机器学习纳米学位——猫狗大战

Jun Peng

2018 年 04 月 15 日

目 录

I. 问题的定义

项目概述

图像分类与识别在日常生活中有着重要的应用，如人脸识别解锁、医疗诊断等。机器学习技术特别是深度学习技术的快速发展给图像分类识别提供了强有力的工具。深度神经网络作为深度学习技术的一种，由于相对于传统机器学习技术如 adaBoost[1]等具有更高的准确度，目前已经成为解决图像分类识别问题最主要的工具。在近几年举行的 ImageNet[2]挑战中，获得冠军的参赛队伍都采用了深度神经网络。

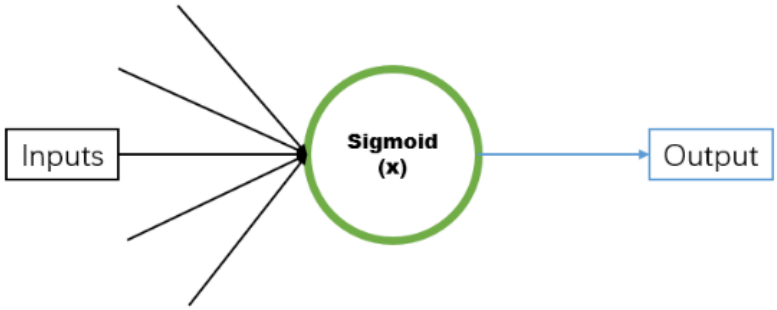

图 1 一个采用 Sigmoid 函数作为激活函数的神经元

神经网络顾名思义是由神经元组成的网络。一个神经元根据输入值通过激活函数计算输出值。通常采用的激活函数有 sigmoid 函数、Rectified Linear Unit（ReLU）函数等：

Sigmoid 函数：

$$\theta(\vec{x}) = \frac{1}{1+e^{-\vec{\omega}\bullet\vec{x}+b}}$$

ReLU 函数：

$$\mathrm{Re}\,Lu(\vec{x}) = \begin{cases} \vec{\omega}\cdot\vec{x}+b, \vec{\omega}\cdot\vec{x}+b > 0 \\ 0, \vec{\omega}\cdot\vec{x}+b \le 0 \end{cases}$$

其中输入量 x 既可以是标量也可以是向量，ω 为权重向量，b 为偏移量。通过设计或学习权值 ω 和 b 可使神经元获得不同的输出。

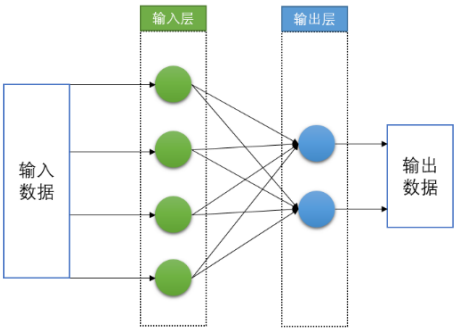

图 2 神经网络示意图

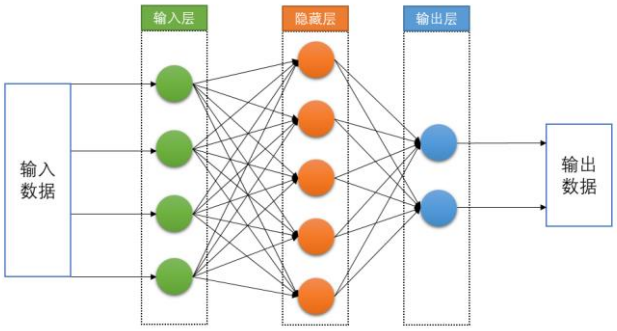

图 3 深度神经网络示意图

将多个神经元彼此连接起来就构成了神经网络。经典的神经网络由输入层和输出层组成，每一层都可以有多个神经元构成。如果在输入层和输出层之间加入更多的层（隐藏层）则构成了深度神经网络。

为了让神经网络能够完成既定的任务（如图像的分类和识别），需要对其进行训练：通过给神经网络提供训练数据，不断优化神经网络中的权重和偏移，使得其能够给出正确的分类或识别结果。神经网络的训练是通过反向传播（backpropagation）算法[13]进行的。根据给定的损失函数，利用链式求导法则获得损失函数关于各个神经元权重系数的导数，并采用梯度下降（Steepest Descent）算法等优化算法对网络参数进行优化，从而实现了对神经网络的训练。

对于图像识别分类问题，目前主流的方法是采用卷积神经网络（Convolutional neural net，CNN）[3]。卷积神经网络相对于传统的深度神经网络，主要区别在于增加了卷积层（convolution layer）和池化层（Pooling layer）用于图像特征的提取和筛选。在此基础上，一系列的卷积神经网络被提出，使得图像分类问题的正确率得到了极大的提升。较为知名的网络结构有 VGG[5]、ResNet[7]、Xnception[10]、DenseNet[23]等。

为了能够快速使用和部署深度神经网络，高效率的深度学习程序框架是必不可少的。目前比较流行的框架有 TensorFlow[17]、Caffe[18]、MXnet[16]、Keras[15]等。本项目中将采用将 TensorFlow 作为后端的 Keras 框架，同时配合 Python 的其他工具箱（如 Scikit-Learn[19]）构建解决问题所需的程序。

问题陈述

区分猫和狗对于人类来说这是个简单的任务，但是对于机器来说这并不简单。猫狗大战[20]是 Kaggle 在 2013 年举办的一场趣味竞赛。选手们通过设计算法教会计算机区分猫和狗。

根据给定的图片，所设计的算法需要判断出图片中的是猫还是狗：

- 输入：一张包含猫或狗的图片。
- 输出：图片中是猫还是狗。

这是一个二分类问题，因此输出是两个值中的一个：0 或者 1。输出 0 代表是猫，输出 1 代表是狗。

本项目中作者将构建一个卷积神经网络，使它读入图片并判断图片中的是猫还是狗。

评价指标

为了能够训练神经网络识别猫狗，需要提供损失函数。对于二分类问题可以采用交叉熵函数：

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

其中 n 是样本数，y 是目标标签，y=0 代表猫，y=1 代表狗，$\hat{y}$ 是样本为狗的概率。损失函数越小表明神经网络对于数据集有着较好的分类效果。交叉熵函数对于错误的分类结果非常敏感，错误的分类结果会使损失函数变大。因此，交叉熵是一个评估模型正确性和准确性的很好方法。

除了损失函数，还可以通过分类准确度来衡量神经网络的好坏。准确度的定义为：

$$accuracy = \frac{N_r}{N}$$

其中 N_r 是正确分类的样本数， N 是总样本数。模型分类效果越好，准确率越接近 1。

需要注意的是，对于两个准确度相近的模型，此时需要依据交叉熵损失函数来判断模型的好坏。交叉熵损失函数越小，模型的效果越好。

II. 分析

数据的探索

输入数据为图片文件，可以认为是尺寸为（width，height，3）的数组。每张图片的名称为 cat.x.jpg 或者 dog.x.jpg，其中 cat 或 dog 是该图片的标签，x 是该图片的编号。图 4 中给出了一些图片样本。Kaggle 给出了 25000 张图片作为训练集。其中包含了 12500 张包含猫的图片和 12500 张包含狗的图片。另外还有 12500 张包含猫或者狗的图片作为测试集。

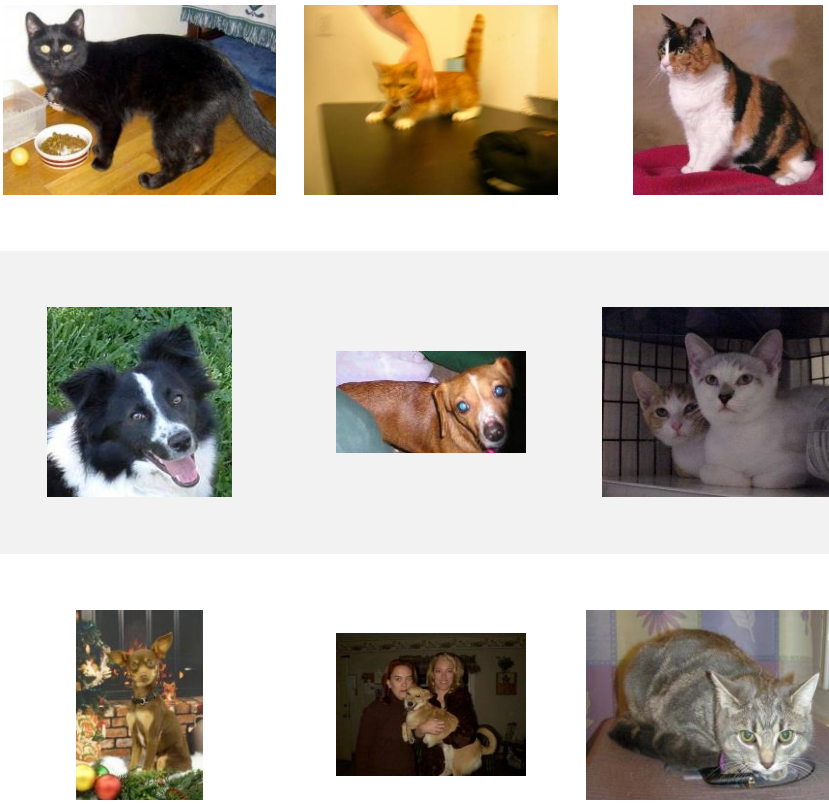

图 4 训练数据样本

图 5 给出了训练数据中图片尺寸的分布，可以看出图片从小到大各个尺寸都有。

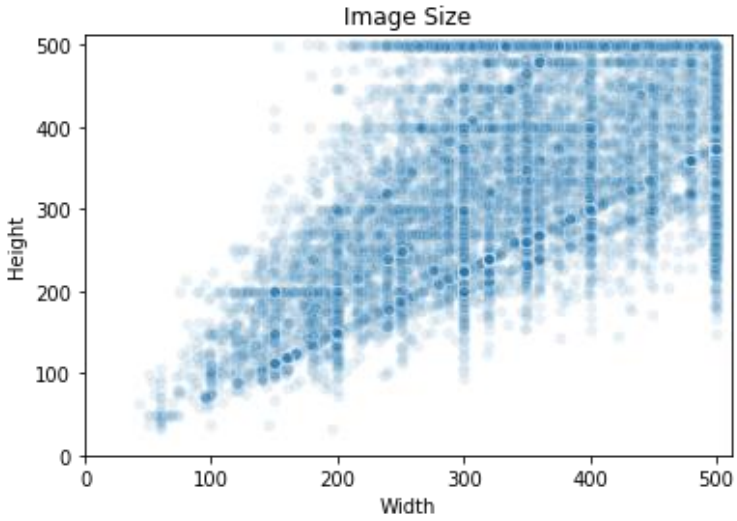

图 5 图片尺寸分布

从图 5 中可以看出，训练图片的尺寸是不一致，因此需要将其变换为相同的尺寸。为了使输入图像的尺寸一致，项目中将用到 Keras 的图片生产器（ImageDataGenerator）对图像尺寸进行变换。如对于尺寸小于指定尺寸的图片，将对其空白区域进行填充，而对尺寸大于指定尺寸的图片，将进行缩放和填充。当然，这样做的代价是有可能丢失图片中的细节信息，同时可能引入额外的误差（如填充方法选取不同而在边界处进入的新的特征）。

就图像中的拓扑特征来说，只有通过神经网络的提取才能够判断是否有意异常值。事实上训练数据集中确实存在一些错误标记的图片（如 dog.12376）。但考虑到神经网络非常强的容错能力，这里不做特别的处理，因为基本不可能将错误标记的图片从 25000 张图中全部挑选处理。

因此，从目前训练数据的信息看了，不需要做特别的处理。在训练中只需要进行归一化和图片尺寸变换。

探索性可视化

图片数据的要素除了尺寸还有像素中各个颜色通道的分布。如果训练数据的颜色分布不是均匀的或者是正态分布的，那么说明这个训练数据本身是有偏向的。

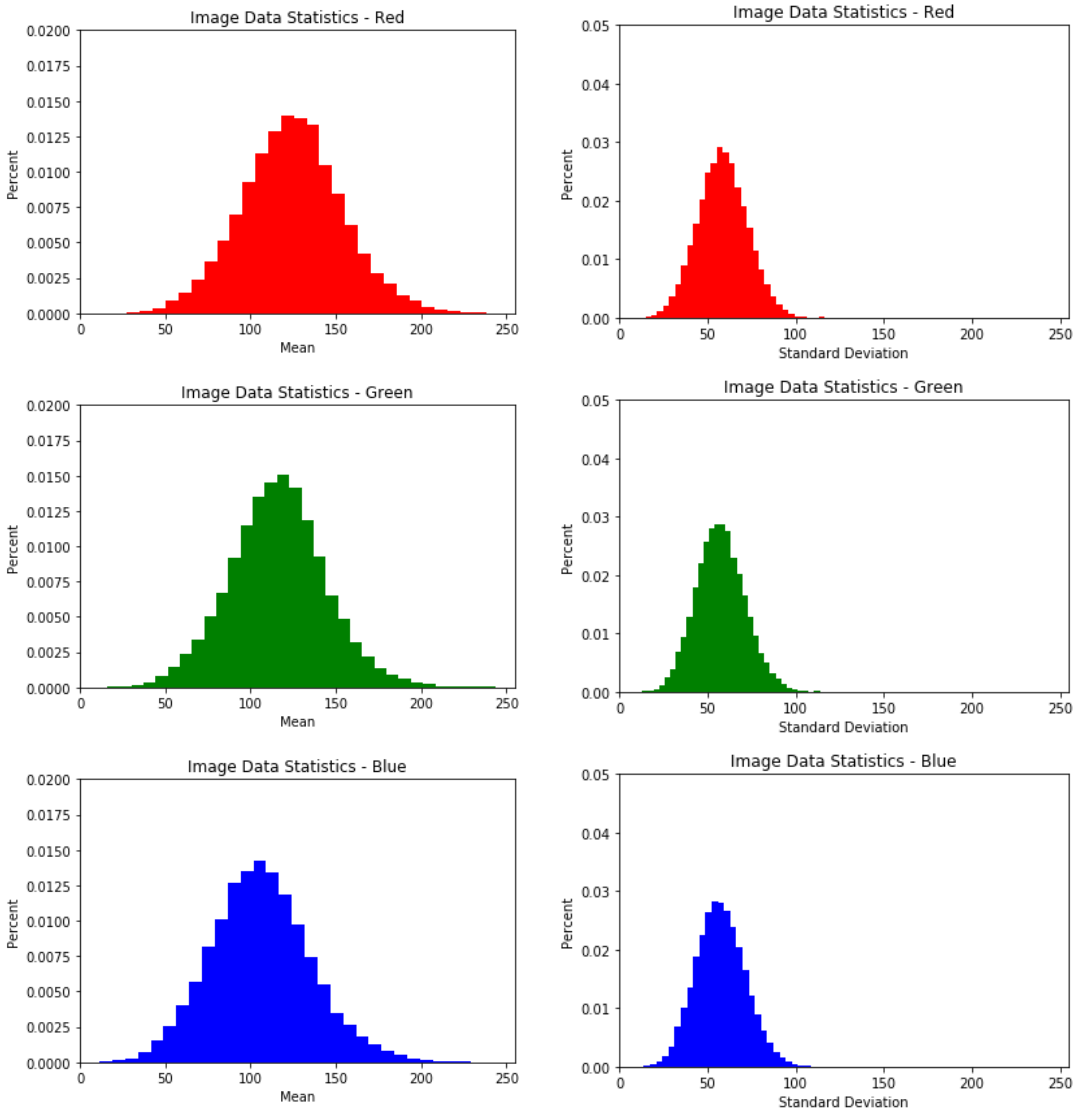

图 6 训练数据不同颜色通道的统计信息

为了了解训练数据中各个颜色通道中平均值的分布，在图 6 中给出了训练数据中每个图片各个颜色通道的统计分布柱状图（Histogram）。可以看出各个颜色通道的分布都类似于正态分布。红色和绿色分布的峰值大约位于 120 附近，而蓝色最大值在 110 附近。虽然不是完全的正态分布，但整体上看各个训练样本的颜色平均值偏差也不是很大。因此就颜色来说，该训练数据集没有在颜色上有特别的偏向。

算法和技术

本项目中将采用迁移-融合学习的方法构建猫狗分类网络。迁移学习是指对已经能够完成特定任务的模型进行修改再训练使其能够完成相似任务的过程。基于一些预训练的卷积神经网络，本项目中将对其输出层进行改造和重新训练，使其能够完成识别猫狗的任务。

项目中将对几个已经在 ImageNet 上完成训练的网络模型进行迁移。由于 ImageNet 挑战是让模型对 1000 中物体进行分类，在 ImageNet 数据上训练的卷积神经网络对于各种自然图像特征都有非常好的提取和筛选能力。对于猫狗分类问题，即使训练数据图片中出现了如人物、物体等其他非猫狗的客体，这些预训练的模型能够很好的提取出来并筛除同时不会影响到分类的准确性，而不像从头训练的猫狗分类专用网络模型可能会把一些图片中常出现的事物（如人手、人脸、桌子等）当作一个分类特征而影响分类准确性。因此，基于这些已经训练好的模型迁移构造猫狗分类网络是可行的并且是相比构造专门模型更好的选择。

基准模型

基准模型选用的是[22]中给出的一个基于 ResNet50-InceptionV3-Xception 的融合模型。其结构如图 7 所示。

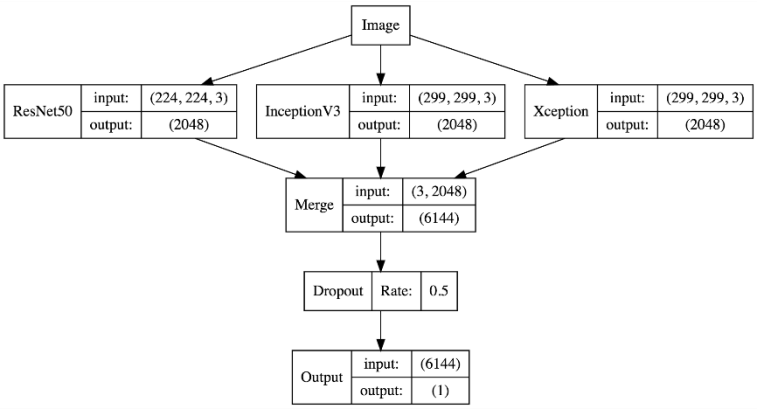

图 7 基准模型结构[22]

各个子模型的权重采用 ImageNet 权重，但去掉了各个模型的分类器（Top 层），并将其 GlobalAveragePooling2D 层的输出合并传递给分类器（output layer）。训练时，固定除了分类器层以外的权重。

作者按照[22]中的步骤复现了一遍该模型，并获得了测试样本 0.03852 的 Kaggle 得分，比原文中更好（0.04141），说明该模型的正确性。实现过程见附件（cat_and_dog_mergenet.ipynb）。

按照毕业项目要求，所构造的模型需要达到 Kaggle 天梯前 10%的分数，即得分应小于 0.0617。显然，该基准模型已经达到了这一要求。本项目中将要做的是获得比基准模型更好的得分。

III. 方法

数据预处理

一方面，对于训练数据，由于每个训练数据图片尺寸都不太一样，需要对读入的数据尺寸进行调整，使其满足各个模型的输入要求。另一方面，图像数据实际上是三维的数组，其尺寸为（Width，Height，Channel），每个数组元素的取值范围为 0~255。而所使用到的预训练模型需要归一化到（0~1）或者（-1,1）的输入数据。因此，在数据预处理阶段主要是完成两部分工作：先对图片尺寸进行调整，然后对数据进行归一化。

为了调整图片数据的尺寸，作者有两种方法，借助 PIL 模块的 image.resize 方法，或者使用 Keras 图像生成器 ImageDataGenerator 指定数据尺寸。考虑到在后期对网络进行 Fine-tuning 时，作者需要使用到 ImageDataGenerator 的一些其他功能，为了简便程序，作者采用 ImageDataGenerator 进行数据输入同时调整图片尺寸。

图 8 原始图片



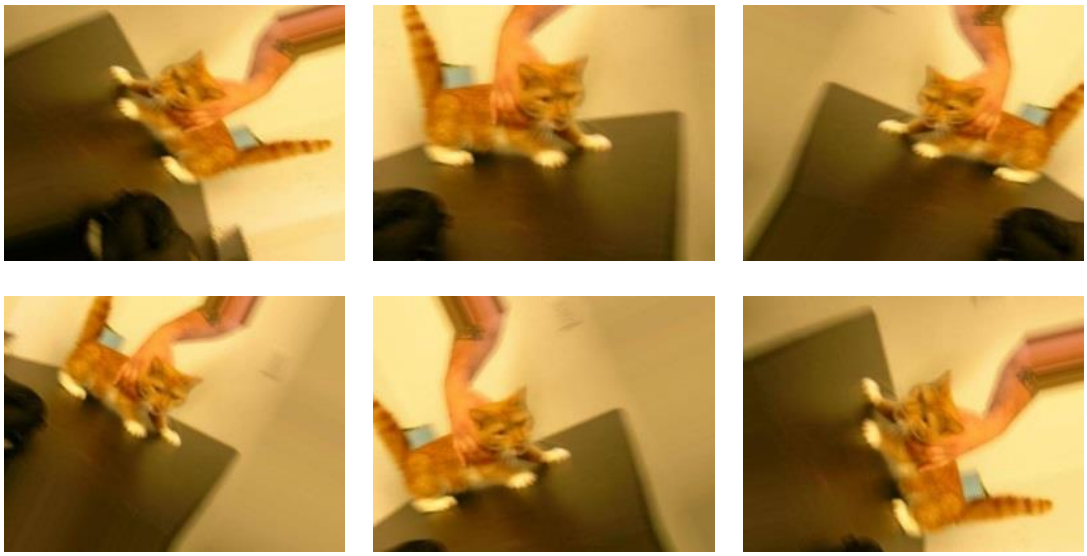

图 9 经过图像增强后的图片

图片生成器除了能改变图像的尺寸外，还能对图片进行随机旋转、拉伸和缩放。图 8 和 9 中给出了图片生成器对图片增强的一个例子[24]。在 Fine-tuning 中，为了获得更多的数据输入，作者将使用图像生成器的以上功能。

在 Application 模块中，Keras 为一些模型（如 Xception 模型）提供了图片预处理函数 preprocess_input。该函数会将存储图片数据的数组按照一定规则进行归一化。在此，作者使用该函数对输入数据进行归一化。

执行过程

本项目将要进行迁移的候选网络模型包括：ResNet50[7]、Xception[10]、InceptionV3[8]、DenseNet169[23]以及 NASNetMobile[25]。

第一步，作者将遵从[22]中的步骤，实现 ResNet50-Xception-InceptionV3 的融合模型构建（称之为 Mergenet1）。

使用 prepare_data_file()函数，作者将为训练数据和测试数据建立符号链接。训练数据位于/data/train/cat 和/data/train/dog 文件夹中，测试数据位于/data/test/test/文件夹中。

```
#为数据连理symbol-link
train_data_dir, test_data_dir = prepare_data_file()
```

```
100%|██████████| 25000/25000 [00:00<00:00, 159992.19it/s]
100%|██████████| 12500/12500 [00:00<00:00, 185895.98it/s]
```

图 10 为训练数据和测试数据建立符号链接

为了能够将各个模型 GlobalAveragePooling2D 层获得的数据拼接起来，作者使用 helper 模块中的 write_feature_data()函数将各个模型在每个数据上得到的特征导出并存储在相应的 feature_*.h5 文件中。具体过程参见 helper.py 文件。

```
#导出ResNet特征数据
write_feature_data(ResNet50, (224, 224),
                   train_dir = train_data_dir,
                   test_dir = test_data_dir,
                   batch_size=1)
```

```
Found 25000 images belonging to 2 classes.
Found 12500 images belonging to 1 classes.
25000
12500
25000/25000 [==============================] - 734s 29ms/step
12500/12500 [==============================] - 367s 29ms/step
```

图 11 导出候选模型的特征数据（以 ResNet50 为例）

在完成所有候选模型的特征导出后，作者将 ResNet50、Xception 和 Inception_V3 的特征数据读入进来，并采用 np.concatenate()函数进行拼接，形成最终的融合模型的特征数据。这个数据将作为输入数据传递给接下来构建的分类器。

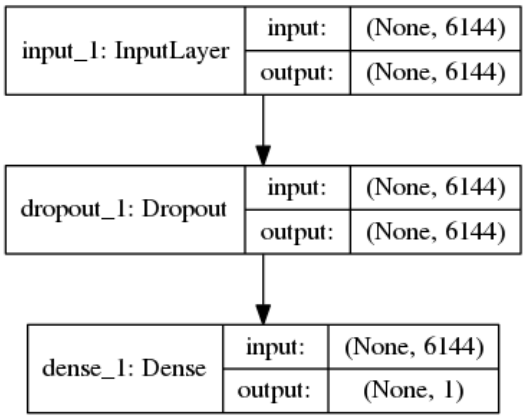

图 12 融合模型分类器结构

融合模型分类器的结构如图 12 所示，其中输入数据为合并了的特征向量。模型采用 adadelta 优化器训练，保留 20%的数据作为验证集。在训练过程中保存在验证集上损失函数最小的模型权重。经过 20 代训练，在验证集上最小损失函数能达到 0.0091，训练准确度为 99.66%，验证准确度为 99.70%。

在测试集上进行预测并提交到 Kaggle 进行评估，这个模型最后的得分为 0.03852。需要注意的是，为了避免预测数据为 0 或者 1 时带来的 log 函数溢出问题，作者采用了 clip 的方式，使得整个预测结果最小数为 0.005，最大为 0.995[22]。

模型	得分
ResNet50	0.07529
InceptionV3	0.04117
Xception	0.04091
DenseNet169	0.04590
NASNetMobile	0.12040

表 1 各个模型简单迁移后的 Kaggle 得分

采用这种融合方法得到的模型效果如此好，原因是什么呢？作者想弄清楚如果按照上述方式单独训练融合模型中的每一个子模型是否也能达到上述效果？为了回答这个问题，作者将分别对每一个子模型进行如上过程的训练，即导出该模型在训练集上的特征向量，并将其作为输入给一个分类器。通过训练该分类器得到最终的分类模型。作者分别测试了 ResNet50 模型、InceptionV3 模型、Xception 模型、DenseNet169 模型和 NASNetMobile 模型。各个模型在测试集上的得分如表 1 所示。可以看出融合方法比单独的模型得到了更准确的分类结果。

完善

为了进一步提升分类效果，作者将从两个角度对模型进行优化。一方面，作者选择单独得分最高的三个模型：InceptionV3、Xception、DenseNet169，进行融合看看效果如何。另一方面，作者分别对 ResNet50、DenseNet169 和 Xception 模型进行 Fine-Tuning，即放开一些层的权重进行训练，同时对输入数据进行增强，以达到更好的效果。

采用 InceptionV3、Xception 和 DenseNet169 的模型作者称之为 Mergenet2。该模型的最终测试数据得分为 0.03797。比之前的结果又有了提上。

接下来作者分别对 ResNet50、DenseNet169 和 Xception 模型进行 Fine-tuning。之所以选择这三个网络是因为他们分别代表了三种不同的网络结构。

- ResNet50 的 Fine-tuning

作者将加载了 ImageNet 权重的 ResNet50 模型的分类器去掉，取而代之的是一个 Dropout 层和只有一个输出的分类器。在此作者没有采用导出特征向量的方法是因为考虑到接下来的调参需要改变 ResNet50 一些层的权重，特征向量将发生改变。

第一步作者固定所有 ImageNet 权重，只允许分类器被训练。经过 10 代训练，作者得到的测试集得分为 0.12619。第二步，作者允许新模型 162 层以上

的层被训练（具体网络结构，见 cat_and_dog_Resnet_transfer.ipynb 文件），同时引入数据增强。模型权重的初始值采用前一步调参的结果。经过 5 代训练，其最终得分为 0.07575。接下来，作者依照上述方法，分别放开 152 和 140 以上层的权重进行训练。各阶段 Fine-Tuning 模型的得分见表 2。

训练层数	仅输出层	162 层以上 (Fine-tuning-1)	152 层以上 (Fine-tuning-2)	140 层以上 (Fine-tuning-3)
得分	0.12619	0.07575	0.07564	0.07259

表 2 ResNet50 模型 Fine-tuning 结果

- DenseNet169 的 Fine-tuning

Fine-tuning DenseNet169 模型的步骤与 ResNet50 的相应过程一致，在此不再重复，具体的优化过程参见 cat_and_dog_Densenet169_transfer.ipynb 文件。在此，作者只放开一个 DenseBlock 进行训练，这差不多是一半的卷积层。表 3 给出了调参的结果。

训练层数	仅输出层	370 层以上 (Fine-tuning-1)
得分	0.08211	0.04314

表 3 DenseNet169 模型 Fine-tuning 结果

- Xception 的 Fine-tuning

对于 Xception 模型，作者依次放开了 97 层以上、107 层以上以及 87 层以上的权重进行训练。结果如表 4 所示。

训练层数	仅输出层	97 层以上 (Fine-tuning-1)	107 层以上 (Fine-tuning-2)	87 层以上 (Fine-tuning-3)
得分	0.07127	0.03600	0.03780	0.03770

表 4 Xception 模型 Fine-tuning 结果

与之前两个模型调参方法不同的是，作者在训练 107 层以上和 87 层以上这两个模型是，权重的初始值采用的都是 97 层以上训练的结果。之所以这么做是想看看，增加或者减少可训练层对于结果提升是否有影响。从结果来开，增加和减少可训练层并没有比 97 层以上这个模型得到更好的结果。

最后，作者将以上优化过的几个网络再次融合为一个猫狗分类模型（称为 Mergenet3）。具体过程与基准融合模型相似（见 cat_and_dog_mergenet3.ipynb 文件）。最终其在测试集上的得分为 0.03546。该模型是作者的最终模型。这个分数达到了 Kaggle 竞赛天梯前 0.5%的水平。

IV. 结果

模型的评价与验证

本项目的最终模型如前所述，是基于 Fine-tuning 过的 Xception、ResNet50 和 DenseNet169 的基础上建立的融合模型，相比于基准模型和单独优化的各个模型，其在测试集上的得分有明显的提升。该模型最终在测试集上的得分为 0.03546，满足了设计要求，并且达到 Kaggle 天梯前 0.5%的排名水平。

Mergenet3 含有两个隐含的可变参数，一个是 Dropout 概率，在本项目取值是 0.5。另一个是三个模型特征向量拼接时的顺序（比如 Xception-ResNet50-DenseNet169 或者 Xception -DenseNet169-ResNet50）。这两个参数对于模型最终的得分有一定的影响。对于 Dropout 概率，作者发现更大的概率有时能获得更好的得分（大约 0.03490 左右）。而不同的拼接顺序影响则没有规律，时好时坏。由于 Mergenet3 的准确度已经非常高了（验证集上正确率为 99.88%），作者认为，考虑到 Dropout 概率和特征拼接顺序对模型的影响是有很大随机性的，而事实证明取一个折中的 Dropout 概率（0.5）和任何一种拼接顺序给模型最终结果带来的变化本身不是很大（小于 0.1%的正确率变化），因此这个问题暂时先不考虑。

一方面，在进行预测时，Dropout 概率设为 0，即没有随机的因素；同时，输入参数顺序在模型训练好后也必须是固定的。另一方面，在后面会看到，训练数据中本身就有错误标记的数据，而模型仍然能具有非常强的预测能力。因此，如果不考虑训练阶段的种种变数，最终 Mergenet3 模型本身是十分鲁棒的。

图 13 部分测试样本及其预测值

图 13 中给出了部分测试样本的预测结果，这些样本为随机挑选（具体见 cat_and_dog_mergenet3.ipynb 文件）。可以看出，最终模型对于这些样本的预测结果基本是正确的。特别的，对于图 No.6, 作为人类的作者尚不能肯定是狗，不过看起来很可能是狗。从这些结果可以看出，该模型对于猫狗的识别能力是非常强的，即使图片中出现了很多其他物体，如人体、笼子等等，都能做出很好的分辨。

综上所述，作者得到的 Mergenet3 模型是可信的。

合理性分析

序号	模型名称		交叉熵损失函数
1	Mergenet1		0.03852
2	Mergenet2		0.03797
3	Mergenet3		0.03546
4	Xception	导出 Feature 作为输入	0.04091
		连接全连接层	0.07127
		连接全连接层-Fine-tune-1	0.03600
		连接全连接层-Fine-tune-2	0.03780
		连接全连接层-Fine-tune-3	0.03770
5	ResNet50	导出 Feature 作为输入	0.07529
		连接全连接层	0.12619
		连接全连接层-Fine-tune-1	0.07575
		连接全连接层-Fine-tune-2	0.07564
		连接全连接层-Fine-tune-3	0.07259
6	DenseNet169	导出 Feature 作为输入	0.04590
		连接全连接层	0.08211
		连接全连接层-Fine-tune-1	0.04314
7	Inception_V3		0.04117
8	NASNetMobile		0.12040

表 5 所有模型的比较

为了比较基准模型和最终优化的模型，作者将各个模型的损失函数放在表 5 中。从数据可以看出，Mergenet3 是目前获得的最好的模型，它的损失函数比所有模型的都小。同时，这个得分达到 Kaggle 天梯前 0.5%的水平，达到了在项目最初时的设计要求。这一分数也表明，Mergenet3 能够非常好的确定图片中的一只动物是猫还是狗（前提是这个动物必须是猫狗中的一种）。Mergenet3 的胜出表明，融合模型能够比单独的模型获得更好的效果。

V. 项目结论

结果可视化

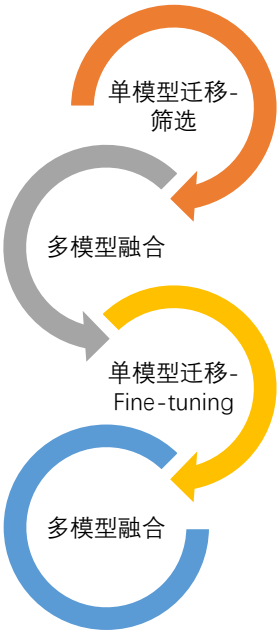

图 14 本项目构造思想

图 14 中展示了本项目构造模型的核心思想。通过迁移-筛选，作者从众多可能的模型中选择最适合目前所处理问题的几个模型作为融合的候选模型。此时作者筛选模型的评判标准是对它进行直接迁移后得到的模型的损失函数值。之后，利用这些候选模型，作者能够构造出一个基准模型作为后续调参优化的参考。随后，作者分别对这些候选模型进行深度的调参（Fine-tuning），使得各个模型分别尽可能的提高准确率，降低损失函数值。最后，将调参后得到的各个候选模型再次融合起来，可以得到比单独模型更好的效果。

对项目的思考

在这个项目中，作者构造了一个能够确定一只是猫或狗的动物的确切种类的卷积神经网络。作者采用融合几个预训练模型的方法，将原本用于分类 1000 类物体的卷积神经网络迁移为只对猫狗进行分类的网络。

首先，作者确定了一个基准模型。这是一个由 ResNet50、InceptionV3 和 Xception 组成的融合模型。这几个模型的权重采用的是 ImageNet 分类模型权重。将这些模型的分类器去掉后，作者导出了训练数据和测试数据的全部特征向量。通过拼接各模型导出的特征向量，作者就得到了用作迁移的输入数据。这些输入数据被输入到只有一个 Dropout 层和一个全连接层的二分类器。通过

训练这一二分类器就获得了最终的基准模型。基准模型的测试结果表明，通过融合和迁移，能够得到一个非常准确的猫狗二分类模型。

遵循这样的过程，作者考虑分别对被融合的各个模型进行深度调参（Fine-tuning），使它们本身对猫狗分类问题有了更好的分类精度后再将其融合起来，希望这样能够获得更好的分类效果。作者分别对 ResNet50、Xception 和 DenseNet169 模型进行了调参，使得他们的精度得到提升。随后将这些经过调参后的模型作为子模型，采用融合方法构造最终的分类模型。结果表明，经过这样方式的“迁移-融合-迁移-融合”，作者得到了分类效果极好（非常小的测试集交叉熵损失函数）的模型。这一模型的最终评分可以达到当时 Kaggle 猫狗大战比赛天梯前 0.5%的排名。

本项目中采用的这种方法对于其他问题的迁移学习也有着指导意义。研究者不需要对于特定问题从零开始训练网络，也不需要在进行迁移时花费大量的精力在单个模型上进行调参。他们需要的是，融合不同的模型，选择最好的组合，然后分别优化最好组合中的几个模型。当这几个模型优化取得了一定效果后就不需要进行更深度的迁移或优化，而是将他们再次融合起来。这样能够获得意想不到的好效果。当然，前提是这些被迁移-融合的模型对所考虑的问题是有效的。

本项目最困难的地方还是对单个模型的迁移/调参。随着深度卷积神经网络研究的深入，研究者们构造的模型越来越复杂，动辄三四百个卷积层。想要通过调参将这些巨大的网络迁移用来解决一些稍微小一些规模的问题（如从 ImageNet 的 1000 分类问题迁移到猫狗大战的 2 分类问题）所需要付出的时间和精力是巨大的。而采用本项目中的选择非最优调参加融合的方法，能够节约不少时间。作者在做项目过程中，实际大部分时间花费在对各个模型的 Fine-tuning 上。

作者所花费的时间和精力相比于深度调参单个模型是少的多的并且收益是非常大的。本项目的最终模型完全达到了预期的效果，并且还有极大的优化空间。

目前这一模型完全可以直接部署到有趣的平台上进行猫狗识别。当然，如果想要识别任意一个物体是猫还是狗，这个模型还是无法办到的，因为它只是一个二分类模型。

需要做出的改进

首先，正如前文提到的，作者在项目中并没有对训练数据进行清洗，在此假设了所有训练数据都是正确的。然而事实并非如此，[26]中提到了训练数据中实际上有一定量错误的图片，既不是猫也不是狗。图 15 给出了一个最匪夷所思的例子。

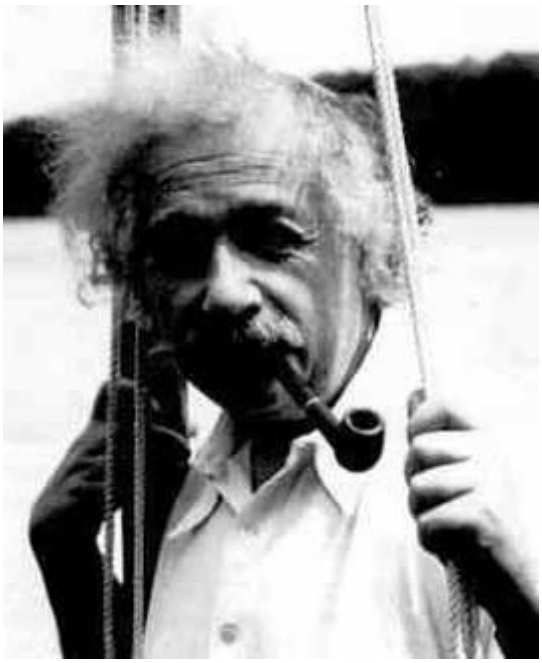

图 15 一个错误的训练数据（dog.1773.jpg）

尽管未经过数据清洗，作者仍然获得了相当好的分类模型。这再次证明了所构造模型的鲁棒性。当然，如果能将全部错误的训练数据从数据集中去除，相信能够达到更好的分类效果。

其次，在候选融合模型选择过程中，作者并没有做太多的尝试。作者根据的一个简单的原则是各个融合模型结构差异越大越好。之所以这样想，作者考虑的是差异越大的模型所提取出的特征越广泛，更能泛化描述目标特征。这一假设是有待考证的。如果精心选取几个在训练数据上做简单迁移就能达到非常好精度的模型作为候选模型，相信最终融合模型的效果将更好，完全能够超越 Kaggle 猫狗大战项目天梯当时的第一名。

再次，在 Fine-tuning 技术策略上作者目前还是比较生硬的一个模块一个模块的放开权重和训练。相信有更好的调参策略存在。同时，本项目在 Fine-tuning 时没有在分类器中尝试正则化调参。正则化主要是用来限制某些权重过大带来的过拟合等问题，考虑到我们用了较大的 Dropout 概率以及正则化调参收益的问题，因此放弃了正则因子调参。如果进一步考虑了以上诸因素，模型的准确性能够更上一个台阶。

最后，由于时间进度原因，作者没有完成模型的应用部署。在完成项目后，作者将着手学习 IOS 开发，希望在 IOS 平台上部署一些有趣而且可训练的轻量级网络，如培神（杨培文）的神经网络 for iphone app[27]。

参考文献

[1] Freund Y, Schapire R E. A decision-theoretic generalization of on-line learning and an application to boosting[J]. Journal of computer and system sciences, 1997, 55(1): 119-139.

[2] ImageNet, http://image-net.org/

[3] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[C]//Advances in neural information processing systems. 2012: 1097-1105.

[4] LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324.

[5] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.

[6] Szegedy C, Liu W, Jia Y, et al. Going Deeper with Convolutions[J]. arXiv preprint arXiv:1409.4842, 2014.

[7] He K, Zhang X, Ren S, et al. Deep Residual Learning for Image Recognition[J]. arXiv preprint arXiv:1512.03385, 2015.

[8] Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the Inception Architecture for Computer Vision[J]. arXiv preprint arXiv:1512.00567, 2015.

[9] Szegedy C, Ioffe S, Vanhoucke V, et al. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning[J]. arXiv preprint arXiv:1602.07261, 2016.

[10] Chollet F. Xception: Deep Learning with Depth wise Separable Convolutions[J]. arXiv preprint arXiv:1610.02357, 2016.

[11] Xie S, Girshick R, Dollár P, et al. Aggregated Residual Transformations for Deep Neural Networks[J]. arXiv preprint arXiv:1611.05431, 2016.

[12] Hu J, Shen L, Sun G. Squeeze-and-excitation networks[J]. arXiv preprint arXiv:1709.01507, 2017.

[13] Backpropagation Algorithm, https://en.wikipedia.org/wiki/Backpropagation

[14] Kaggle Cat vs. Dog. https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition#evaluation.

[15] Keras. https://keras.io/

[16] MXNet. https://mxnet.apache.org/

[17] TensorFlow. https://www.tensorflow.org

[18] Caffe. http://caffe.berkeleyvision.org/

[19] Scikit Learn. http://scikit-learn.org/stable/

[20] Kaggle Cat vs. Dog 2013. https://www.kaggle.com/c/dogs-vs-cats

[21] https://www.kaggle.com/jeffd23/catdognet-keras-convnet-starter

[22] https://github.com/ypwhs/dogs_vs_cats

[23] Huang G, Liu Z, Maaten L V D, et al. Densely Connected Convolutional Networks[J]. arXiv preprint arXiv: 1608.06993, 2016.

[24] http://keras-cn-docs.readthedocs.io/zh_CN/latest/blog/image_classification_using_very_little_data/

[25] Zoph, Barret, et al. “Learning Transferable Architectures for Scalable Image Recognition.” ArXiv Preprint ArXiv:1707.07012, 2017.

[26] https://zhuanlan.zhihu.com/p/34068451

[27] 神经网络 for iphone, https://itunes.apple.com/cn/app/id1116487840