

猫狗识别算法

----机器学习纳米学位毕业项目

张济强

2019 年 7 月 18 日

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.....4
- 2 分析.....5
 - 2.1 数据的探索.....5
 - 2.2 数据可视化.....5
 - 2.3 算法和技术.....7
 - 2.3.1 深度学习.....7
 - 2.3.2 卷积神经网络.....8
 - 2.3.3 卷积运算.....9
 - 2.3.4 优化器 Adam.....10
 - 2.3.5 Dropout 丢弃法.....10
 - 2.3.6 VGG 模型介绍.....11
 - 2.3.7 ResNet 模型介绍.....13
 - 2.3.8 Inception 系列模型介绍.....14
 - 2.3.9 Xception 模型介绍.....19
 - 2.3.10 集成学习-组合模型策略.....20
 - 2.3.11 技术实现框架.....21
 - 2.4 基准模型.....21
- 3 具体算法实现.....23
 - 3.1 数据清洗.....23
 - 3.2 数据预处理.....25
 - 3.3 迁移学习.....25
 - 3.3.1 初始模型选择.....26
 - 3.3.2 输入数据和调整.....26
 - 3.3.3 模型的训练.....26
 - 3.3.4 评估结果.....27
 - 3.4 算法改进.....27
 - 3.5 算法进一步改进.....29
- 4 结果.....31
 - 4.1 模型的评价与验证.....31
 - 4.1.1 单一模型选择.....31
 - 4.1.2 组合模型策略.....31
 - 4.1.3 权重比例分配策略.....31
 - 4.2 实验结果分析.....32
 - 4.2.1 单一模型实验结果.....32
 - 4.2.2 组合模型的实验结果.....32
 - 4.2.3 权重比例分配方案的实验结果.....32
- 5 项目结论.....33
- 参考文献.....35

1 问题的定义

1.1 项目概述

本项目是 Kaggle 猫狗识别竞赛项目,用来解决识别给定图像是猫还是狗的问题。是一个常见的图像识别分类问题,图像识别也是计算机视觉中只要研究方向之一。

项目使用的是 Kaggle 网上提供的猫狗数据集,包含 25000 张训练数据图片和 12500 张测试图片。训练集中图片给出了 dog 和 cat 的标签属性,而测试集中每张图片给出了 id 号,每个 id 号对应一个关于这张图片是猫还是狗的概率,最终会生成一个 id 和概率的表格,提交到 Kaggle 上获得模型的评分。

本项目我们将使用深度学习的卷积神经网络来构造模型,从而来识别图片猫狗分类问题。卷积神经网络 (Convolutional Neural Networks, CNN) 是一类包含卷积计算且具有深度结构的前馈神经网络 (Feedforward Neural Networks), 是深度学习 (deep learning) 的代表算法之一。

卷积神经网络研究始于二十世纪 80 至 90 年代,随着深度学习理论的提出和数值计算设备的改进,使得卷积神经网络的计算得到了巨大的提升,也被广泛应用于各种领域。例如计算机视觉,自然语言处理,语音识别,无人驾驶,机器人等。

1.2 问题陈述

本项目本质是一个图片分类问题,我们需要从训练集中读取图片,利用这些图片数据来训练我们的模型,再使用这个模型来识别测试集合中的图片是猫还是狗。

我们采取的策略分为数据预处理,特征提取,设计模型架构,训练模型,评估模型,测试模型这几个步骤。我们会将读取训练集图片数据,对每个像素使用卷积运算,从而获取图片的特征,随着网络深度的不断加深,卷积运算帮我们提取简单到复杂的各种特征,然后将特征交给分类器来进行分类,分类器会给出每个类别的对应概率,根据概率的大小我们就知道每个图片的类别了。

1.3 评价指标

我们使用的模型评估指标和 Kaggle 的评估标准一致，因为是个判断猫狗的问题，所以只有两个类别，因此我们使用二元交叉熵对数损失函数，计算公式如下：

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

使用这个评价指标能够更精确的评价模型的性能，如果使用传统的分类准确率来衡量的话，我们测试发现模型的分类准确率都很高，在损失不同的情况下分类的准确率几乎没有分别，因此使用损失函数来能够更加细致的衡量模型的优劣。其根本原因在于交叉熵是衡量基于概率分布的差异。

2 分析

2.1 数据的探索

我们从 kaggle 上下载数据并解压压缩包, 看到数据分为两部分, 一部分是训练集, 放在 train 文件夹下, 一部分是测试集方式 test 文件夹下, 我们发现数据本身没有提供验证集, 因此我们会从提供的训练集中随机抽出一部分作为验证集。我们将图片读取图片数据, 将图片的原始像素值作为初始的输入样本。

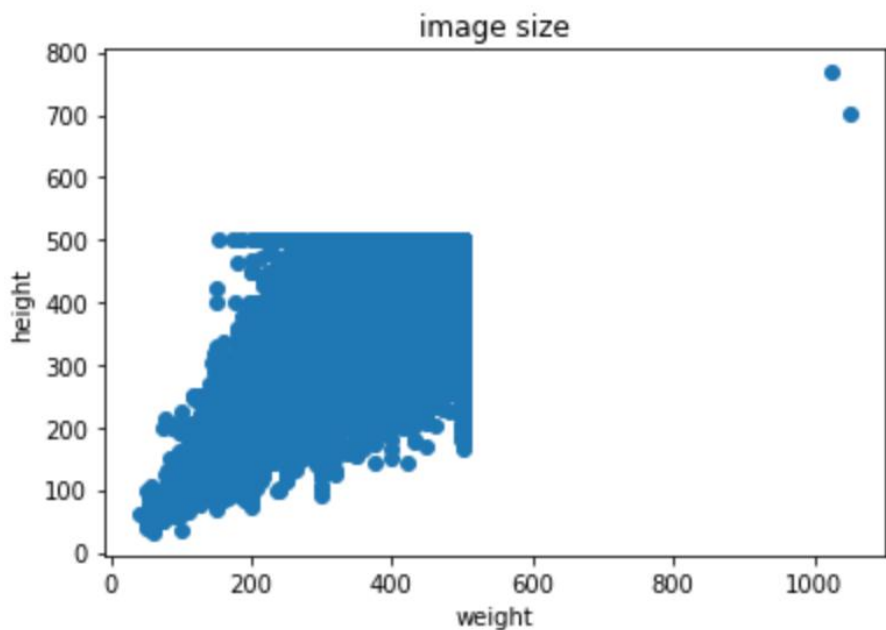
2.2 数据可视化

训练数据集有两个类别, 猫和狗, 标签类别在文件名中, 从文件名中可以看到买猫狗的分类名称。下面我们看下读取的猫狗图片:



图 2.1 训练集展示

训练集总共有 25000 张图片, 猫和狗各有 12500 张。猫狗类别数量相同, 所以不需要因为类别数量不均衡而在训练模型时候调整权重。测试集有 12500 张图片。注意到图片的尺寸不一致, 有大有小。我们抽取训练集 5000 张图片, 绘制一下图片尺寸散点图看下:



5000 张图片尺寸分布图

从图片尺寸分布散点图可以看到，图片的尺寸是有一定差异的，个别图片的尺寸明显和其它尺寸有很大不同，例如右图上那两个点。我们需要对输入图片的尺寸做一个统一的处理。另外我们需要对图片做归一化处理，其实后边章节中我们选用预训练的模型时候，预训练的模型都会对输入图片做预处理处理，其中就包含调整图片尺寸，归一化处理等。我们在具体实现方法中具体讨论。

我们接着再观察一组图片，这是我们从训练集中人工发现的一些异常图片，如下图所示：



部分异常图片

显然在训练数据集中存在一些异常数据，实际类别和标签不符合的图片，或者最右边这张一堆人抱着狗的照片，这很难判断为类别是狗，因为照片中占据主

主导地位的是人，针对这些情况，我们需要对数据训练集中的数据做一个清洗，我们试着从训练集中筛选出类似这样的异常图片，将它们从训练集中删除掉。具体方法将在具体实现方法章节中介绍

2.3 算法和技术

我们实现识别猫狗算法的技术是基于深度学习理论的，具体来说是使用深度学习的卷积神经网络来构造模型架构。而为了节省训练时间和效率我们将使用基于迁移学习的预训练模型来识别进行猫狗的分类。

2.3.1 深度学习

深度学习是基于机器学习的一个分支，它从数据中进行学习，从而提取出数据的规则或者映射。强调从层(layer)中来学习，随着层数的加深，将会学习到越来越有意义的规则。深度学习的“深”指的就是这个层的不断加深，模型中包含的层数就叫做模型的深度(depth)。

深度学习中的理论基础来自神经网络，神经网络的结构是逐层堆叠的。神经网络来自于生物学领域，很多文章说是模拟大脑的神经单元，传递信息的方式类似于神经元的树突。这是个误区，我们构造的基于神经网络的深度学习模型并不是仿照大脑的结构，而是通过学习数据形成的数学框架。

深度学习总体来说就是通过“层”对数据不断运算变换，最终实现输入到输出的映射。神经网络中每层对输入数据所做的具体操作保存在该层的权重中(weight)，也就是说输出结果是由权重来左右的。学习的意思就是求出最适合的权重参数，数据就是输入值，通过数据输入值来就得合适的权重值。

那么问题来了，到底什么才是合适的权重值呢。其实就是我们得到的输出和已知输入对应的标签之间的距离，换句话说就是求得的预测值和真实值之间的误差最小的时候，此时的权重就是我们想要的和合适的权重值。我们定义真实值和预测值之间的误差叫做损失，我们用损失函数来表示。求取最小损失值的方法就是深度学习的理论基础，叫做“反向传播”。通过反向传播算法来调整权重值，从而得到最小的损失值，这个过程由“优化器”来完成。这个深度学习的调节权重的过程如下图所示：

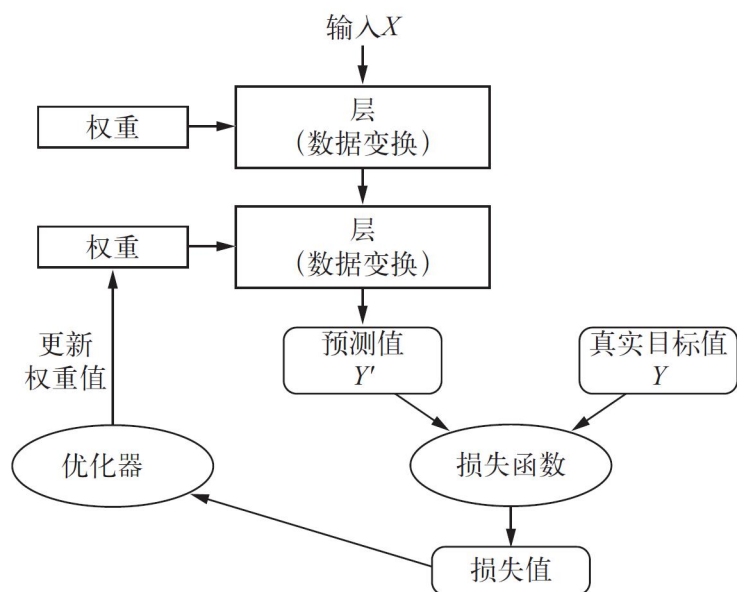


图 2.2 通过损失来调整权重

2.3.2 卷积神经网络

卷积神经网络(Convolutional Neural Networks)简称 CNN，是一类包含卷积计算且具有深度结构的前馈神经网络，是深度学习的代表算法之一。卷积神经网络具有表征学习能力，能够按其阶层结构对输入信息进行平移不变分类。因此也被称为“平移不变人工神经网络”。

卷积层和普通的密集层之间的根本区别在于，Dense 层从输入特征空间中学习到的是全局模式。而卷积层学到的是局部模式。卷积神经网络具有以下两个重要的性质：

(1) 平移不变性

卷积神经网络在图像某个区域学习到某个特征模式，那么在其它区域如果存在这个特征模式，它也会发现。而对于密集层网络来说，如果新的位置出现相同模式，它也必须要从头学习。这就使卷积神经网络可以更高效地使用数据。这也和我们现实中的视觉世界一致。打个比方，我们在美国认识了迪士尼乐园，那么到了中国也会认出迪士尼乐园的。我们并不需要在中国重新去认识迪士尼乐园。

(2) 空间层次结构

卷积层会从最开始的学习局部简单的特征模式，比如边缘检测，随着网络深度的加深，一层层的卷积层会学到越来越复杂得模式，这和我们视觉世界中认识事物从简单到复杂，从一般到特殊是吻合的。

2.3.3 卷积运算

卷积实质上是对两个实变函数的一种数学运算，下面给出离散型是的卷积计算公式如下：

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i - m, j - n)$$

其中 I 可以视为图片的输入,K 是核函数。我们列举一个二维卷积运算的例子来说明。在二维卷积层中，一个二维输入数组和一个二维核数组通过互相关运算最后输出一个二维数组。我们给出一个宽和高均为 3 的二维数组最为输入，核数组高和宽为 2. 这个和数组在卷积网络中被称之为过滤器。卷积核窗口的形状取决于卷积核的高和宽。我们具体操作下：

0	1	2		
3	4	5	0	1
6	7	8	2	3
输入数组			卷积核	

在二维卷积运算中，卷积窗口从输入数组的最左上方开始，按照从左至右，从上往下的顺序，依次在输入数组上滑动。当卷积窗口滑到某一位置上的时候，窗口中的输入子数组与核数组按照元素对应位置相乘并求和，得到输出数组中相应位置的元素。计算过程如下：

$$\begin{aligned} 0 * 0 + 1 * 1 + 3 * 2 + 4 * 3 &= 19 \\ 1 * 0 + 2 * 1 + 4 * 2 + 5 * 3 &= 25 \\ 3 * 0 + 4 * 1 + 6 * 2 + 7 * 3 &= 37 \\ 4 * 0 + 5 * 1 + 7 * 2 + 8 * 3 &= 43 \end{aligned}$$

最终得到输出数组如下：

19 25
37 43

该输出结果也叫作输出特征图。

2.3.4 优化器 Adam

我们选择 Adam 算法，而不是之前 kaggle 上经常使用的 RMSProp 算法。Adam 是从 2 个算法脱胎而来的：momentum 和 RMSProp，它集合了两个算法的主要优点，同时也做了自己的一些创新，主要有以下几点

- 计算高效，方便实现，内存使用也很少
- 更新步长和梯度大小无关
- 对目标函数没有平稳要求，即 loss function 可以随着时间变化
- 能较好的处理噪音样本，并且天然具有退火效果
- 能较好处理稀疏梯度，即梯度在很多 step 处都是 0 的情况

计算公式如下：

$$V_{dw} = \beta_1 V_{dw} + (1 - \beta_1) dW, V_{db} = \beta_1 V_{db} + (1 - \beta_1) db \quad \leftarrow \text{momentum}$$
$$S_{dw} = \beta_2 S_{dw} + (1 - \beta_2) (dw)^2, S_{db} = \beta_2 S_{db} + (1 - \beta_2) (db)^2 \quad \leftarrow \text{RMSprop}$$

$$V_{dw}^{correct} = \frac{V_{dw}}{(1 - \beta_1^t)}, V_{db}^{correct} = \frac{V_{db}}{(1 - \beta_1^t)}$$
$$S_{dw}^{correct} = \frac{S_{dw}}{(1 - \beta_2^t)}, S_{db}^{correct} = \frac{S_{db}}{(1 - \beta_2^t)}$$

$$W = W - \alpha \frac{V_{dw}^{correct}}{\sqrt{S_{dw}^{correct} + \epsilon}}, b = b - \alpha \frac{V_{db}^{correct}}{\sqrt{S_{db}^{correct} + \epsilon}}$$

Adam 计算公式

2.3.5 Dropout 丢弃法

我们在卷积层执行完毕之后，在使用 sigmoid 激活函数之前，会使用 dropout 方法，进一步抑制过拟合。Dropout 方法是神经网络最有效也最常用的

10

正则化方法之一。是深度学习巨头 Geoffrey Hinton(恭喜获得图灵奖)提出。对某一层使用 dropout，就是在训练过程中随机将该层的一些输出特征舍弃(设置为 0)。假如在训练过程中，某一层对给定输入样本的返回值应该是向量 $[0.12, 0.25, 0.35, 0.86, 0.01]$ 。使用 dropout 后，这个向量会有几个随机的元素变成 0，比如 $[0, 0.25, 0.35, 0, 0.01]$ 。dropout 比率(dropout rate)是被设为 0 的特征所占的比例，通常在 0.2 到 0.5 的范围内。测试时没有单元被舍弃，而该层的输出值是需要按 dropout 比率缩小，因为这时比训练时 有更多的单元被激活，需要加以平衡。

Dropout 方法的思路是在每个样本中随机删除不同的部分神经元，因此可以降低过拟合。核心思想是在层的输出值中引入噪声，打破不显著的偶然模式(Hinton 称之为阴谋)。如果没有噪声的话，网络将会记住这些偶然模式，从而导致过拟合。

2.3.6 VGG 模型介绍

VGG 模型是 2014 年 ILSVRC 竞赛的第二名，第一名是 GoogLeNet。但是 VGG 模型在多个迁移学习任务中的表现要优于 googLeNet。而且，从图像中提取 CNN 特征，VGG 模型是首选算法。VGG 的架构如下：

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

VGG 架构图

从上边架构图可以看到 VGG 是基于卷积网络的深度模型，我们使项目中使用 VGG16 的架构,VGG16 共包含 13 个卷积层,用 con3-xxx 表示,3 个全连接层，用 FC-xxx 表示，5 个池化层。其中，卷积层和全连接层具有权重系数，因此也被称为权重层，总数目为 13+3=16，这即是 VGG16 中 16 的来源。(池化层不涉及权重，因此不属于权重层，不被计数)。

VGG 的特点如下：

- 小卷积核。作者将卷积核全部替换为 3x3(极少用了 1x1)，小卷积核有助于捕获更多细节变化
- 小池化核。相比 AlexNet 的 3x3 的池化核，VGG 全部为 2x2 的池化核
- 层数更深特征图更宽。基于前两点外，由于卷积核专注于扩大通道数、池化专注于缩小宽和高，使得模型架构上更深更宽的同时，计算量的增加放缓

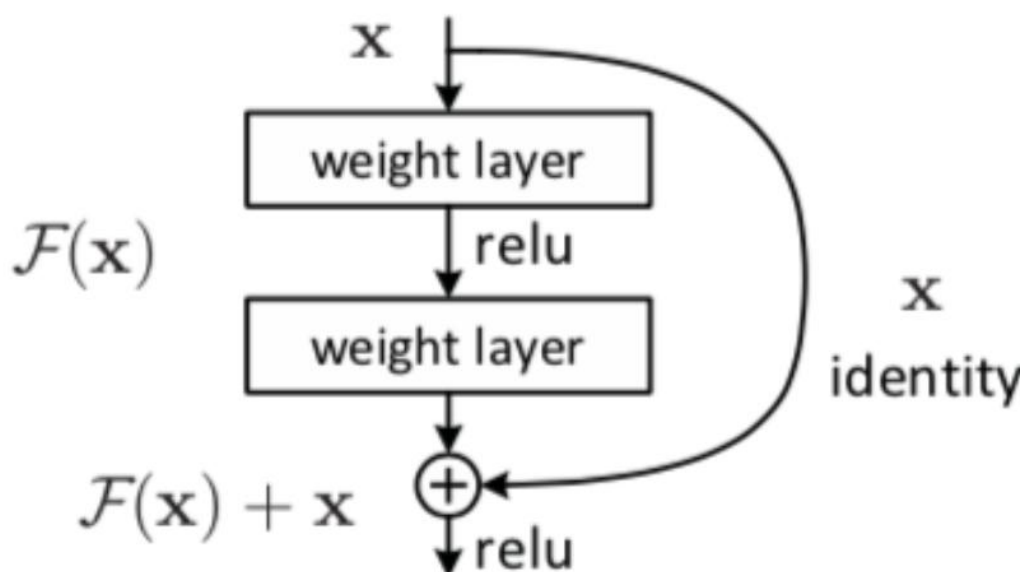
总结起来就是 VGG 采用更小的卷积核和池化层来堆叠更深的网络结构，也证实了模型的精度和网络的深度有很大关系，增大网络深度提高模型性能，这是 VGG 最大的贡献。为日后其它模型的构建奠定了基础。

2.3.7 ResNet 模型介绍

这里我们介绍下 ResNet，ResNet (Residual Neural Network) 由微软研究院的 Kaiming He 等四名华人提出，通过使用 ResNet Unit 成功训练出了 152 层的神经网络，并在 ILSVRC2015 比赛中取得冠军。ResNet 核心就是提出基于恒等映射理论的残差网络结构。

在 VGG 架构提出之后，我们知道通过加深网络的深度可以提高预测精度。但是随着深度的加深，发现在训练集上模型的表现却变差了，模型的性能开始退化，这不是过拟合导致的问题，因为如果是过拟合的话，应该在训练集上准确率提高才对。这反应一个问题，在较深的网络结构中模型并不是很好训练，主要原因还是深度的加深，在反向传播的时候，误差逐层递减，最终为 0，导致无法求导。因此提出了残差网络结构来解决这个深度导致的模型退化问题。

如果深层网络的后面那些层是恒等映射，那么模型就退化为一个浅层网络，即使深度再大，至少效果也不会低于浅层网络的预测效果。也就是说如果我们在深度网络中能够构造恒等映射，那么至少模型不会因为深度的加深导致退化。那当前要解决的就是学习恒等映射函数了。但是直接让一些层去拟合一个潜在的恒等映射函数 $H(x) = x$ ，比较困难，这可能就是深层网络难以训练的原因。但是，如果把网络设计为 $H(x) = F(x) + x$ ，如下图所示，我们可以转换为学习一个残差函数 $F(x) = H(x) - x$ 。只要 $F(x) = 0$ ，就构成了一个恒等映射 $H(x) = x$ 。而且，拟合残差肯定更加容易。



残差模块结构：

通过这种残差结构，相当于简单执行了同等映射，不会产生额外的参数，也不会增加计算复杂度。ImageNet 上的实验证明了作者提出的加深的残差网络能够比简单的叠加层所生产的深度网络更容易优化，而且，因为深度的增加，结果得到了明显提升。残差网络的提出虽然没有从根本上解决深度导致的梯度消失，但是却在一定程度上避免了因为深度的叠加导致的梯度消失，从而缓解了退化问题。残差的理念极大推动了深度学习模型架构的设计，后续的 Inception 和 Xception 上启用了残差块。

2.3.8 Inception 系列模型介绍

这里我们介绍下 Inception 家族的模型。Inception 网络是 CNN 分类器发展史上一个重要的里程碑。在 Inception 出现之前，大部分流行 CNN 仅仅是把卷积层堆叠得越来越多，使网络越来越深，以此希望能够得到更好的性能。

VGG 模型的泛化性能非常好，常用于图像特征的抽取目标检测候选框生成等。但是 VGG 最大的问题就在于参数数量。在 Inception 架构提出之前，卷积神经网络都是靠提高模型的宽度和深度来完提升模型的性能。这种方式会导致以下的缺点：

- 会导致更大的参数空间，更容易过拟合
- 需要更多的计算资源
- 网络越深，梯度容易消失，优化困难

Inception 从网络架构设计入手，其目的是在提高计算资源的利用率算量不变的情况下，提升模型的训练效率和精度。Inception 思想通过设计一种具有优良局部拓扑结构的网络，即对输入图像并行地执行多个卷积运算或池化操作，并将所有输出结果拼接为一个非常深的特征图。因为 1×1 、 3×3 或 5×5 等不同的卷积运算与池化操作可以获得输入图像的不同信息，并行处理这些运算并结合所有结果将获得更好的图像表征。

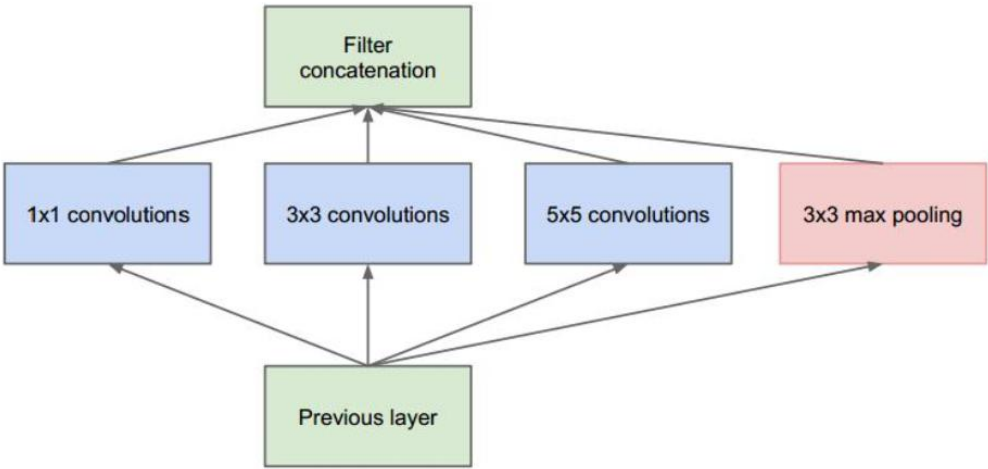
Inception 是一个非常复杂的网络，它使用很多技巧来提升模型的训练速度和准确率。它的不断改进也带来了多个 Inception 版本。常见的版本主要有 InceptionV1, InceptionV2, InceptionV3, InceptionV4。

InceptionV1:

InceptionV1 出现之前，图像识别中存在一些问题。例如要识别图片中的猫，有的图片中猫占得比例很大，有的图片中猫所占得比例很小。这就为卷积操作选择合适的卷积核带来了困难。猫在图片中占得比例较大，信息分布也更具有全局性，就需要较大的卷积核，而猫在图片中占得比例较小，信息分布就比较局部，那就更适合小的卷积核。非常深的网络更容易过拟合。将梯度更新传输到整个网络是很困难的。简单地堆叠较大的卷积层非常消耗计算资源。

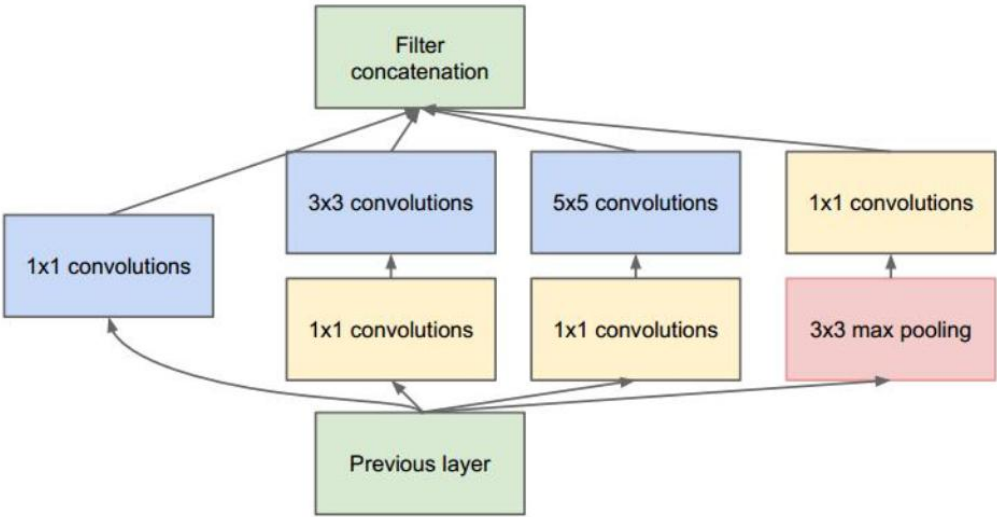
针对这些问题，我们想到为什么不在同一层级上运行具备多个尺寸的滤波器呢？网络本质上会变得稍微「宽一些」，而不是「更深」。作者因此设计了 Inception 模块。

下图是「原始」Inception 模块。它使用 3 个不同大小的滤波器(1×1 、 3×3 、 5×5) 对输入执行卷积操作，此外它还会执行最大池化。所有子层的输出最后会被级联起来，并传送至下一个 Inception 模块。



Inception 模块原始架构

卷积神经网络需要耗费大量的计算资源，为了降低算力成本，作者在卷积层之前添加了额外的卷积层，来限制输入信道的数量。尽管添加额外的卷积操作似乎是反直觉的，但是卷积比卷积要廉价很多，而且输入信道数量减少也有利于降低计算成本。需要注意的是，卷积在最大池化层之后，而不是之前。添加了这些额外的卷积之后，就构成了可实现降维操作的 Inception 模块，如下图所示



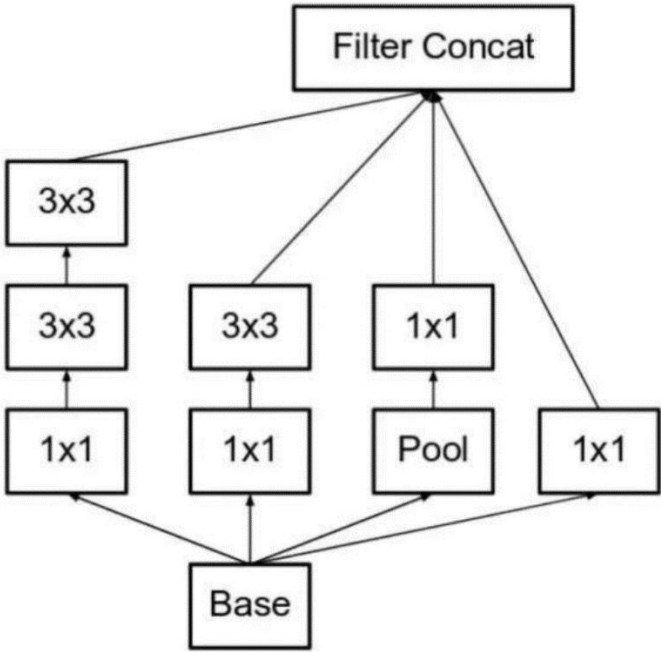
可降维的 Inception 模块

InceptionV2

InceptionV2 在之前的基础上继续探索网络架构，通过分解卷积来提高效率。当卷积不会大幅度改变输入维度的时候，神经网络可能会表现得更好，

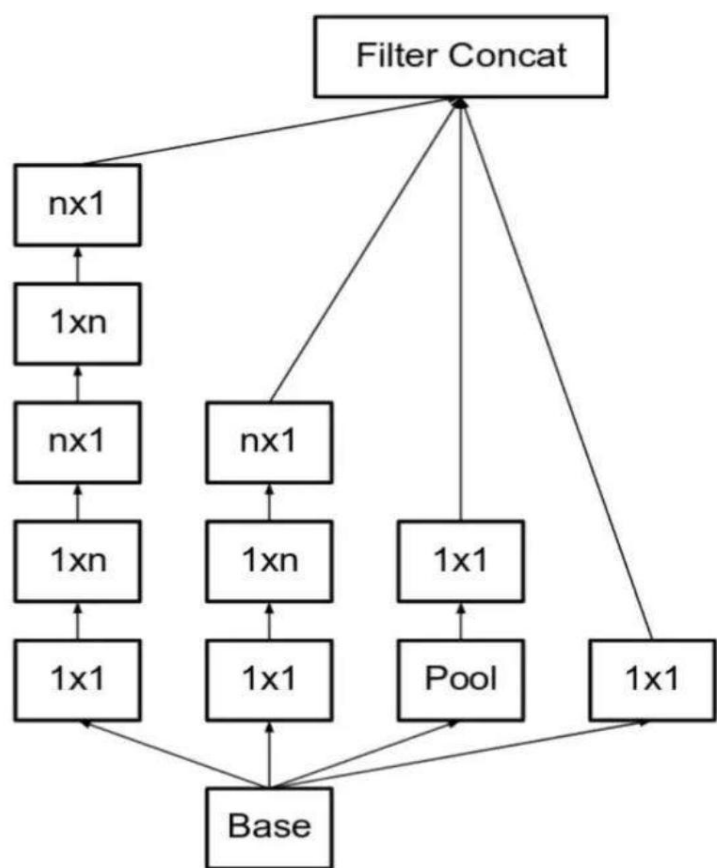
过多的减少维度会造成信息的丢失。InceptionV2 通过巧妙的因子分解方法，来提高计算的效率。

将 5×5 的卷积分解为两个 3×3 的卷积运算以提升计算速度。尽管这有点违反直觉，但一个 5×5 的卷积在计算成本上是一个 3×3 卷积的 2.8 倍。所以叠加两个 3×3 卷积实际上在性能上会有所提升，如下图所示：

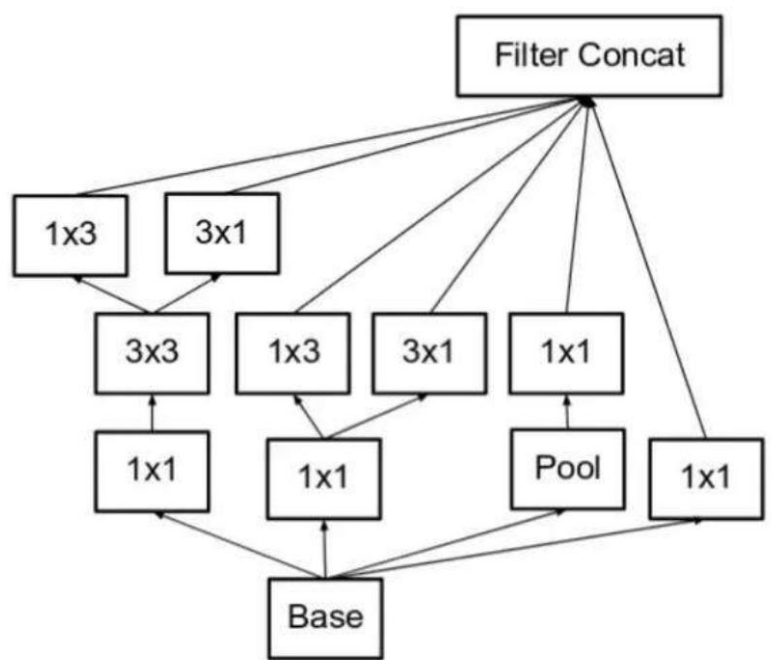


前一版 Inception 中的 5×5 的卷积变成了两个 3×3 卷积的堆叠

此外，作者将 $n \times n$ 的卷积核尺寸分解为 $1 \times n$ 和 $n \times 1$ 两个卷积。例如，一个 3×3 的卷积等价于首先执行一个 1×3 的卷积再执行一个 3×1 的卷积。他们还发现这种方法在成本上要比单个 3×3 的卷积降低 33%，这一结构如下图所示：



模块中的滤波器组被扩展（即变得更宽而不是更深），以解决表征性瓶颈。如果该模块没有被拓展宽度，而是变得更深，那么维度会过多减少，造成信息损失。如下图所示：



InceptionV3

InceptionV3 和 V2 的架构差不多，只是增加了下面几个改进：

- (1) 使用 RMSProp 优化器
- (2) 分解 7x7 卷积
- (3) 辅助分类器使用 BatchNorm
- (4) 标签平滑 (添加到损失公式的一种正则化项, 旨在阻止网络对某一类别过分自信, 即阻止过拟合)

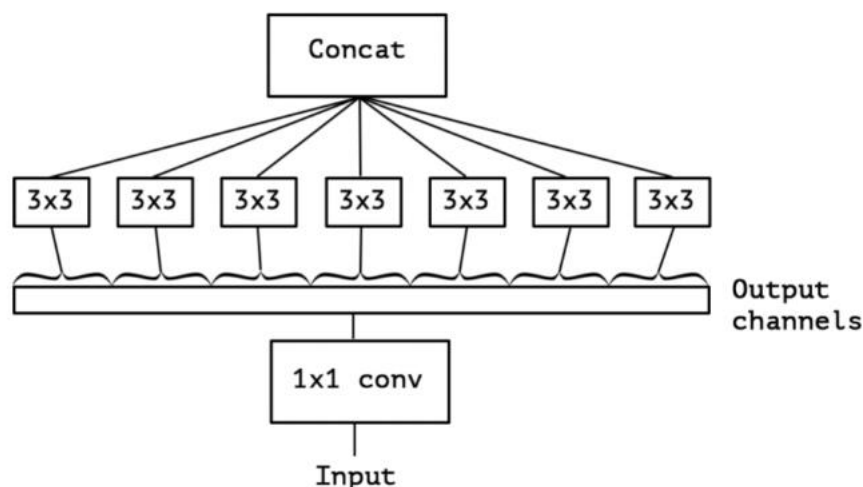
InceptionV4

在该论文中, 研究者介绍道, Inception 架构可以用很低的计算成本达到很高的性能。而在传统的网络架构中引入残差连接曾在 2015ILSVRC 挑战赛中获得当前最佳结果, 其结果和 Inception-v3 网络当时的最新版本相近。这使得人们好奇, 如果将 Inception 架构和残差连接结合起来会是什么效果。在这篇论文中, 研究者通过实验明确地证实了, 结合残差连接可以显著加速 Inception 的训练。也有一些证据表明残差 Inception 网络在相近的成本下略微超过没有残差连接的 Inception 网络。研究者还展示了多种新型残差和非残差 Inception 网络的简化架构。这些变体显著提高了在 ILSVRC2012 分类任务挑战赛上的单帧识别性能。作者进一步展示了适当的缩放激活值大小如何稳定非常宽的残差 Inception 网络的训练过程。通过三个残差和一个 Inception v4 的模型集成, 作者在 ImageNet 分类挑战赛的测试集上取得了 3.08% 的 top-5 误差率。

Inception 的思想就是把全连接改成稀疏连接, 卷积层也是稀疏连接, 但是不对称的稀疏数据数值计算效率低下, 因为硬件全是针对密集矩阵优化的, 所以, 我们要找到卷积网络可以近似的最优局部稀疏结构, 并且该结构下可以用现有的密度矩阵计算硬件实现, 产生的结果就是 Inception

2.3.9 Xception 模型介绍

Xception 主要是对 InceptionV3 的改进。主要思想是将通道之间的相关性和空间之间的相关性独立分开处理。Xception 提出深度可分离卷积 (Separable Convolution) 来替代原 InceptionV3 的卷积操作。Xception 可以看作是一种极限方式的 Inception。极限的 Inception 的模块架构如下图所示:



极限的 Inception

从图中可以看到先是对输入数据做普通的 1×1 卷积操作，然后再对每个 1×1 卷积结果的每个 channel 做 3×3 卷积操作，最后将结果合并。

Xception 提出的深度可分离卷积和这种极限的 Inception 是一个思想，只是执行顺序相反，先进行 3×3 卷积，再进行 1×1 卷积。论文的作者认为这个顺序没有什么影响。

2.3.10 集成学习-组合模型策略

集成学习（ensemble learning）是机器学习中一类学习算法，通过训练多个学习器并将它们组合起来使用的方法。这类算法通常在实践中会取得比单个学习器更好的预测结果。而组合模型策略就是来自于集成学习方法。通过多个模型的组合起到单一模型达不到的效果。

组合模型的集成策略一般采用多层特征融合的方式。多层特征融合是针对单模型的一种模型层面的集成方法。由于深度卷积神经网络特征具有层次性的特点，不同层的特征富含的语义信息可以相互补充，在图像语义分割，细粒度图像检索，基于视频的表象性格分析等任务中常见的多层特征融合策略的使用。一般的，多层特征融合操作时可将不同层的网络特征级联。而对于特征融合应选取哪些网络层，一个实践经验是：最好使用靠近目标函数的几层卷积特征。因为越深层特征包含的高层语义越强，分辨力也会越强。相反，网络较浅层的特征较普遍使用，用于特征融合可能起不到作用，或者甚至会起到相反作用。

本项目我们使用不同的模型来提取特征，最后将这些不同模型提取的特征组合起来，起到增加样本特征数的效果，这样就会提供更多的信息给我们训练的模型参考，增强模型的泛化能力。

2.3.11 技术实现框架

我们使用 Keras 框架来完成整个算法，Keras 是一个 Python 深度学习框架，可以方便地定义和训练几乎所有类型的深度学习模型。Keras 为深度学习模型提供了高层次的构建模块，它并不处理张量操作和求微分等底层运算。而是通过在后台使用一个高度优化的张量库来上实现这些运算。这个张量库就是 Keras 的后台引擎，这个后台引擎可以使用使用 Theano，CNTK 和 tensorflow 这三个框架。本项目我们使用 tensorflow 作为后台引擎。由于训练模型涉及大规模矩阵运算，我们使用基于 gpu 进行计算。我们的开发项目使用的显卡是 gtx1070。因此我们需要安装 CUDA 和 cuDNN。下面列出我们开发项目使用的技术列表：

- Keras
- Tensorflow
- Stick-learn
- Opencv
- Numpy
- Pandas
- Matplotlib

2.4 基准模型

我们模型采取的是二元交叉熵对数损失函数来计算损失的大小，项目要求损失得分必须进入 kaggle 排名的百分之十，kaggle 猫狗识别的排名榜总共有 1314 个名次，所以如果要进入前百分之十的名次。则至少要排到第 131 名。下面我们看下 kaggle 排名进入百分之十前后的损失得分情况，以下截图来自 Kaggle 网猫狗识别大赛的公开排名页面。














119	Pim Nijdam		0.05994	6
120	mhebi		0.05999	5
121	ValentinBrasso		0.06006	7
122	EvenOldridge		0.06009	42
123	TimSmole		0.06024	9
124	Chase		0.06026	6
125	John Vial		0.06037	18
126	tmuzap		0.06054	21
127	icodingc		0.06068	12
128	Wilson Sun		0.06082	23
129	Jeremy Howard		0.06086	11
130	RaviKiranK		0.06114	35
131	Reziproke		0.06127	4

图 2.3 Kaggle 猫狗识别排名

从图中可以看到 131 名的损失分数是 0.06127，所以我们的模型得分不能低于这个值。我们将模型损失得分基准控制在至少 0.06。

3 具体算法实现

3.1 数据清洗

我们在上一章节中提到训练数据集中有一些异常图片，例如标签和实际类别不同的图片，或者图片中主次不分的，比如一群人抱着一只很小的猫或者狗，这种图片，猫狗并不是图片的主要成分，这些就很难判断为猫或者狗。针对这些异常情况，我们需要对数据做出清洗，这样有助于我们更好的训练模型。

训练集有 25000 张图片，我们用眼睛观察这种人工方式是没效率且不明智的。我们需要采用一点技巧来帮助我们。这个方法就是使用已经预训练的模型。这些模型是经过实践检验的优良模型，已经调整好权重等参数了。可以很好地帮我们完成任务。我们使用基于 ImageN 数据集训练的模型，VGG, ResNet, Inception 和 Xception 都有基于 ImageNet 训练的版本，我们可以从网上下载。ImageNet 是要对 1000 个物品进行分类，当然这个分类不是物种的分类，举例：狗，是物种的分类，但是具体是哪种狗，比如泰迪，比格等，ImageNet 数据集则给出了具体的品种。ImageNet 总共对 118 个物种做了详细的细分。

ImageNet 分类是具体的某个动物的品种，比如如果是狗，他会给出狗的具体品种，例如拉布拉多，金毛等。所以我们需要判断出使用 ImageNet 数据的预训练模型给出的具体类别，到底是猫还是狗，或者非猫非狗。我们需要做一个映射，将预训练模型给出的具体的狗或者猫的品种，我们映射到仅仅是猫或者是狗就足够了。

基于 ImageNet 数据集训练给出预测结果格式包含具体物种 id 号，具体物种名称，以及是该品种的概率。我们做了个统计，将属于猫 id 号放在一起，将属于狗的 id 号集中到一起。我们将整理好的 id 和物种的集合如下所示：

```
Cats = [
    'n02123045', 'n02123159', 'n02123394', 'n02123597'
    , 'n02124075', 'n02125311', 'n02127052']
```

将属于猫 id 集中到一起，狗的结构与之类似。通过这两个列表，我们就可以做到从拉布拉多到狗这种关系的映射了。我们选取 Xception 作为我们的异常图片筛选器。选取理由是 Xception 在 ImageNet 上训练效果是最佳的，官网给出的各模型评估列表如下：

3.2 数据预处理

这里我们将下载的数据解压, 数据分成 `tain` 和 `test` 两个文件夹, 文件的目录结构大概是这样:

Train	Test
Cat. 0. jpg	0. jpg
Cat. 1. jpg	1. jpg
Dog. 13000. jpg	10. jpg
Dog. 13010. jpg	100. jpg

文件命名格式为 `class.id.format`, 例如 `dog. 10. jpg`, `cat. 12. jpg`, 并且训练集和测试集数据直接存放在 `train` 和 `test` 下, 这种方式不利于我们后边使用 `keras` 的 `ImageDataGenerator` 做数据增强, 因为 `flow_from_directory` 方法需要将不同种类的图片分类在不同的文件夹中, 然后数据集本身还没有验证集. 所以我们将 `train` 下文件夹下的数据拆分为训练集和测试集, 并且我们将每个图片按照类别存放, 训练集的狗放到 `image_data/train/1` 下, 猫放到 `image_data/train/0` 下, 同理验证集里狗和猫存放分别是 `image_data/valid/1` 和 `image_data/valid/0`. 我们从原始训练数据中随机抽取 20% 作为验证集使用。

为了适合模型的输入, 我们需要对图片做归一化处理。我们将在下一部分的数据调整中做详细说明。

我们对训练集数据使用采用数据增强, 数据增强是从现有的训练样本中通过改变图像的显示状态来增加数据样本, 丰富图像的多样化。从而使模型可以观察到更多的数据。因此使用数据增强可以提升模型的泛化能力, 从而可以抑制过拟合。我们在编写代码发现适当的数据增强可以降低损失。

3.3 迁移学习

我们使用迁移学习的方式来训练模型, 迁移学习是一种使用预训练网络的高效方法。它基于一个已经训练完毕的网络模型, 这个模型通常在大规模图像分类数据上已经训练完成。使用这个预训练网络学到的特征结构可以适用于不同的分类问题, 即使这些新问题涉及的任务完全不同。如果从头开始训练一个卷积神经网络, 那么需要大量的训练时间并且需要反复调整超参数, 并且准确度也不

会保证高于目前公开发布的模型架构，因此采用迁移学习的方式是最适合的，可以保证准确度的情况下，极大缩小训练时间。

3.3.1 初始模型选择

既然已经确定使用迁移学习的方式来训练模型，那么就行选择哪个模型呢。这个需要我们做出选择，我们使用的策略是选取几个比较适合该分类问题的模型，然后进行对比，看下各个模型在猫狗数据集上的训练测试效果来决定下一步的策略。我们选取 Keras 提供的 VGG16, ResNet, InceptionV3 和 Xception 这四个预训练模型。这四个也是非常经典的深度学习模型。非常适合我们这个项目任务。

3.3.2 输入数据和调整

我们确定了选出的模型之后，需要对输入数据进行调整，以用来匹配这些预训练模型的输入数据格式。VGG16 和 Resnet50 使用 224x224 的分辨率，InceptionV3 和 Xception 使用 299x299 的分辨率。我们需要对原始数据做出 Resize 操作。

因为我们使用预训练模型做迁移学习，需要将输入数据进行处理好匹配预处理模型需要的输入格式。我们首先要调整图片尺寸，Xception 和 InceptionV3 需要的图片尺寸为 (299, 299)，并且将图片像素从 (0, 255) 缩放到 (-1, 1) 区间内，而 VGG16 和 ResNet50 需要的是 (224, 224)，并且对所有图像的所有像都减去像素均值 [103.939, 116.779, 123.68] (根据所有的 ImageNet 图像算出)，并将通道顺序从 RGB 调整为 BGR。Keras 中的每个模型对应 Preprocess 函数帮助我们实现了这种归一化相关预处理。

我们使用的四个预训练模型是只包含卷积网络部分，不包含 Flatten 和 Dense 层，仅用作特征提取，因此在输出的时候我们要对所有模型加上一个 GlobalAveragePooling2D 层来进行输出格式调整。GlobalAveragePooling2D 相比 Flatten，可以大量减少模型参数，降低计算成本。

3.3.3 模型的训练

我们使用数据增强进一步降低过拟合，水平和垂直的偏移系数为 0.2，水平翻转，随机缩放 0.2，剪切变换角度 0.2。图片旋转 30 度。

优化器使用 Adam，dropou 系数为 0.5，采用批量随机梯度下降。Ecoph=15。根据训练日志绘制各个模型损失折线图如下：

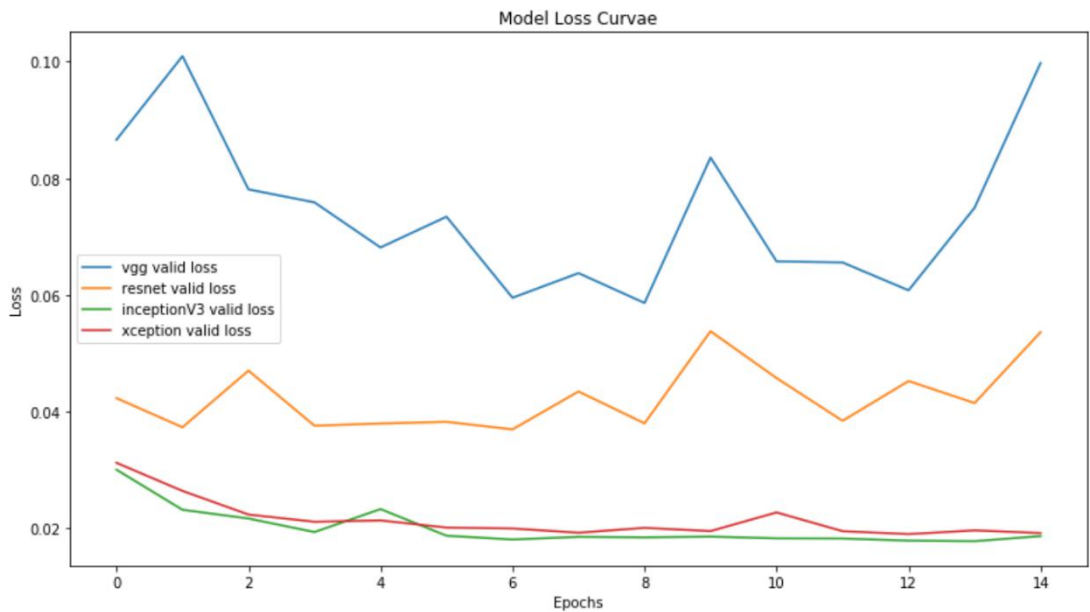


图 3.1 损失曲线图

3.3.4 评估结果

我们从验证损失曲线中可以看到，基于 InceptionV3 和 Xception 的模型效果明显比基于 vgg16 和 resnet50 的模型好很多，在 kaggle 上面的得分情况也可以看到，采用 vgg 和 resnet 做迁移学习，得分情况比较落后，都无法进入 10% 的名次，尤其是 vgg16 排名非常靠后。而 InceptionV3 和 Xception 得分情况则非常好，都轻松排进 10% 的名次，InceptionV3 可以排在 34 名，而 Xception 可以排在 22 名。

3.4 算法改进

之前我们都是基于单一模型使用迁移学习，这些模型虽然也取得了不错的效果，但是由于这些网络的网络架构又有很大的差异，因此图像特征的提取也必然存在差异。于是我们想到如果将这些模组合起来，那就是使用多个不同模型提取特征，在将这些特征合并。既然一种模型无法达到理想的效果，我们可以使用多

个模型一起努力，人多力量大的原则来解决问题,这种方式增加了特征数量，有助于降低过拟合，给模型更多的信息。下面看下我们构造的模型的架构：

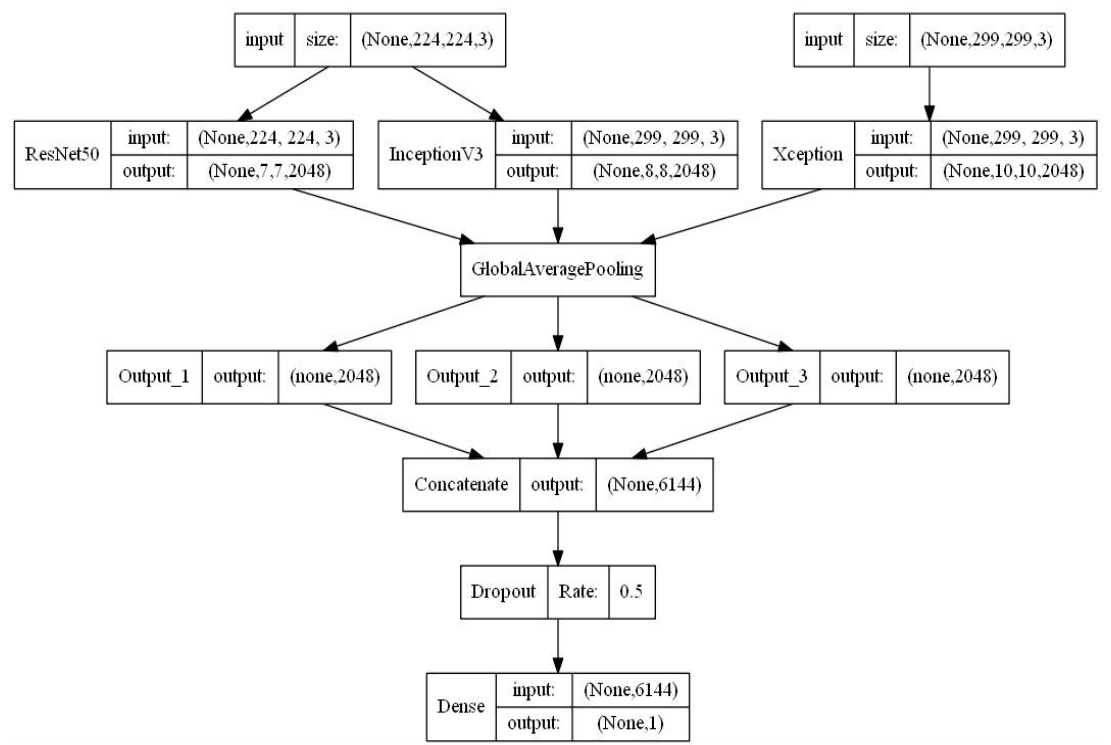


图 3.2 组合模型架构

我们采用分类效果组好的 resnet50，inceptionV3 和 Xception 这是模型进行特征组合。下面我们使用这种组合模型和单模型分类最好的 Xception 做一下对比。我们分别绘制 Epochs = 15 和 Epochs = 8 的验证损失曲线如下：

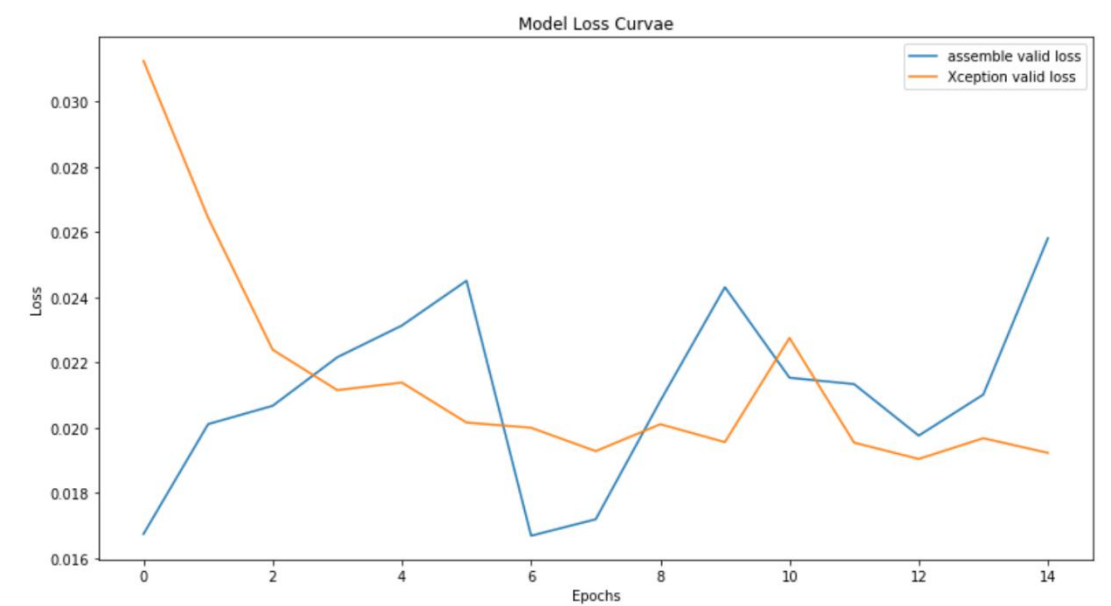


图 3.3 Epochs=15 损失曲线图

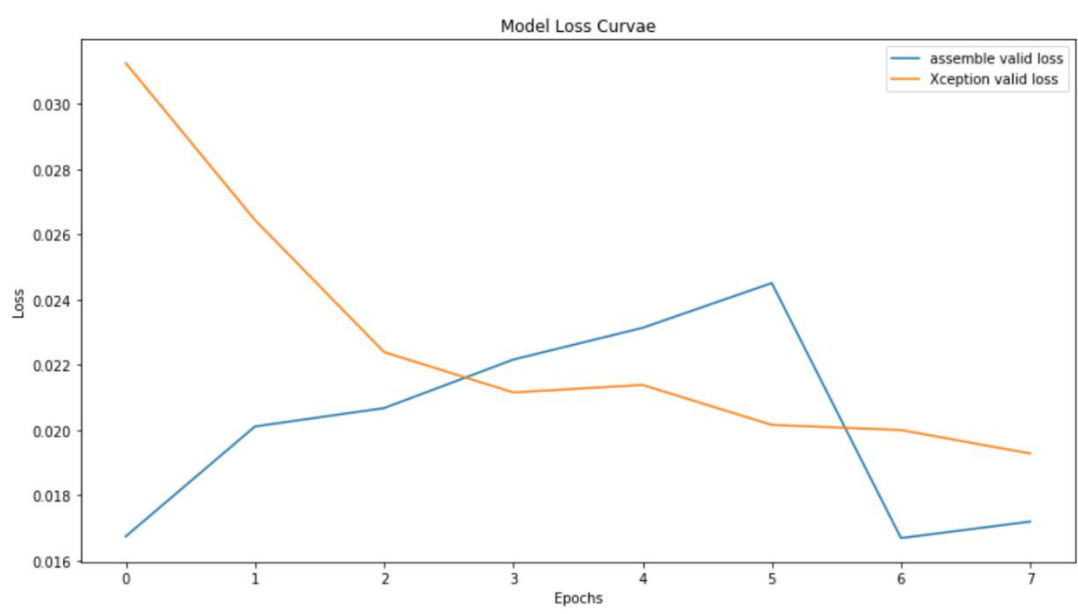


图 3.4 Epochs=8 损失曲线图

从图中可以看出组合模型的方式相比单一的 Xception，使用更少的训练时间就能达到比较理想的效果，并且损失得分也更好。在 kaggle 上组合模型得分为 0.04101，排名 18。优于 Xception 的 0.04273，排名 22。

3.5 算法进一步改进

我们之前使用组合模型的方式来进行特征的融合,这种方式取得了一定效果，比单一使用一种模型效果要好,但是我们也看到了模型之间架构存在差异，导致各个模型的能力大小不同,xception 就强于 inceptionv3,而 inceptionv3 也优于 resnet。而我们之前那种组合特征的方式,是基于模型能力基于均衡的方式来进行特征融合的，这就导致了模型特征不平衡的问题,因为能力强的模型应该占据更好的地位，能力较弱的应该处于较低的地位。所以我们想到给不同模型设置不同的特征权重比例，性能强的模型占据高权重,反之性能弱的比重占据小一些。权重比例分配方式如下：

$$0.6 * \text{Xception} + 0.1 * \text{ResNet50} + 0.3 * \text{InceptionV3}$$

以这种方式进行训练，我们对比下这种按照权重比例分配的方式和普通的组合模型，看看谁的效果更理想一些。绘制损失曲线如下：

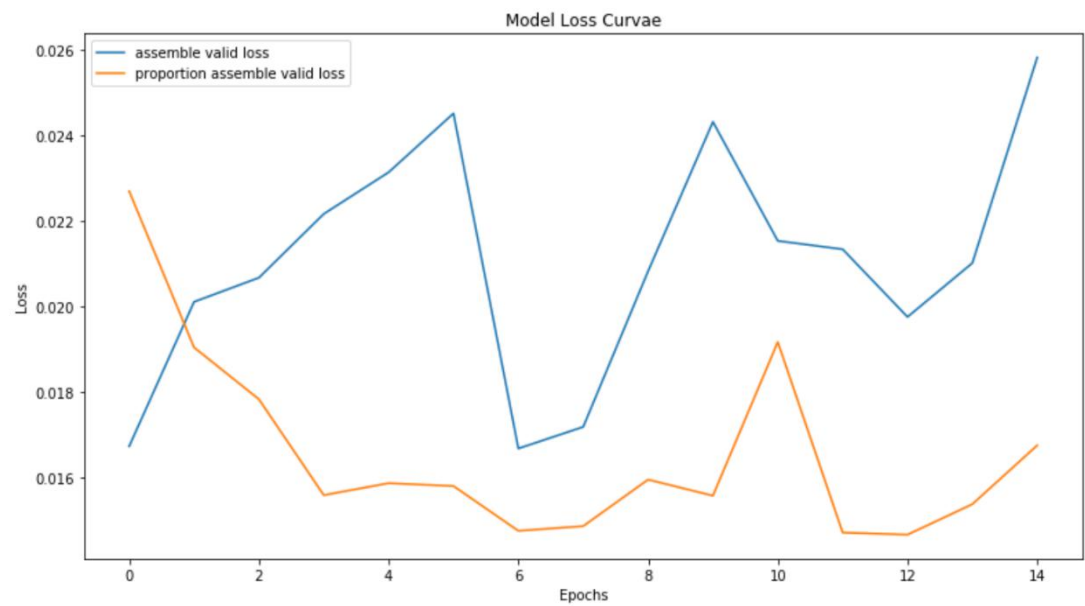


图 3.5 损失曲线图

图中黄色曲线是按照权重比例方式的组合模型，蓝色曲线是普通的组合模型。可以看到我们采取的权重比例方式还是要优于普通组合模型的。而在 kaggle 上权重比例方式分配的模型得分为 0.03939，排名 15，高于普通组合模型的 18 的排名。

4 结果

4.1 模型的评价与验证

4.1.1 单一模型选择

我们选用了 VGG16, ResNet50, InceptionV3, 和 Xception 这四个模型, 经过测试发现 Xception 是最优的, 因为 Xception 集成了其余三个模型所有的优点, VGG 的提出的通过多卷积核和池化堆叠加深网络深度, Resnet 的残差网络概念, Inception 系列提出的多尺寸卷积和多个小卷积核替代大卷积核思想提升了模型的训练效率。Xception 集成这些优点, 并在此基础上提出了深度可分离卷积 (DepthwiseSeparableConv), 将通道和空间分开进行映射, 可以进一步提升模型的性能。

4.1.2 组合模型策略

我们的思路是将多个模型提取的不同形式的特征进行合并, 这样可以起到增加特征数目, 将不同的优点集成的效果。我们在实验过程中抽取了三个最好的模型, ResNet50, InceptionV3 和 Xception 而使用 InceptionV3 和 Xception 的组合, 发现不如三个模型组合的效果, 这也反映出多模型起到互补的作用, 进一步降低过拟合。

4.1.3 权重比例分配策略

我们在提出组合模型进行特征提取的方式取得了一定的进步, 考虑到模型之间架构存在差异, 导致各个模型的能力大小不同, xception 就强于 inceptionv3, 而 inceptionv3 也优于 resnet。所以我们想到给不同模型设置不同的特征权重比例, 性能强的模型占据高权重, 反之性能弱的比重占据小一些。这种方式取得了一定的进步。

4.2 实验结果分析

4.2.1 单一模型实验结果

我们从验证损失曲线中可以看到, 基于 InceptionV3 和 Xception 的模型效果明显比基于 VGG16 和 ResNet50 的模型好很多, 在 kaggle 上面的得分情况也可以看到, 采用 vgg 和 resnet 做迁移学习, 得分情况比较落后, 都无法进入 10% 的名次, 尤其是 vgg 排名非常靠后。而 InceptionV3 和 Xception 得分情况则非常好, 都轻松排进 10% 的名次, InceptionV3 可以排在 34 名, 而 Xception 可以排在 22 名, 因此基于 xception 的迁移学习方式是优秀的。

4.2.2 组合模型的实验结果

输出日志来看, 组合模型验证损失最优值为 0.0167, 在第二次迭代就达到, 很快就收敛到最优, 而基于 Xception 的预训练模型最优损失在 0.0193, 在第 1 次迭代达到该最优值。组合模型的损失相对于 Xception 降低了 12.47%, 并且可以更早训练到最优解。从损失曲线图可以明显看到, 组合模型在迭代次数更少的情况下达到比较理想的效果, 基于 Xception 的预训练模型在迭代次数逐渐增加下也损失也开始越来越小, 但在训练时间和损失值上都比不上组合模型, 这也验证了我们的想法, 可以通过将一些优秀的模型组合起来提升训练效果。最后, 我们在 Kaggle 上看到组合模型的得分更加优秀, 排名 18。

4.2.3 权重比例分配方案的实验结果

我们通过设置权重比例, 从日志中看到, 不设置权重比例的最佳损失为 0.0167, 带权重比例的损失为 0.01467, 提升了约 12.15%。我们看下两种不同设计的损失曲线, 从图中可以明显看到带比例的方式要优于不带比例的方式。将结果提交到 Kaggle 中, 看到带比例组合模型得分为 0.03939, 而不带比例方式的组合模型得分为 0.04101, 提升了大约 4 个百分点。从 kaggle 排名看, 带比例的组合模型排名为 15, 优于不带比例的组合模型的排名 18。

5 项目结论

我们完成了猫狗识别算法和设计，在整个项目研究过程中，我们首先是熟悉数据集结构，并将数据集按照训练集，验证集，和测试集按照类别重新划分。然后使用数据增强方式扩大训练数据，进一步降低过拟合。然后我们采取迁移学习的方式使用预训练模型，为了确定哪些模型效果更好，我们对各单一模型进行对比，确定了各个模型在猫狗分类问题的表现水平。

之后，我们为了进一步提高模型的水平，在单一模型能力有限的情况下，提出将多模型组合进行特征合并的方法来进一步抑制过拟合，提高模型的泛化能力。实验证明这种组合模型的方式确实起到了一定的效果。

最后我们考虑到个模型的架构差异和能力的不同，又提出了基于权重分配方式的组合模型策略，实验证明这种方式比普通的组合模型方式要更加有效，损失得分也更好。

我们使用最终训练好的模型对测试集的图片进行了预测，我们随机抽取几个例子展示如下图所示：



可以看到我们的模型可以正常使用，预测的几个测样样本也给出了非常肯定的答案。通关不断改进我们训练出了一个可以在现实生活中使用的识别算法。

后期如何进一步改进模型，可以考虑增加训练数据，让模型获得更多的学习数据，进一步提高模型的泛化能力。或这使用更加优秀的模型来进行特征组合。也可以使用更好的集成方法或者学习率递减等手段提高性能。个人觉得进对图像处理技术将会起到很大的作用，通过图像处理技术对图像做进一步的预处理改进，会让网络模型更好学习图片特征，比如针对一些模糊程度很高的图片做一些处理，让图片更加清晰。鉴于本人目前图像处理知识的缺乏，在本文没有过多图像处理方面的讨论，这留作后期研究了。

总提来说，猫狗识别项目目的在于使用深度学习的方法，通过卷积神经网络来进行图像分类。事实证明使用深度学习的方法，在处理计算机视觉问题是有一定优势的。

参考文献

- [1] K. Simonyan and A. Zisserman. Very deepconvolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.
- [2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition. arXiv:1512.03385
- [3] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of The 32nd International Conference on Machine Learning, pages 448 – 456, 2015.
- [4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1 – 9, 2015
- [5] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. arXiv:1512.00567
- [6] François Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. arXiv:1610.02357
- [7] Francois Chollet. Deep learning with Python
- [8] Ian Goodfellow, Yoshua Bengio, Aaron Courville. Deep Learning