机器学习工程师纳米学位毕业项目
基于卷积神经网络实现猫狗识别

陈亚俊

2018 年 6 月 10 日

目录

1 定义

1.1 项目概述

计算机视觉是一门研究如何使机器学会“看”的科学，也就是指用摄像头和电脑代替人眼对目标进行识别、跟踪和测量等机器视觉，并进一步做图形处理，使电脑处理成为更适合人眼观察或传送给仪器检测的图像；计算机视觉研究相关的理论和技术，试图建立能够从图像或者多维数据中获取信息的人工智能系统。

该项目解决的问题就涉及计算机视觉的图象检测识别领域，做该项目是为了加深对卷积神经网络的理解和实践经验，并试图探讨将其运用到多种不同的场景。

该项目使用的数据集主要是 kaggle 网站上的“Dogs vs. Cats Redux: Kernels Edition”比赛中所给的数据，该数据集中有 25000 张训练数据，其中猫的图像有 12500 张，狗的图像有 12500 张，另外还有 12500 张测试数据。项目使用卷积神经网络来实现图像的识别，卷积神经网络(Convolutional Neural Network)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，对大型图像处理有出色的表现，其包括卷积层和池化层。多年来，卷积神经网络已经发展出了很多模型，例如经典的 LeNet-5、具有里程碑意义的 AlexNet、由牛津大学著名研究组 VGG 提出的 VGG-Nets 等，这些模型的出现推动了卷积神经网络的快速发展，该项目将尝试使用其中的一些模型来实现猫狗识别，并尝试使用多个模型融合的方法来进一步提高识别预测的性能。

1.2 问题陈述

该项目需要根据数据集中的数据，构建卷积神经网络模型，输入一张彩色图片，识别出该图片中是猫还是狗，用数据集中的图片输进模型，使模型不断训练，然后调节超参数，不断优化模型，直至在测试中模型的正确率达到一定程度。

实现这个项目的方法有很多种，可以使用传统的机器学习方法，如支持向量机(SVM)，考虑到 SVM 的复杂性较高，运算起来效率很低，可以考虑使用主成分分析(PCA)来对图像进行降维操作，由此来提高 SVM 的运行效率。也可以使用深度学习的方法来解决该识别问题，因为卷积神经网络相比普通神经网络在图像上的优势明显，所以考虑使用卷积神经网络来解决该问题；卷积神经网络的模型众多，LeNet-5 模型的复杂度较低，可以先尝试自建该模型来对图像进行识别，如果效果并不理想，可以考虑使用更复杂、更现代化的模型，例如 VGG-16，Xception 等，因为这些模型的复杂度较高，所以可以考虑使用迁移学习的思想来简化操作，提高工作效率；理论上，这些复杂模型的表现应该是要优于 LeNet-5 模型的，如果效果还是达不到预期，则可以考虑使用迁移学习+混合模型的方案，使用多个卷积神经网络模型，通过特征提取的方式将它们进行融合来对图像进行识别，但由于使用了多个模型混合的方案，该方法在运行效率上是要低于之前方法的，在多次尝试后，可以对各种方法进行权衡，选择最适用

于该问题的方法。

最终的期望结果是模型能够准确的识别出一张图像是猫还是狗，并且其正确识别的准确率达到较高水平，错误识别的准确率处在较低水平。在面对大量数据，包括一些不太标准的图像时，模型的准确率也能达到较高水平。

1.3 评价指标

该项目的目的是实现猫狗识别，结合数据集来分析，这是一个二分类问题，由于正确率无法反映出模型误判的具体程度，显然一个误判程度更低的模型要优于误判程度较高的模型，因此可以使用 Logloss 来作为评估标准会更合适，它可以避免正确率的局限性。在 kaggle 上，用作评分的 Logloss 的计算公式如下:

$$\text{Logloss} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(p_i) + (1-y_i)\log(1-p_i)]$$

其中 N 为样本数，y_i的值取决于图像的种类，如果是图像是 dog，y_i是 1，如果是 cat，y_i是 0；p_i是图像预测为 dog 的概率，根据这个公式就可以算出 Logloss 的值作为评估标准。

2 分析

2.1 数据探索

数据集分为训练集和测试集，训练集中有 25000 张图像，其中猫的图像有 12500 张，狗的图像有 12500 张，训练集中的所有图像都有标签，即图像的文件名,文件名的形式为 xxx-##.jpg,其中 xxx 为图像的类别,”cat”或者”dog”，##为图像的编号,从 0 到 12500,图像按编号顺序排列;测试集有 12500 张图像，图像没有标签，没有顺序。训练集数据示例如图 2-1 所示，测试集数据示例如图 2-2 所示。

图 2-1 训练集数据示例

图 2-2 测试集数据示例

在将图像传输到模型中训练之前，需要先对数据进行预处理。因为原始数据输入一般都是以卷积神经网络模型难以表示的形式出现，首先图像应该被标准化，从而使得它们的像素都在相同并且合理的范围内，比如[0,1]或者[-1,1]。将[0,1]中的图像与[0,255]中的图像混合，通常会导致失败。将图像格式化为具有相同比例，是唯一一种必要的预处理。另外，许多模型需要标准尺寸的图像，因此必须裁剪或者缩放图像以适应该尺寸，有时候这种重新调整比例的操作不是必要的，一些卷积模型接受可变大小的输入，并动态地调整它们的池化区域大小以保持输出大小恒定，不过在该项目中，将会对所有图像进行缩放，以适应不同模型对输入的需求。

归一化处理和图像尺寸的调整是对训练集、测试集都需要操作的预处理方式，但还有一些操作是只需要对训练集进行的，该操作可以统称为数据集增强，数据集增强是减少大多数计算机视觉模型泛化误差的一种极好方法。在该项目中使用的数据集增强的方式是将图像进行水平或者竖直翻转，在经过多个 Epoch 迭代的

过程中，模型所接收的图像的状态是不一样的，因此隐性的增加了训练集的大小，进而改进分类器的泛化能力。

对于数据集，肯定会出现一些异常数据，这是不可避免的，在大量的数据面前，想要一一排查也是不可能的，所以在对正常数据做图像预处理之前，需要先熟悉数据集的构成和质量。因此在项目中做了对图像尺寸分布的可视化，通过可视化可以了解到数据的质量，然后通过分析图展示的内容来进行适当的删减。很显然，尺寸过小的图像在进行训练时，相比大尺寸图像其轮廓是不清晰的，在经过多层的网络，尤其是池化层后，其特征变得不可描述，从而会影响到分类器的判断，影响神经网络的拟合效果，造成过拟合等现象，如果这种数据大量出现，将对模型的效果有很大影响，如图 2-4 和图 2-5 所示，图像尺寸的分布中存在着一些尺寸过小和少量尺寸过大的图像，基于上面的分析，应该将这些图像进行删除。如图 2-6 所示，右侧的图像是一个异常数据，不仔细看，可能发现不了图中有一只猫，如果猫是前景，那该图像的背景太复杂，会影响训练的效果，除此之外，训练集中还有一些其他异常数据，完全不是猫也不是狗的图像，这类异常数据在训练集中大概有 45 张左右，通过人工审核的方法去寻找这些异常数据不太可能，所以是通过代码来寻找的，具体的实现方法在后面会讲到，可以先看一下找出的这 45 张图像，如图 2-7 所示，对于这些异常数据，我将他们和尺寸过小的数据一起直接删除掉了，不让其进入模型训练。

2.2 数据可视化

在建立模型之前，需要先对数据进行观察，了解数据的特点，掌握数据的结构，这样才能更好的利用数据，才能更有效的进行数据预处理。首先，要对数据集进行示例展示，分别从训练集和测试集中随机选取若干图像，将其可视化，并且将其标签，即文件名进行对应展示，如图 2-3 所示。

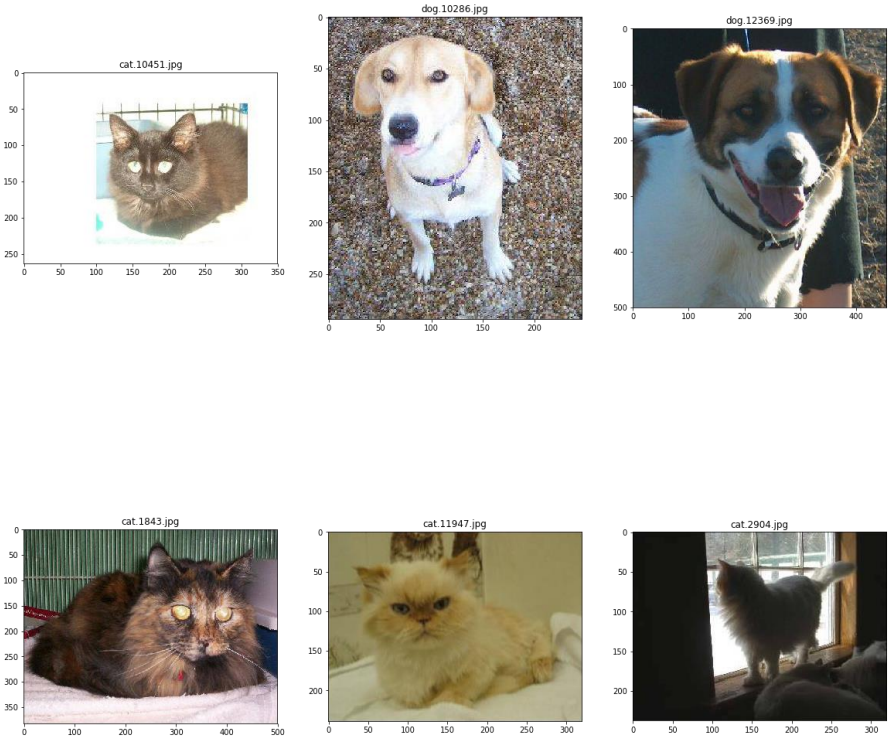

图 2-3 训练数据的可视化

了解数据集的基本结构之后，需要进一步了解数据集的特点。可以对数据集中图像大小的分布进行可视化，将每张图像的尺寸作为一个点，在坐标轴上进行尺寸分布的展示，了解图像的尺寸集中程度，掌握数据集的特点，其中猫数据集的尺寸分布如图 2-4 所示，狗数据集的尺寸分布如图 2-5 所示。

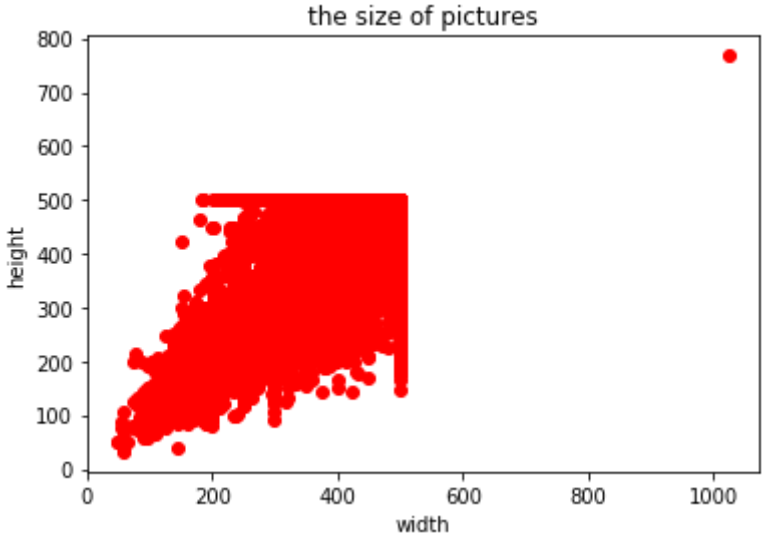

图 2-4 猫数据集的尺寸分布

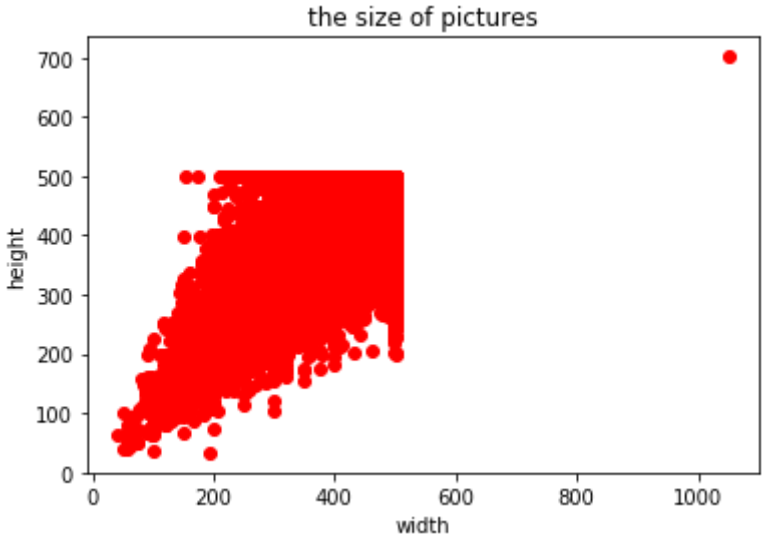

图 2-5 狗数据集的尺寸分布

对于训练集中的一些异常数据，如尺寸过小的数据和非猫非狗的数据进行可视化展示，图中展示了一张尺寸过小的图像和一张异常的图像，其效果如图 2-6 所示，对于所有的异常数据，这些数据在预处理时被认为既不是狗，也不是猫，一共有 45 张左右，在可视化过程中，我将这些图像全部展示了出来，如图 2-7 所示。

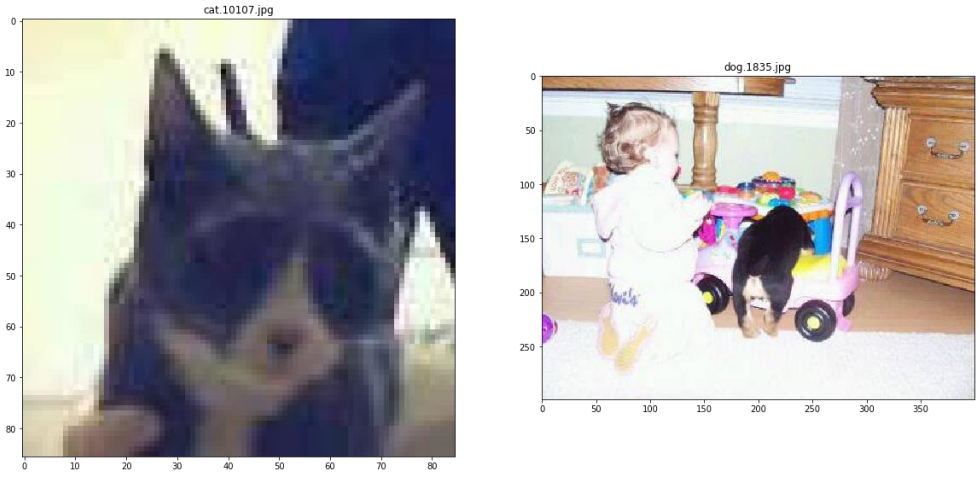

图 2-6 异常数据展示

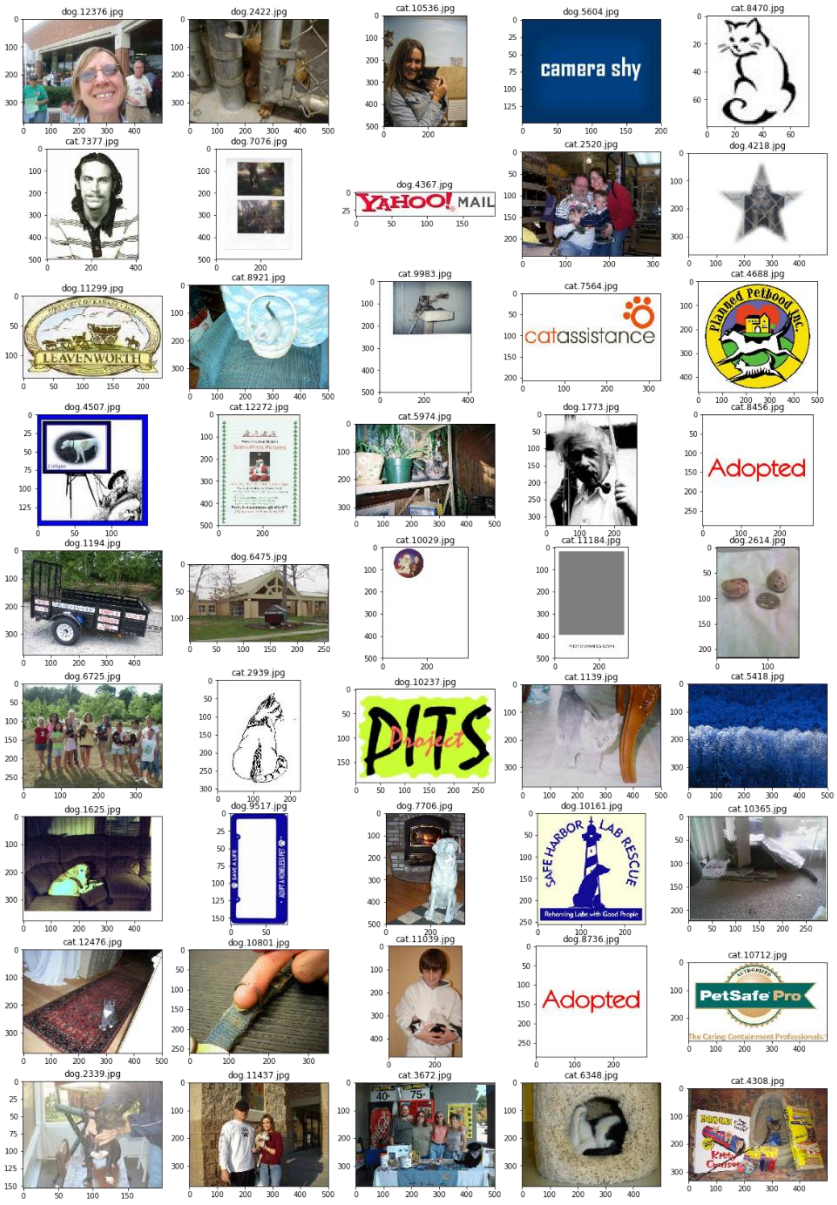

图 2-7 所有异常数据展示

2.3 算法和技术

该项目使用的算法是卷积神经网络，该网络是一种专门用来处理具有类似网络结构的数据的神经网络。由于 PCA+SVM 的方案在实践中被证明，其运行效率和最终效果比不上卷积神经网络，所以本项目中不在进行尝试。取而代之是其他三种尝试，这三种尝试是：1、自建 LeNet-5 模型进行图像的识别；2、使用迁移学习的思想，引用 Xception 模型，进行图像的识别；3、使用迁移学习和模型混合的方式，引用 Xception 、Inception V3、InceptionResNet V2 三个模型，对图像进行识别。

在通常形式中，卷积是对两个实变函数的一种数学运算，其基本形式如下：

$$y(n) = \sum_{i=-\infty}^{\infty} x(i)\,h(n-i) = x(n) * h(n)$$

2.3.1 卷积层

卷积运算用*号表示。通常，卷积被定义在满足上述积分式的任意函数上。

在卷积网络的理论中，卷积的第一个参数通常叫做输入，如上式中的函数 x，第二个参数叫做核函数,如上式中的函数 w,输出有时被称作特征映射。

卷积运算通过三个重要的思想来帮助改进机器学习系统:稀疏交互、参数共享、等变表示。

1、稀疏连接

传统的神经网络使用矩阵乘法来建立输入与输出的连接关系，这将使得每一个输出单元与每一个输入单元都产生连接。但是卷积神经网络具有稀疏连接的特征，这是使核的大小远小于输入的大小来达到的。这意味着需要存储的参数更少，不仅减少了模型的存储需求，还提高了它的统计效率，这会极大的减少计算量，显著的提高了效率。

2、参数共享

参数共享是指在一个模型的多个函数中使用相同的参数。在传统的神经网络中，当计算一层的输出时，权重矩阵的每一个元素只使用一次，当它乘以输入的一个元素后就再也不会用到了。在卷积神经网络中，核的每一个元素都作用在输入的每一位置上。卷积运算中的参数共享保证了只需要学习一个参数集合，而不是对于每一位置都需要学习一个单独的参数集合。因此，卷积网络在存储需求和统计效率方面极大的优于稠密矩阵的乘法运算。

3、等变表示

对于卷积网络，参数共享的特殊形式使得神经网络层具有对平移等变的性质。如果一个函数满足输入改变，输出也以同样的方式改变这一性质，就可以称其为等变的。对于图像而言，卷积产生了一个二维映射来表明某些特征在输入中出现的位置。如果移动输入中的对象，它的表示也会在输出中移动同样的量。当处理多个输入位置时，一些作用在邻近像素的函数是非常有用的。

2.3.2 池化层

卷积网络的一个经典层包含三级。第一级中，这一层并行地计算多个卷积产生一组线性激活响应。第二级中，每一个线性激活响应将会通过一个非线性的激活函数，例如整流线性激活函数(ReLU)。第三极中，使用池化函数来进一步调整这一层的输出。

池化函数使用某一位置的相邻输出的总体统计特征来代替网络在该位置的输出。如最大池化函数计算相邻矩形区域内的最大值并输出，其他常用的池化函数还有平均池化函数等。使用池化可以看作增加了一个无线强的先验：这一层学得的函数必须具有对少量平移的不变性。当这个假设成立时，池化可以极大的提高网络的统计效率。

2.3.3 项目使用的技术概述

如本章开篇所述，项目使用三种方法进行模型的建立。出于时间、操作性和展示性的考虑，在技术的使用上，将采用两种不同的深度学习开源框架，即 Tensorflow 和 Keras。

Tensorflow 是由 Jeff Dean 领头的谷歌大脑团队基于谷歌内部第一代深度学习系统 DistBelief 改进而来的通用计算框架。DistBelief 是谷歌 2011 年开发的内部深度学习工具，这个工具在谷歌内部已经获得巨大的成功。基于 DistBelief 的 ImageNet 图像分类系统 Inception 模型赢得了 ImageNet2014 年的比赛(ILSVRC)。虽然 DistBelief 已经被谷歌内部很多产品所使用，但是 DistBelief 过于依赖谷歌内部的系统架构，很难对外开源。为了将这样一个在谷歌内部已经获得了巨大成功的系统开源，谷歌大脑团队对 DistBelief 进行了改进，并于 2015 年 11 月正式公布了基于 Apache2.0 开源协议的计算框架 Tensorflow。相比 DistBelief，Tensorflow 的计算模型更加通用、计算速度更快、支持的计算平台更多、支持的深度学习算法更广泛而且系统的稳定性也更高。除了在谷歌内部大规模使用，Tensorflow 也受到了工业界和学术界的广泛关注。

Keras 是 Python 中一个以 CNTK、Tensorflow 或者 Theano 为计算后台的深度学习建模环境。相对于常见的几种深度学习计算软件，比如 Tensorflow、Theano、Caffe、CNTK、Torch 等，Keras 在实际应用中有显著的优点。如 Keras 在设计时以人为本，强调快速建模，用户能快速地将所需模型的结构映射到 Keras 代码中，尽可能减少编写代码的工作量，特别是对于成熟的模型类型，从而加快开发速度。另外，Keras 的特点是高度模块化的，用户几乎能够任意组合各个模块来构造所需的模型。在 Keras 中，任何神经网络模型都可以被描述为一个图模型或者序列模型，其中的部件被划分为以下模块：神经网络层、损失函数、激活函数、初始化方法、正则化方法、优化引擎。这些模块可以以任意合理的方式放入图模型或者序列模型中来构造所需的模型，用户并不需要知道每个模块后面的细节。这种方式相比其他软件需要用户编写大量代码或者使用特定语言来描述神经网络结构的方法效率要高得多，也不容易出错。

2.4 基准测试

输入数据的类型(图像)，是一种已知的拓扑结构，在这种前提下，使用卷积神经网络是最好的选择。具有衰减学习率以及动量的 SGD 是优化算法的一个合理的选择。因为数据集的数量没有达到千万级别，所以应该在一开始就包含一些温和的正则化。提前终止是常用的一种方案，Dropout 也是一个容易实现的，兼容性强并且出色的正则化项。参数共享是一个可以提高效率的有效方法，该方法可以显著的减小模型所占用的内存。此外，该模型应该具备基本的输入层、卷积层、最大池化层、扁平化层、全连接层和输出层。其中每一层之间的激活函数都应当选用修正线性单元(ReLU)。

在测试集上，对于单张图像的识别率应该在 98%以上，对于整体的识别正确率应该在 99%以上，按照 kaggle 网站对测试集上的 LogLoss 的要求要在排名前 10%,即 LogLoss 要小于 0.06，因此对于整体的 LogLoss 应该小于 0.06，这样的模型才是一个合格的模型。

3 方法

3.1 数据预处理

该数据集需要做的预处理并不多，主要是对图像的一些基本变换操作。在该模型中，主要做了如下预处理。

1、图像的大小调节，由于大部分的图像的大小是不固定的，但很多的神经网络模型的输入节点的个数是固定的，在本次项目中使用的三个模型，Xception、InceptionV3 和 InceptionResNetV2 都是 Inception 系列的模型框架，接收的都是(299,299,3)的图像输入结构。因此，在数据预处理中，需要将图像的大小调整为上述的图像大小。

2、图像的旋转和翻转，在很多识别问题中，图像的翻转不会影响识别的结果。于是在训练图像识别的神经网络模型时，可以随机地翻转训练图像，这样训练得到的模型可以识别不同角度的实体。比如假设在训练数据中所有的猫头都是向右的，那么训练出来的模型就无法很好的识别猫头向左的猫。虽然这个问题可以通过收集更多的训练数据来解决，但是通过随机翻转训练图像的方式，可以在零成本的情况下很大程度的缓解该问题。在该项目中，对图像进行了随机的水平翻转和竖直翻转。

3、图像色彩的调整，和图像翻转类似，调整图像的亮度、对比度、饱和度和色相在很多图像识别应用中都不会影响识别结果。所以在训练神经网络模型时，可以随机调整训练图像的这些属性，从而使训练得到的模型尽可能小的收到无关因素的影响。

4、异常数据的处理，数据集的异常数据不可避免，但也要通过一定方法来抑制异常数据的出现。在该项目中的异常数据主要是尺寸过小的图像，因此在可视化图像的尺寸分布后，可以看出尺寸过小的图像的一个分布情况，然后对这部分的图像进行删除。除了尺寸过小的数据外，还有少量既不是猫也不是狗的图像，对

于这些异常数据，应该和尺寸过小的图像一样，进行删除处理。在 2 万多张的图像中找出少量的异常数据是不容易的，因此在寻找了一些方法后，决定采用 keras 的预处理模型来解决这个问题，在 keras 中由一些已经调试好的分类器，并且提供了模型和权重文件，在 ImageNet 上，这些模型对 1000 个物品进行分类，其中狗有 118 个子分类，猫有 7 个子分类，最初，采用了 ImageNet 的评价指标 Top-1，来这次项目的训练集进行分类，试图将训练集中的非猫非狗的图像找出，然而发现找出的图像有 3000 多张，然后对这些进行随机的可视化，发现其中有很多猫狗图像被误判，因此提高评价标准至 Top-5，此时，检测出异常数据有 500 多，依然误判很多，此后，尝试了 Top-10、Top-20、Top-30、Top-40 和 Top-50 逐渐提高评价指标，Top-40 有 55 张，Top-50 有 45 张，因为数量较小，就将他们全部展示出来，发现 Top-40 有一两张误判的情况，已经很少了，不过最终时采用了 Top-50 的指标。在发现了这些数据后，将他们和尺寸过小的图像一起，进行了删除，详细的异常数据检测过程可以参见代码 DataPre.ipynb。

5、数据的批量输出，将数据分成若干个 batch 是必要的，面对巨大的数据集，如果同时放入模型训练，这将是灾难性的，没有多少硬件能够承受这么的压力，对内存来说，是极大的考验，因此，将数据分批次输入模型就显得非常有效；在合理范围内，增大 batch size 可以提高内存的利用率，矩阵乘法的并行化效率提高，跑完一次 epoch 所需的迭代次数减少，对于相同数据量的处理速度进一步加快，在一定范围内，一般来说 batch size 越大，其确定的下降方向越准确，引起的训练震荡越小。因为没有使用 AWS 的 GPU 服务器进行训练，所以本机的内存没有那么大，对 batch size 的设定就会被限制很多，在尝试了 batch size 为 64 后，出现错误内存不足，设置为 32 后，依旧是内存不足，设置为 16 后，程序不再因为内存不足而崩溃，结合上面所说的适当提高 batch size 带来的好处，因此选用 16 作为最终的 batch size。

3.2 实现

项目最初使用的是 Tensorflow+LeNet5 模型的方案，通过 Tensorflow 来自己搭建 LeNet5 的神经网络模型，然后通过该模型来对数据集进行训练和预测。LeNet5 模型最早在 1994 年由 Yann LeCun 提出，该模型充分考虑了图像的相关性，是较早出现的神经网络模型之一。当时的结构有如下特点：

1、每个卷积层包含三个部分：卷积(Conv)、池化(pooling)、非线性激活函数(sigmoid)。

2、MLP 作为最终的分类器。

3、层与层之间稀疏连接减少计算复杂度。

它的结构模型如图 3-1 所示。

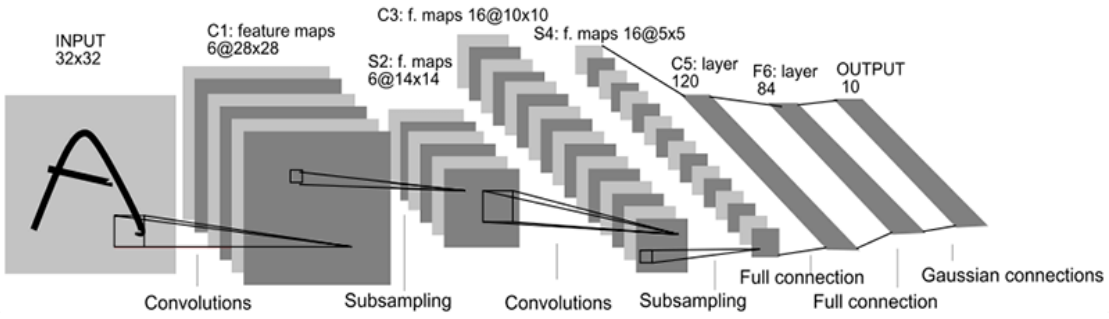

图 3-1 LeNet5 模型结构图

观察 LeNet5 模型的网络结构可知，网络的输入是 32×32 的单通道图像，C1 层为卷积层，共包含 6 个特征图。因为卷积核尺寸为 5×5，步长为 1，padding 方式为 VALID，所以特征图尺寸为 28×28。S2 层为下采样层(也叫池化层)。每个单元连接与 C1 对应的特征图上 2×2 的区域，这四个输入相加后乘以一个可训练的系数,然后加上可训练的偏置后,应用 Sigmoid 函数。S2 的特征图尺寸为 14×14。C3 为卷积层，共 16 个特征图。因为卷积核尺寸为 5×5，步长为 1，padding 方式为 VALID，所以特征图尺寸为 10×10.C3 中的每个单元连接到 S2 特征子集的 5×5 区域。S4 是下采样层，每个单元连接与 C3 对应的特征图上 2×2 的区域，这四个输入相加后乘以一个可训练的系数,然后加上一个可训练的偏置后应用 sigmoid 函数。2×2 的感受野不重叠，所以 S4 的特征图尺寸为 14×14。C5 层是卷积层，共 120 个特征图。每个单元连接到 S4 所有特征图上的一个 5×5 区域。因为卷积核尺寸为 5×5，步长为 1，padding 方式为 VALID，所以特征图尺寸为 1×1。

F6 是全连接层，共包含 84 个神经元。该层的输入与权重做点积，然后加上偏置后应用 sigmoid 进行压缩。

输出层包含 10 个神经元，对应于 10 种分类结果。该层由欧几里得径向基函数(RBF)组成，每个 RBF 单元的输出由以下公式得出：

$$y_i = \sum_j (x_i - w_{ij})^2$$

在使用 Tensorflow 对 LeNet5 模型进行构建的时候,对原始的模型进行了一定的调整，如：所有的卷积层都连接于前一层的所有特征图；所有的下采样层都使用 maxpooling 代替；使用 ReLU 替换 sigmoid 激活函数；RBF 输出使用 softmax 函数替换。由此通过 Tensorflow 实现了 LeNet5 模型，并对数据进行了训练和测试。

InceptionV3 模型的结构图如图 3-2 所示。

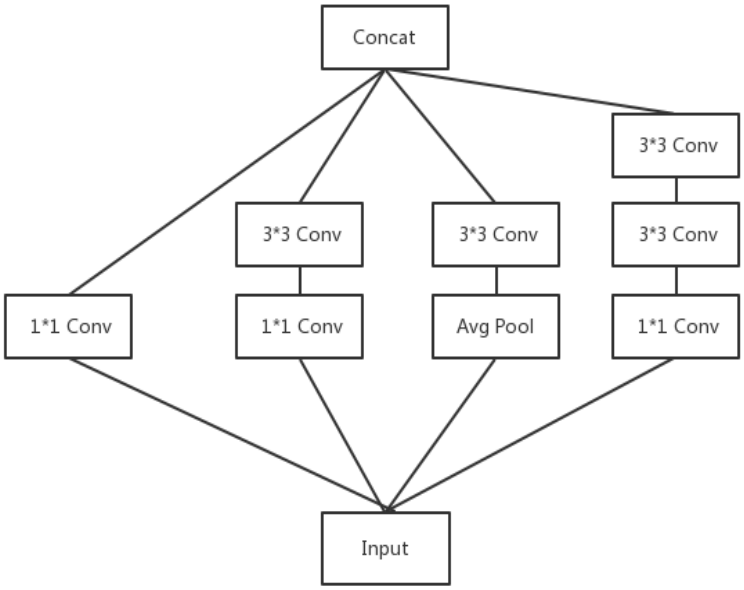

图 3-2 InceptionV3 结构图

Inception 结构最初提出是认为，特征的提取和传递可以通过 1×1 卷积，3×3 卷积，5×5 卷积，pooling 等，Inception 结构将这一个输入同时输给上面这几种提取特征方式，然后 concat。Inception 的作用就是代替人工确定卷积层中的过滤器类型或者确定是否需要创建卷积层和池化层。该架构的主要思想是找出如何用密集成分来近似最优的局部稀疏结。采用不同大小的卷积核意味着不同大小的感受野，最后拼接意味着不同尺度特征的融合。网络越到后面特征月抽象，且每个特征涉及的感受野也更大，随着层数的增加，3×3 和 5×5 卷积的比例也要增加。InceptionV3 网络引入了 Factorization into small convolutions 的思想，将一个较大的二维卷积拆成两个较小的一维卷积，比如将 7×7 卷积拆成 1×7 卷积和 7×1 卷积。这种思路一方面节约了大量参数，加速运算并减轻了过拟合，同时增加了一层非线性扩展模型表达能力。这种非对称的卷积结构拆分，其结果比对称的拆为几个相同的小卷积核效果更明显，可以处理更多、更丰富的空间特征，增加特征多样性。

Xception 中主要采用了 depthwise separable convolution.该方法将传统的卷积操作分成两步，假设原来是 3×3 的卷积，那么 depthwise separable convolution 就是先用 M 个 3×3 的卷积核一对一卷积输入的 M 个 feature map，不求和，生成 M 个结果；然后用 N 个 1×1 的卷积核正常卷积前面生成的 M 个结果，求和，最后生成 N 个结果。Xception 的结构图如图 3-3 所示。

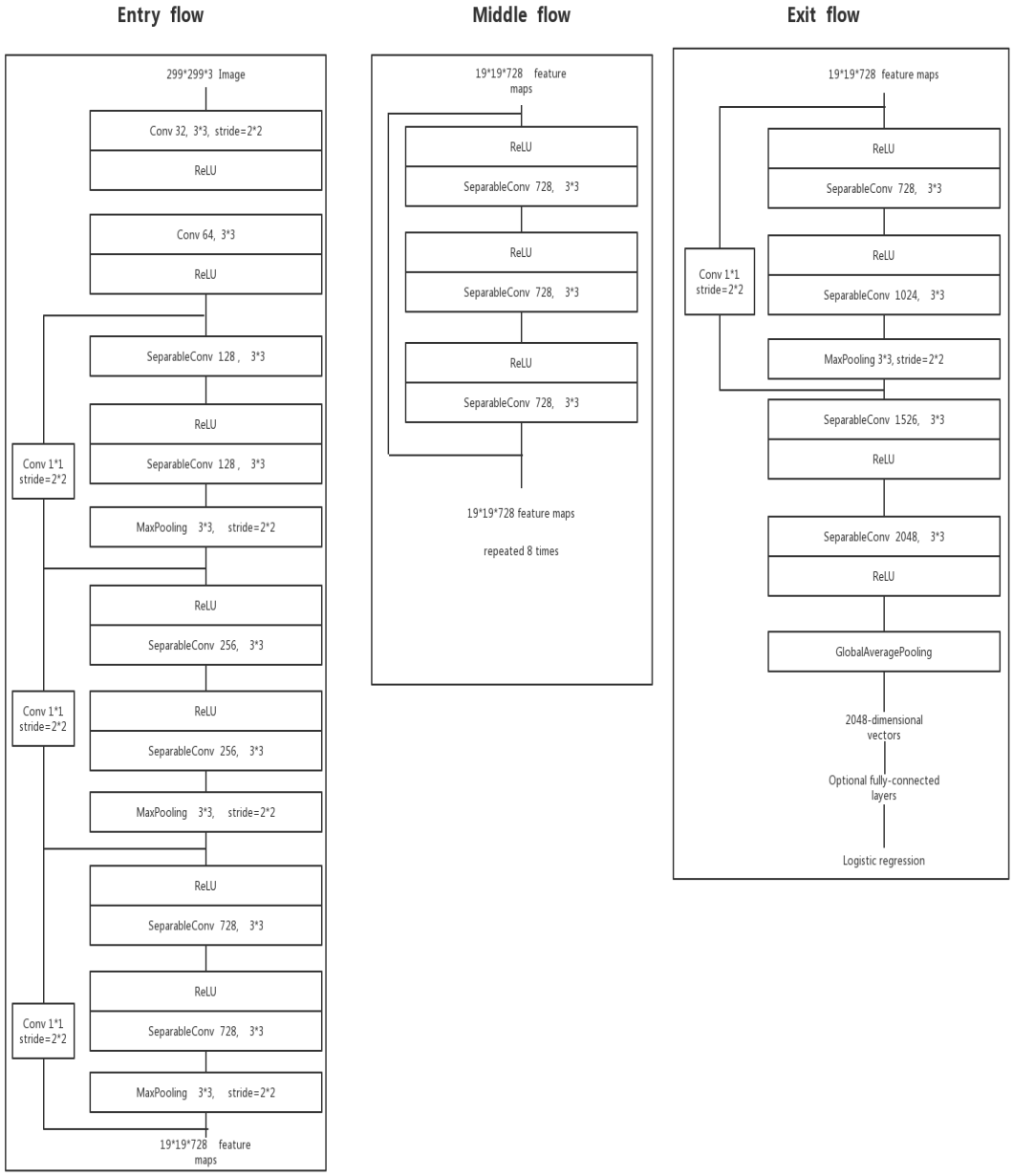

图 3-3 Xception 网络结构

另外，Xception 中每个小块的连接采用的是 residule connect，而不是原 Inception 中的 concat。

InceptionResNetV2 是从 InceptionV3 模型变化而来，该模型从微软的残差网络(ResNet)论文中得到了一些灵感。残差连接(Residual connections)允许模型中存在 shortcuts，可以让研究者成功地训练更深的神经网络，这样也能明显的简化 Inception 块。InceptionResNetV2 的模型结构如图 3-4 所示。

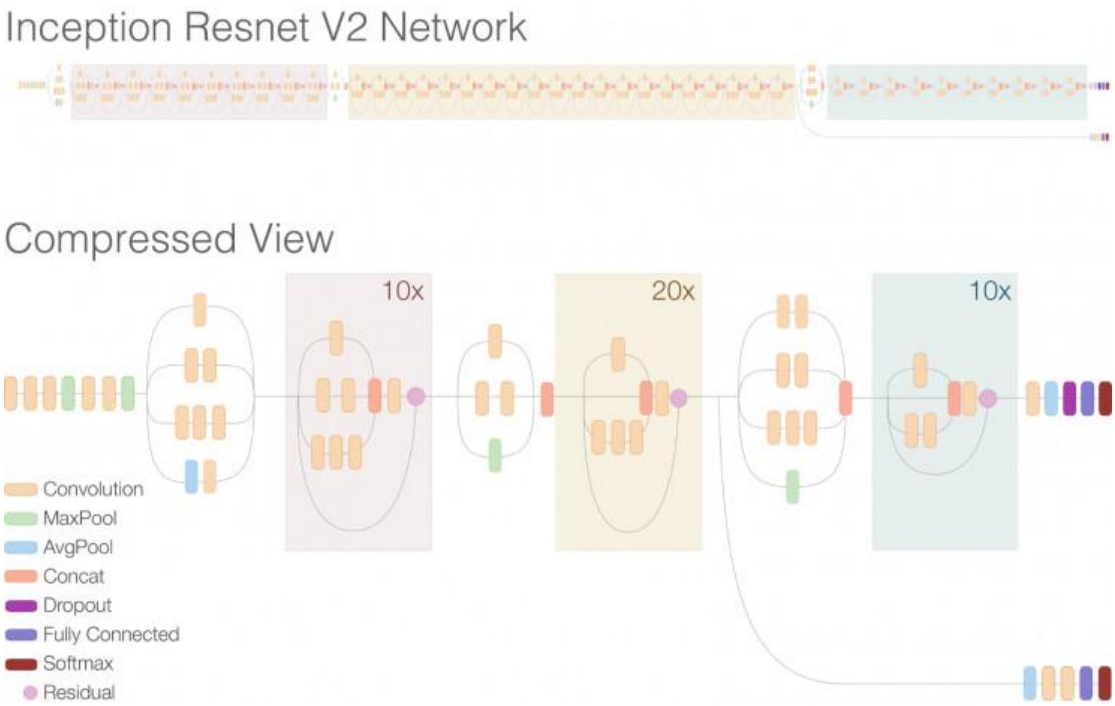

图 3-4 InceptionResNetV2 网络结构

该网络比先前的 InceptionV3 还要深一些。主要 Inception 模块的池化运算由残差连接替代。InceptionResNetV2 的精度要比之前的最优模型更高。

在使用 keras 实现三个模型的混合时，先将数据通过生成器进行预处理，并进行分 batch 输出，以防止内存不足而导致训练失败。然后数据将分别进入三个网络模型中训练，将三个网络模型中提取出来的特征分别存入 h5 文件中，这样可以方便以后使用时快速的导入特征，也可以防止训练中断导致重新训练，可以极大地提高效率，节省训练时间。最后，将 h5 文件中的特征提取出来，拼接在一起后，放入 sigmoid 函数中进行分类，这个时候，可以适当的对比结果，在 sigmoid 函数分类前，加入全连接层之类的有利于提高模型精准度的操作。

3.3 改进

在项目的推进过程中，多次改进了算法，最初的方案是使用 Tensorflow+LeNet5 的方法，通过自建模型，来实现训练和预测，但是当模型搭建好训练完成后，对测试集进行预测时，发现模型的最高 LogLoss 只能达到 0.08 左右，无法满足最初预定的基准模型和评价指标。即使在多次调整超参数之后，模型依旧无法达到预定的标准，在分析原因之后，认为可能是模型的复杂度不够，导致最终的效果不理想。

在分析完结果后，迅速调整方法，放弃使用自建 LeNet5 模型的方法，转而使用多模型融合的方法，但是由于时间原因，使用 Tensorflow 来实现的话可能需要的时间比较多，所以选择使用更易于实施的工具 keras。在构建完整体的结构后，开始进行代码的实施，将数据放入模型进行训练，最终的测试结果很理想，整体的 LogLoss 达到了 0.03 左右，不仅可以达到预期的评价指标，还可以超出

指标很多。

所以，选用了 Keras+混合模型的方案作为了最终的实现方案。

4 结果

在对混合模型进行超参数调节的时候，主要调整了优化器的类型和学习率。最初的学习率调的比较高，在 0.01 附近，此时模型在 10 个 Epoch 的迭代后，并不能达到很好的 LogLoss，大概在 0.1 左右，此时调用写好的 Loss 可视化模块来查看 10 次 Epoch 中 Loss 的变化，发现验证集上，val_loss 出现震荡，推断可能是学习率太高所致，由此降低学习率至 0.001，再次进行训练，发现此时的 LogLoss 有明显的下降，效果要好于原来学习率为 0.01 时，再次观察 Loss 的下降曲线，发现在前期 Loss 的下降是正常的，但是再后期的迭代中，Loss 还是出现了轻微的震荡，在最终的多次尝试后，将学习率定在了 0.0002 左右时，验证集上的效果比较理想。

优化器的选择，也做了多次的尝试，使用过 Adam 优化器、Adadelta 优化器、Nadam 优化器、SGD 优化器等，发现对最终效果的影响并不是很大，相对于学习率的影响，优化器的选型对结果影响不大，因此选择了效果相对要好一些的 SGD 优化器，随机梯度下降算法(SGD)每次只随机选择一个样本来更新模型参数，因此每次的学习是非常快速的，并且可以进行在线更新，但是 SGD 的最大缺点在于每次更新可能并不会按照正确的方向进行，因此会带来优化波动，但是这些波动并不全是缺点。

在测试集上，按照 kaggle 网站对测试集上的 LogLoss 的要求要在排名前 10%，即 LogLoss 要小于 0.06，因此对于整体的 LogLoss 应该小于 0.06。

在项目中，在测试集上，最初方案使用 Tensorflow 和 LeNet5，能达到的最好的 LogLoss 是 0.08，对比基准测试的阈值，该方案的单张图像的识别正确率能达到 98%以上，整体的识别率可以达到 97%，LogLoss 最低可以达到 0.08，整体达不到基准测试结果的要求。最终方案使用 Keras 和混合模型，对比基准测试的阈值，该方案的单张图像的识别正确率能达到 98%，整体识别率能达到 99%，LogLoss 最低可以达到 0.03 左右，达到了基准测试的结果，满足评价指标的要求，该结果在 kaggle 网站上的成绩如图 4-1 所示。

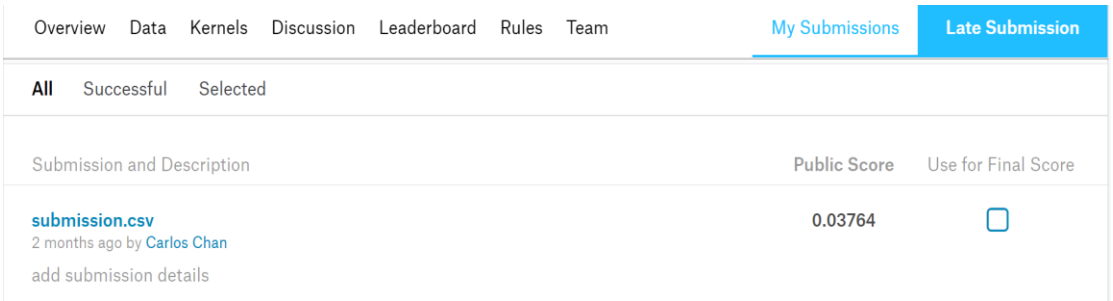

图 4-1 kaggle 成绩截图

5 结论

5.1 数据的可视化

对混合模型进行代码实现过程中，编写了两个结果可视化的模块，结果可视化模块和图像可视化模块，结果模块用于可视化训练模型过程中的 Accuracy 在迭代时的变化趋势，再将数据可视化以后，可以方便容易的分析出模型的好坏，有益于对超参数的调节。具体的可视化坐标图如图 5-1 和图 5-2 所示。

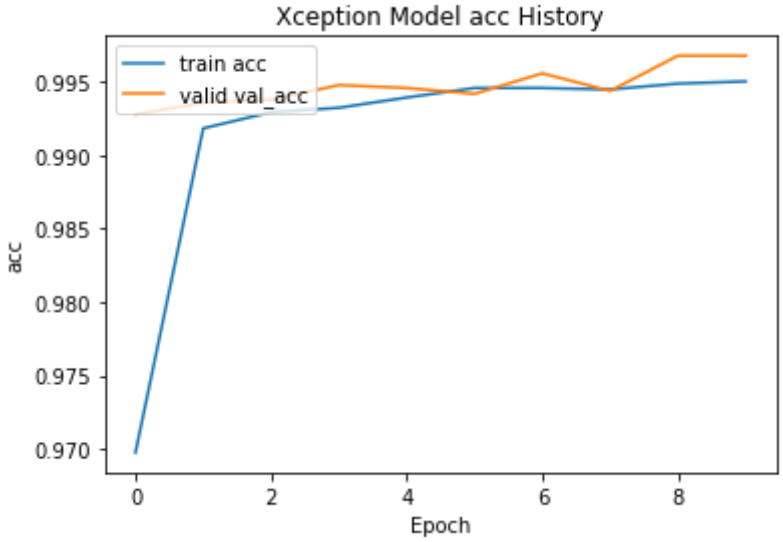

图 5-1 混合模型的 Accuracy 可视化

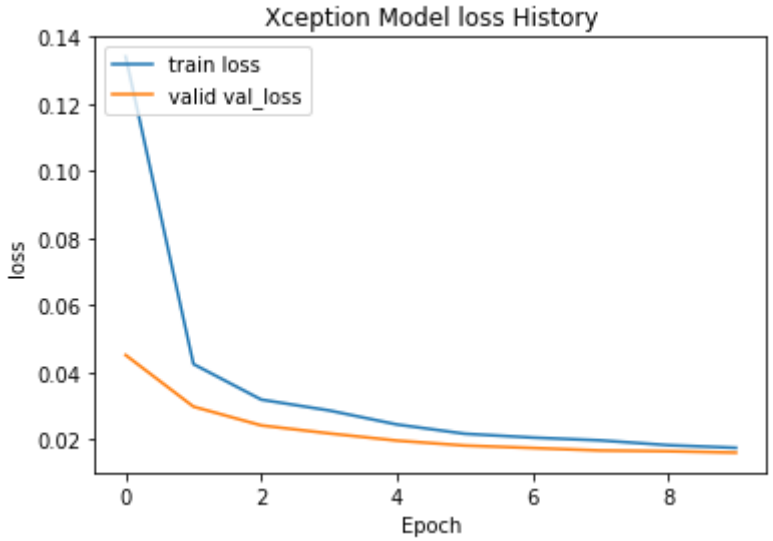

图 5-2 混合模型的 Loss 可视化

图像可视化模块是用来检测测试结果的，该模块可以随机输出测试图像，并在图像的上方显示该图像的类别和准确率。使用这个模块可以清晰易懂的表现出模型

的实际效果，让即使不懂代码的人也能对模型结果一目了然。其展示效果如图 5-3 所示。

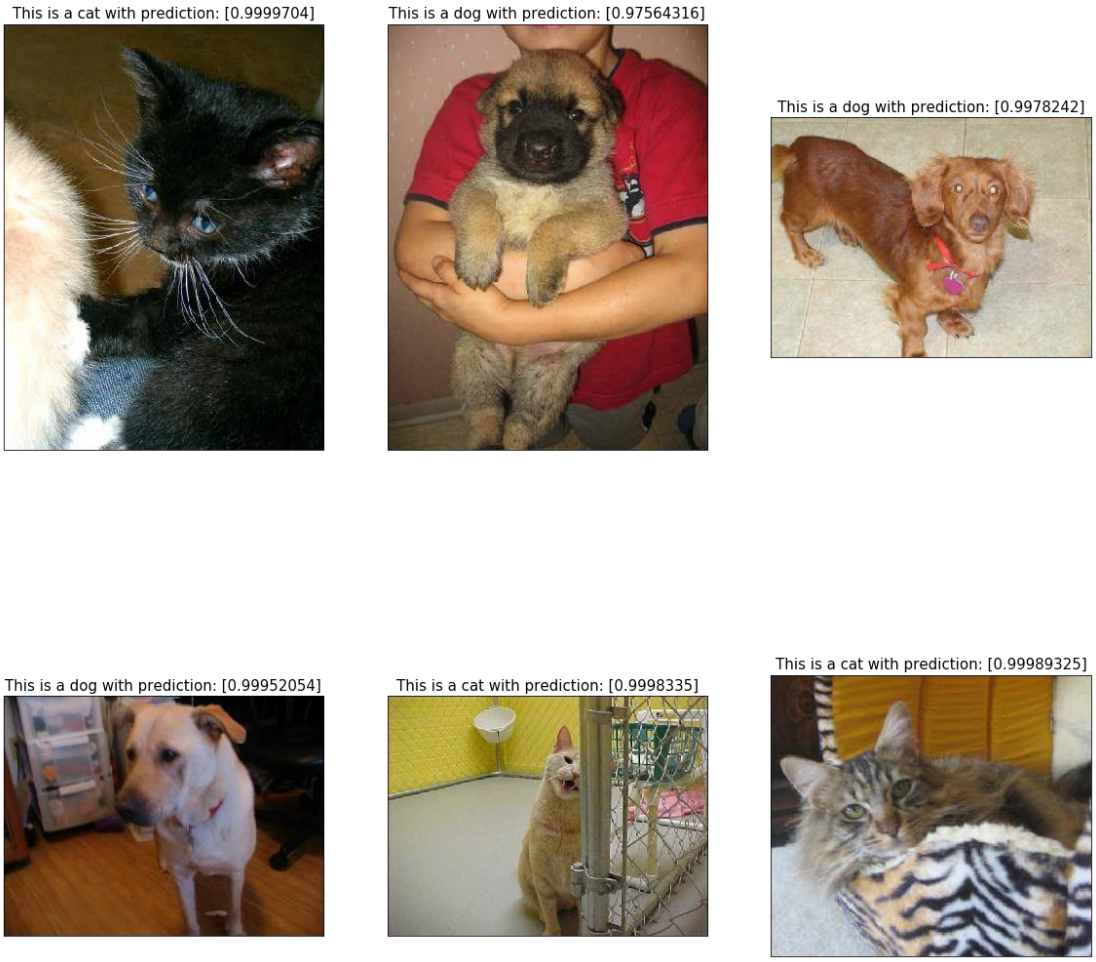

图 5-3 图像可视化模块效果图

5.2 对项目的一些思考

一个端到端问题的解决，首先要考虑问题是属于分类问题还是回归问题，是图像识别问题还是目标检测问题，在给问题定性之后，才能准确精准的选择解决问题的方案，在选择方案后，就是对方案的实施，对比，优胜劣汰，选区最优的方案，然后对方案进行检测，确定最终方案，并将最终方案调试到最优，这样才是一个完整的解决方案。

该项目的难点在于超高的评价指标，如果一个普通的机器学习或者深度学习的项目，能够达到一定的识别率就可以，那这个项目的挑战和难度都会大大降低，但是现实并不是这样，该项目的评价指标非常的高，这也是优达学城高标准所能带来的高品质。因此要实现这么高的指标，就不能使用普通的思路和方法，需要使用大部分人没有使用过的方法，这也是最初方案失败的原因。这个项目有趣的地方在于方法的无限性，无论使用什么方法，都可以去尝试，可以使用迁移学习，可以尝试自建模型，也可以尝试使用特征提取和支持向量机的组合，这让项目的乐趣大大提高，每一种方法的实现都是对个人能力的极大提高。

5.3 还需要的改进

虽然最终方案的模型可以达到最终的评价指标，但是对于模型的内部结构和原理的理解可能并不够深入，对于模型实施的结果是可预见的，但对于模型实施的过程无法理解。这也许需要使用特殊手段来进入模型训练的内部，可视化其提取的特征，可视化其模型内部的数据，而这些可能需要更多的精力和时间，项目的时间有限，在项目中可能无法更好的实现，但是项目外，这是可以探索的一个很好的方向。在可视化这些内容后，其带来的改进和提升肯定是显而易见的，对模型的观察可以更深入，对数据的了解更彻底，这样无论使调试、调参还是对模型的把握上，都可以更加容易掌握。