

深度学习

猫狗大战

Chen Jiangdong
leechan.rv@hotmail.com
2018.5.24

目录

1 问题定义.....	1
1.1 项目概述.....	1
1.2 问题陈述.....	1
1.3 评价指标.....	1
1.4 机器学习流程.....	1
2 分析.....	2
2.1 数据的探索.....	2
2.2 算法和技术.....	7
2.2.1 深度学习.....	7
2.2.2 迁移学习(Transfer learning).....	8
2.2.3 深度学习框架 Keras 和 GPU 计算.....	8
2.3 基准模型.....	8
3 方法.....	14
3.1 数据预处理.....	14
3.2 执行过程.....	15
3.2.1 流程(Pipeline).....	15
3.2.2 模型的训练.....	17
3.2.3 训练结果生成.....	17
3.3 完善.....	19
4 结果.....	20
4.1 模型的评价与验证.....	20
4.2 合理性分析.....	21
5 项目结论.....	21
5.1 结果可视化.....	21
5.2 对项目的思考.....	23
5.3 需要做出的改进.....	23
参考文献.....	24

1 问题定义

1.1 项目概述

项目属于计算机视觉领域中的图像分类问题。图像分类的过程非常明确^[1]: 给定已经标记的数据集，提取特征，训练得到分类器。在图像分类领域，比较著名的数据集有：MNIST、CIFAR-10、ImageNet 和 MS COCO。项目使用 Kaggle 竞赛^[2]提供的 Dogs vs. Cats 数据集，任务是对给定的猫和狗的图片进行分类，因此是二分类问题（Binary Classification）。

1.2 问题陈述

项目要解决的问题是使用 12500 张猫和 12500 张狗的图片作为测试集，训练一个合适的模型，能够在给定的 12500 张图像中分辨猫和狗。在机器学习领域，用于分类的策略，包括 K 均值聚类、支持向量机等，均能够用于处理该分类问题。但在图像分类领域，神经网络技术具有更加明显的优势，特别是深度卷积神经网络，已成功地应用于图像识别领域。

1.3 评价指标

分类准确度（accuracy）和代价函数（Cost Function）是常用的分类评估指标。为了对模型进行更细致的评价，代价函数更加合理。通过代价函数计算结果的值越小，就代表模型拟合的越好。神经网络的代价函数是用于 logistic 回归的一个泛化^[2]。Kaggle 官网在此次竞赛中对预测结果的评估采用的是对数损失函数 log loss：

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))] \quad (1)$$

式中， m 为测试集图像总数量； $h_\theta(x^{(i)})$ 指图像类别为狗的预测概率；如果图像类别为狗， $y^{(i)}$ 为 1，如果为猫， $y^{(i)}$ 为 0。

项目预定的目标是将对数损失控制在 0.05 以内，这在 Kaggle 的 Leaderboard 上的排位已进入 Top-5%。

1.4 机器学习流程

熟悉机器学习的基本流程对更顺利地解决该项目是有帮助的。机器学习流程包括问题定义、评价指标选择、特征工程和处理过拟合等阶段。以下是机器学习较规范化的流程：

（1）问题定义和数据集获取

定义问题时需要对输入数据和即将预测的对象（输出）非常清晰。接着，分析问题类型，确定是否为二分类问题、多分类问题，还是标量回归等问题？确定问题的类型有助于后面对模型结构、损失函数等具体技术的选择。

（2）选择成功的度量

在模型训练的过程中，我们可能需要实时地观察模型的训练情况，以控制某些因素的变化。有了这个衡量标准，我们才能使模型在训练时往正确的方向收敛，而不会像无头苍蝇，晕头转向！这个衡量标准可以是模型准确度（accuracy），精确率（precision）和召回率（recall）等。作为对模型结果成功与否的评判标准，这个衡量标准可以指导我们选择损失函数（Loss Function）和优化方法（Optimizer）等。

（3）确定评估方法

三种常见的评估方法 (evaluation protocols): (1) 维持一个验证集不变, 这通常在数据充足的情况下使用; (2) k-折交叉验证, 数据量较少情况下; (2) 迭代的 k-折交叉验证, 数据量较少而要求高精度模型评价时使用。

(4) 准备数据

首先, 应该把数据转化成一种机器学习模型能够输入的格式。假设模型是一个深度神经网络, 则数据应该被格式化为张量。这些张量所取的值通常应该被缩放成较小的值, 例如 Xception 和 InceptionV3 要求输入值在 [-1, 1] 范围内。较常见的是缩放至 [0, 1] 范围内。如果不同的特征在不同的范围取值, 那么数据应该被归一化。而对于小数据问题, 可能做一些特征工程。一旦你的输入数据和目标数据的张量准备好, 你就可以开始训练模型了。

(5) 设计一个优于随机预测的模型

假设我们的输出和输入存在某些关联, 即输出能够由输入提供的信息来推测, 而且我们有足够的数据用于训练, 那么我们才可能建立有效的分类器模型。搭建模型有 3 个需要关注的点: (1) 选择 last-activation; (2) 选择损失函数; (3) 选择优化器并设置学习率。下表列出的内容对于选择常见机器学习问题 last-activation 和损失函数具有指导意义。

表 1-1 机器学习问题类型及激活函数与损失函数

Problem type	Last-layer activation	Loss function
Binary classification	sigmoid	binary_crossentropy
Multi-class, single-label classification	softmax	categorical_crossentropy
Multi-class, multi-label classification	sigmoid	binary_crossentropy
Regression to arbitrary values	None	mse
Regression to values between 0 and 1	sigmoid	mse Or binary_crossentropy

(6) 设计一个过拟合模型

当创建完一个具有预测能力的模型之后, 我们需要做的就是对模型进行改进, 使模型足够强大。例如对于神经网络模型, 可以添加更多的卷积层、使各个层更巨大, 或者增加训练的 epochs 数量等。在训练的时候, 我们需要时刻监测训练损失值和验证损失值。当模型在验证集上的表现开始下滑时, 通常就意味着模型已经过拟合了。

(7) 调整模型并调整超参数

这一步主要是为了获得理想的模型, 即模型既不欠拟合也不过拟合。因为上一步我们得到的模型是过拟合的, 所以这一步主要解决的问题是过拟合问题。常见的调整模型的方法有: 添加 Dropout 层、移除某些层、增加 L1/L2 正则化、尝试不同的超参数寻找最优配置, 以及迭代性地选择特征 (尝试新的 filter, 去除不具信息性的 filter)。

2 分析

2.1 数据的探索

项目使用的数据可以从 Kaggle 官网上获取, 它包括一个训练集和一个测试集。训练集包括 25000 张被标记的猫和狗的图片 (各占一半)。测试集为 12500 张未被标记的图片, 同样是猫狗图片各占一半。测试集将作为输入来训练分类器。在数据用于训练前须进行预处理, 使数据格式与模型相匹配。另外, 数据预处理还可以应用数据增强技术, 包括图片裁剪或填充、归一化。如果计算资源有限,

还可以将图像转化为灰度图像存储。数据集的部分图片如图 1 所示：



图 1. 训练集样本

训练集图像的尺寸分布如图 2 所示。通过该散点图，我们可以非常直观地了解数据的构成：绝大部分图像的尺寸都分布在 500×500 内。猫和狗的图片各自都只有一张图像的尺寸偏离正常范围内，属于异常值。于是下一步，我们首先找出这两幅图像。

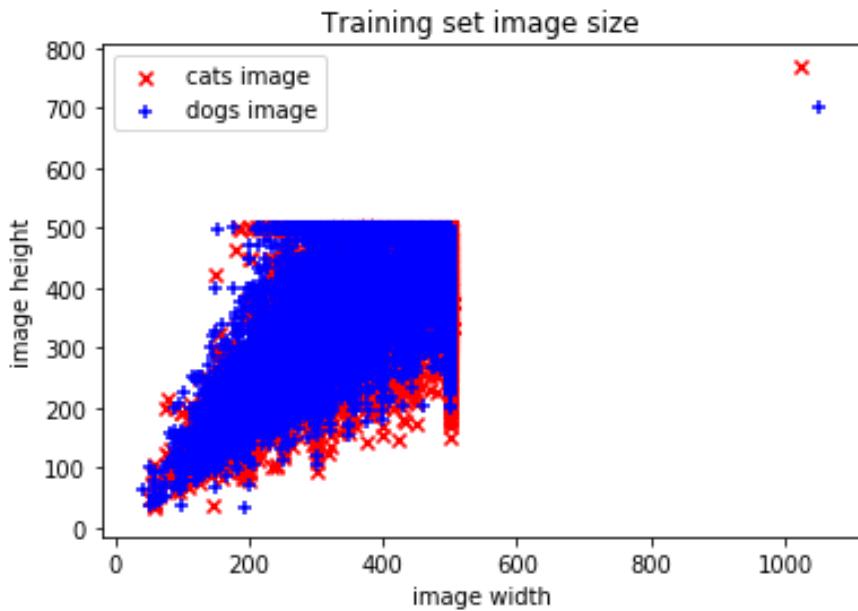


图 2. 训练集图像尺寸分布

这两幅图像如图 3-a 所示。但是，我们的图像在输入模型训练前，是需要进

行数据预处理的，例如我们可能对图像进行尺寸缩放或裁剪以使输入数据格式统一。图像裁剪可能会影响模型的训练，因为它可能将图像的有用信息丢弃。而在该项目中，我将只使用尺寸缩放。缩放后的猫狗图像如图 3-b 所示，我们还是可以很确定地辨别出猫和狗，这说明了图像缩放这种预处理方式并不会对模型分类带来坏影响，并不影响其学习，所以我认为这对异常值是没有必要处理的！

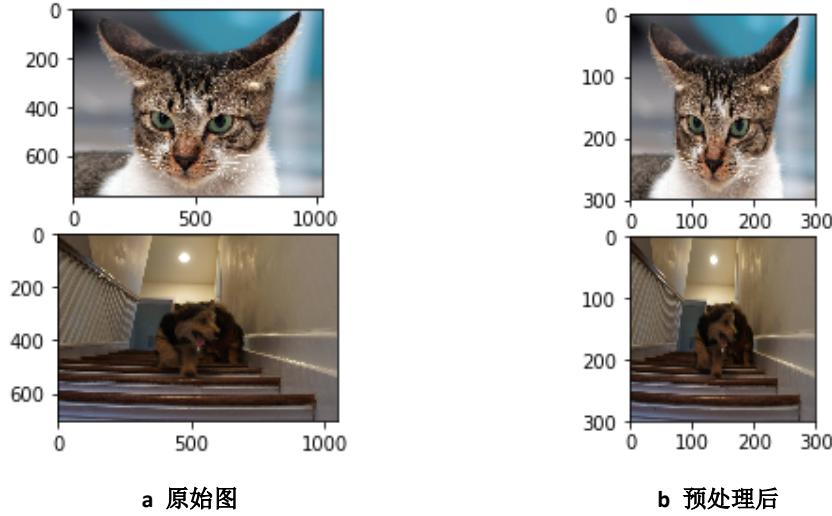


图 3. 尺寸异常图

既然图像的尺寸对模型的训练不会带来影响或者影响甚微，那么便接着探索数据集是否存在其他异常值。由于读取图像时三个通道的值都限制在 $[0,255]$ 内，分布范围较小，因此图像通道值的最大值和最小值和均值都不会成为异常值。因而，我将目光投向通道值的标准差，原因是通过标准差更可能发现异常值。设想一下，假如一幅图像中通道值的标准差很小，这意味着图像越接近于纯色，则其包含的信息量越少，越不容易用于提取可用于分类的特征。为了简化问题，将图像读取为灰度图，这样就只需要分析单通道了。

下面两幅图像分别是所有猫狗图像中，灰度图模型下，通道值标准差最小的情况。可以看到，虽然图像颜色比较单一，但是我们还是能够通过人眼清晰地识别出猫和狗，因此这部分不存在异常值！

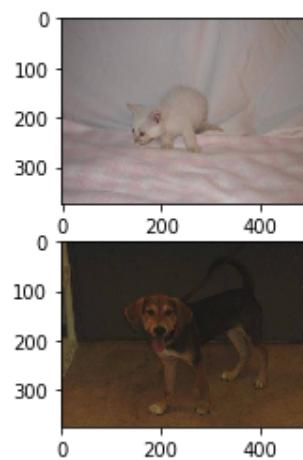


图 4. 颜色较单一的图

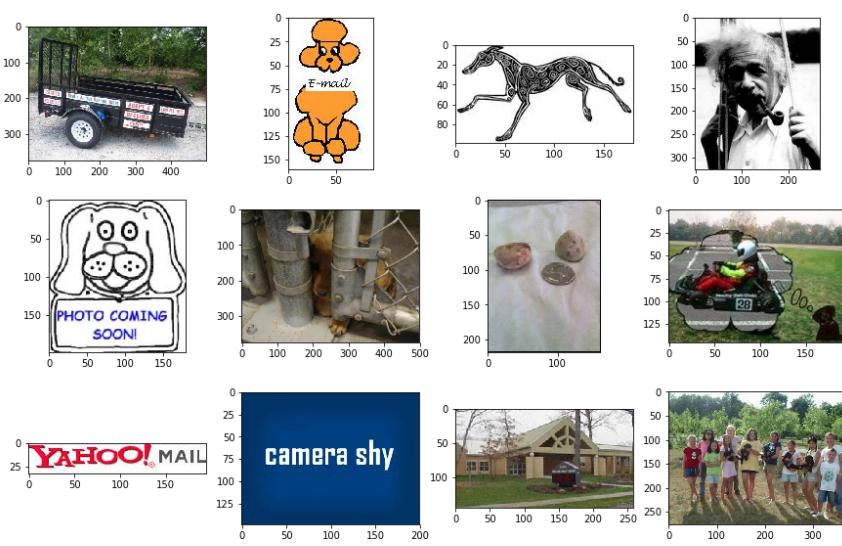
训练数据的异常值还有另一种情况，那就是类别标签标注上的异常值。寻找

这些异常值最直接但同时也是最繁琐的方法，就是耐心地将训练集里面的图片都手动检查一遍！但事实上我们可以用其他方式解决，那就是使用预处理模型来进行排查，这将大大减少我们的工作量。基本思路是，使用表现最佳的预处理模型（这里选择在 Imagenet 上表现很好的 Xception）对训练集的图片进行预测，给出预测正确分类概率最高的 n 个分类。检验猫、狗标签是否在这 n 个类别之内，若不在则将该图片视为可能的异常值。

利用 Xception 预测后，使用 `decode_predictions`，设置 `top` 参数为 n （一开始均设置为 30），将预测结果转化，获得预测正确分类概率最高的 n 个分类。检验猫、狗标签是否在这 n 个类别之内，若不在则将图像视为可能的异常值。此时，猫狗图片中的可能异常值数量分别为 80 和 23，如图 5 所示为部分查找得到的可能异常值。



a 猫图片可能异常值



b 狗图片可能异常值

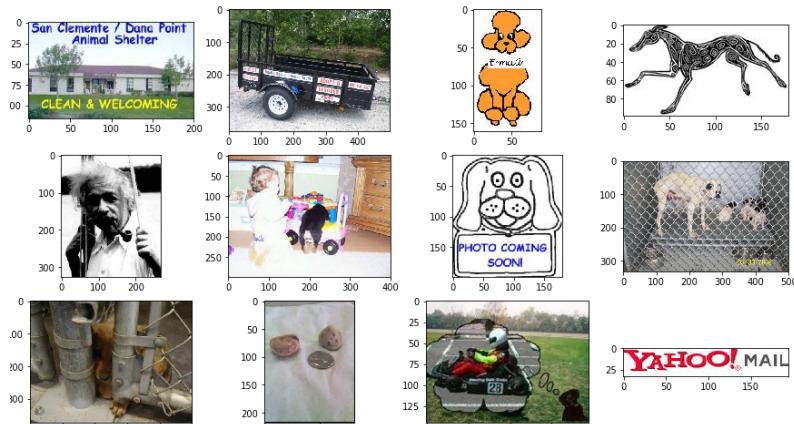
图 5. Top 参数均为 30 时的可能异常值

可以看到，当 `top` 参数均设置为 30 时，猫图片的异常值统计数量较狗图片

多，同时错误统计也多。对于狗图片的异常值统计情况，则几乎没有失误。这可能是由于 `imagenet` 在构建数据集时对狗的归类做的比较详细（`imagenet` 上狗的类别有 118 种，而猫的类别只有 7 种）。因此，我认为设置 `top` 参数时，应该将猫和狗分开讨论，不能一概而论。经过一番尝试，我将猫的 `top` 参数设置为 35，狗的 `top` 参数设置为 10，最后获得的部分可能的异常值如图 6 所示，其中猫有 62 幅，狗有 36 幅。相较于完全手动检查一遍，这已经大大降低了工作量。



a 猫图片可能异常值



b 狗图片可能异常值

图 6. 调整 Top 参数后的可能异常值

从上面的分析可以看到，猫狗大战数据集的异常值主要是标注上的异常值。在将训练数据输入模型训练之前，需要将这些异常数据进行预处理。我将考虑一下处理方式：(1) 将与主题完全无关的图片删除；(2) 对于分类错误的图片，修改类别标注（例如本来是狗的图片却标注为猫，这时候就需要将标注修改）；(3) 将背景复杂的图片进行裁剪。

这些异常值很难通过数据科学的方法（均值、平均值等）进行描述和发现，在处理的时候选择也只能手动处理。在报告的第三部分“方法”中将介绍具体的处理过程。

2.2 算法和技术

2.2.1 深度学习

鉴于深度卷积网络 (CNN) 在图像分类领域中的优势，我们将构建这样一种分类器，并逐步改善它的性能。CNN 学习识别基本的直线，曲线，然后是形状，点块，最后是图片中更复杂的物体，最终 CNN 分类器把这些大的，复杂的物体综合起来识别图片。CNN 通过正向和反向传播，自己学习识别物体，而不需要我们设定特定的特征。CNN 可能有几层网络，每个层可能捕获对象抽象层次中的不同级别，如图 7 所示。

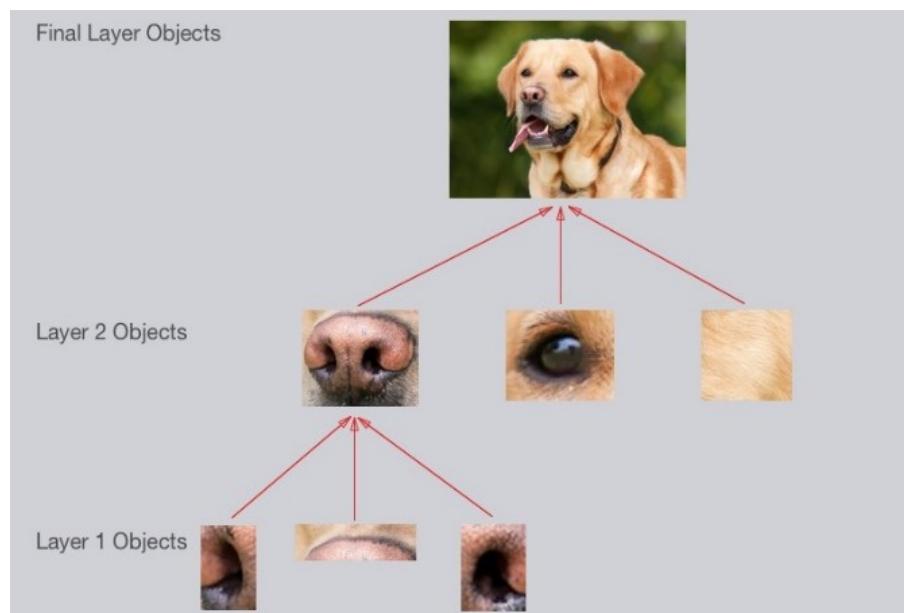


图 7. 对狗图片每一层 CNN 可能识别物体的示意图

为什么 CNN 要强于传统的计算机视觉分类方法呢？我们知道，图像分类的传统流程包括两部分，即特征提取和分类^[3]。特征提取指的是从原始图像中提取出更具体和高级的特征，这些特征携带着具体信息并能够用于区分各个类。这种特征提取的方式是无监督模式，因为从图像的像素点中提取信息时并没有使用类别标签，例如方向梯度直方图 (HOG) 等特征提取算法。

完成特征提取后，再使用这些特征与类别标签训练一个分类模型，常用的模型有随机森林、SVM 和 LR 等。传统的图像分类流程存在一个很明显的缺点，那就是特征提取的类型必须在模型训练前确定，而不能根据图像和分类标签进行调整。如果这些提前选择的特征不足以区分各个类别，即缺乏代表性时，训练获得的模型的准确性将会是不理想的。传统图像分类流程一个较好的方法是使用多种特征提取器，然后组合这些特征以得到一种更好的特征，但这本身就是比较困难和复杂的。

而 CNN 就完全不同了，它并没有建立使用硬编码的特征提取器，而是将特征提取和分类两个模块融合在一起。CNN 通过自动识别图像的特征来进行特征的提取，并基于分类标签进行分类。本项目将借助 Keras 构建 CNN 网络，它是 TensorFlow 的上层封装，提供了更多高级的 API 并且能够加快开发速度。

2.2.2 迁移学习(Transfer learning)

迁移学习通俗地讲就是应用已有的知识来学习新的知识，在深度学习领域体现在把预训练的模型之参数搬迁至新模型中，以加快新模型的训练。因此，应用迁移学习时模型便不用从零开始学习（*starting from scratch*）。迁移学习要求预训练模型和新模型之间存在相似性。Andrew Ng 曾说过，迁移学习将会是继监督学习之后的下一个机器学习商业成功的驱动力！可见迁移学习的重要性。

本项目将应用迁移学习技术，使用预训练的 CNN 特征，具体就是使用在 ImageNet 上大型卷积神经网络预训练的权重。因为 ImageNet 中的标签包含本项目中的分类标签——猫和狗，因此任务具有相关性，符合应用迁移学习的前提。

2.2.3 深度学习框架 Keras 和 GPU 计算

Keras 是一个高层神经网络 API，Keras 使用纯 Python 语言编写而成，并基于 Tensorflow、Theano 以及 CNTK 后端。在本项目中使用的 Keras 版本为 2.1.5，且基于 Tensorflow-gpu 后端。Keras 让使用者能够简易和快速地进行原型设计，因为它具有高度模块化、极简和可扩充特性。Keras 主要用于解决以下三类问题^[4]：

- 1) 二分类问题 (Binary Classification); 2) 多分类问题 (Multi-class Classification);
- 3) 标量回归 (Scalar Regression)。

GPU 加速计算是指同时利用图形处理器 (GPU) 和 CPU，以加快科学、分析、工程、消费和企业应用程序的运行速度。GPU 加速器于 2007 年由 NVIDIA 公司率先推出，现已在世界各地为政府实验室、高校、公司以及中小型企业的高能效数据中心提供支持^[5]。GPU 加速计算的原理是将应用程序计算密集部分的工作负载转移到 GPU 中运行，同时仍由 CPU 运行其余程序代码。

2.3 基准模型

对于图片分类问题，存在一些分类效果很好的深度卷积网络，例如 AlexNet、Inception，其中 AlexNet 的结构图如 8 所示，它由 5 个卷积层和 3 个全连接层构成。但在一开始，我们会先使用简单的分类模型进行训练并观察其表现。

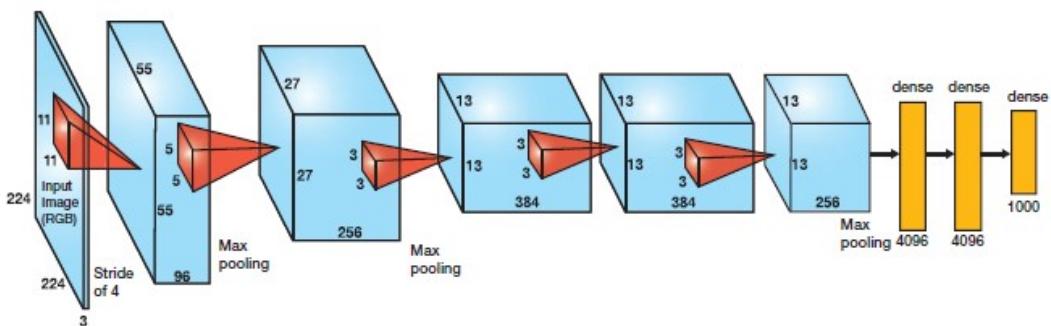


图 8. AlexNet 结构图

在该项目中，我首先建立了如下结构的简单模型，它的分类准确性确实要明显优于随机分类：

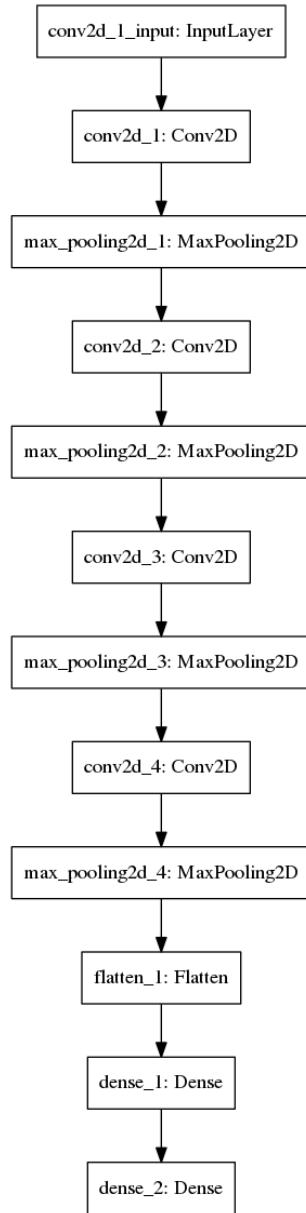


图 9. 搭建的第一个模型的结构图

可以看到，该简单 CNN 模型由若干个层“堆叠”而成，且可以分成这三部分：

(1) 输入层；(2) 卷积层和池化层的组合；(3) 全连接的多层感知机分类器。在需要处理过拟合问题时通常还会引入 dropout 层。那这些层是以什么样的原理工作的，又发挥着什么作用呢？

首先是最基本的卷积操作。卷积是信号和图像处理中最重要的操作之一，在 1D（如语音处理）、2D（如图像处理）及 3D（如视频处理）的数据处理中都存在卷积的应用。鉴于该项目的应用场合为图像分类，因此仅讨论图像处理中主要用于提取特征的 2D 卷积，它是卷积神经网络的核心模块。每一个卷积运算都需要一个卷积核，它是一个高和宽比原始图像小的矩阵。每个卷积核针对特定任务，例如锐化、模糊和边缘检测等。以锐化为例，卷积核为：

$$Kernel = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

卷积计算过程：首先，水平和垂直地翻转卷积核，由于此处卷积核是对称的，因而翻转后卷积核不变。接着，将卷积核的第一个元素和图像矩阵的第一个元素对齐，将卷积核的每个元素和图像矩阵的对应元素相乘并求和，将所得结果置于输出矩阵对应位置上，如图 10 所示。然后将卷积核在图像上滑移并用同样的方法计算，重复上述过程，最后便得到一个完整的输出矩阵。

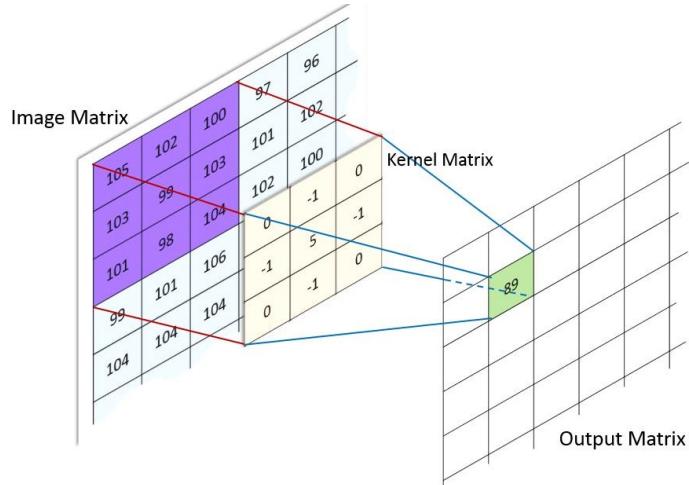


图 10. 卷积计算

上述锐化的卷积计算作用在实际图像上的效果如图 11 所示。



图 11. 图像的卷积操作

接下来介绍的是池化（pooling），池化最常用的是平均池化（mean / average pooling）和最大值池化（max pooling）两种形式。池化可以看作一种特殊的卷积过程，它的应用能够有效地减少模型的参数。通过卷积获得的特征一般维度较大，事实上为了描述一幅图像，我们可以对图像不同位置的特征进行聚合统计，只要统计的信息足够用于分类那就是可行的。这些统计不仅具有较低的维度，甚至还能改善分类效果，因为它更不容易过拟合。这种聚合统计的操作就是池化。池化的作用机制如图 12 所示。

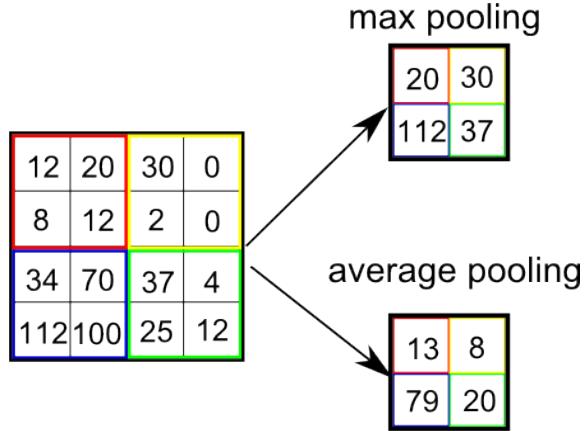


图 12. 池化机制

全连接层（fully-connected layers, FC）在整个卷积神经网络扮演着“分类器”的作用。全连接层的神经元与前一层中的所有激活具有完全连接。

Dropout 指的是在深度学习网络训练过程中（验证和测试过程中无效），按照一定的概率将网络中的神经元暂时丢弃。**Dropout** 层的目的是为了防止 CNN 模型过拟合，其作用机制如图 13 所示。在这个简单模型中并未引入 Dropout 层，但在对预训练模型进行微调时，我们便需要引入 Dropout 以有效处理过拟合问题。

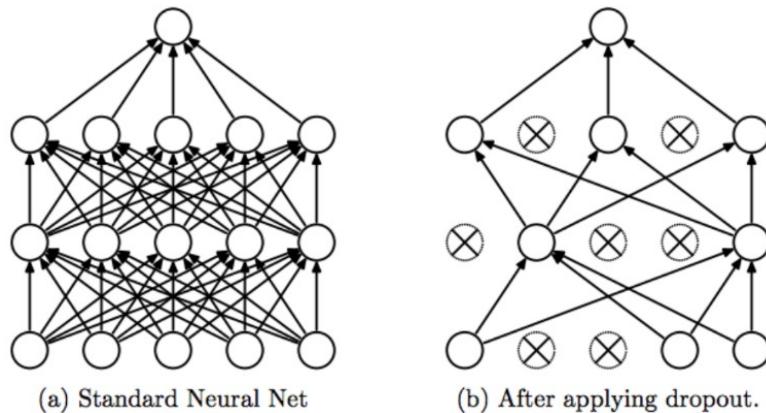


图 13. Dropout 原理

现在，已经介绍了 CNN 模型的特点、主要模块，以及通过这些模块构建一个简单网络。但自己涉及 CNN 有时并不是一个好的决定，它既费时而且又不一定能获得优秀的表现。所以，在该项目将采用迁移学习的方法，其中使用的用于 fine tune 的预训练模型包括 VGG16, ResNet50, Inception v3 和 Xception。

VGG16 和 AlexNet 想比，采用了连续的几个 3×3 卷积核来代替 AlexNet 中的较大卷积核，VGG 结构如图 14 所示。 3×3 卷积的引入是一种改进，因为多个非线性层的堆叠可以增加网络深度，进而实现更复杂特征的提取。从下面的结构图中还可以发现，VGG-D 多次重复使用同一大小的卷积核来提取更复杂核更具代表性的特征^[6]。VGG 最终其在 ImageNet 上的 Top-5 准确度为 92.3%。

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					

图 14. VGG 结构图

“ Inception ” 微架构由 Szegedy 等人在 2014 年论文 "Going Deeper with Convolutions" 中首次提出^[7]。尽管 VGG 在 ImageNet 上表现很不错，但是其卷积层的通道数过大，需要的计算量是巨大的。例如一个 3×3 的卷积核，若输入输出的通道数都为 512，那么计算量将是 $9 \times 512 \times 512$ 。我们知道在卷积操作中，输出特征图和输出特征图的位置具有映射关系。但是，深度卷积网络中大部分的激活值是不必要的，即前面说的映射关系有大部分是冗余的。Inception 的搭建便是基于这样的理念，它体现了最高效的深度网络架构应该是激活值之间为稀疏链接的。虽然存在一些技术可以对网络进行剪枝来获得稀疏权重，但是稀疏卷积核的乘法并没有得到优化，反而更慢。因此，GoogleNet 设计了一种如图 15 所示的 Inception 模块，其巧妙之处便是使用密集结构来近似一个稀疏的 CNN！另外，Inception 还使用 1×1 卷积核实现瓶颈层，而且最后还用全局池化层替换了全连接层，这两种做法都降低了计算量。Inception 的整体结构如图 16 所示。Inception 最终在 ImageNet 上的 top-5 准确度为 93.3%，但是速度比 VGG 还快。

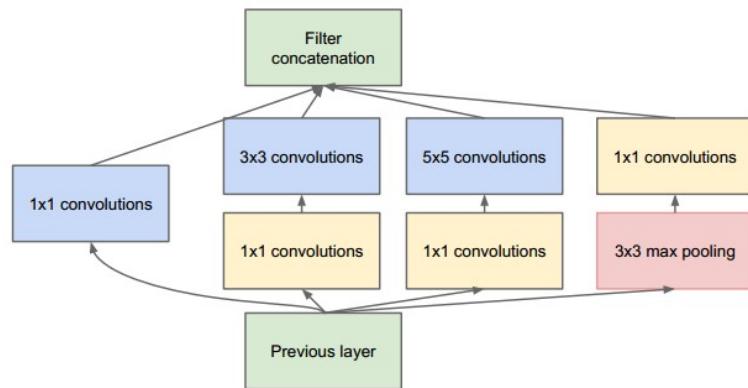


图 15. Inception 模块

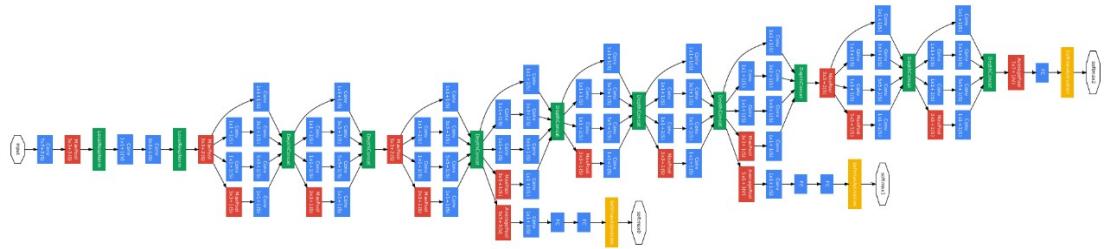


图 16. GoogLeNet, 2014

在 2015 年的 “Deep Residual Learning for Image Recognition” 论文中，He 等人首先提出 ResNet，ResNet 架构已经成为一项有意义的模型，其可以通过使用残差模块和常规 SGD（需要合理的初始化 weight）来训练非常深的网络^[8]。随着网络深度的加深，过拟合问题会越突出。而残差网络便设计了一种残差模块以允许我们训练更深的网络。残差网络与传统的顺序网络架构（AlexNet、VGG）不同，它加入了 $y=x$ 恒等映射层，可以让网络在深度增加情况下却不退化。图 17 所示为残差网络的一个构建块(building block)，输入 x 经过两个 weight layer 层后，再和输入 x 相加，形成一个微架构模块。ResNet 最终由许多这种模块组成。

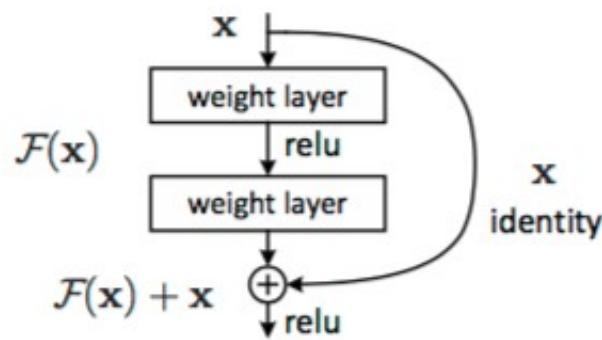


图 17. 残差网络的一个 building block

Xception 模型是由 François Chollet(Keras 维护者)于 2016 年在论文“Xception: Deep Learning with Depthwise Separable Convolutions”上提出的^[9]。Xception 是 Inception 架构的扩展，它用深度可分离（depthwise separable convolution）的卷积代替了标准的 Inception 模块，在基本不增加网络复杂度的前提下提高模型准确性。在 Inception 中，开始将图像区域和通道稍微分开。使用 1×1 的卷积将原始输入投射到多个分开的更小的输入空间，而且对于其中的每个输入空间，都使用一种不同类型的 filter 来对这些数据的更小的 3D 模块执行变换。Xception 则更进一步。它不再只是将输入数据分割成几个压缩的数据块，而是为每个输出通道单独映射空间相关性，然后再执行 1×1 的深度方面的卷积来获取跨通道的相关性^[10]。

从这些流行的 CNN 模型的结构设计可以看出，CNN 的设计趋向于模块化和高效率，特别是在 Inception 和 ResNet 中。

3 方法

3.1 数据预处理

在数据探索阶段，我们已经对异常值进行了仔细地排查。因为在接下的数据预处理阶段，我们会通过图像尺寸缩放将尺寸异常值图像处理为正常尺寸图像，因此尺寸异常值不会对模型的学习带来负面影响。但是数据中存在的标注异常值则不一样，标注异常值将影响模型的学习，必须最大限度地排除。

在第一章中，我们已经将这些可能的标注异常值查找出来了。在这里必须进行处理，基本思路是：(1) 将与主题完全无关的图片删除；(2) 对于归类错误的图片，修改类别标注；(3) 将背景复杂的图片进行裁剪。这一步将完全通过人工方式手动处理。部分图片的处理方式如图 18 所示：



图 18. 异常值处理

最终将狗图像中序号为 1043、1194、1773、1835、2614、4367、5604、6475、6725、8736、9517、9931、10161、10237、10747、10801、11299、12376 的图片删除，没有狗图像需要标注修改，将序号为 2422、3889、4507 的狗图像进行裁剪；将猫图像中序号为 372、2520、3216、3672、3868、4338、4688、5351、

5418、7377、7564、8456、9090、9171、10029、10712、11184、12272 的图片删除，没有猫图像进行标注修改，将文件名为 252、503、724、2150、2337、3731、4190、4308、5347、5355、6429、7599、7920、9494、9596、10270、10636、12227、12424、12476、12493 的猫图像进行裁剪。通过这一步的异常值处理，获得新的训练集。处理后训练集中猫图像剩下 12482 张，狗图像剩下 12482 张，所以训练集一共剩下 24964 张图片。在进一步地进行数据预处理前先将文件中猫狗图像的名称重新命名，使名称的序号是连续的。

不同卷积网络模型对数据预处理的要求是有差别的，例如，模型对图片的尺寸的要求：**VGG16** 和 **ResNet50** 默认的输入图像的分辨率为 224×224 ，而 **Xception** 和 **Inception V3** 则默认为 299×299 。另外，**Xception** 和 **Inception V3** 都要求输入的张量的值必须在 $[-1,1]$ 内。所以，在进行数据预处理时一定要具体问题具体分析，而不能一刀切。

接着，需要考虑将训练数据划分为训练集和验证集。这里，使用 10000 张猫的图像和 10000 张狗的图像作为训练集，将剩下的分别为 2482 张猫和狗的图像作为验证集。**Keras** 的 `model.fit` 函数提供了 `validation_split` 参数来方便地实现这个过程，但因为数据集是平衡的，手动将用于训练的猫狗图像中各前 $n\%$ 作为训练集，后 $(100-n)\%$ 作为验证集也是可取的（在参数微调时我便是这样处理数据的）。

另外，由于在使用预训练的模型时容易发生过拟合，因此在数据预处理阶段使用特征工程技术是一个十分可行的手段。特征工程（**Feature Engineering**）指的是在将数据输入模型之前，应用我们具备的机器学习算法知识和对数据的掌握，给数据加上一些硬编码。在该项目中，为了防止过拟合，我采用了数据增强技术（**Data Augmentation**），使用 **Keras** 内置的 `ImageDataGenerator` 就能够快速地实现。数据增强在针对图像分类问题中具体指，在训练的数据量相对较少时，通过平移、翻转、镜像和添加噪点等方法从已有的数据中创造出一批“新”的数据。这样，模型在训练的时候，在每一个 `epoch` 中都不会面对完全一样的一批数据。虽然这些新增加的数据不能媲美原始训练数据（和原始训练数据存在关系），但对防止过拟合确实有帮助。

3.2 执行过程

3.2.1 流程(Pipeline)

本项目中，在使用各个预训练模型时，都采取了同一个流程。首先，尝试使用特征提取方法。第一步，加载去掉顶层分类器的模型（留下卷积层）之结构和权重，并将训练数据输入入模型，提取特征向量；第二步，为卷积层添加全连接层和 `drop out` 层，输入上一步提取的特征向量进行分类，观察分类结果并评价。第三步，为卷积层添加一个包含卷积核的顶层分类器，将加载的卷积基础层冻结（不冻结顶层分类器的卷积核），重新编译模型。输入图片数据进行训练，得到一个训练过的顶层分类器。之后，尝试使用参数微调方法。在使用特征提取手段时，我们已经得到了一个包含卷积层的顶层分类器。现在，可以将冻结的卷积层解开部分层，与顶层分类器一起训练。参数微调就是微调加载的模型的部分卷积层以及新加入的顶层分类器的权重。

这里需要注意的是：为了顺利地进行参数微调，模型的各个层都需要以预先训练过的权重为初始值。所以，不能将随机初始化权重的全连接层放在预训练过的卷积层之上，否则会破坏卷积层预训练获得的权重。体现在前面谈到的流程中，就是先使用特征提取方法训练顶层分类器，再基于这个顶层分类器进行参数微调。

另外，参数微调时不会选择训练整个网络的权重，而只微调位于模型中较深的部分卷积层，这在一定程度上可以防止过拟合。在前面已经提到，因为由底层卷积模块学习到的特征更具一般性，而不是抽象性。

对于 VGG16 模型，我首先尝试选择将其卷积模块的末尾 3 个卷积层冻结然后再进行参数微调，但效果并不理想。于是我又将冻结层扩大至末尾 4 个卷积层，进行参数微调；对于 InceptionV3，选择将 249 层之前的层冻结；对于 Resnet50，选择将 168 层之前的层冻结；对于 Xception，选择将 126 层之前的层冻结。下图所示表示了 VGG16 参数微调的设置方法，其他模型的设置原理与此类似。

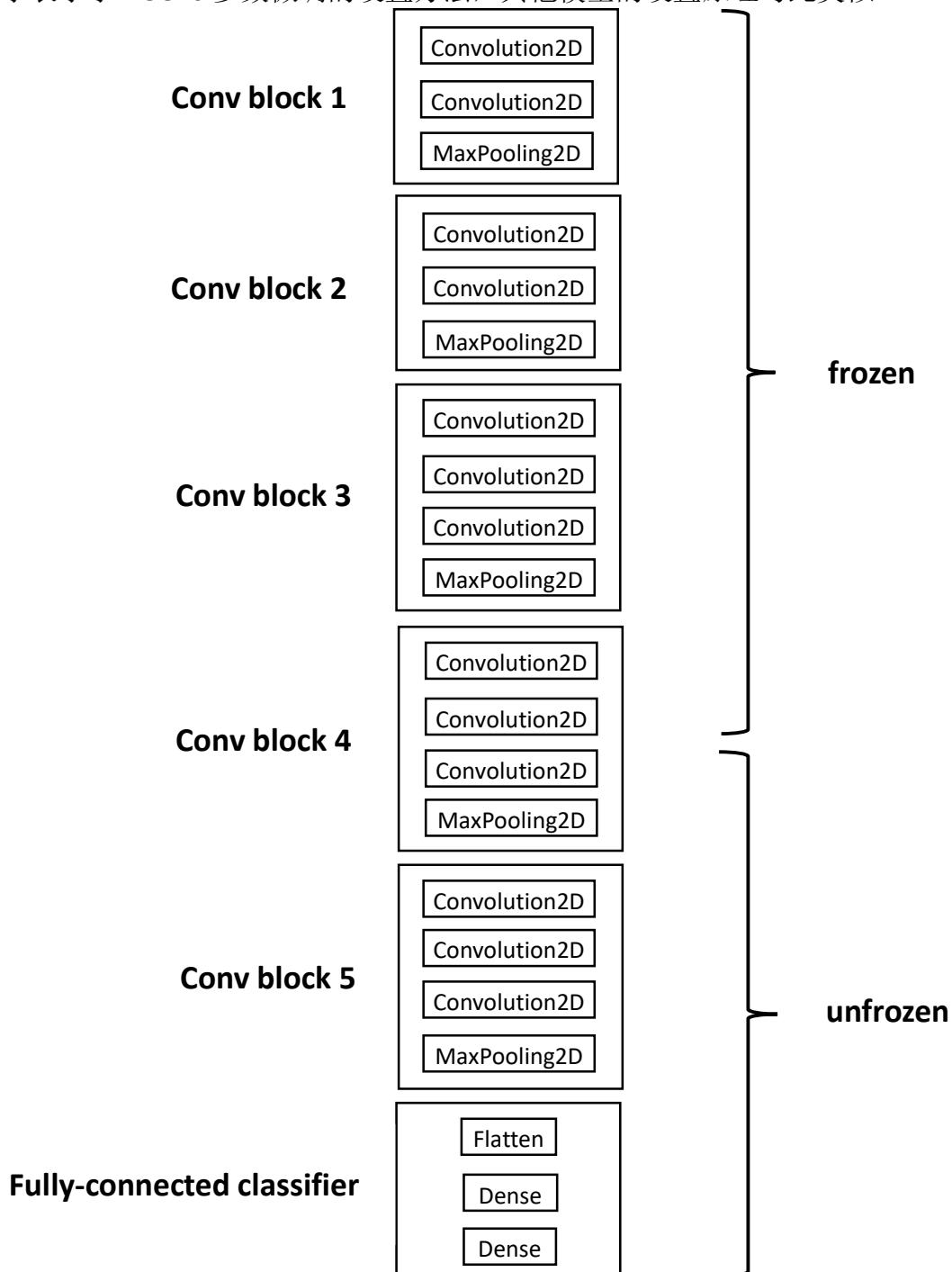


图 19. VGG16 参数微调设置

3.2.2 模型的训练

应用参数微调技术时应该在比较低的学习率下训练，这里使用的学习率为 0.00001，优化器为 RMSprop。较低的学习率可以使训练过程中保持较低的更新幅度，以防止破坏模型卷积层预训练的特征。

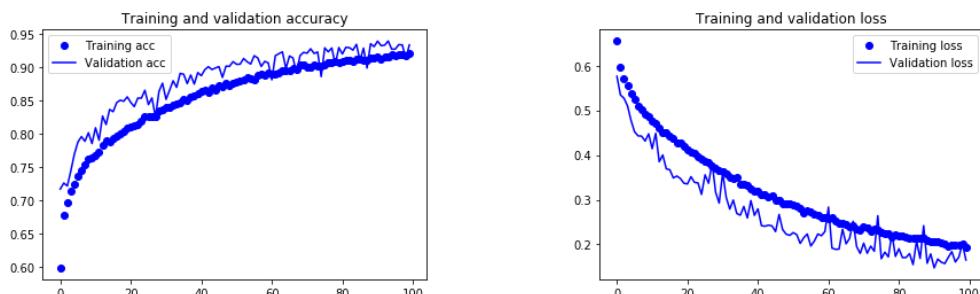
在一开始，我将 Batch size 设置为 50，但是在训练时发现模型不容易收敛，而且在测试集和验证集上的损失函数值波动较大，引入 Keras 的回调函数 EarlyStopping（EarlyStopping 能够让模型在训练时根据设定的条件停止）时，需将 patience 参数设置为较大的值。故后面将 Batch size 调整为较大值。但 Batch size 太大对 GPU 显存资源也是一个很大的负担，最终将训练时的 Batch size 调整为 100。

为了应对模型过拟合问题，我使用了数据增强技术来训练一个新的网络，所以所有 epoch 中是不会有两次相同的输入的。但是，这些输入仍然是相互关联的，因为它们来自有限的原始图像，数据增强并不能产生新的信息，而只能重新混合现有的信息。因此，这可能不足以完全摆脱过度拟合。为了进一步克服过度拟合，我还将在模型中全连接层分类器的前面添加一个 dropout 层这是一种正则化手段，不过跟 Regularization 不同的是，它是通过将训练的层中的部分神经元的输出置零来实现的。在这里，使用的 Dropout 参数为 0.5。

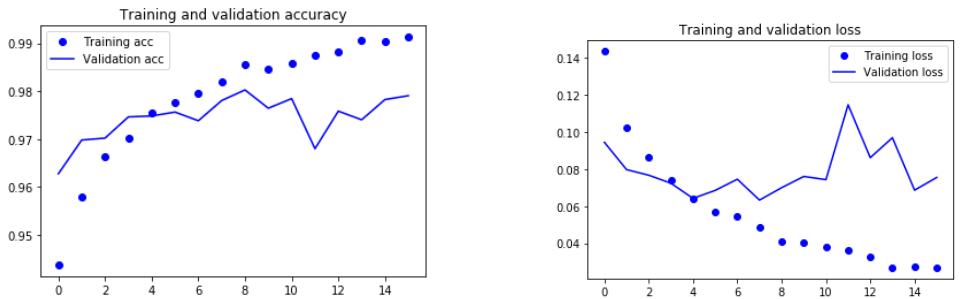
但在参数微调过程中，我发现 Xception 很快就停止训练了，因为使用了 EarlyStopping 回调函数，而且 val_acc 和 val_loss 都呈现出模型的性能在下降的趋势，最后参数微调效果也不理想。于是我增大了数据增强的幅度，重新跑了一遍程序，这时 Xception 模型依然很快就停止训练，但是 val_acc 和 val_loss 却是往好的趋势变化。在这之后，我果断将 EarlyStopping 中的 stop 参数调整为较大的整数，从而增加 Xception 训练的 epoch 数量。

3.2.3 训练结果生成

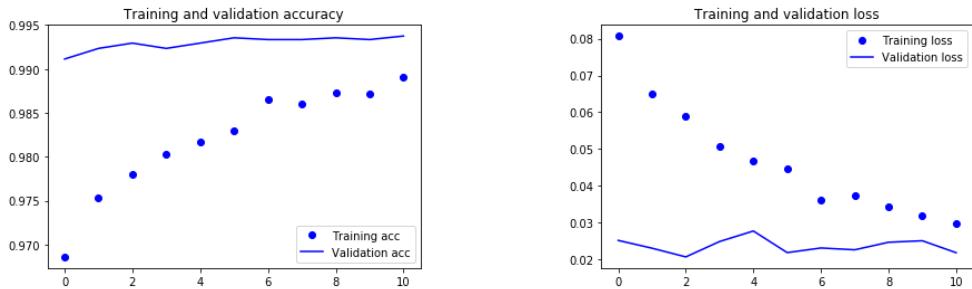
所有模型的训练都在 Jupyter Notebook 中完成。程序批量读取图片，并根据具体模型进行相应的数据预处理，包括数据增加技术。各模型训练过程如图 20 中所示，由于参数微调时的模型的顶层分类器之权重是由特征提取过程中训练过的，因此在训练的一开始，模型就已经表现得比较好。后三个模型在整个参数微调过程中，模型的改进幅度并不明显。最终训练得到的结果准确率已经很高了。



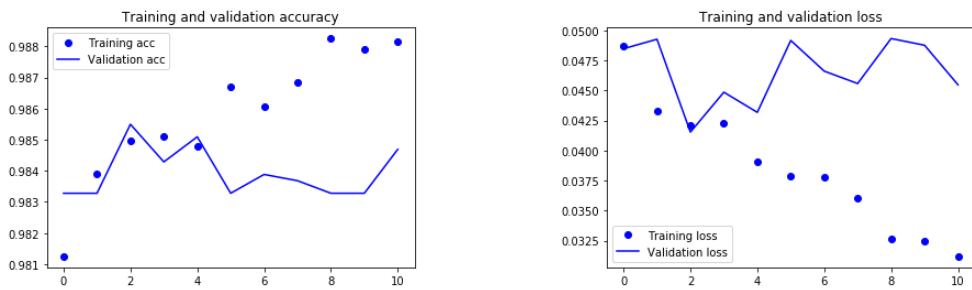
a. Basic model



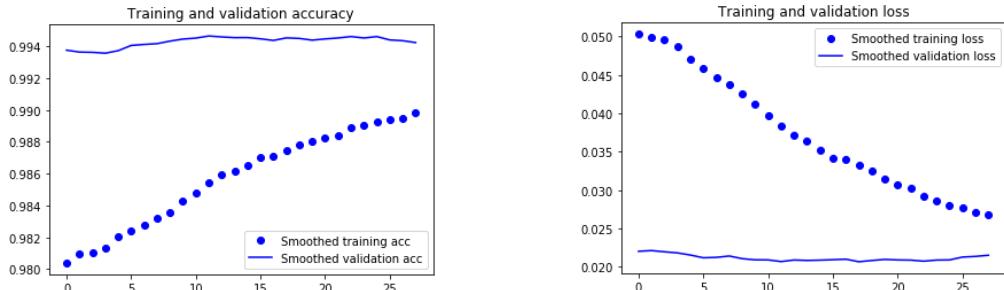
b. VGG16



c. InceptionV3



d. ResNet50



e. Xception

图 20. 模型训练曲线

可以看到模型表现相对较优的是 Xception、Inception V3，它们在验证集上的分类准确率均高于 99%，损失值均低于 0.03。

而 Inception V3 在验证集上的分类准确率已经达到了 99.3%，logloss 值为 0.025 左右，参数微调效果非常理想，和 Xception 十分接近！

ResNet50 虽然也达到了不错的分类准确性，但结果却和 Inception V3 相比却处于劣势。初步猜测是模型训练时参数设置不够合理导致的，例如数据增强的设置，在往后的学习中会进行验证。

VGG16 的表现相对于前面三种模型则处于明显的劣势，毕竟 VGG16 相对较老，网络结构也较浅。

3.3 完善

使用预训练的模型进行参数微调确实比使用自己搭建的模型能够获得更小的损失函数值，但是训练过程还是十分缓慢的。在对每一个模型进行参数微调前，我都有都尝试直接使用特征向量来进行分类，而且个别模型的分类结果十分理想，包括 ResNet50、InceptionV3 和 Xception。因此，尝试将提取的多个较优模型的特征进行组合，是一个非常可行的方案！

整个构建过程很简单。首先，我们选择在前面的训练中表现最好的三个模型 ResNet50、InceptionV3 和 Xception 作为特征融合的对象模型。接着，先将数据输入各个模型中，提取特征向量并保存下来。然后，将对应同一个训练样本的来自这三个模型的三个特征向量在长度上进行堆叠，最终得到共 12500 个维度为 $3 \times 2048 = 6144$ 的特征向量。

最后，构建一个简单的神经网络，输入上一步特征融合得到的特征向量集合，进行训练。应用特征融合技术构建的模型如图 21 所示^[11](这部分代码和思路借鉴自杨培文老师，由衷感谢！)。

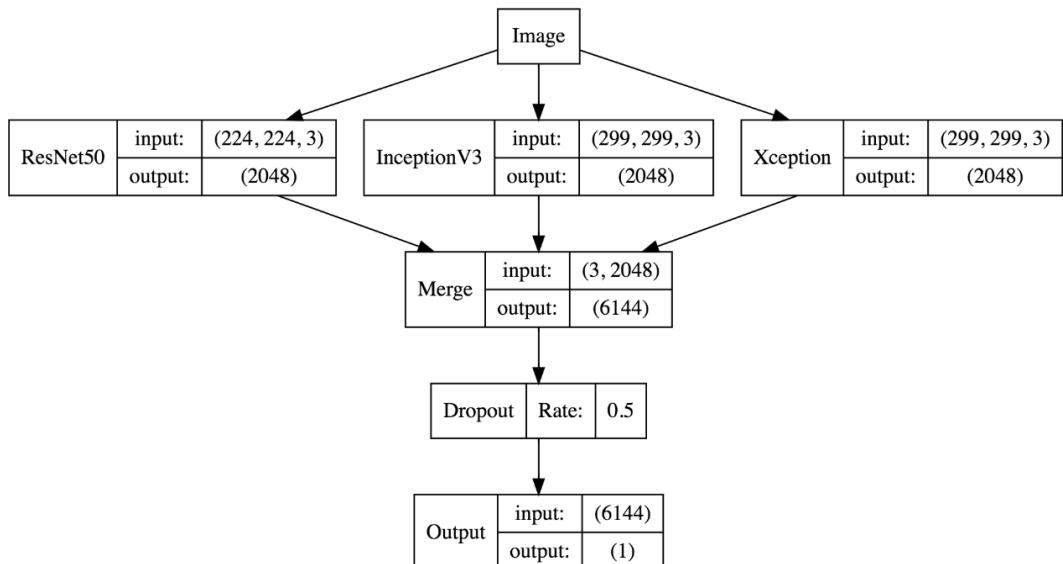


图 21. 特征融合模型

训练过程如图 22 所示。最终在验证集上的准确率很轻松地就达到了 99.6%，`val_loss` 也只有 0.012。

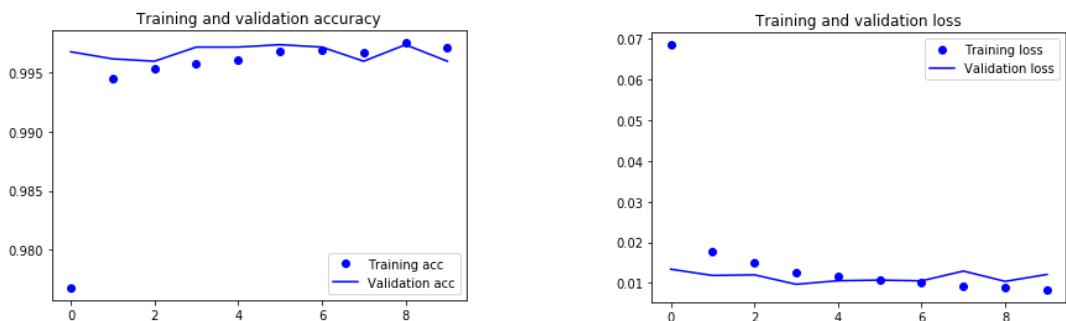


图 22. 特征融合训练曲线

4 结果

4.1 模型的评价与验证

在模型训练过程中，我反复对各个模型的参数进行调整，力求获得较优的分类器模型。首先是学习率，一开始我将参数微调的学习率设置得太大（0.001），导致训练曲线波动较大，而且模型优化的趋势不明显，最后获得的模型分类准确性不理想。通过查找资料才发现，参数微调时如果学习率设置太大容易将预训练模型的权重破坏。因此我将学习率设置为 0.00001，果然，模型训练曲线不再剧烈波动，而且最后获得的模型分类准确性明显升高，`val_loss` 变小。

使用数据增强时也需要注意！如果数据增强的幅度不够，那重新训练的模型质量并不会得到质的提高。但是如果数据增强的幅度太大，“新的图像”和原始图像没有任何关联，那会导致模型学习错误的信息，影响模型的成绩。因此在使用 Keras 的 `ImageDataGenerator` 时我一开始的设置就偏于保守，将 `rotation_range` 参数设置为 10, `width_shift_range` 等参数（具体参数见.ipynb 文件）设置为 0.1。然后逐渐增大参数设定值，反复训练并观察结果。最后发现对于 InceptionV3 和 Xception 模型 `rotation_range` 设置为 30, `width_shift_range` 等参数设置为 0.2 时参数微调效果很好。例如对于 Xception 模型，设置较优的数据增强参数与一开始的保守参数相比较，得分从原来的 0.5076 变为现在的 0.4457。但是对于 ResNet50 模型，上述参数设置的变化并没有对结果产生明显的改进。

在训练时我也根据模型的拟合情况，尝试改变 `Dropout` 的参数，包括设置为 0.3、0.5。但是发现，对于 Xception、InceptionV3 以及 ResNet50 模型，这个改变带来的影响不明显，于是最后决定将 `Dropout` 参数固定为 0.5，将调节过拟合的重心放在数据增强上。

模型训练时的 `epoch` 数量对模型的最终结果也很重要。在该项目中具体表现在两个地方：一方面，在对预训练模型进行参数微调前，我都会先训练一个顶层分类器，作为参数微调的基础。一开始我并没有太在意这个阶段的 `epoch` 数量，认为只要稍微训练一下就行，于是这个阶段的 `epoch` 数量均设置为 20。但后面我才发现事实并非如此，顶层分类器对后面的参数微调也很重要。于是我具体模型具体分析，力求在这个阶段也找到最佳的 `epoch` 数量，使顶层分类器越理想。另一方面，在参数微调的过程中，我引入了 Keras 的 `EarlyStopping` 函数，对于 VGG16 和 ResNet50，训练曲线的波动较大，而且 `val_loss` 在一开始下降趋势不明显，所以需要将 `patience` 参数设置得大一点，以防止模型训练过早的结束。

在训练完模型之后，最后一步便是对测试集进行预测了。由于本项目的评价指标是对数损失函数 `LogLoss`，对于预测正样本，0.995 和 1 相差无几，但是对于预测负样本 0 和 0.005 却相差甚远。因此对模型预测输出结果进行截取可能会小幅地改善最终结果。所以在产生最后的预测结果数据时，都使用了 `numpy.clip` 函数将预测值限制在[0.005 和 0.995]这个区间。

在使用与训练模型时，由于使用参数微调的分类结果要优于只使用特征提取的分类结果，因此，我们只把参数微调得到的模型在测试集上的分类结果上传至 Kaggle 网站，各模型获得的得分分别为：

表 2-模型在测试集的预测表现

Model	Score
VGG16	0.09509
Inception V3	0.04241
Resnet50	0.06505
Xception	0.04625
Feature Integration	0.03993

可见，在单模型中 Inception V3 和 Xception 的参数微调效果最好，得分均优于设定的目标 0.05。Resnet50 次之，VGG16 与其他单模型相比差距较明显。而特征融合的得分最优，为 0.03993，在全球排名中可以排到 15/1314，比所有单模型的表现都要好。

4.2 合理性分析

对单模型进行参数微调的合理性：对预训练模型进行参数微调属于迁移学习的一种方式。由于预训练模型先前在大数据集(这里加载的权重是在 Imagenet 训练得到的)上训练过，而且权重基本上是最优化的值，因此这些预训练模型本身就很可靠。如果从零开始设计和训练模型会花费很大的时间和精力，而且得到的结果不一定优于参数微调结果。

特征融合技术的合理性：特征融合技术的实现可以有多种方式。使用堆叠特征向量的方法，非常简单快捷，而且效果也比较理想。

5 项目结论

5.1 结果可视化

我们还可以将模型学习的结果进行可视化。卷积网络可视化方式中最主要和常用的有这三种：(1) 可视化中间卷积层的输出 (intermediate activations)；(2) 可视化卷积神经网络滤波器 (convnets filters)；(3) 可视化图像中类激活的热图 (heatmap)。

我们在这里只使用第三种方式 (热图, heatmap) 来观察模型学习的情况，这类技术一般称为“类激活映射” (Class Activation Map, CAM) 可视化。heatmap 对于理解图像中的哪个部分被识别为某一类是非常有帮助的，它也因此可以用于在图像中对某物体进行定位。

在这里，使用一开始构建并训练的简单模型进行可视化，简单模型准确性不是很高，但这样反而可能观察到模型的缺陷。模型的 CAM 可视化过程如下：首先，选择一张猫和一张狗的图片作为用于模型可视化的图片，如图 23 所示。



图 23. 原始图片

接着，应用 CAM 技术生成热图，如图 24 所示。。

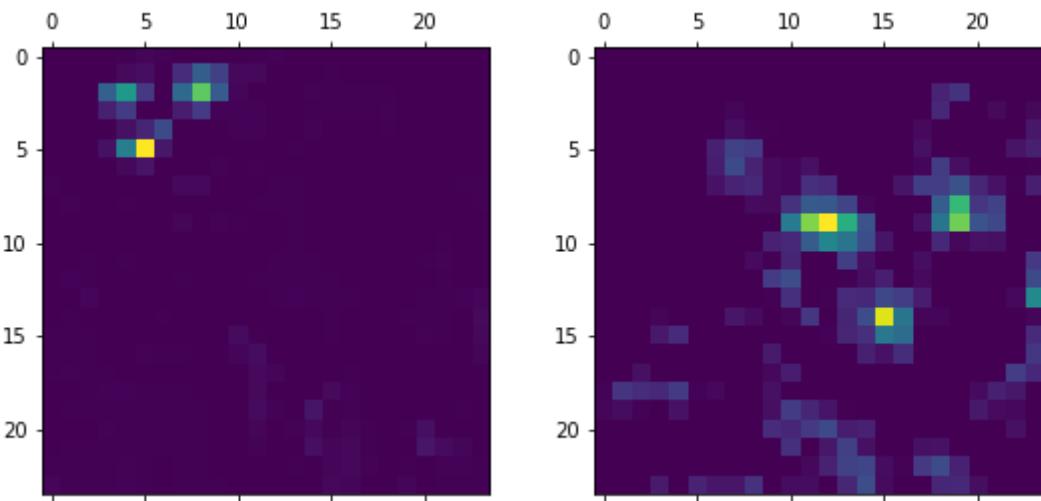


图 24. CAM 热图

最后，将热图叠加在原始图像上，得到最终结果如图 25 所示。可以直观地观察到简单分类器主要依靠眼睛和鼻子特征来区分猫狗，简单模型较好地对物体进行了识别和定位。



图 25. 叠加结果

5.2 对项目的思考

猫狗大战是一个非常有趣同时又充满挑战的竞赛。识别图片中的猫狗对人类来说很容易，但交给计算机判断有一定的难度，通常需要比较复杂和庞大的卷积网络才能做出比较理想的分类效果，这在一定程度上也衬托出人脑的强大。

在图像分类领域，**CNN** 发挥着重大的作用。**CNN** 是一个非常灵活的模型，而且一直在进步，可以应用的新方法也在不断出现。同时，训练和调节 **CNN** 以获得更优的准确性也是一个充满挑战的过程，需要一定的技巧和充足的耐心。通过这个项目的实践，使得我对 **CNN** 的理解和应用水平得到了大步地提升，开始能够独立地写深度学习代码并解决实际问题。

5.3 需要做出的改进

由于时间仓促，以及计算资源的限制。在进行参数微调时，各个模型冻结层和微调层的选择组合总数很有限。接下来，可能会尝试研究冻结层数量对参数微调效果的影响，以寻求各个模型相对最优的参数微调方法。

另外，如果要继续优化模型表现，可以使用更加复杂的特征融合技术。例如，使用参数微调后的预训练模型来导出特征向量，而不是直接使用从 **Keras** 中加载的预训练模型来导出。而且从特征融合训练曲线中可以看到，模型已经趋于过拟合，因此对于特征融合，还可以尝试在数据预处理阶段使用数据增强技术。

本项目中，模型优化的空间事实上已经十分有限了！所以为了获得更好的分数，可以减小验证集的比例，将绝大部分数据投入到训练集中。或者尝试使用其他评估方法，如 **K**-折交叉验证。

最后，可以手动地筛选分类错误的图片，针对具体模型，探索模型存在的缺陷。再根据这些缺陷，探究对模型进行进一步优化的方法。

参考文献

- [1] D. A. Forsyth and J. Ponce. Computer Vision: A Modern Approach. Prentice Hall, August 2002.
- [2] <https://www.coursera.org/learn/machine-learning/supplement/afqGa/cost-function>
- [3] <http://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/>
- [4] François Chollet. Deep Learning with Python[M]. Manning, 2017.
- [5] <https://www.nvidia.cn/object/what-is-gpu-computing-cn.html>
- [6] Simonyan K, Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition[J]. Computer Science, 2014.
- [7] Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the Inception Architecture for Computer Vision[J]. Computer Science, 2015:2818-2826.
- [8] He K, Zhang X, Ren S, et al. Deep Residual Learning for Image Recognition[J]. 2015:770-778.
- [9] Chollet F. Xception: Deep Learning with Depthwise Separable Convolutions[J]. 2016:1800-1807.
- [10] <https://towardsdatascience.com/an-intuitive-guide-to-deep-network-architectures-65fdc477db41>
- [11] https://github.com/ypwhs/dogs_vs_cats