猫狗大战

机器学习纳米学位

谭维

2017年12月29日

I. 问题的定义

项目概述

项目需解决一个计算机视觉领域中的图像分类问题。通过对神经网络模型的训练，使得模型能够识别图片是猫还是狗。

项目使用的模型是 Google Inception Net 中 2015 年 12 月提出的 Inception V3 来进行这次猫狗大战项目。此模型在 ILSVRC（ImageNet Large Scale Visual Recognition）中的 top-5 错误率为 3.5%，超过人工标注的错误率（5%）。

项目选择数据集来自于 kaggle 上猫狗大战比赛的数据集，数据集分为训练数据集和测试数据集。训练集中有猫狗图片各 12500 张，共 25000 用来训练模型。测试数据集中有猫狗图片共 12500 用来测试模型的预测准确率。

问题陈述

这是一个在 kaggle 上的图像分类比赛，通过对大量猫、狗图片，训练一个神经网络模型来识别测试数据中的每一张图片是猫还是狗，通过对测试数据中图像分类的错误率来判断神经网络模型的训练效果。

评价指标

评估指标使用 kaggle 的评估标准：LogLoss

其公式为：

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

其中 n 为测试集中图片数量，如果图像为狗 y_i 为 1，否则为 0。$\hat{y}_i$ 为模型预测值，如果是狗为 1，是猫为 0。Log() 表示自然对数 Ln()。

II. 分析

数据的探索

从 kaggle 上下载训练数据和测试数据。训练数据中有猫狗图片各 12500 张，命名格式为 feature.num.jpg 。可以通过命名特点将图片进行分类，划分对应的标签。将其中20%的数据做验证集，来验证模型的训练准确率。测试数据中有猫狗照片共 12500 张，当模型训练好时，将测试数据的图像划分类别通过 csv 格式提交至 kaggle 获得 Log Loss 分数。

图 1：猫狗图片样本

从图 1 的样本图片中很明显能够分辨出宠物类别，而且图片尺寸不一致，需要将图片尺寸统一。通过对第一版模型对训练数据进行预测，将其中分类错误明显的图片进行人工检查，并没有发现标签标注错误的图片。

算法和技术

Google Inception Net 首次出现在 ILSVRC 2014 的比赛中，就以较大优势取得了第一名。那届比赛中的 Inception Net 通常被称为 Inception V1，它最大的特点是控制了计算量和参数量的同时，获得了非常好的分类性能——top-5 错误率 6.67%。Inception V1 降低参数量的目的有两点，第一、参数越多模型越庞大，需要供模型学习的数据量就越大，而目前高质量的数据非常昂贵；第二、参数越多，耗费的计算资源也会更大。Inception V1 参数少但效果好的原因除了模型层数更深、表达能力更强外，还有两点：一是去除了最后的全连接层，用全局平均池化来取代它。二是 Inception V1 中精心设计的 Inception Module 提高了参数的利用效率，其结构如图 2 所示。

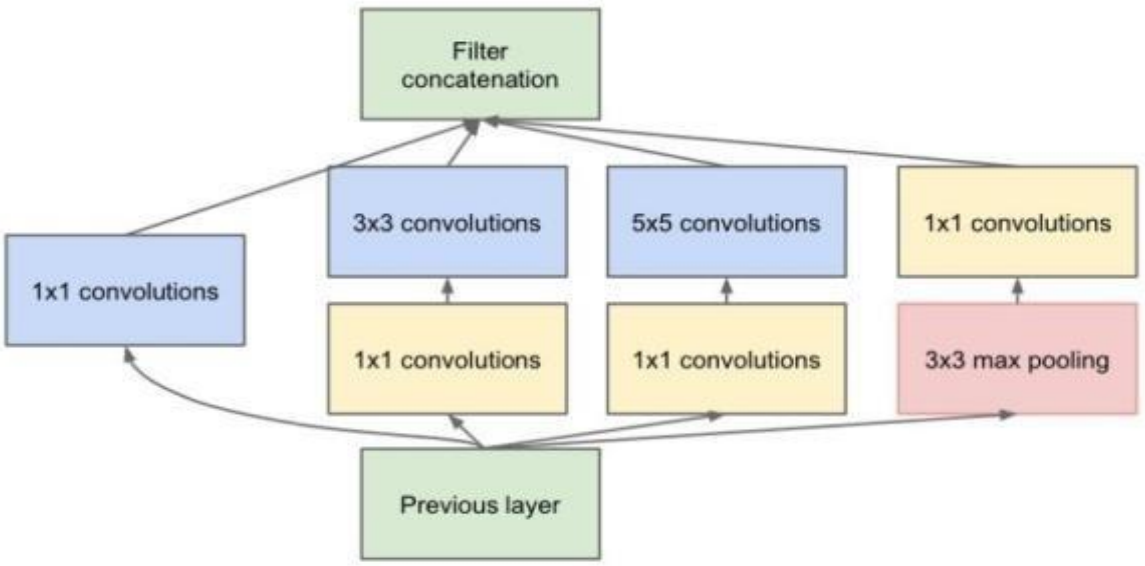

图 2：Inception Moduel 结构图

Inception Module 借鉴了 *Network In Network（以下简称NIN）* 论文的思想，形象的解释就是 Inception Module 本身如同大网络中的一个小网络，其结构可以反复堆叠在一起形成大网络。不过 Inception V1 比 NIN 更进一步增加了分支网络。

在看下 Inception Module 的基本结构，其中有 4 个分支：第一个分支对输入进行 1x1 的卷积，这其实也是 NIN 中提出的一个重要结构。1x1 的卷积是一个非常优秀的结构，它可以跨通道组织信息，提高网络的表达能力，同时可以对输出通道升维和降维。可以看到 Inception Module 的 4 个分支都用到了 1x1 卷积，来进行低成本（计算量比 3x3 小很多）的跨通道的特征变换。第二个分支先使用了 1x1 卷积，然后连接 3x3卷积，相当于进行了两次特征变换。第三个分支类似，先是 1x1 的卷积，然后连接 5x5 卷积。最后一个分支则是 3x3 最大池化后直接使用 1x1 卷积，这是因为 1x1 卷积的性价比很高，用很小的计算量就能增加一层特征变换和非线性化。Inception Module 的 4 个分支在最后通过一个聚合操作合并（在输出通道数这个维度上聚合）。Inception Module 中包含了 3 种不同尺寸的卷积和 1 个最大池化，增加了网络对不同尺度的适应性。

Inception V2 学习了 VGGNet，用两个 3x3 的卷积代替 5x5 的大卷积（用以降低参数量并减轻过拟合），还提出了著名的 Batch Normalization （以下简称BN）方法。BN是一个非常有效的正则化方法，可以让大型卷积网络的训练速度加快很多倍，同时收敛后的分类准确率也可以得到大幅提高。BN在用于神经网络某层时，会对每一个 mini-batch 数据的内部进行标准化处理，使输出规范化到 N(0,1) 的正态分布，减少

了 Internal Variance Shift（内部神经元分布的改变）。

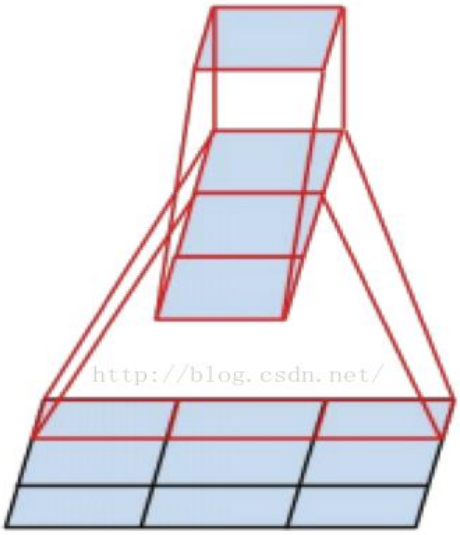

图 3：将一个 3x3 卷积拆成 1x3 卷积和 3x1 卷积

而 Inception V3 网络则主要有两方面的改造：一是引入了 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 7x7 卷积拆成 1x7 卷积和 7x1 卷积，或者将 3x3 卷积拆成 1x3 卷积和 3x1 卷积，如图 3 所示。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的表达能力。

另一方面，Inception V3 优化了 Inception Module 的结构，现在 Inception Module 有 35x35、17x17 和 8x8 三种不同结构，如图 4 所示。这些 Inception Module 只在网络的后部出现，前部还是普通的卷积层。并且 Inception V3 除了在 Inception Module 中使用分支，还在分支中使用了分支，可以说是 Network In

Network In Network。

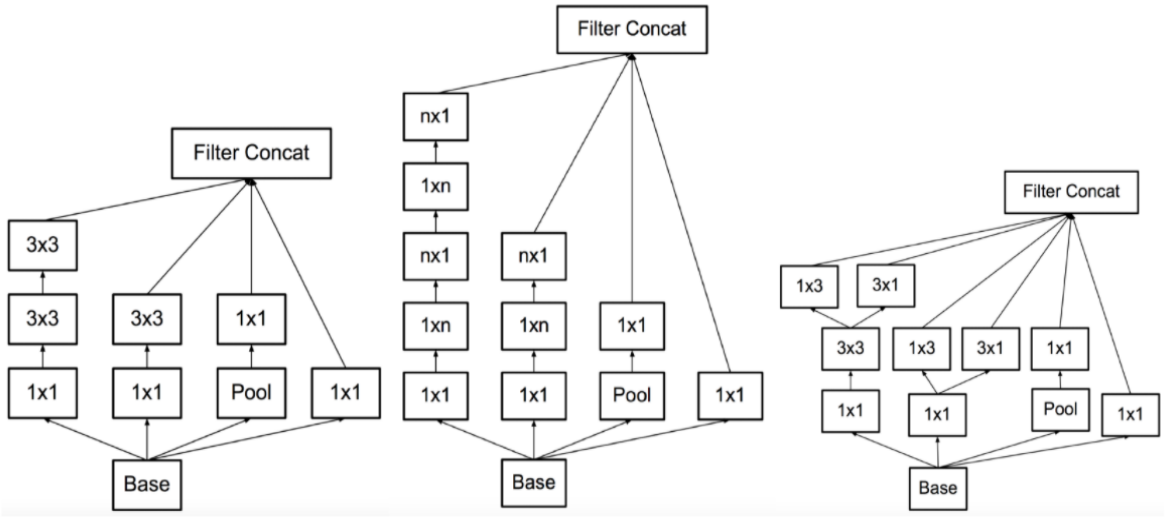

图 4：Inception V3 中的三种结构的 Inception Module

基准模型

在 kaggle 上有很多根据模型训练后提交上去的 Log Loss 分数，我的基准模型为排行榜上第 100 名 Karthik Kannan，他的 Log Loss 分数为0.05629。

III. 方法

数据预处理

项目通过使用 keras 库中的图片预处理来处理图片，先将图片按照命名特征将图片分类到不同文件夹下。再通过图片生成器（ImageDataGenerator）下的 flow_from_directory 方法读取文件夹下的图片，并将文件大小调整至 Inception V3 的输入尺寸（299，299，3），然后在模型第一层中使用 inception_v3 中的输入预处理（preprocess_input）方法将图片中 RGB 值由 [0,255] 归一化至 [-1,1]。

项目在通过一个训练后的模型来预测训练数据中的图片，将分类错误明显的进行人为检查，看数据集中是否由标签错误的图片需要进行清洗。

执行过程

类型	kernel 尺寸/步长（或注释）	输入尺寸
卷积	3*3 / 2	299*299*3
卷积	3*3 / 1	149*149*32
卷积	3*3 / 1	147*147*32
池化	3*3 / 2	147*147*64
卷积	3*3 / 1	73*73*64
卷积	3*3 / 2	71*71*80
卷积	3*3 / 1	35*35*192
Inception 模块组	3 个 Inception Module	35*35*288
Inception 模块组	5 个 Inception Module	17*17*768
Inception 模块组	3 个 Inception Module	8*8*1280
池化	8*8	8*8*2048
线性	logits	1*1*2048
Softmax	分类输出	1*1*1000

图 5：Inception V3 网络结构

如图 5 所示，项目使用的 Inception V3 模型在前面使用了几个普通的非 INception Module 卷积层。网络的输入数据尺寸为 299x299x3，在经历 3 个步长为 2 的层之后，尺寸最后缩小为 35x35x192，空间尺寸大大降低，但是输出通道增加了很

多。这部分一共有 5 个卷积层，2 个池化层，实现了对输入图片数据的尺寸压缩，并对图片特征进行了抽象。

接下来就将是三个连续的 Inception 模组块，这三个模组块中各自分别有多个 Inception Module，每个 Inception 模组块内部的几个 Inception Module 结构非常类似，但是存在一些细节不同。

第 1 个 Inception 模组块包含了 3 个结构类似的 Inception Module，它们的结构和图 4 中的第一幅图非常相似，其中第 1 个 Inception Module 的名称为Mixed_5b。这个 Inception Module 中有 4 个分支，从 Branch_0 到 Branch_3 第一个分支为 64 输出通道的 1x1 卷积；第 2 个分支为有 48 输出通道的 1x1 卷积，连接有 64 输出通道的 5x5 卷积；第 3 个分支为有 64输出通道的 1x1 卷积，连接有 96输出通道的 3x3 卷积；第 4 个分支为 3x3 的平均池化，连接有 32 输出通道的 1x1 卷积。最后，使用 tf.concat 将 4 个分支的输出合并在一起，生成这个 Inception Module 的最终输出的 tensor，尺寸为 35x354x256。

第 1 个 Inception 模组块的第 2、3 个Inception Module——Mixed_5c、Mixed_5d 与 Mixed_5b 非常相似，唯一的不同就是第 4 个分支最后接的是 64 输出通道的 1x1 卷积，而 Mixed_5b 是32输出通道，比之前增加了 32。因此第一个 Inception Module 模组块最后的输出尺寸为 35x35x288。

第 2 个 Inception 模组块是一个非常大的模组块，包含了 5 个 Inception Module，其中第 2 个到第 5 个 Inception Module 的结构非常相似，它们的结构如图 4 中的第二幅图所示。其中第 1 个 Inception Module 名称为 Mixed_6a，它包含 3 个分支。第一个分支是一个 384 输出通道的 3x3 卷积，这个分支的通道数以下就超过了之前的通道数之和。不过步长为 2，因此图片尺寸将会被压缩，且 padding 模式为

VALID，所以图片尺寸缩小为 17x17；第 2 个分支有三层，分别是一个 64 输出通道的 1x1 卷积和两个 96 输出通道的 3x3 卷积。最后一层的步长为 2，padding 模式为 VALID，因此图片尺寸也被压缩，本分支最终输出的 tensor 尺寸为 17x17x96；第 3 个分支是一个 3x3 最大池化层，步长同样为 2，padding 模式为 VALID，因此输出的 tensor 尺寸为 17x17x256。最后依然是使用 tf.concat 将三个分支在输出通道上合并，最后的输出尺寸为 17x17x(384+96+256)=17x17x768。在第 2 个 Inception 模组块中，5 个 Inception Module 输出 tensor 的尺寸将全部定格为 17x17x768，即图片尺寸和输出通道数都没有发生变化。

接下来是第 2 个 Inception 模组块的第 2 个 Inception Module——Mixed_6b，它有 4 个分支。第一个分支是一个简单的 192 输出通道的 1x1 卷积；第 2 个分支由 3 个卷积层组成，第 1 层是 128 输出通道的 1x1 卷积，第 2 层是 128 通道数的 1x7 卷积，第 3 层是 192 输出通道数的 7x1 卷积。这里运用了 Factorization into small convolutions 思想，串联的 1x7 卷积和 7x1 卷积相当于合成了一个 7x7 卷积，不过参数量大大减少了（只有后者的 2/7）并减轻了过拟合，同时多了一个激活函数增强了非线性特征变换；第 3 个分支一下子拥有了 5 个卷积层，分别是 128 输出通道的 1x1 卷积，128 输出通道的 7x1 卷积，128 输出通道的 1x7 卷积，128 输出通道的 7x1 卷积和 192 输出通道的 1x7 卷积。这个分支可以算是利用 Factorization into small convolutions 的典型，反复将 7x7 卷积进行拆分；最后，第 4 个分支是一个 3x3 的平均池化层，再连接 192 输出通道的 1x1 卷积。最后将 4 个分支合并，这一层输出的 tensor 的尺寸是 17x17x(192+192+192+192)=17x17x768。

第 2 模组块的第 3~5 个 Inception Module——Mixed_6c、Mixed_6d、Mixed_6e 和 Mixed_6b 非常相似，只是第 2 个分支和第 3 个分支中前几个卷积层的输出通道数量不同，从 128 变为了 160，但是这两个分支最终输出通道数不变，都是

192。其他地方则完全一致。而网络每经过一个 Inception Module，即使输出 tensor 尺寸不变，但是特征都相当于被重新精炼了一遍，其中丰富的卷积和非线性化对提升网络性能帮助很大。

第 3 个Inception 模组块包含了 3 个 Inception Module，其中后两个 Inception Module 的结构完全一致，它们的结构如图 4 中第三幅图所示。其中第 1 个 Inception Module 的名称为 Mixed_7a，包含了 3 个分支。第一个分支是 192 输出通道的 1x1 卷积，再接 320 输出通道数的 3x3 卷积，不过步长为 2，padding 模式为 VALID，因此图片尺寸缩小为 8x8；第 2 个分支有 4 个卷积层，分别有 192 输出通道的 1x1 卷积、192 输出通道的 1x7 卷积、192 输出通道的 7x1 卷积，以及 192 输出通道的 3x3 卷积。最后一个卷积层同样步长为 2，padding 为 VALID，因此最后输出的 tensor 尺寸为 8x8x192；第 3 个分支则是一个 3x3 的最大池化层，步长为 2，padding 为VALID，而池化层不会对输出通道产生改变，因此这个分支的输出尺寸为 8x8x(320+192+768)=8x8x1280。从这个 Inception Module 开始，输出图片尺寸又被缩小，同时通道数也增加了，tensor 的总 size 在持续下降中。

接下来是第 3 个 Inception 模组块的第 2个 Inception Module——Mixed_7b，它有 4 个分支。第 1 个分支是一个简单的 320 输出通道的 1x1 卷积；第 2 个分支先是 1 个 384 输出通道的 1x1 卷积，随后在分支内开了两个分支，这两个分支分别是 384 输出通道的 1x3 卷积和 384 输出通道的 3x1 卷积，然后使用 tf.concat 合并两个分支，得到的输出 tensor 尺寸为 8x8x(384+384)=8x8x768；第 3 个分支更复杂，先是 448 输出通道的 1x1 卷积，然后是 384 输出通道的 3x3 卷积，然后同样在分支内拆成两个分支，分别是 384 输出通道的 1x3 卷积和 384 输出通道的 3x1 卷积，最后合并得到 8x8x768 的输出 tensor；第 4 个分支是在一个 3x3 的平均池化层后接一个

192 输出通道的 1x1 卷积。最后，将这个非常复杂的 Inception Module 的 4 个分支合并在一起，得到的输出 tensor 尺寸为 8x8x(320+768+768+192)=8x8x2048。

第 3 个 Inception 模组块的最后一个 Inception Module——Mixed_7c 与 Mixed_7b 完全一致，最后输出的 tensor 也是 8x8x2048。然后通过对最后一个卷积层的输出进行一个 8x8 全局平均池化，padding 模式为 VALID，得到一个 1x1x2048 的 tensor，使用 tf.squeeze 去掉输出 tensor 中维数为 1 的维度。

此模型先使用 inception_v3 中的输入预处理方法将图片归一化，再使用 keras 中已进行预训练的 Inception V3 模型来提取猫狗图片中的特征，得到一个 2048 的特征向量。并对特征向量使用 relu激活函数，将负权重变为 0，并将特征向量尺寸收敛至 1024。最后使用 sigmoid 激活函数，得到 0 到 1 之间的预测结果。接下来通过 compile 配置模型的学习过程，优化器（optimizer）使用 adam，损失函数（loss） 因为使用 Log Loss（对数损失）进行评估，所以使用 binaty_crossentropy 损失函数。

完善

当模型训练验证集准确率达到 99.2%，通过预测测试数据集，导出 csv 文件提交到 kaggle 上。得到 2.1 左右的评估分数，相当于 50% 左右的准确率。检查发现电脑中的文件排序不是按照从小到大数字排序，更改导出文件代码，得到 0.05047 的得分，超过基准模型。

IV. 结果

模型的评价与验证

模型得到的验证准确率为 99.2% ，具有不错的训练效果，对于验证集中的猫狗图片有着很好的识别能力。模型中通过随机打乱训练数据，保证了模型对于预测结果

的稳定性。而且验证集中的数据都是模型没有参与训练的数据，可以很好的检验模型的泛化能力。

合理性分析

项目最终模型训练效果得分为 0.05047，相当于 kaggle 排行榜上第 68 名，超过了设立的基准模型第 100 名。模型具备了很好分辨猫狗的能力，顺利完成猫狗分类问题。

V. 项目结论

结果可视化

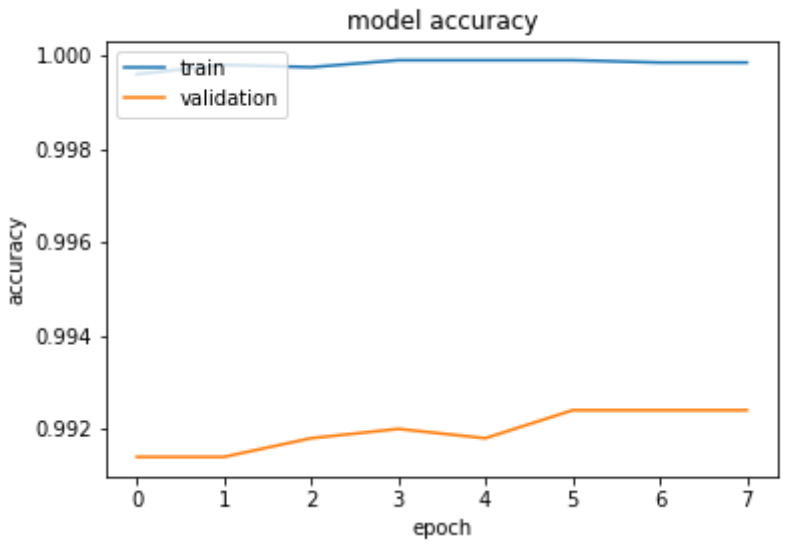

图 6：模型训练准确率图像

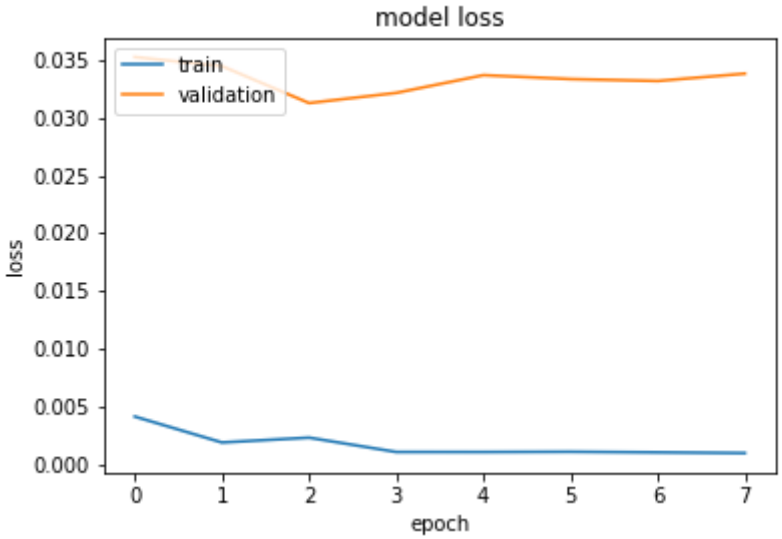

图 7：模型训练损失图像

从图 6 中可以看到模型对 Inception V3 提取的特征进行训练所得到的训练准确率。训练集准确率非常高，达到了 99.9% 以上。而验证集的准确率由 99.1% 左右，在第 5 个 epoch 稳定提升到 99.2% 以上。

图 7 显示为模型损失图像，训练集模型的损失不断下降，验证集损失在第 2 个 epoch 达到最低点，之后有所回升，并趋于平稳。

对项目的思考

项目先将不同标签图像分类到不同文件夹下。再通过图片生成器（ImageDataGenerator）下的 flow_from_directory 方法读取文件夹下的图片，并调整图片大小，通过 inception_v3 中的输入预处理方法将图片中 RGB 值归一化至 [-1,1]。然后使用 keras 中预训练的模型提取图片的特征向量，通过 relu、sigmoid 激活函数来输出结果。

通过此项目的学习，掌握了通过 keras 库来进行图像分类问题，并且取得了不错的效果。通过将图片分类后加以训练，完全可以在有足够数据的情况下解决各种各样的图像分类问题，而不只是猫狗图像分类。

此项目只使用了 Inception V3 模型，可以使用如 ResNet50、Xception、InceptionResNetV2 等模型来获得更好的训练效果。

需要作出的改进

可以将此识别类别扩大，并且制作一些可使用的应用，如应用到手机上，使手机可以识别各种宠物。可以尝试使用各种不同品种的宠物狗、宠物猫图片，训练一个可以识别不同品种宠物狗、宠物猫图像的模型，并将其应用到一个手机 APP 上来进行不同品种的猫狗识别。

参考文献

1. **ImageNet: A Large-Scale Hierarchical Image Database**, CVPR 2009, Authors: J.Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei
2. **ImageNet Classification with Deep Convolutional Neural Networks**, NIPS 2012, Authors: Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton
3. **Network In Network**, Authors: Min Lin, Qiang Chen, Shuicheng Yan
4. **Rethinking the Inception Architecture for Computer Vision**, Authors: Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna
5. **Xception: Deep Learning with Depthwise Separable Convolutions**, Authors: Francois Chollet
6. **Deep Residual Learning for Image Recognition**, Authors: Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun

7. **Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning**, Authors: Christian Szegedy, Sergey loffe, Vincent Vanhoucke, Alex Alemi15
8. **Adam: A Method for Stochastic Optimization**, Authors: Diederik Kingma, Jimmy Ba