

机器学习工程师纳米学位毕 业项目

猫狗大战



Allen Ren

DECEMBER 12, 2019

1. 问题定义

1.1. 项目概述

项目将训练一个从自然图像中区分猫、狗的模型，模型可以通过一定方式应用到移动端App，网页应用程序中，实现实时解决分类猫狗的问题。

猫狗分类是生活中很常见的问题，是宠物识别问题的一个子集，自从神经网络应用于传统计算机视觉（computervision）领域，图像识别的准确率大大提高，从而可以使用程序替代人类实现很多常见动物、物品的分类问题，比如在Udacity课程中的狗狗品种分类问题。而将算法应用于移动端App和Web应用程序，更是加快了实验理论向实际应用转化的过程。

项目使用的数据集由kaggle提供，包含训练集（train.zip）和测试集（test.zip）。训练集25,000张图片，猫、狗各12,500张，图片名称分别包含dog, cat字符串。测试集12,500张图片，图片名称为1~12,500的数字。项目将使用全部的训练集数据对模型进行训练、验证、测试准确率的工作，测试集不会给出准确率统计，但是会形成报告文件，给出每张图片对应猫、狗的概率以及分类情况。

1.2. 问题陈述

项目采用的数据集是kaggle提供的真实世界采集的数据集。真实场景中，图像识别经常会受到光照、遮挡以及采集过程中由于运动导致的模糊、畸变等影响，给识别带来不可预知的问题，增加识别的难度。

由于采用深度神经网络，随着网络的加深，模型的训练速度会明显降低，这给项目的顺利进行带了很大问题。并且，同一个网络模型在训练过程中是否收敛，收敛速度都是不可预测的，所以需要进行多次训练，这也是需要考虑的问题。

项目最后会给出模型的训练准确率，在训练集子集（从训练集分离出来的测试集）上的准确率，以及在测试集上对图像的分类结果，由于测试集数目庞大，本项目不会统计在测试集上的准确率。

1.3. 评价指标

在对项目模型进行评价时，使用 Logarithmic loss 即 Log loss 作为模型的评价指标。与准确率不同的是，Log loss 会根据预测值与实际标签的差异，考虑模型预测的不确定性，让我们对模型的性能有更细致的了解，而准确率只是预测对的数量占总数的百分比，它只能告诉我们是否的比例。下图是真实的 Label=1 时的 log loss 曲线图

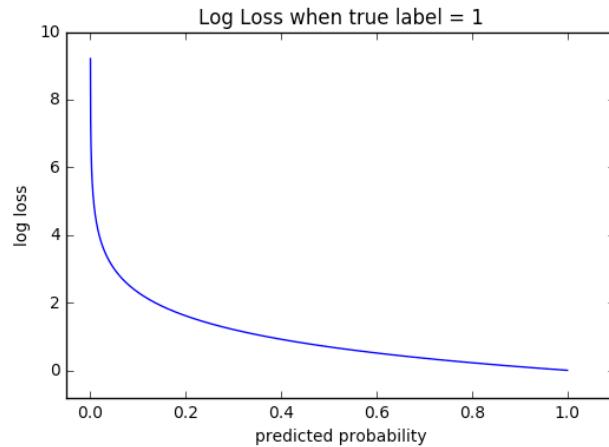


图1 真实值为1时的Log loss曲线图

2. 分析

2.1. 数据探索可视化

影响图像分类的因素有很多，比如图像的大小，图像的清晰度，图像的亮度，图像中是否存在分类种类以外的对象，是否存在严重遮挡等。

首先从最基本的图像大小入手，对数据集进行探索。下图是所有训练集图像尺寸分布的散点图，横轴代表图像宽度的像素值，纵轴代表图像高度的像素值。

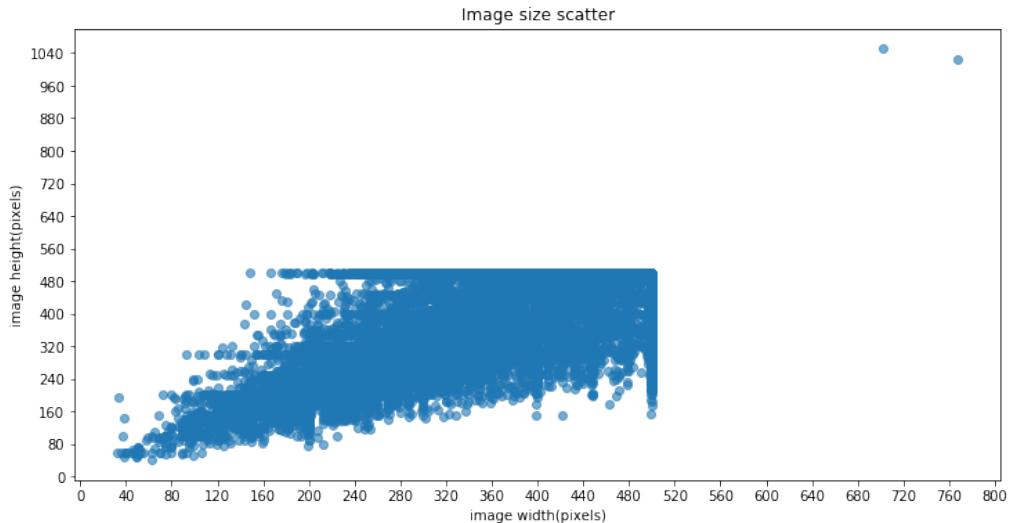


图2 图像尺寸分布散点图

从图中可以明显观察到有2张图像的宽、高远远大于其他图像，而且有一部分图像的宽、高比较小，都小于80pixels。这些数据都可能是异常数据，需要根据后面所选模型的输入条件进行清洗。

其次，图像的清晰度也是影响分类结果的重要因素，接下来从图像清晰度角度探索数据集。有意思的是，严格意义上来说，图像的清晰度是一个主观概念，没有绝对的客观衡量标准。最常见的是数字信号处理中，对图像做傅立叶变换，然后观察图像的高低频特征分布。如果图片

有少量的高频成分，那么该图片就可以被认为是模糊的。然而，区分高频量多少的具体阈值却是十分困难的，不恰当的阈值将会导致极差的结果。

这里，我们的期望是可以使用一个单一的数值来直观表示图像的清晰度度量。Pech-Pacheco 在 2000 年模式识别国际会议提出将图片中某一通道（一般用灰度值）通过拉普拉斯

(Laplacian) 掩模做卷积运算，然后计算标准差，出来的值就可以代表图片清晰度。这种方法之所以有效在于拉普拉斯算子是用于测量图像的二阶导数，突出图片中快速变化的区域。此算法基于以下假设：如果图片具有较高方差，那么它就有较广的频响范围，代表着正常，聚焦准确的图片。但是如果图片具有较小方差，那么它就有较窄的频响范围，意味着图片中的边缘数量很少。正如我们所知道的，图片越模糊，其边缘就越少。

但在下面的探索中可以看到，一些分辨率，具有大量马赛克状的图像也很会有比较高的 Laplacian 卷积值，但是它们绝对是模糊的图像，同时，一些背景模糊但是前景清晰的图像也会由于其较低的 Laplacian 值被错杀。

下图是将训练集图像归一化到 224*224 尺寸下时，数据集的 Lap-value 曲线图，横轴代表图像在数据集中的索引，纵轴是图像对应的 Lap-value。

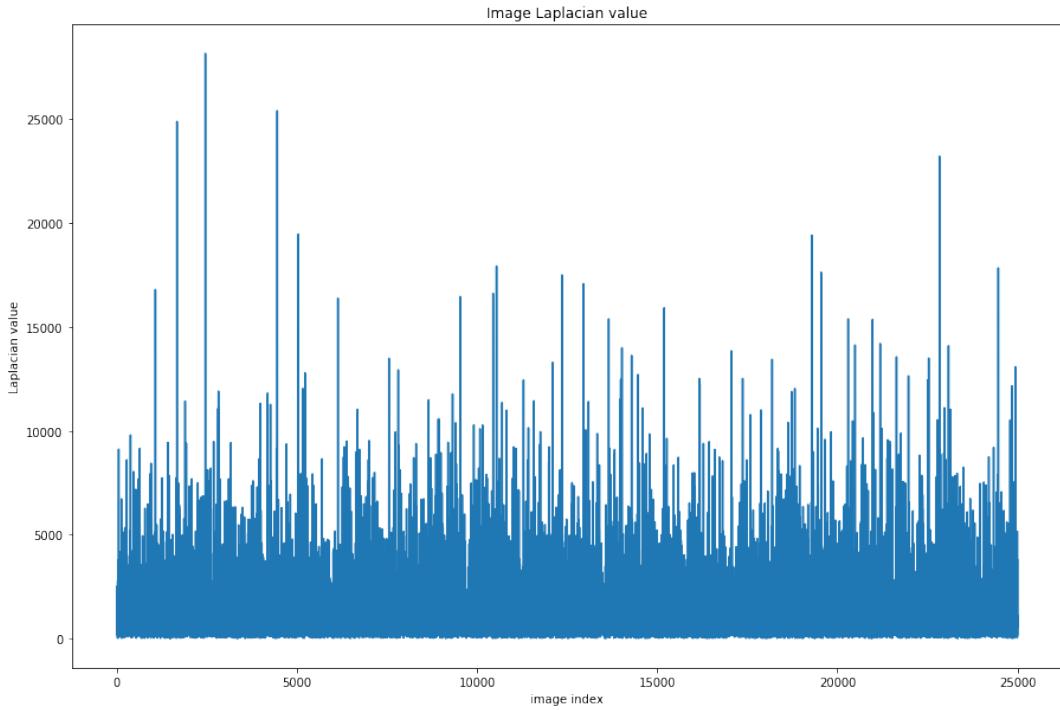


图3数据集Lap-value曲线图

数据非常密集，图3看起来更像是密集的柱状图。而密集的数据导致很难从图中找到数据集的 Lap-value 的最大值和最小值。事实上，通过对 Lap-value 集的直接查找获得数据集 Lap-value 的最大值为：28154.32，最小值为：4.99。

从项目介绍中可以知道，项目要解决的是猫狗分类的二分类问题，但是由于数据集庞大，我并不知道在数据集中（训练集，测试集）是否包含其他的对象，比如人，鸟类，家畜类等，所以，我首先采用随机抽样的方式，从训练集中抽出8张图片进行显示，观察它们包含的对象。



图 4 随机抽取显示训练集图像

虽然在随机抽取的图片中，并未明显的看到有其他种类的动物，但这并不能代表整个训练集的数据都如此理想，在后面的数据清理阶段，会进一步以图像中的对象为目标进行数据探索和清理的工作。

在此如此大量（25000张）的训练集中，显然不适合人工手动标记删除这类数据，所以在后续执行过程中，我使用现有的预训练分类模型，通过对训练集的分类，找出最不合理的图片。

2.2. 算法和技术

由于项目涉及猫狗分类问题，有必要在此简单讨论一下分类算法。

2.2.1. 分类算法

在机器学习项目学习初期，最先接触到的就是分类算法，比如支持向量机，罗辑回归，决策树和随机森林等以及后期可用于分类的神经网络算法。这些算法的输入都是特征向量，对于图片来说，就是图片的像素数据组成的向量，因为图片中每个像素都包含图像的一部分特征。输出是分类的数字类别。

罗辑回归和支持向量机都是通过求解分类超平面的方法实现对数据的分类，不同之处在于逻辑回归是在原特征空间内直接求解分类超平面，而支持向量机是通过核函数将原特征空间数据映射到高维空间进行分类平面的求解，等价于在原特征空间进行分类曲面的求解。但是这类算法对于图像识别分类并不适用，原因在于图像的特征空间就是图像所有像素的集合，也就是说图像的特征分布于整个数据空间，直白点说，要识别的分类对象和图像的背景并不存在明显的数据界限。

决策树和随机森林中，森林的基本组成单元就是决策树，所以随机森林和决策树解决的是同类问题。但是决策树会遇到维度灾难，即在维度增加时，决策树的分割区域会以指数方式增加。随机森林虽然可以在一定程度上解决这个问题，但这种解决只是相对而言，并不是从根本上解决了维度灾难。但即便是质量很低的图像（ $64*64*3$ ），也会有至少10,000个数据，即10,000个维度，所以，决策树和随机森林并不适合高纬分类问题。

2.2.2. 神经网络

人工神经网络（artificial neural network，缩写ANN），简称神经网络（neural network，缩写NN）或类神经网络，是一种模仿生物神经网络(动物的中枢神经系统，特别是大脑)的结构和功能的数学模型或计算模型，用于对函数进行估计或近似。神经网络由大量的人工神经元联结进行计算。

一个典型的神经网络由输入层、中间层（隐藏层）和输出层构成，如下图：

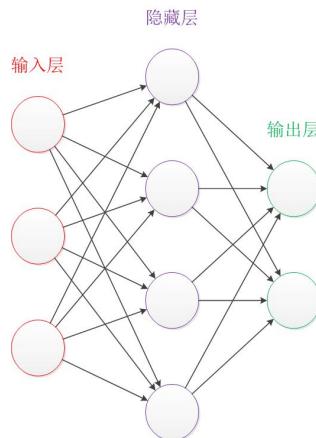


图 5 神经网络结构

图5是典型的三层神经网络结构图。在更大型、更复杂的网络结构中，会包含更多的隐藏层来处理数据。

在神经网络中，输入层和输出的层的节点数往往是固定的，由具体的使用场景确定，中间层（隐藏层）是可以自由指定的；图中每个圆点代表一个神经元，但是图中最重重要的部分是神经元间的连接线，每个连接线代表一个不同的权重，这是需要训练得到的。

神经元是神经网络最基本的组成部分。按照生物学定义，一个神经元通常包含多个树突，用于接受传入信息；而轴突只有一条，轴突尾端有多个轴突末梢用于给其他神经元传递信息。轴突末梢和其他神经元的树突相连从而传递信号；信号的处理在细胞核中完成。如下图所示：

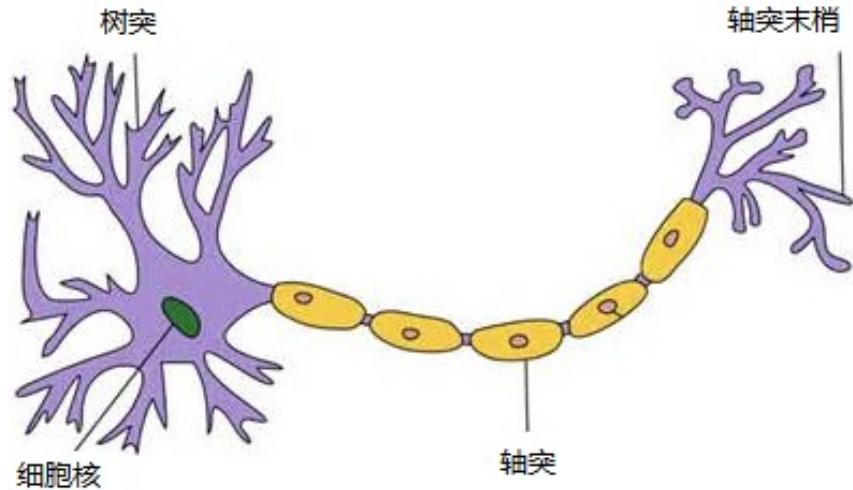


图 6 神经元

在人工神经网络中，神经元模型是一个包含输入，输出与计算功能的模型。输入可以类比于神经元的树突，输出类比于神经元的轴突，计算则类似于神经元的细胞核，信息的处理在此完成。如下图：

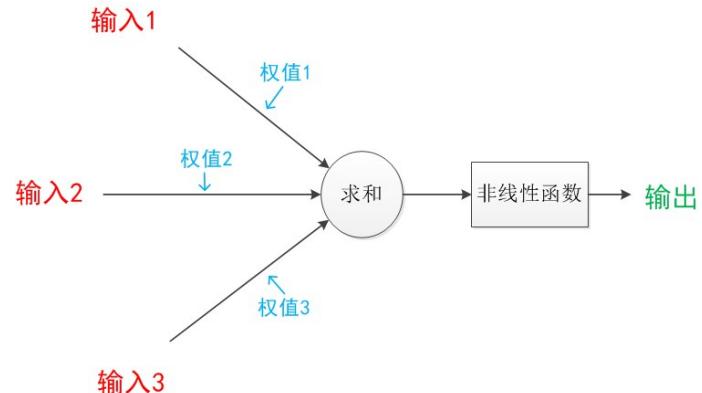


图 7 神经元模型

图中是一个典型的神经元模型，包含3个输入，1个输出和两个计算功能，中间的连接线称为连接，每个连接代表一个权重。每一个神经元模型是一个简单的非线性函数，输入首先被加权求和，然后通过非线性函数（通常所用为sigmoid或relu）得到输出。神经网络中的每个神经元具有相同的结构，不同之处在于输入的权重有差别。

神经网络具有自适应性，使用固定的权重很难很好的逼近数据到类别的映射关系，神经网络通过反向传播来更新权重，完成分类工作。在线性分类问题中，通常使用平均平方差函数作为损失函数对模型进行参数约束。对于逻辑分类问题，最标配的是交叉熵函数，即预测结果与样本真实类别之间计算得出的交叉熵，此函数对于每一个权重是可

导的，因此能使用梯度下降方法更新权重值。这为神经网络的反向传播算法提供了一个可以快速更新权重的方法，使模型可以更快的收敛到最优。

根据神经元连接方式不同，神经网络可分为前馈神经网络（feed-forward neural network）和递归神经网络（recurrent neural network）。

本项目中使用的卷积神经网络属于前馈神经网络。前馈神经网络的特点是网络由不同的层组成，同层之间的神经元没有连接，只与下一层神经元相连。一个神经网络中的层数称为该网络的深度，信息从输入层开始延深度方向一层一层向前传播，此亦即深度学习的由来。

2.2.3. 卷积神经网络（CNN）

卷积神经网络的研究始于二十世纪 80 至 90 年代，最早出现的卷积神经网络是时间延迟网络和 LeNet-5，在二十一世纪后，随着深度学习理论的提出和计算设备的发展，卷积神经网络快速发展，并应用于计算机视觉，自然语言处理等领域。卷积神经网络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。卷积神经网路由一个或多个卷积层和顶端的全连通层（对应经典的神经网路）组成，同时也包括关联权重和池化层（pooling layer）。这一结构使得卷积神经网路能够利用输入数据的二维结构。与其他深度学习结构相比，卷积神经网路在图像和语音识别方面能够给出更好的结果。这一模型也可以使用反向传播算法进行训练。

卷积神经网络名称中，“卷积”代表数学卷积运算，卷积层通过卷积核对输入数据进行采样，是一种特殊的线性运算。其核心组成部分如下图所示：

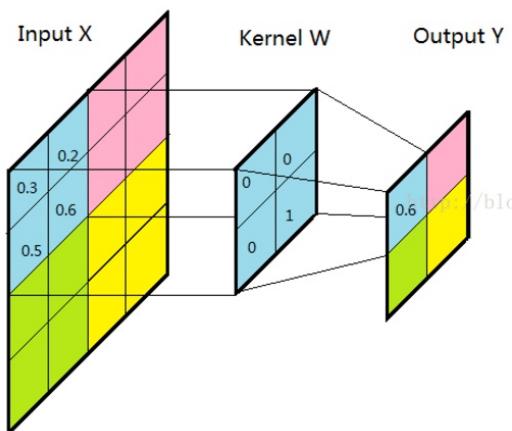


图 8 卷积神经网络核心部分—卷积

4x4 的输入经过 2x2 卷积核采样后，输出 2x2 的数据

和传统的前馈神经网络不同的是，卷积神经网络中，后一层每个神经元只与前一层的部分神经元相连，相连的部分取决于卷积核的大小。因此，卷积神经网络的前几层只处理局部

的输入，但是全部的隐藏层会具有全局性。这等价于强制约束传统的全连接神经网络中的一些权重为零。

卷积神经网络具有平移不变性，这个特性是由于同层网络中的神经元具有相同的权值，这就决定了同层神经元提取的特征是相同的，所以不管特征出现在图像的什么位置，都能被这一层的神经元检测到。一般一层神经元具有多组权值用来检测多种特征。

卷积神经网络的局部连接和权值共享大大减少了模型的参数，也就是说，使用相同的硬件资源（内存，CPU，GPU），可以训练深度更大，包含神经元更多的网络，从而提高模型的准确率。

2.2.4. 算法

在Keras包含的众多模型中，几乎每个模型都包含庞大的参数，使用ImageNet数据集进行训练时，往往需要使用高性能的硬件经过数天或者数十天的训练才能完成。在本项目环境下，显然不具备这样的硬件条件，而且时间上也来不及。基于这些客观条件的限制，最好的办法是使用迁移学习来完成项目。

所以，本项目使用的算法是基于**迁移学习的模型融合**。

2.2.5. 技术

2.2.5.1. 迁移学习

如上面算法中所讨论的那样，迁移学习是算法中使用的基础技术。什么是迁移学习？迁移学习的使用方法和使用场景是什么？如下进行一一细说。

迁移学习 是一个使用已经训练好的模型来解决相似任务的过程。这个已经训练好的模型被称为预训练模型。相似任务是指当前任务与已经训练好的模型具有类似的任务目标，比如识别，分类等，并且数据集有一定的相似性，如训练集都是包含花、草、猫、狗、人等对象的图像，而不是使用在图像集上训练的模型进行自然语言的处理。在迁移学习过程中，可以将预训练模型的一个或多个层应用于新模型中来解决当前的新任务。

Keras库中有很多预训练的经典模型，如：VGG，ResNet，Xception等，都可以作为迁移学习的基本模型使用。使用这些经典模型作为新问题模型的基础模型，有很多好处：

- 1、**已经学习的有用特征**：这些模型已经从ImageNet提供的含有1,000,000张图像的数据集中学习了如何从图像中检测数据的通用特征。
- 2、**最前沿的技术**：这些模型在设计解决相应问题时，都使用了当前最前沿的技术，并且在它们各自的图像识别领域有很好的表现。
- 3、**容易获取**：这些模型都已经包含在本项目使用的Keras神经网络库中，并且它们预训练的权重可以很容易的通过网络下载。

最后，最重要的一点，使用迁移学习可以大大降低模型的训练时间，并且可以有效的控制在泛化数据上的误差。

使用迁移学习非常灵活，比如可以将模型照搬直接集成到具有图像识别、分类功能的App中，也可以将预训练模型作为特征提取的模型使用，这时候，预训练模型顶层以下层的输出将作为新的分类模型的输入。

总的来说，预训练模型的使用方法主要有以下几种：

1、直接作为分类器使用：直接将预训练模型及其参数作为分类器执行图像图像分类任务。

2、作为独立的特征提取工具使用：使用模型或模型的一部分用来执行图像的预处理和相关的特征提取任务。通过调用模型中不同的网络层来提取不同的特征，越靠近输入层（底层）的网络层，提取的特征越基础，比如直线、点等特征，越靠近输出层（顶层）的神经层提取的特征越复杂。通过调用这些层，可以将它们的输出保存，作为后续处理的输入使用。

3、作为集成的特征提取器使用：将模型或模型的一部分集成到新的模型中使用。此时，集成到新模型中的预训练模型层的权重会被冻结，在新模型的训练过程中这些参数不会被更新。

4、权重初始化：将模型或模型的一部分集成到新的模型中使用，并在训练过程中作为新模型的一部分，调整之前训练过的权重。

上面四种使用预训练模型的方法，没有优劣高下之分，不管使用哪种方法，往往都要根据不同的任务进行模型的调优并经过多次实验才能得到理想的效果。

本项目中将使用VGG19、ResNet50、Xception三个预训练模型，它们的基本信息如下

Model	VGG19	ResNet50	Xception
Layers	26	177	134
Total params	143,667,240	25,636,712	22,910,480
Trainable params	143,667,240	25,583,592	22,855,952
Non-trainable params		0	53,120
Input size	(224,224,3)	(224,224,3)	(299,299,3)

图 9 预训练模型基本信息

2.2.5.2. 模型融合

在介绍迁移学习时提到，每个预训练模型在设计之初都是侧重不同的任务目标，这就决定了对统一训练集而言，每个模型训练得到的权重偏重是有差别的。在使用迁移学习时，不同模型在同一数据集上的表现就会不同。为了尽可能的对模型扬长避短，可以综合不同预训练模型的特征，达到最好的效果。这就是模型融合的初衷。模型融合是一种非常有效的技术，它可以明显提升ML任务的表现成绩。通过把多个单模型融合在一起，能够降低bias, variance, 控制Overfitting, 提高准确率。模型融合有多重方法，但是基本分成两个不同阶段。1、对不同模型预测结果进行融合。可以采取投票，加权平均等方式，2、在模型构建时对不同模型进行融合。具体到本项目中，可以将不同预训练模型的不同层嫁接在一起，组成新的模型，也可以根据迁移学习使用方法的提示，将预训练模型的权重倒出，对不同模型的预训练权重进行融合，然后最为新的权重最为新模型的输入。

本项目中，将使用第二种方法，将不同的预训练模型权重倒出，进行简单的拼接融合作为新模型的输入使用。

2.2.5.3. 数据增强

在图像识别、分类任务中，数据的质量至关重要，关系到模型在训练中对整个数据集特征的提取。在本项目中，会讨论数据的一般增强方法，比如图像的直方图均衡化对数据的影响。在实际使用中，会使用Keras的ImageDataGenerator方法，此方法会对数据进行随机的缩放/旋转，以增强数据的多样性，避免数据同质化。

2.2.5.4. 优化器

模型的训练过程就是通过不断优化模型的损失函数，将模型损失降到最低，来无限逼近理论最优解。而降低模型损失，优化损失函数就是通过优化器执行的。在机器学习发展过程中，有众多的优化器。基本都是通过对损失函数梯度进行优化来达到目的。常用的优化器有SGD, Momentum, Adadelta, Rmsprop, Adam等。虽然它们采用的技术各不相同，但是它们有共同的基本参数：学习率，学习率衰减。SGD最简单，只有这两个参数；Momentum算法参考了物理学中动量的概念，是对SGD的改进，在一定程度上可以避免模型陷入局部最优解，但是没有解决在训练中学习率消失的问题，这导致模型可能提前结束训练过程；Adadelta, Rmsprop, Adam三个算法都存在动态改变学习率的机制，在训练过程中不再使用单一的学习率，通过考虑训练时的时序关系动态改变学习率，这有效避免了学习率消失带来的影响，而且将每次迭代的学习率控制在一定范围内，从而也避免了模型训练过程不平滑的问题。

目前，最常用的是Adam优化器，它能更快的决定损失函数的梯度方向，从而将模型更快的优化到最优。

本次项目中，将使用SGD 和Adam优化器，并通过调整它们的参数来优化模型。

2.2.5.5. Dropout

在模型训练过程中，由于模型参数数量庞大，并不是所有的参数都能得到差不多的训练机会，这就回导致一部分参数被过度训练，而一部分参数得不到训练，最后导致模型的过拟合。所以在模型中加入 Dropout 层，输入参数是一个(0,1)的概率，在训练过程中，模型会以输入的概率随机丢弃一部分参数，即将这部分权重置 0，重点训练另一部分参数，如此重复不断的对训练的参数进行调整，可以在一定程度上放置模型的过拟合。

2.2.6. 基准指标

项目使用kaggle提供的训练数据和测试数据，本项目的最低指标是kaggle Public Leaderboard的前10%，根据kaggle数据，目前最新的参赛队伍有4452支，所以最低目标是项目的模型表现能排到至少第445名，对应的数据是模型的loss score不高于0.05602。

3. 方法

3.1. 数据预处理

3.1.1.1. 图像增强—直方图均衡化

前面的讨论中提到，数据的清晰度对图像识别、分类任务的准确率有很大影响，这里通过对低 Lap-value 的图像做直方图均衡化，来观察直方图均衡化对图像清晰度的影响。

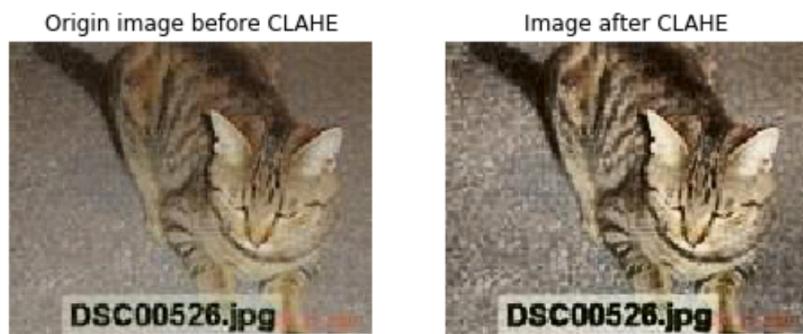


图 10 直方图均衡化前后

可以明显看出，使用 CLAHE 增强后的图像，明显比原图有更高的对比度，图中动物的细节更明显。有理由猜测这会对之后将要训练的模型表现有很大提升。

3.1.1.2. 数据清洗

在训练模型前，确保数据集中不存在严重干扰数据是数据清洗的目的。但是在大批量的数据集中，很难通过手动标记在规定的时间内完成对数据的人工分类。本项目中，通过使用预训练模型对数据集进行分类，来完成干扰数据的查找工作。具体步骤如下：

- 1、 使用 ImageNet 数据集的分类标签对训练集数据进行分类。ImageNet 数据集中，狗的分类共 118 种，猫的分类共 7 种，如下图：

- 2、确定预训练模型的分类范围。在 Keras 的官方文档中，可以发现，预训练模型，如 ResNet50 的 top-5 > top-1。top-1 是指一张狗的图片，模型判断它是狗的概率最高，即最大概率是狗的准确率。top-5 是指，模型给出 5 个可能分类，5 个概率中包含正确分类的准确率。由 top-5 > top-1 可以简单推断，top-N > top-(N-1)，这给我们确定使用那个层级的准确率提供的思路。
- 3、使用 top-10 准确率，两个预训练模型 VGG19, ResNet50 分别对数据集进行预测，将确定的可疑图像合并。
- 4、使用 top-30 准确率，三个预训练模型 VGG19, ResNet50, Xception 对上一步中产生的数据集进一步分类，生成最终的疑似图像库。
- 5、人工筛选疑似图片。

通过以上步骤，共筛选出 82 张疑似图片，随机选取 8 张进行显示，如下：

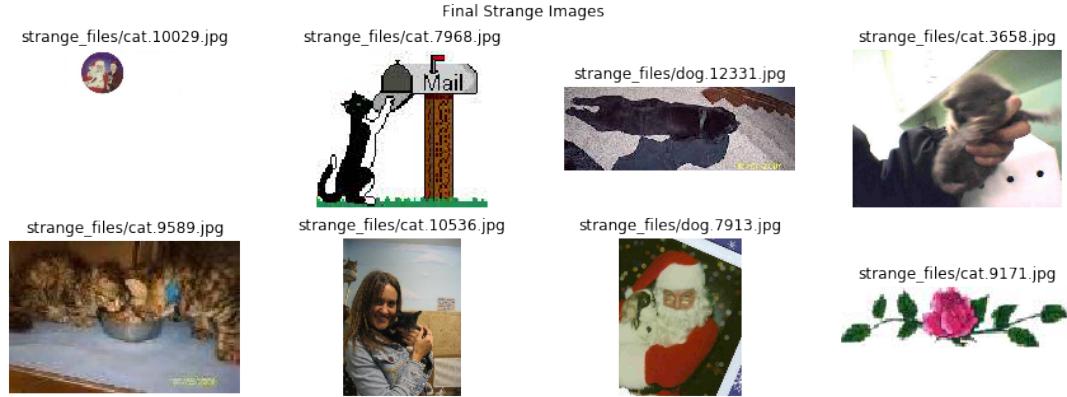


图 11 异常数据展示

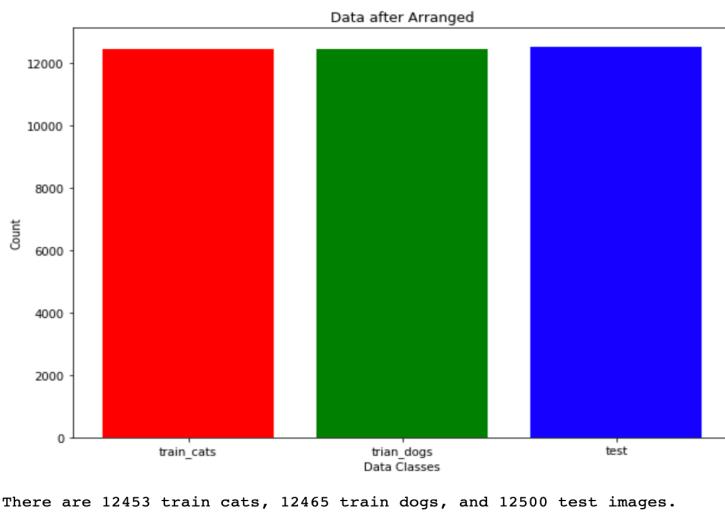


图 12 最终数据集

清理后最终得到 12453 张猫类图片，12465 张狗类图片，12500 张测试图片。

3.2. 执行过程

3.2.1. 准备数据

本项目中，使用 Keras 的 ImageDataGenerator 来进行数据的生成。此函数要求不同种类的图片存放在不同的文件夹里，这部分工作在代码里完成，整理后的文件夹目录如下图：

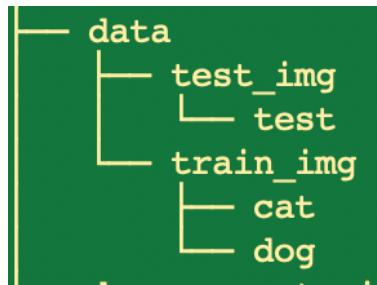


图 13 数据目录

3.2.2. 提取特征向量

VGG19, ResNet50, Xception 三个模型对输入数据都有要求，比如：VGG19 和 ResNet50 的图像默认尺寸为 224×224 , 要求必须有 3 个通道，且图像最小宽、高不能小于 32；Xception 模型的图像默认尺寸为 299×299 , 要求必须有 3 个通道，且图像最小宽、高不能小于 71，并且默认输入数值在 $(-1, 1)$ 范围内。当输入数值不符合默认要求时，使用每个模型的预处理函数 `preprocess_input` 即可将输入图片处理成该模型的标准输入。当要输入与默认图片大小不同的图片时，只需传入当前图片大小即可，但是要保证输入的图片大小在允许的范围内。

一般的卷积神经网络在前面的若干层都是卷积池化层及其各种变种，后面几层是全连接层。全连接层之前的网络层被称为瓶颈层（bottleneck）。将图片输入训练好的卷积神经网络直到瓶颈层的过程可以看做是对图像进行特征提取的过程。通常为了减少内存的消耗，加快计算的过程，再将瓶颈层的结果输入全连接层之前，做一次全局平均池化，比如 ResNet50 瓶颈层输出结果是 $7 \times 7 \times 2048$ ，如果直接输入到全连接层，参数会非常多，所以进行一次全局平均池化，将输出矩阵调整为 $1 \times 1 \times 2048$ ，这种做法额外的好处是可以通过减少参数数量降低模型的过拟合程度。

在 Keras 中载入预训练模型，设置 `include_top=False` 去掉顶层的分类层，设置 `pooling='avg'` 指明使用全局池化层，在使用全局池化层后，VGG19 输出的特征向量是 1×512 ，ResNet50 和 Xception 输出的特征向量是 1×2048 ，将这三个向量拼接，最后得到 1×4608 的特征向量。

```
from keras.models import *
from keras.layers import *
from keras.preprocessing.image import *
import h5py

def extract_bottleneck(input_model, image_size, size=None, func = None):
    width = image_size[0]
    height = image_size[1]
    input_tensor = Input((width, height, 3))
    x = input_tensor
    if(func):
        x = np.expand_dims(x, axis=0)
        x = Lambda(func)(x)

    base_model = input_model(input_tensor, weights='imagenet', include_top=False)
    model = Model(base_model.input, GlobalAveragePooling2D()(base_model.output))

    # image enhancement
    generator = ImageDataGenerator()
    train_gen = generator.flow_from_directory('data/train_img', image_size, shuffle=False,
                                                batch_size=size)
    test_gen = generator.flow_from_directory('data/test_img', image_size, shuffle=False,
                                              batch_size=size, class_mode=None)

    train_data = model.predict_generator(train_gen, train_gen.samples//size)
    test_data = model.predict_generator(test_gen, test_gen.samples//size)

    with h5py.File("bottleneck_%s.hdf5" % input_model.__name__) as h5:
        h5.create_dataset('train', data=train_data)
        h5.create_dataset('test', data=test_data)
        h5.create_dataset('label', data=train_gen.classes)
```

图 14 提取特征向量

3.2.3. 加载特征向量

经过上面的代码以后，将预训练模型 VGG19, ResNet50, Xception 的特征向量保存在了三个文件中，分别是：

```
bottleneck_VGG19.hdf5
bottleneck_ResNet50.hdf5
bottleneck_Xception.hdf5
```

现在载入这些特征向量，并且将它们合成一条特征向量，然后把 `X_train` 和 `y_train` 打乱，设置 numpy 的随机数种子为 2019126。如下图：

```
from sklearn.utils import shuffle
np.random.seed(2019126)

X_train = []
X_test = []

for filename in ["bottleneck_VGG19.hdf5", "bottleneck_ResNet50.hdf5", "bottleneck_Xception.hdf5"]:
    with h5py.File(filename, 'r') as h:
        X_train.append(np.array(h['train']))
        X_test.append(np.array(h['test']))
        y_train = np.array(h['label'])

X_train = np.concatenate(X_train, axis=1)
X_test = np.concatenate(X_test, axis=1)

X_train, y_train = shuffle(X_train, y_train)
```

图 15 加载特征向量

3.2.4. 构建模型

载入预处理的数据之后，先进行一次概率为 0.5 的 dropout，然后直接连接输出层，激活函数为 sigmoid，按之前的讨论，优化器首先使用默认参数的 SGD，输出一个零维张量，表示某张图片中有狗的概率。如下图：

```
input_tensor = Input(X_train[1:])
x = Dropout(0.5)(input_tensor)
x = Dense(1, activation='sigmoid')(x)
model = Model(input_tensor, x)

model.compile(optimizer='sgd', loss='binary_crossentropy', metrics=['accuracy'])
```

图 16 构建模型

模型的拓扑图如下：

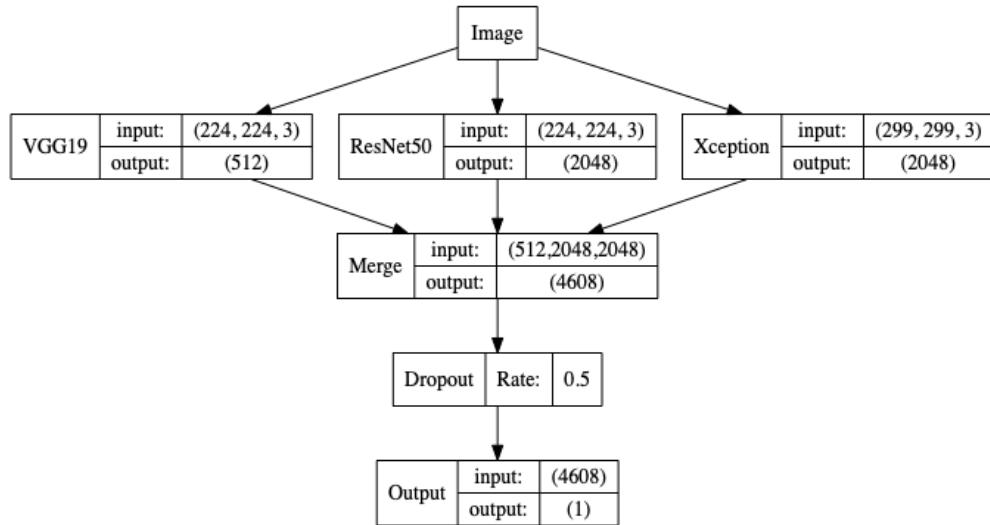


图 17 模型拓扑图

4. 结果

4.1. 模型评价

通过前面的项目完成过程，可以看出，从最开始的数据集探索，数据集清理，再到预训练模型的特征提取，然后是模型的构建以及模型的调优过程，最终得到模型的调优训练结果如下：

params:	lr	decay	β_1	β_2	train loss	train acc	val loss	val acc
SGD	default	default	/	/	0.0303	0.9896	0.0251	0.9928
SGD_opt1	0.02	default	/	/	0.0247	0.9911	0.0206	0.9940
SGD_opt1	0.02	10^{-7}	/	/	0.0216	0.9928	0.0194	0.9948
params:	lr	decay	β_1	β_2	train loss	train acc	val loss	val acc
Adam	default	default	default	default	0.0173	0.9935	0.0170	0.9954
Adam_opt1	0.002	default	default	default	0.0167	0.9947	0.0183	0.9958
Adam_opt2	0.002	default	0.92	default	0.0176	0.9952	0.0291	0.9948
Adam_opt3	0.002	default	0.92	0.995	0.0164	0.9961	0.0365	0.9952
Adam_opt4	0.002	10^{-8}	0.92	0.995	0.0091	0.9978	0.0362	0.9960

图 18 模型参数优化记录

根据上面的结果和可视化图表，可以看出，不同的优化器对模型影响很大。在 SGD 调优时，可以看到 SGD 的 train loss 和 val loss 几乎是同步降低的，这说明模型性能得到了提升，并且保证了很好的泛化能力，模型没有过拟合，可以继续沿着当前思路进行模型优化。而在使用 Adam 优化器时，在参数调优时，模型的 train loss 和 val loss 是背离的，train loss 在不断下降，而 val loss 却在上升，说明模型的泛化能力在降低，模型正在过拟合，需要改变参数调优思路。

4.2. 模型选择

最终，选择 Adam 优化器的默认值参数训练结果，对测试集进行分类，生成 pred_final.csv 文件，上传到 kaggle 进行评分，模型的最终得分为 0.04243，这达到了预期的目标。

[pred_final.csv](#) 0.04243
25 minutes ago by Allen Ren
renren finally upload for udacity graduation program

图 19 模型在 kaggle 评分

模型在测试集上的结果，前 15 张展示如下：

可以看到在前10张图中，第6张图像分类错误，将猫分类到了狗的类别当中。由于测试集数量庞大，12,500图片，在项目环境中没有条件制作分类标签，所以没有统计识别准确率，根据预训练模型 top-1 和 top-5 的数据，模型的分类准确率大概在 85%左右。



图 20 测试集分类结果

5. 结论

5.1. 对项目的思考

猫狗分类问题在深度学习应用于计算机视觉领域时就已经是各类算法模型验证模型表现的典型问题。这在神经网络的应用中属于经典问题。首先猫、狗是生活中很常见的动物，很容易收集数据；其次，猫、狗有很多共同的特征，比如都是四肢行走动物，毛发相似度很高，都有尾巴，个头儿虽然有差异，但是个别品种的狗和猫相差不大等等。所以对猫、狗能否正确分类，就对算法模型提出的很高的挑战，毕竟从图像上看，人类本身也很容易犯错误。

深度学习无疑是处理图像问题的最佳机器学习模型，并且在 CNN 提出以后的各种图像处理赛事中，深度学习模型都取得了很好的成绩。但是相比于传统的机器学习模型，深度学习需要更多的数据，更强大的算力和资源。这也给深度学习的普及和推广设置了瓶颈。本项目是在 AWS 云平台上完成，主要步骤虽然也比较耗时，但是相对于个人笔记本电脑来说，性能还是相当强大的。

Kaggle 上猫狗大战前几名的 loss 达到了 0.0330，相比于本文中的 0.04243，绝对值减少了 0.0943，说明模型算法还有很大的改进空间。本项目中只是简单的使用了 VGG19,Xception,和 ResNet50 这三个模型进行了提取特征向量，然后将特征向量直接拼接，忽略了模型的差异性

和特征向量间的关系。而且 VGG19 所产生的 512 维度的特征向量相对于 Inception_V3 等模型少很多，这也是有待提高的地方。除了这三个模型，还可以增加更多新的模型，或者使用 stacking 的方法进行模型融合，进一步降低方差，提高分类的准确率。

另外可以将样本增强之后再使用，正如前面讨论中对图像增强方法——直方图均衡化的论述，图像对比度的增加肯定会对模型的表现有很大影响。

还可以在使用 `ImageDataGenerator` 数据生成器时，调整默认参数，给数据增加随机的偏移，旋转等，增加数据丰度。

5.2. 需要改进的工作

回顾整个项目过程，需要改进的工作主要集中在以下几点：

1、数据预处理。

数据预处理对模型的训练很关键，在本次项目中，对数据的预处理做的不足，也可能是导致模型准确率提高不明显的关键原因。对图片数据的预处理和数据集的预处理不同，对图片数据的预处理，主要是对图片施加随机的旋转，缩放等操作，从而丰富数据集，消除图片同质化。

2、模型训练。

在模型训练方面，需要学习更多人的经验，对模型的训练进行调优，这主要包括模型的输入是否有最优区间，模型融合时不同模型之间的关系等。

3、模型调优。

在模型调优工作中，主要集中在对优化器超参数的调优上，还可以从模型本身入手，对模型不同层级的输入输出进行调整。

6、参考资料

- [1] F. Chollet. Keras. <https://github.com/fchollet/keras>, 2015
- [2] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
- [3] Chollet F. Xception: Deep learning with depthwise separable convolutions[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 1251-1258.
- [4] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.

- [5] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770–778.
- [6] Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the inception architecture for computer vision[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 2818–2826.
- [7] Canziani A, Paszke A, Culurciello E. An analysis of deep neural network models for practical applications[J]. arXiv preprint arXiv:1605.07678, 2016.
- [8] Building powerful image classification models using very little data
- [9] Dogs vs. Cats: Image Classification with Deep Learning using TensorFlow in Python.
- [10] ImageNet: VGGNet, ResNet, Inception, and Xception with Keras.
- [11] 杨培文, Udacity 手把手教你如何在 Kaggle 猫狗大战冲到 Top2%. 2019