猫狗大战

机器学习工程师 纳米学位 开题报告

张洪阳 2018年4月15日

目录

1. 问题定义

1.1 项目概述

猫狗大战源自 Kaggle 于 2013 年举办的一个娱乐竞赛项目，要求分辨给定测试图片是猫还是狗，属于计算机视觉领域图像分类问题，适合使用以卷积神经网络（Convolutional Neural Network，CNN）构建模型、最小化对数损失函数作为策略、梯度下降与反向传播作为算法的深度学习方法。

卷积神经网络根据生物视觉神经中感受野（Receptive Field）概念的启发，使用卷积核与上层的稀疏连接取代传统 DNN（Deep Neural Network）层与层之间的全连接权重，极大降低了网络参数数量，而理论上相比全连接网络表示能力只是略微降低，并且更容易训练。由于计算机视觉相关问题数据结构的特殊性（图像、视频等），往往每个像素只与附近区域的像素高度相关，非常适合使用 CNN 模型。

1.2 问题陈述

Kaggle 竞赛提供的数据包含了一个训练集和一个测试集，训练集有 25000 张图片，猫狗各一半，测试集有 12500 张图片。这些从真实世界中采集而来的猫狗图片，图像分辨率不同，背景环境复杂，猫狗的品种繁多，这些都为我们的分类增加了难度。我们需要搭建模型，用训练集里面得 25000 张图片来训练模型，最后再使用测试集来验证我们的模型，看看对于猫狗类型的判断准确率。

1.3 评价指标

模型输出单元使用 Sigmoid 激活函数，损失函数采用对数损失函数，即伯努利

分布下的交叉熵：

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

这是一个二分类问题的通用评价指标，

n 代表测试集中的图片数量；

$\hat{y}_i$为测试图片是狗的概率；

y_i如果图片是狗为 1，是猫为 0。

2 分析

2.1 数据的探索

训练数据集包括 25000 张图片，并在文件名中标注了图片为猫还是狗，猫狗数量各占一半，测试数据集包括 12500 张图片，没有标注类别，文件以数值 ID 命名。

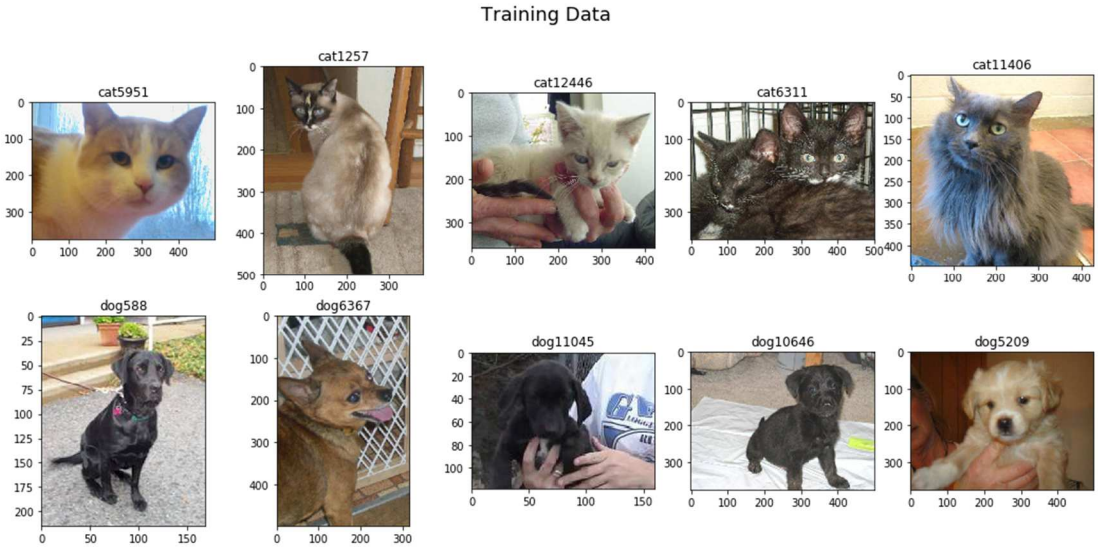

图 1. 训练数据按类别随机抽样

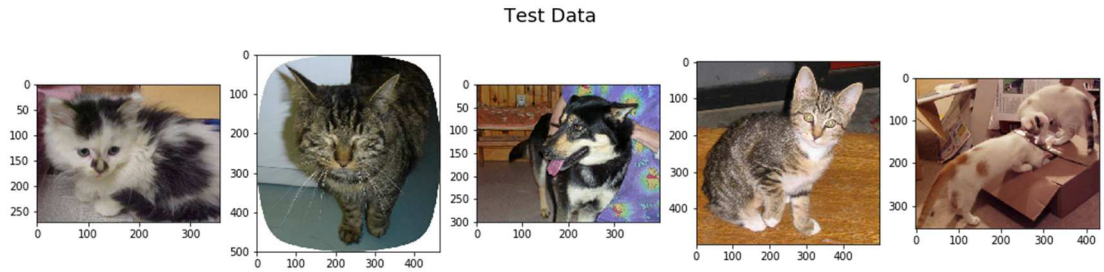

图 2. 训练数据随机抽样

如上图所示，图 1 为从训练数据集中随机抽样猫狗各 5 个样本，图 2 为从测试数据集中随机抽样 5 个样本，可见数据内容具备多样性与复杂性。如：图片尺寸参差不齐，不同图片包括了猫/狗身体不同角度，不同的猫/狗数量，有的图片没有包含全身，甚至没有正面脸部，图片背景元素也具备充分的干扰性。

2.2 算法和技术

根据分析可知，猫狗识别本是上是一个二分类问题。适用于分类的机器学习算法很多，比如支持向量机 SVM，随机森林 RF，决策树 DT， 然而这些方法在注明数据及上 Imagenet 数据集上取得的效果一直不是很好，并不能有效的解决图像识别问题，人们急需要一种新的方法来克服这一计算机视觉难题。直到 2013 年 Alex 发明 Alexnet 深度学习神经网络模型取得 Imagenet 比赛第一名，图像识别问题才算解决了。深度学习其实并不是一个全新的概念，早在上世纪 40 年代，就已经提出，但是由于当初的计算级的计算能力小，数据集稀缺，深度学习一度处于低谷。近年来，得益于 GPU 运算能力的爆发，数据集的增多，以及有效算法的开发，深度学习这一技术大放异彩，解决了许多问题，特别是计算机识别领域，使用卷积神经

网络（cnn），一种专门用来处理具有类似网格结构的数据的神经网络，可以说基本解决了图像识别的问题。所以在此次项目中，我也采用 CNN 来搭建我的模型框架。

另外，各大机构公司也推出了很多的深度学习平台， 比如 Google 的 tensorflow， Facebook 的 pytorch， 贾杨清开发的 Caffe。本项目中，本着简单快速上手的原则，使用基于 Google tensorflow 作为 backend 的 keras 来搭建深度学习神经网络模型。

2.3 基准模型

使用 ResNet50、Inception v3、Xception 组合模型输出特征向量进行逻辑回归（Logistic Regression），可以在验证集上达到 99.6%的正确率，在测试集的损失函数值达到 0.4141，提交 Kaggle 排名为 20/1314，要求项目结果不低于此标准。

3 方法

3.1 数据预处理

剔除异常样本,异常值判定使用四分位数法,判定的统计指标可以是图片尺寸、像素均值等，也可以使用预训练模型输出图像特征向量，根据特征向量到均值向量的欧氏距离作为判定值。异常样本删除需慎重，轻微异常的样本有益于模型的泛化性能，最好对判定为异常的样本图片进行人为判断是否保留。 图像尺寸统一，一般统一为所使用模型需要的输入尺寸，对于放大的图像需要进行插值处理。

新建一个文件夹 train2，包含两个子文件夹 dog 和 cat，将 train 文件夹中的图片，按类别放入 dog 和 cat 文件夹中，方便之后的训练。

3.2 实现

项目使用以 Tensorflow 为框架封装的 Keras API 进行模型实现：

卷积神经网络的核心是对图像特征的提取表示，从最基本的边缘特征逐级提升到抽象层次的对象，CNN 通过各卷积层及其相应的权重来记录这些特征。 从零开始训练一个卷积神经网络，需要非常精心的网络设计，大量的参数优化，和长时间的运算，很耗费资源。其实经过长久的发展，现在已经有很多优秀的网络结构模型可以利用，这样既能很好的表征猫狗的特征，也能节约计算资源。所以如之前所说，我直接应用了 4 个已经预训练好的模型,ResNet50,VGG19,InceptionV3,Xception。代码中只需要载入这些特征向量，并且将它们合并成一条特征向量，这样节得到了我们的 X_train，X_test 和 y_train。代码如图所示：

```
for filename in ["gap_ResNet50.h5", "gap_VGG19.h5", "gap_InceptionV3.h5", 'gap_Xception.h5']:
    with h5py.File(filename, 'r') as h:
        X_train.append(np.array(h['train']))
        X_test.append(np.array(h['test']))
        y_train = np.array(h['label'])
        X_train = np.concatenate(X_train, axis=1)
        X_test = np.concatenate(X_test, axis=1)
        X_train, y_train = shuffle(X_train, y_train)
```

最后构建模型。

4 结果

4.1 模型的评价与验证

然后就可以训练模型了：

```
In [35]: model.fit(X_train, y_train, batch_size=128, epochs=10, validation_split=0.2)

         Train on 20000 samples, validate on 5000 samples
         Epoch 1/10
         20000/20000 [==============================] - 1s - loss: 0.1772 - acc: 0.9305 - val_loss: 0.0266 - val_acc: 0.9910
         Epoch 2/10
         20000/20000 [==============================] - 1s - loss: 0.0430 - acc: 0.9846 - val_loss: 0.0171 - val_acc: 0.9944
         Epoch 3/10
         20000/20000 [==============================] - 1s - loss: 0.0286 - acc: 0.9909 - val_loss: 0.0160 - val_acc: 0.9940
         Epoch 4/10
         20000/20000 [==============================] - 1s - loss: 0.0260 - acc: 0.9913 - val_loss: 0.0132 - val_acc: 0.9952
         Epoch 5/10
         20000/20000 [==============================] - 1s - loss: 0.0217 - acc: 0.9931 - val_loss: 0.0125 - val_acc: 0.9956
         Epoch 6/10
         20000/20000 [==============================] - 1s - loss: 0.0201 - acc: 0.9937 - val_loss: 0.0120 - val_acc: 0.9958
         Epoch 7/10
         20000/20000 [==============================] - 1s - loss: 0.0188 - acc: 0.9938 - val_loss: 0.0115 - val_acc: 0.9962
         Epoch 8/10
         20000/20000 [==============================] - 1s - loss: 0.0182 - acc: 0.9940 - val_loss: 0.0112 - val_acc: 0.9962
         Epoch 9/10
         20000/20000 [==============================] - 1s - loss: 0.0173 - acc: 0.9947 - val_loss: 0.0108 - val_acc: 0.9964
         Epoch 10/10
         20000/20000 [==============================] - 1s - loss: 0.0161 - acc: 0.9950 - val_loss: 0.0106 - val_acc: 0.9960

Out[35]: <keras.callbacks.History at 0x1a72c3bdba8>
```

可以看到，经过 10 epochs 的训练，loss 已经降到 0.0106 了，识别准确率到达 0.9960，还不错。模型训练完了之后，使用测试集来验证，将结果输出为 pred_csv 文件。

4.2 合理性分析

最后可以看到，经过结合四种不同模型，合并它们的特征向量，我得到较好的识别结果，其效果大于单独使用其中的一种模型。因为这里的每一个模型都是久经检验的，可以说更有所长，综合它们到一起，可以高度概括出图片当中的内容，所以最后得出的结果较好。

5 结论

5.1 思考

整个工作的流程可以概括为：数据预处理，提取特征向量或者使用预训练的特征向量，综合不同的模型载入特征向量，构建模型并训练，测试集预测最终结果。决定最后结果的关键点在于第二步，如何提取特征向量，我直接使用了已训练的，久经考验的几个模型，综合使用它们，这样可以节约我自己的训练时间，且可得到较好的结果。我还考虑过这几种模型不同的组合使用，发现四种全用时，效果最好。我还考虑过 dropout 数值对于结果的影响。我们知道 dropout 技巧实际上是一种模型平均，就是把来自不同模型的估计或者说预测通过一定的权重平均起来。每次以一定的概率忽略一些隐层节点，这样每次训练的网络就是不一样的，每次训练都可以看做一个新的模型；此外，隐层节点以一定的概率出现，不能保证哪两个节点同时出现，这样就阻止了某些特征关联与其他特征的情况。最后发现，dropout 在 0.5 的时候，效果最好。总的来说，目前的结果还是不错的，高效的识别率，解决了猫狗识别问题。

5.2 改进

想要进一步提高识别率，似乎就需要深入钻研一下神经网络基本结构，不断练习调节参数，积累对于调参的心得。众所周知，深度神经网络的参数非常之多，而且内部具体是如何工作的，目前还是一个黑箱状态，人们了解的并不够透彻，所以不同参数之间对结果有着不同的影响，这个需要更多的时间和经验来掌握。

参考文献

[1] Karen Simonyan, Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

[2] Christian Szegedy, Wei Liu, Yangqing Jia. Going Deeper with Convolutions, 2014.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren. Deep Residual Learning for Image Recognition, 2015.

[4] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe. Rethinking the Inception Architecture for Computer Vision, 2015.

[5] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning, 2016.

[6] François Chollet. Xception: Deep Learning with Depthwise Separable Convolutions, 2017.

[7] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, Kaiming He. Aggregated Residual Transformations for Deep Neural Networks, 2017.