 epoch 为 2 的时候，验证误差曲线开始有着明显的上升趋势，这是过拟合的表现之一。因此，我选择添加回调函数防止过拟合，回调函数主要有 EarlyStopping、ModelCheckpoint。设置回调函数可以让模型在每一个 epoch 结束时计算选择的指标，EarlyStopping 的作用是当指标在连续 n（n 是该方法参数 patience 的值）个 epoch 不再降低或上升时模型就终止训练。ModelCheckpoint 的作用是每一个 epoch 结束，每当模型的性能上升，则保存模型。因模型收敛速度较快，所以在不同的参数下，loss 曲线达到最低点的 epoch 数有很大差异。因此，本项目使用 EarlyStopping 观察验证集的 loss 值，如果在 5 个 epoch 内验证集的 loss 值不再下降则终止学习，并同时使用 ModelCheckpoint 记录 loss 曲线达到最低点的对应模型的参数。

方案	验证集最小 loss	验证集准确率	测试集 loss	测试集准确率
初始参数：学习率 1.0，epoch 为 50，batch_size 为 64，全连接层神经元数量为 1024				
初始	0.1952	96.35%	0.200	96.12%
学习率组参数： batch_size 为 64，全连接层神经元数量为 1024				

lr = 1.0	0.1022	96.27%	0.1054	96.12%
lr = 0.5	0.0992	95.91%	0.1084	95.76%
lr = 0.2	0.1036	95.91	0.1084	95.56%
lr = 0.1	0.1007	95.78%	0.1069	95.72%
lr = 0. 05	0.0992	96.27%	0.1063	95.48%
batch_size 组参数：学习率 1.0，全连接层神经元数量为 1024				
batch_size = 32	0.1113	95.64%	0.1170	95.16%
batch_size = 64	0.1076	95.69%	0.1118	95.48%
batch_size = 128	0.1016	96.18%	0.1087	95.88%
batch_size = 256	0.1028	95.91%	0.1027	96.20%
batch_size = 512	0.0982	95.82%	0.1031	96.08%
全连接层神经元数量组参数：学习率 1.0， batch_size 为 256				
nb_neurons = 256	0.1004	96.22%	0.1076	95.92%
nb_neurons = 512	0.1078	95.60%	0.1155	95.68%
nb_neurons = 1024	0.1073	95.55	0.1140	95.52%
nb_neurons = 2048	0.0981	96.27%	0.1025	96.08%
nb_neurons = 4096	0.1000	96.13%	0.1047	95.96%

表 3-1 调整参数对分类验证集、测试集的准确率和误差的影响。依次调整了优化器学习率、训练组样本数量，batch_size，全连接层神经元数量

首先查看学习率对模型效果的影响，让训练集以不同的优化器学习率训练模型，分别为[1.0,0.5,0.2,0.1,0.05],其余参数不变，并加入了 patience 为 5 的 early_stopping。随着学习率的逐渐下降，验证集和测试集最小 loss 的值也大致相同。当学习率为 1.0，测试集的最小 loss 达到 0.1054，相应测试集的准确率达到 96.12%，均优于其他学习率测试出来的模型。因此，我选择使用 1.0 的学习率进行下一步测试。

训练使用批梯度下降法，对于参数 batch_size，batch_size 越大，其优点是可以提高内存的利用率，提高训练效率，梯度下降的方向越准。缺点是每一次 epoch

的迭代次数减少，权值收敛的速度变慢。本项目分别使用[32,64,128,256,512]的 batch_size 训练模型，随着 batch_size 的增加，验证集和测试集的 loss 均呈下降趋势，且当 batch_size 为 256 的时候测试集的 loss 达到最小。而相对应的测试集的准确里也是所有测试组中最高的。因此，我选择使用 256 的 batch_size 进行下一步测试。

全连接层中的神经元数目是一个非常重要的参数。一般来说，一个足够大的单层网络是接近于任何神经网络的。全连接层中神经元的个数对神经网络的训练效果影响很大。神经元个数少了，会容易造成模型欠拟合，反之容易造成模型过拟合。固定其他参数，使用不同的神经元个数训练模型(如表 3-1)，随着神经元数量的增多，验证集的 loss 和测试集的 loss 均大致呈下降趋势，直到全连接层神经元数量到了 4096 是有所回升。当全连接层神经元的个数为 2048 时，验证集和测试集的 loss 均低于其他测试组。因此，我将全连接层的神经元个数上调至 2048 个。

因神经网络的参数众多，导致非常容易发生过拟合的现象。解决过拟合的方式有很多种。比较常见的方法有设置 Dropout 和设置最大范数权值约束。

验证集和测试集的误差值	Dropout = 0.3	Dropout = 0.4	Dropout = 0.5
weight_constraint=1	0.0981，0.1048	0.1022，0.1061	0.1009，0.1085
weight_constraint=2	0.1000，0.1053	0.0994，0.1046	0.0991，0.1035
weight_constraint=3	0.0945，0.1043	0.0993，0.1059	0.0976，0.1055
weight_constraint=4	0.1030，0.1116	0.1114，0.1219	0.0994，0.1044
weight_constraint=5	0.1014，0.1066	0.0983，0.1047	0.0983，0.1086

表 3-2 调整 dropout 正则化参数对分类验证集、测试集的准确率和误差的影响。

有两种防止过拟合的方式。一是 Dropout，设置 dropout 会使模型在训练中每一次的迭代都随机（临时）删掉网络中一定比例的隐藏神经元，输入输出神经元保持不变。被隐藏的神经元不会参与正向传播过程，因此在反向传播时神经元

的权值也不会有任何更新。dropout 的过程相当于对很多个不同的神经网络取平均，从而达到了防止过拟合的效果。第二个方式是增加权值约束，模型在训练的过程中，为了使损失值最小化，模型会不断更新权值去拟合每一个样本。当模型的权值过多的时候，而样本的数量不足够约束这些权值的时候，模型就会出现过拟合的问题。这种情况下无论添加什么样的样本，模型都会很好的拟合数据，使损失值十分接近 0。给模型设置最大范数权值约束可以很好的约束模型的拟合能力，防止模型的过拟合。

使用不同的 dropout 和使用不同的最大范数权值约束训练模型(如表 3-2)，当 dropout 为 0.5，weight_constraint 为 2 时，测试集的最小 loss 达到最低值 0.1035，其对应的验证集的 loss 为 0.0991。因此，将最小 dropout 调整至 0.5，weight_constraint 为 2。

最终将参数组合调整为：训练集样本数 22483，学习率 1.0， batch_size 为 256，神经元数量为 2048，dropout 为 0.5，weight_constraint 为 2。训练完成后，测试集的最低 loss 为 0.0988，测试集的 loss 为 0.1035，测试集的准确率达到 96.00%，比起初次训练测试集的准确率没有明显的提升，但测试集的 loss 减少了且误差曲线没有明显的过拟合。

4 结论

4.1 模型的评价与验证

最终模型的测试集准确率达到了 96.00%，比基准模型的准确率低了 0.43%。模型先后对优化器的学习率、batch_size、全连接层神经元数量、dropout、正则化等参数进行多值比较并选出最优级，参数设置合理。

为了验证该模型的可靠性，使用模型对 The Oxford-IIIT Pet Dataset[10]进行预测，该数据集中具有 25 个种类的狗、11 个种类的猫的照片，每个种类的照片数量均在 200 左右，一共有 7390 张照片。将数据集随机分为 10 份，每一部分 739 张照片。随后分别对 10 份数据集进行预测。表 4-1 中可看到十组数据的平均 LogLoss 为 0.2553，平均 Accuracy 为 91.489%，每组的 LogLoss 和 Accuracy 与平均值的差距不大，皆在合理的范围内。在该数据集中模型的准确率没有原数据集中的高，误差也提升了，我猜测是因为原数据集的照片来源于网络，每种狗的样本数量有所差异，对一些样本数量少的种类模型没有很好的学习到，所以在种类样本数量分布均匀的数据集中模型的表现会有所下降，但仍然处于比较高的水平。综上所述，我认为模型具有很好的稳定性，预测的结果很可信。

	1	2	3	4	5	6	7	8	9	10	平均值
LogLoss	0.2540	0.2197	0.2897	0.1890	0.2395	0.3162	0.3040	0.2482	0.2413	0.2514	0.2553
Accuracy	90.26%	92.56%	90.26%	93.64%	92.15%	90.66%	90.26%	92.96%	91.61%	90.53%	91.489%

表 4-1 10 份数据集的预测准确率和误差及平均值

4.2 结果分析

为了让机器能识别猫狗，我们使用了卷积神经网络模型，首先使用不带头部的含权值的 resnet50 模型得出所有样本的瓶颈特征，再将瓶颈特征连接到全连接层进行分类，参数的选择都经过调整。观察图 4-1，训练过程中，模型在前两个个 epoch 中，训练集准确率与验证集准确率急剧上升，随后的训练训练集准确率和验证集准确率仍缓慢上升，且开始差距逐渐增大，再观察验证集误差，没有明

显的增加，并在第 15 个 epoch 达到最小值 0.0988。说明模型没有明显的过拟合。最后模型的测试集准确率达到了 96.00%，比预期的结果 96.43%要略低。但使用该模型仍然可以让机器拥有了较高的识别照片上猫狗的能力，我认为该模型解决了问题。

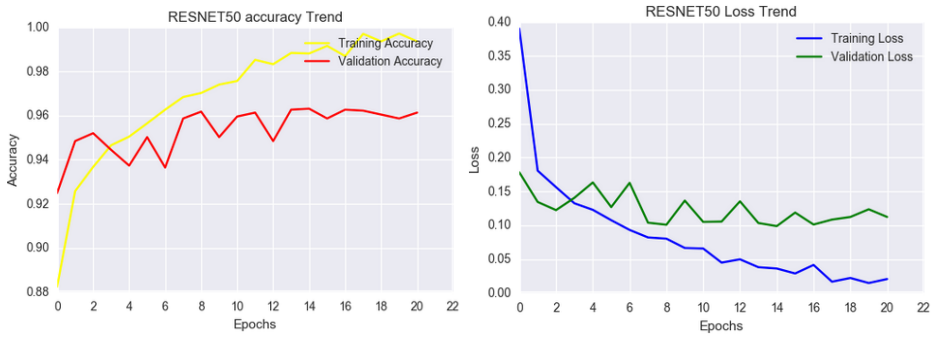

图 4-1 左图为最终模型在训练中的准确率曲线。经 15 个 epoch 训练后，模型的验证集的 loss 达到了 0.0988，对应的验证集的准确率为 96.31%。右图为最终模型的误差曲线，没有明显的过拟合现象。

5 项目结论

5.1 结果可视化（自由把握）

本项目使用的卷积神经网络的主要操作是先使用使用不带头部的含权值的resnet50 模型对图片进行瓶颈特征提取。观察某一样本的卷积特征(图 5-1B)，原图经过'res2a_branch2a'卷积层后输出 16 张特征图，是输入图片分别经 16 个滤波器逐点扫描出来的结果。可以看出，经过一层卷积的特征图还能看出原图的大致轮廓。随着卷积次数的增多，模型提取的特征更加高级化，抽象化。图 5-1C 为模型进过最后一次 merge 层操作后输入的特征图。可以看到 32 张特征图中，很难从图像中观察到特征代表的含义。

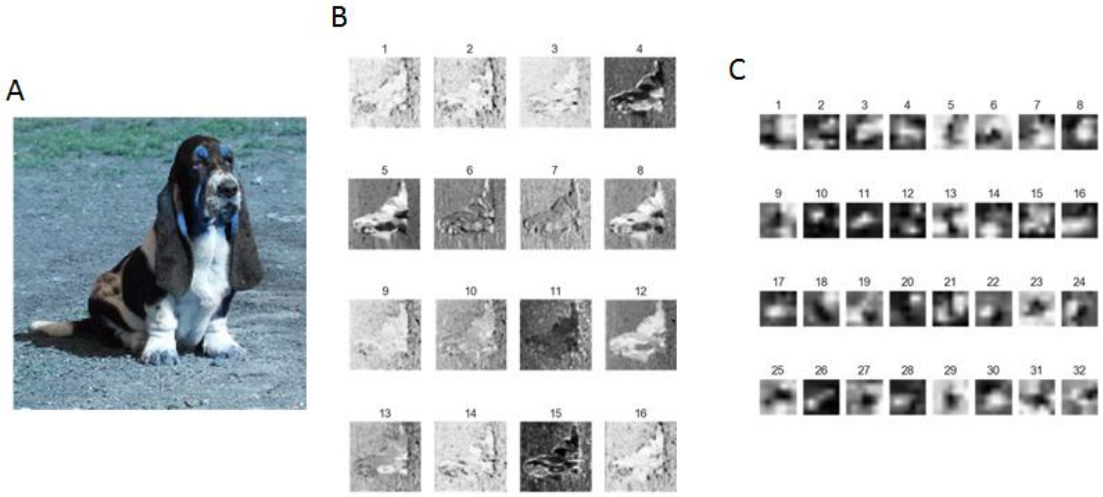

图 5-1 图片的卷积过程，A 为模型的输入数据，B 为经第一层 64 滤波器卷积后的图片（选取了其中 16 个），C 为经最后一层 2048 滤波器卷积后的图片（选取了其中 32 个）。

使用模型对测试集的样本做预测，由于模型的输出层使用的激活函数是 sigmoi 的，sigmoid 函数可以把实数域光滑的映射到[0,1]空间，其输出数字也可看作是输入数据与正类的相似程度。因此可以通过分析预测值与真实标签 0，1 的距离来判断模型对预测结果的可信程度。由图 5-2 的上图可知，模型对大部分样本的预测值都非常靠近 0,1。而在 0.4-0.6 之间的样本不到总样本的 1/15。继续观察左下图和右下图，无论猫组还是狗组，绝大多数的样本的预测值是离真实标签非常接近（差距小于 0.1）。在分类错误的样本中，也是越靠近错误标签

的区域样本分布的数量越多。这说明模型已经找出比较明确的分类依据，但是分类依据的准确性还有提高的空间。

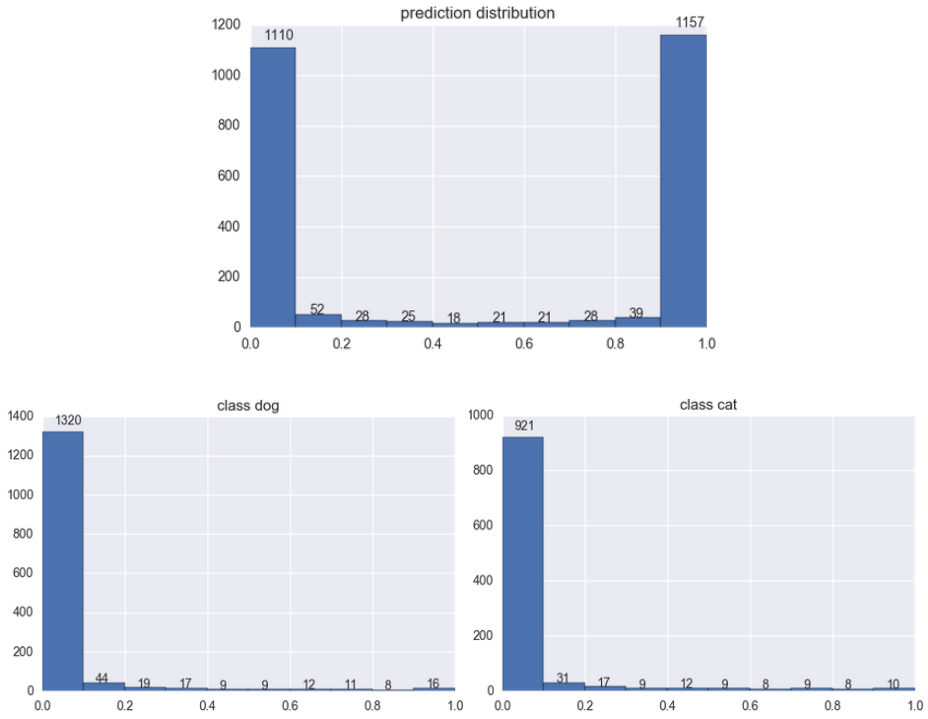

图 5-2 上图为测试集样本的预测值分布，左下图右下图分别为猫组和狗组的样本预测标签与真实标签的差值的分布图

5.2 思考

在让机器识别猫狗个这问题上，本项目使用了卷积神经网络模型，首先搭建不含全连接层的 resnet50 模型，并载入了权值数据，用以提取图像中物体高度抽象化的瓶颈特征。之后建立全连接层将瓶颈特征分成猫狗两类。在训练过程中，项目测试各参数不同数值的对照实验，先后调整了优化器的学习率、batch_size、训练集样本数量、dropout、正则化等参数。最后模型的测试集准确率达到了 96.00%，比预期的结果略差一点，之后使用模型对 The Oxford-IIIT Pet Dataset 进行预测，平均预测准确率达到 91.489%，误差为 0.2553。表明模型具有很好的可靠性。

项目中参数调整这一步骤我认为比较困难，使用所有样本进行一次训练需要数小时的时间，如果使用所有参数组成的多维组合进行网格法搜索来优化参数则

可能需要非常长的时间。但如果逐一对每个参数进行优化，虽然所需时间减少了，但是优化结果并不一定是所有参数组合的全局最优解。经过考虑，本项目最终选择使用逐一对每个参数进行优化的策略。

5.3 后续改进

本项目使用了针对二分类模型改进的 resnet50 模型，最终预测准确率达到了 96.00%，LogLoss 为 0.1035。如今随着硬件的发展，卷积神经网络也有能力使用更多的隐藏层，且层数越多模型参数越多，拟合数据的能力越强，亦可处理更为复杂的数据。但同时层数越多的模型容易发生过拟合现象。因此，选择适当层数的模型以及适当的参数设定成为了提高模型泛化能力的关键。Resnet 模型除了 50 层模型，还具有 18 层、34 层、152 层、1000 层模型，后续的实验可以尝试使用以上模型以提高模型的预测准确率。

本项目对于参数的调整使用的是贪心算法，将多维的参数组合的调整分解成若干个子测试，每个子测试都都单独调整一个参数，其他参数不变，并找出效果最好的参数的值，即局部最优解。之后在使用所有局部最优解的组合进行测试，得出最终的模型。Keras 有 Scikit_Learn 接口包装器，可以让 Keras 的模型使用 grid_serch 接口，如果能对所有参数组成的多维组合进行网格法搜索，可以找出参数的全局最优组合，但也需要要大量的硬件和时间资源。

参考资料

[1]http://baike.baidu.com/link?url=ALi2QaK-EX8PBBqx2Z7I2S4rlOt0kohdytYq-lgO7T4bRPJx1wU2OFuxIJHIp4G-N8b7QqIProGW5ldTeP-d4TxsxTJTak2qvjMSgLn9Naft2wgE_nLiNgODRv28k8z8

[2]https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/data

[3]http://baike.baidu.com/link?url=yVWIpc6AKttIJgffK5ZeUciREIxHoj9ClywNEffy8-892GUIBH_uPZ5-SNh9_qP0oqz_i06vmbTXNqrBx5ftaK

[4]http://baike.baidu.com/link?url=b31Lg-QyzPIuxkoCkDiTncMbw8xCS9HNdOQLCia1G-Eo4Qjtl9Kk-Q0azOumG3vm3yqRlHpjddKzMEi9BBf4ojv4Zeaw-Sf3hgglK2bThlHOULgWtFmLrt7HfKubR1fp7Hu6h0GxLk_cpDyb1R3eOq

[5]http://baike.baidu.com/link?url=leKgAKVu59bgndXaypbZ44_uLK4BOThyKJLOSze1u78kb6mQFT5tvr3UHPufm4d_fwSNQ1yHrrs0qRGeFOMG5HIOWkHdWNOn3smMq-Z64ZiDAplZq5btIrp7vXkvhw-Z

[6] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition. In CVPR 2016

[7]http://www.hackcv.com/index.php/archives/104/?hmsr=toutiao.io&utm_medium=toutiao.io&utm_source=toutiao.io

[8] http://www.image-net.org/challenges/LSVRC/2015/results

[9] https://zhuanlan.zhihu.com/p/22252270

[10] http://www.robots.ox.ac.uk/~vgg/data/pets/