

Capstone Project

狗狗分类器

By Yaping

2020/10/8

Part1 项目概况

背景

图像分类是计算机视觉中最基础也是最常见的问题，也是几乎所有的基准模型进行比较的基准。深度学习模型的发展史就是图像分类任务性能提升的发展历史，在 Imagenet 这样超过 1000 万图像、2 万类的数据集中，计算机的图像分类水准已经超过了人类。但图像分类并不像你想象的那么简单，也没有被完全解决。

项目说明

在我们这个项目中，我将带领你设计一个狗狗分类器，使其在任何时候输入一副图像，都能分辨出是不是狗狗，并输出狗狗的种类（有 133 个狗狗品种）。有趣的是，如果输入一个人类的照片，这个分类器可以辨识出这个人和哪个品种的狗狗相似呢！在这个项目中，我们主要是使用深度学习模型（Deep CNN）来搭建算法，并且会应用迁移学习来提高算法的效率。在算法设计过程中，我们会从最简单的模型开始搭建，逐渐提高算法的复杂度，来提高预测的准确性。

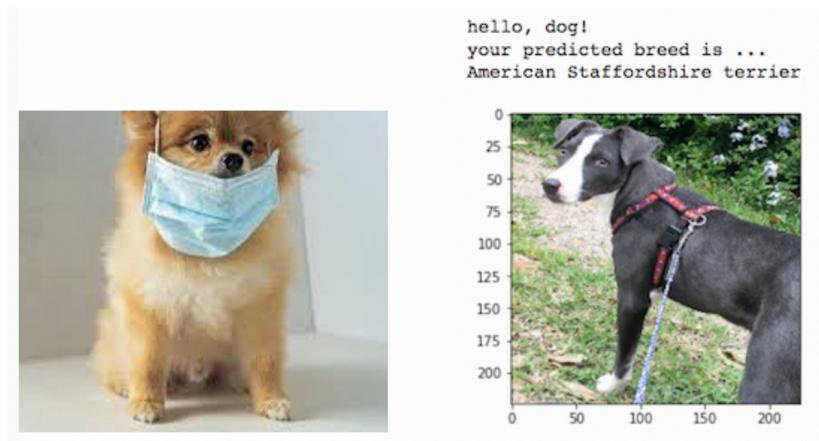


图 1 右图为算法输出示例

解决思路和方案

本项目的任务是利用卷积神经网络（CNN）构建一个狗狗分类器，最终使其能够接受任何用户提供的图像作为输入，如果从图像中检测出小狗，该算法将大致识别出小狗品

种；如果检测出人脸，该算法将大致识别出最相似的小狗品种。基于这些要求，我们得知的解决思路和方法如下：

1. 导入数据集（狗狗数据集和人脸数据集），并将其分为训练数据，校验数据和测试数据。使用 `dog_names` 字符串格式的狗狗品种名称列表，用于标识标签。
2. 使用 OpenCV 实现的 Haar 特征级联分类器（Haar feature-based cascade classifiers）来检测图片中的人脸数据集中的人脸。
3. 用 CNN 从头搭建狗狗分类器：我们使用 Keras 自己搭建，使用了卷积层、非线性、池化层以及 `dropout` 等，但分类效果不好，准确率只有 3.11%。这个结果比随机猜测还是有改进的，但离真正的预测要求还差很远。
4. 使用 Transfer learning（迁移学习）的方法搭建狗狗分类器：迁移学习是就是把已经训练好的模型参数迁移到新的模型来帮助新模型训练的方法，可以在极大提高模型性能的情况下减少训练时间。我们选用了 Resnet50 作为预训练模型，训练的准确率达到了 81.6%，基本达到了可接受的性能。
5. 综合设计分类器：将人类检测算法和小狗分类算法综合起来，以达到输入任意图片，判断图片中是否含有人、小狗或者两者都没有的效果。如果检测到了小狗，模型将进一步输出小狗的种类，如果检测到人脸，将会把与人脸最匹配的小狗品种输出来，否则会输出错误信息。

衡量指标

分类模型的指标有很多，常用的有：

1. 准确率（Accuracy）：指的是预测正确的样本占总样本的比率，取值范围为[0,1]，一般情况下取值越大，代表模型预测能力越好。
2. 精确率（Precision）和召回率（Recall）：精确率也称为查准率，召回率也称为查全率。两个指标相互影响，常一起出现。
3. F1-Score：综合了 Precision 与 Recall 产出的结果，F1 分数对那些具有相近的精度和召回率的分类器更为有利。F1-分数的取值范围是[0,1]，1 代表模型的输出最好，0 代表模型的输出结果最差。

每个指标都不是万能的，有它的适用范围。准确率是评价模型效果最通用的指标之一，描述了找到“真”类别的能力，但它在正负样本极不均衡的情况下是有很大缺陷的。比如在疾病诊断和信托产品客户预测时，这个指标并不适用。在项目中，项目 133 个类别的分布较为均衡，因此是可以采用准确率作为衡量指标的。

本项目采用的评价指标是准确性：`loss='categorical_crossentropy', metrics=['accuracy']`。

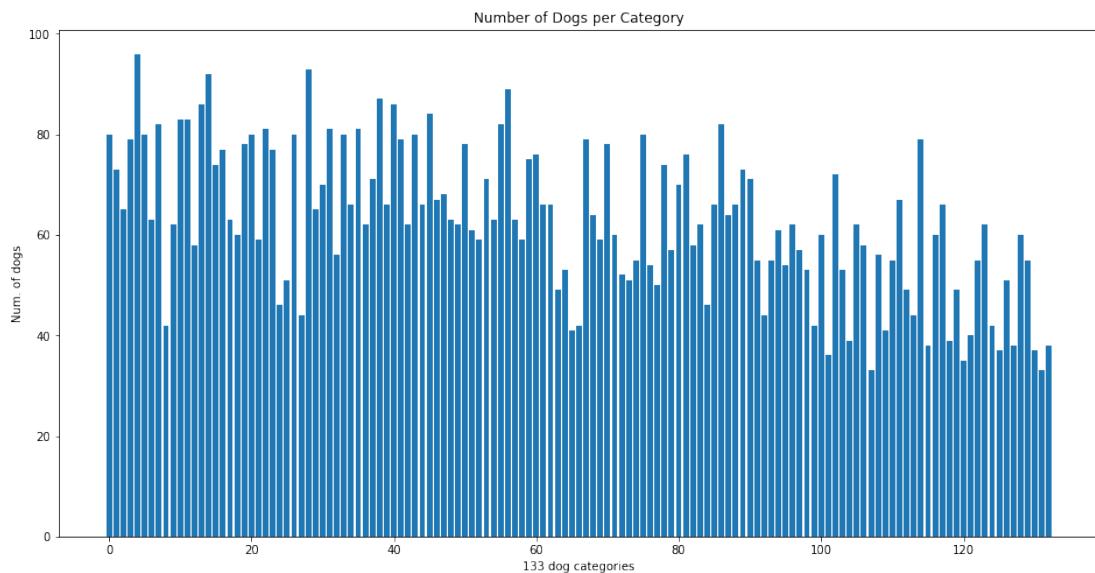


图 2 狗狗品种分布图

Part2 项目分析

数据探索

本次使用的数据有两类：一类是 133 个品种的狗狗图片，一共 8351 张；一类是人类的人脸图片，两类图片都是直接从优达学城 Github 上给出的链接直接下载下来的。狗狗数据集直接进行了训练集、验证集和测试集的划分，具体数目如下：

There are 133 total dog categories.
There are 8351 total dog images.

There are 6680 training dog images.
There are 835 validation dog images.
There are 836 test dog images.

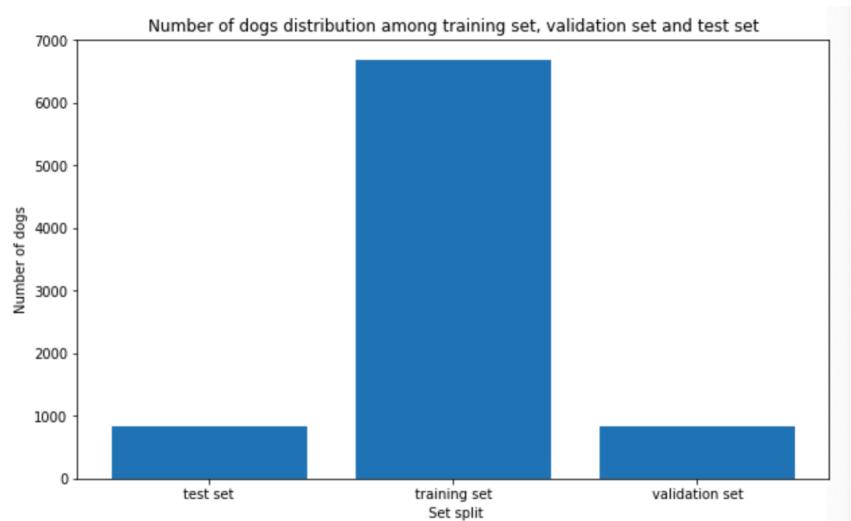


图 3 训练集/校验集/测试集数量分布图

狗狗品种分布的情况如下：一共有 133 个品种，每个品种平均有 63 张图片。

```
Out[16]:  
0  
count    133.000000  
mean     62.789474  
std      14.852330  
min      33.000000  
25%     53.000000  
50%     62.000000  
75%     76.000000  
max     96.000000
```

预测小狗的品种并不简单，是一个非常有挑战性的工作。因为有些狗狗的品种非常相似，即使人类要分辨也非常困难，比如图 4 所示布列塔尼猎（Brittany）和威尔斯激飞猎犬（Welsh Springer Spaniel）。



图 4 相似品种的狗狗

更有挑战的是，有的狗狗品种差异很大，即使是同一个品种，也存在不同的颜色和形态，比如图 5 所示：拉布拉多有黄色、巧克力色和黑色品种。基于视觉的算法需要克服这种同一类别差异很大的问题，并决定如何将所有这些不同肤色的小狗分类为相同的品种。



图 5 不同毛色的同一品种狗狗

在了解了数据集的基本情况之后，观察图片的大小并不相同，这对后续的图像处理显然有不利的影响。因此，我们要先对图片进行预处理，将图片缩放成同一个尺寸，并进行进一步处理。

数据预处理

在本项目的框架中，我们选用了 **Keras**，**Keras** 的后端有 **tensorflow** 做支撑，能提供较为上层的框架，搭建深度学习模型非常高效、方便。**Keras CNN** 要求输入的是一个 4 维的 **numpy** 数组（我们也称之为 4 维张量），其形状为：

$$(nb_samples, rows, columns, channels)$$

其中 **nb_samples** 表示图片（或样本）的数量，**rows**、**columns** 和 **channels** 表示每个图片各自的行数、列数和通道数。

图像加载和处理的大致步骤为：

1. 使用 **path_to_tensor** 函数接受一个彩色图片的文件路径字符串，返回一个适用于 **Keras CNN** 训练的四维张量。这个函数首先加载图片，然后将其重新调整为 224×224 像素的形状。接下来，图片被转成 **numpy** 数组，之后被整理成 4 维向量。在我们的项目中，因为我们处理的是彩色图片，每个图片有 3 个通道。并且，我们处理的是单张图片，因此返回的张量的形状就是：

$$(1, 224, 224, 3)$$

2. 将要输入 **CNN** 模型的 4 维向量准备好后，使用任何 **Keras** 中其他的预训练过的模型需要对数据进行进一步的预处理。首先，**RGB** 图像的通道要重排，以转换成 **BGR** 模式。然后，所有预训练的模型都需要再进行归一化，即必须将每张图片的每个像素都减去平均像素值 $[103.939, 116.779, 123.68]$ ，是用 **ImageNet** 中所有图像的像素值计算得到的）。这些步骤可以通过导入的 **preprocess_input** 函数来实现。

所有准备做好以后，就可以使用模型提取特征了！

Part3 模型搭建和训练

数据准备好以后，我们需要创建一个分类小狗品种的 **CNN**。我们先从头创建一个 **CNN**（因此暂时不能使用迁移学习），并且使测试准确率必须至少达到 1%。

自己搭建 CNN 模型

Keras 提供了非常方便的模块，可以让我们按照自己的想法随意搭建 **CNN** 模型，一般的 **CNN** 网络是一些卷积、非线性、池化层或者 **dropout** 的堆积。卷积的目的是以小的代价（模型参数）来获取图像特征（可以参数共享），池化层主要是进一步减少图像尺寸，有一定的预防过拟合的作用，**Relu** 经常被用来做非线性函数，而 **dropout** 通过随机关闭一些网络节点来达到过拟合的效果，加入 **BatchNormalization** 也可以起到加速计算的作用。网络最后会通过一个全链接层来实现分类，全链接层的输出数量就是类别数。

图 6 的框架基本包含了 **CNN** 的一些基本模块，并进行叠加，可以实现权值共享、减少参数、预防过拟合等作用，是一个基本的 **CNN** 分类网络。

我们的模型取得了 3.1% 的准确率。

Layer (type)	Output Shape	Param #	
conv2d_1 (Conv2D)	(None, 223, 223, 16)	208	INPUT
max_pooling2d_1 (MaxPooling2)	(None, 111, 111, 16)	0	CONV
conv2d_2 (Conv2D)	(None, 110, 110, 32)	2080	POOL
max_pooling2d_2 (MaxPooling2)	(None, 55, 55, 32)	0	CONV
conv2d_3 (Conv2D)	(None, 54, 54, 64)	8256	POOL
max_pooling2d_3 (MaxPooling2)	(None, 27, 27, 64)	0	CONV
global_average_pooling2d_1 (GlobalAveragePooling2D)	(None, 64)	0	GAP
dense_1 (Dense)	(None, 133)	8645	DENSE
Total params: 19,189.0			
Trainable params: 19,189.0			
Non-trainable params: 0.0			

图 6 CNN 模型框架

迁移学习 (Transfer learning) 模型

简单使用自己搭建的 CNN 框架来训练模型，预测结果并不使人满意。为了在不牺牲准确率的情况下减少训练时间，我们将使用迁移学习的方法训练 CNN。

迁移学习是就是把已经训练好的模型参数迁移到新的模型来帮助新模型训练的方法。由于大部分数据或任务是存在相关性的，所以通过迁移学习我们可以将已经学到的模型参数（也可理解为模型学到的知识）通过某种方式和新模型共享，从而加快并优化模型的学习效率，这是一个被业界证实了非常好的方法。

1. 模型选择

在本项目中，有四个迁移模型：VGG-19，ResNet-50，Inception，Xception 可以选择，我们首先选择了 ResNet-50。ResNet 对于解决分类问题已经有非常出色的表现，ResNet 的提出是 CNN 图像史上的一件里程碑事件，ResNet 在 ILSVRC 和 COCO 2015 上取得了 5 项第一，并又一次刷新了 CNN 模型在 ImageNet 上的历史，因此非常适合解决目前的问题。

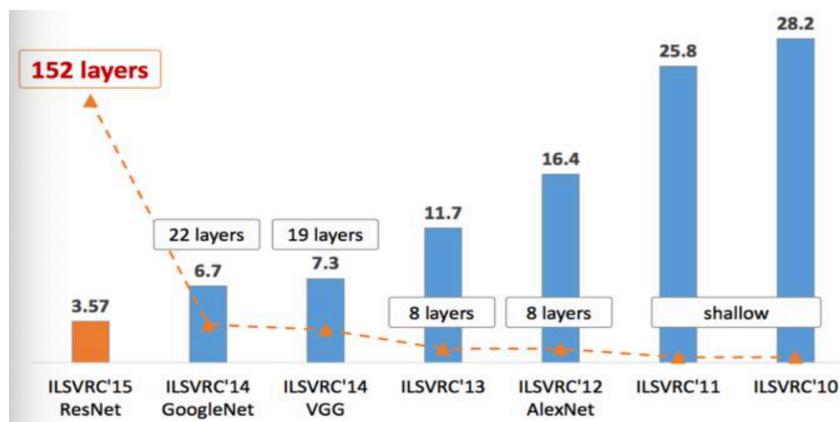


图 7 ILSVRC 不同模型错误率

ResNet 是一个非常有代表性的深度卷积网络 (DNN)，其网络参考了 VGG19，并在其基础上进行了修改，最重要改进是通过短路机制加入了残差单元，如图 8 所示，残差单元极大地解决了深度 CNN 模型难训练困难的问题。其他改进还体现在 ResNet 直接使用 stride=2 的卷积做下采样，并且用 Global average pool 层替换了全连接层。ResNet 的一个重要设计原则是：当 Feature map 大小降低一半时，其数量就增加一倍，这保持了网络层的复杂度。

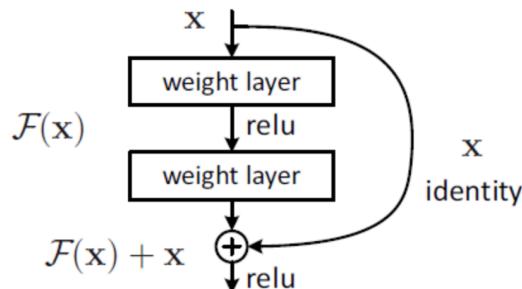


图 8 ResNet-50 残差结构示意图

2. 模型训练

选好了 ResNet-50 之后，因为这个模型是在 IMAGENET 训练集上已经训练好的，我们把卷积层以后的部分去掉，加上全新的未训练的层（相当于把卷积的部分保留并冻结，重新训练分类的部分），然后用我们提供的新的训练集进行二次训练。相比直接在我们的训练集上训练一个全新的模型，迁移学习节省了大量的计算成本，同时因为 IMAGENET 数据集足够强大，可以导致更好的训练效果。

具体方法：抽取特征向量（Extract Feature Vector）

先计算出预训练模型的卷积层对所有训练和测试数据的特征向量，然后抛开预训练模型，只训练自己定制的简配版全连接网络。

在迁移学习中，迁移的模型（如 ResNet-50）本身就具有大量的参数，即使是进行迁移学习，也需要很强大的计算资源和计算时间（即使是使用 GPU 也需要数小时、数天甚至数周的时间）才能训练的动这么深的模型。即使把所有的层都冻结，那么在训练过程中每次更新梯度时，训练集都要和模型中的所有参数进行计算（大量的矩阵相乘），而即使这些参数是定死的、不需要进行更新，这个过程也是非常费时间的。而如果你是使用 CPU 在进行这一步，无疑会需要更多的时间。

抽取特征向量的做法是：把训练集经过预训练模型生成出 bottleneck features，然后直接通过 bottleneck features 进行训练。这种方法相当于是将整个模型拆分成了两个部分：第一步是将所有图片通过 ResNet-50 的卷积结构，将数据“编码”成 bottleneck features；第二步则是用这些 bottleneck features 训练我们后加的新的结构（分类器）。

本项目采用的就是这种方法，因为参数冻结的部分本身就不需要更新，也不需要回传的参数，这种方法和直接训练一个大型的迁移学习网络是差不多的。而这种做法可以省去每次更新过程中和原先模型中参数进行的大量矩阵运算，从而训练的速度就大大加快了。

另外，还可以采用微调（Fine-tune）的方法：冻结预训练模型的部分卷积层（通常是靠近输入的多数卷积层），训练剩下的卷积层（通常是靠近输出的部分卷积层）和全连接层。

Fine-tune 的形式下分不同程度的解冻原有层参数，甚至可以解冻所有层。实际上，预训练模型的每一层都可以自定义解冻，进行二次训练。相比冻结所有预训练模型卷积层，Fine-tune 可以学到更多的特征知识，可以带来更好的效果，这种方法也是现在迁移学习中最为常用的做法。但是 Fine-tune 的代价就是需要大量的计算成本，包括计算时间和计算性能，因此对于计算资源有限的个人来讲，需要谨慎考虑。

Part4 模型评估和验证

人脸检测器评估

理想情况下，我们希望所有人脸图像都能检测到人脸，所有小狗图像都不能检测到人脸。我们的算法不能满足此目标，但是依然达到了可接受的水平。我们针对每个数据集的前 100 张图像提取出文件路径，并将他们存储在'human_files-short' 和'dog_files_short' 中。其中'human_files-short' 中 100 张检测到了人脸，'dog_files_short' 中 11 张检测到了人脸。

小狗检测器评估

小狗检测器的性能比较好，在 200 张图片中，其中'human_files-short' 中 0 张检测到了小狗，'dog_files_short' 中 100 张检测到了小狗，识别效果比较满意。

狗狗分类模型评估

对于狗狗分类器，我们 epochs 设为 40。从下图 9 可以看出，训练集的准确率很快就达到了 96%，而校验集的准确率一直在 82% 附近震荡，测试集的准确率在 81%。从上面的数据可以看出，应该是出现了过拟合。

结果讨论

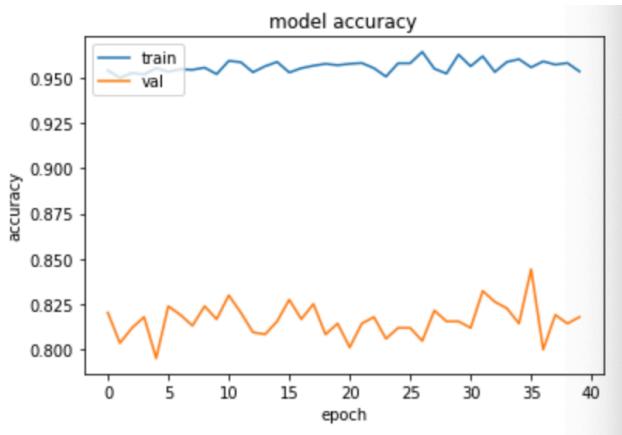


图 9 分步训练准确率

过拟合的产生通常是因为数据集过小而产生的，回头检查一下数据集的大小，数据集一共 8351 个狗狗的图像，133 个狗狗品种，平均每个狗狗有 63 张图片，的确数量比较少。如果想提高模型的准确性，增加数据集的数量是一个很好的方法。除了多采集更多的狗狗图像之外，如果采集条件受限，还可以采用数据增强的方式，把现有狗狗图片做处理后重新加入数据集。

训练中存在的问题

在模型训练中，需要注意的主要问题有：

1. 数据预处理

数据预处理是一个非常重要的事情，数据一定要经过预处理（前面有提到），在送入模型训练时候才能有比较好的效果。

2. 参数的选择

开始训练的时候，我的模型结果非常差，差到完全不能接受，后来仔细观察了 loss 的结果，发现很早就不再下降了。这时候及时的调整了 learning rate，将 learning rate 的值减小，loss 就开始变化了。可见，调参是一个需要耐心的事情。

3. 内存的管理

跑深度学习的算法，最让人崩溃的就是机器跑不动了，有时候还会报错，根本无法运行。这时候一方面要调整自己训练的参数，比如：Batch size，数据格式等，一方面还要寻求更直接的解决方法，比如购买并安装合理配置的 GPU，或者通过远端使用云端的 GPU 等等。

4. 及时地保存结果

我遇到过调了两天的程序，不知道什么原因没有保存，前面的工作全部报废的情况。

所以，我除了设置自动保存外，还会经常人工检查程序是否保存成功，并及时做好备份。通常在训练过程中，用模型检查点来保存验证损失最低的模型，以便后面使用，是非常明智的方法。

```
from keras.callbacks import ModelCheckpoint

checkpointer = ModelCheckpoint(filepath='saved_models/weights.best.Resnet50.hdf5',
                               verbose=1, save_best_only=True)

Resnet50_model.fit(train_Resnet50, train_targets,
                   validation_data=(valid_Resnet50, valid_targets),
                   epochs=20, batch_size=16, callbacks=[checkpointer], verbose=1)
```

图 10 检查点保存

Part5 总结和反思

经过训练，我们的狗狗分类器可以达到输入任意图片，它可以判断图片中是否含有
人、小狗或者两者都没有的效果。如果检测到了小狗，模型将进一步输出小狗的种类，如
果检测到人脸，将会把与人脸最匹配的小狗品种输出来，否则会输出错误信息。

项目的总体实现包括：

1. 读入数据、训练集、验证集集和测试集。
2. 数据预处理（数据格式调整，数据归一化）
3. 用 OpenCV 设计人脸检测器
4. 基于 Keras, 手动设计 CNN 模型来检测狗狗（准确率为 3.1%）
5. 基于迁移学习，选用 ResNet50 模型，设计狗狗分类器（准确率为 81%）
6. 综合人脸检测器和狗狗检测器，对输入的任何图片进行狗狗分类。

我们的狗狗分类器达到了 81% 的准确率，当然这不是最优的结果，可能是出现了过拟合，我们还可以做进一步的优化和提升，如果你也有兴趣的话，就一起来改进吧！本项目的目的只是给你演示了用深度学习的方法，如何一步步根据自己的需要搭建出合理的模型框架，并输出可接受的结果。其中用到了一个很重要的概念就是迁移学习，文章的最后列出了很多很好的参考文献，如果你想更加深入地了解这些概念，可以花时间多读几篇文章，相信一定能给你不少的启发。

反思

我们搭建并测试了基于 ResNet50 的分类模型，但你可以看到，准确率并没有达到预想的效果，要想进一步提高准确率，我们还有很多工作可以做，总结来说包括：

1. 数据增强：增加数据集的数量，更多搜集狗狗的图片，或者将原有数据集做一些剪裁或旋转，以增加图片的多样性。
2. 模型融合：综合各个不同的模型（ResNet-50, Inception, Xception 等），从而得到不错的效果，这个方法非常常见和有效。

3. 进一步地调参：参数调整中深度学习中的地位举足轻重，可以进一步通过仔细观察训练结果的和校验结果的方法，更改学习率，Batch-size 等参数的方法进行微调。

参考文献

你可以参考以下材料来加深对本项目的理解:

- CS231n: Convolutional Neural Networks for Visual Recognition
- Using Convolutional Neural Networks to Classify Dog Breeds
- Building an Image Classifier
- Tips/Tricks in CNN
- Transfer Learning using Keras
- Transfer Learning in TensorFlow on the Kaggle Rainforest competition
- Transfer Learning and Fine-tuning
- Building powerful image classification models using very little data
- 简述迁移学习在深度学习中的应用
- 无需数学背景，读懂 ResNet、Inception 和 Xception 三大变革性架构
- [VGG16] VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION
- [Inception-v3] Rethinking the Inception Architecture for Computer Vision
- [Inception-v4] Inception-ResNet and the Impact of Residual Connections on Learning
- [ResNet] Deep Residual Learning for Image Recognition
- [Xception] Deep Learning with Depthwise Separable Convolutions