优达学城机器学习（进阶）毕业项目

Kaggle：Dog Breed Identification

邓勇

2018 年 2 月 25 日

目录

一、定义

1.1 项目概述

项目是 Kaggle 近期上线的一个 PlayGround 项目。训练数据来自于 ImageNet 关于狗品种的子集。通过这个项目可以让参与者了解图像识别算法，找到自己正在开发或者学习的算法中的问题[1]。

本项目属于图像识别领域新出现的一个研究方向，细粒度视觉分类(Fine-Grained Visual Categorization, FGVC)，其目的是将某一大类事物的子类进行分类。本项目中将狗这一大类事物，识别图片中的狗属于 80 个品种中的哪一种。

研究中，细粒度视觉分类常用的数据集还有：

1. CUB-200-2011（200 类，共 11788 张鸟的图片）；
2. FGVC-aircraft（100 类，共 10000 张飞机的图片）
3. Stanford cars（196 类，共 16185 张汽车的图片）
4. NA birds（555 类，48562 张鸟的图片）

研究中，常用的方法主要有两大类：局部识别和整体识别。局部识别，基于局部的图像特征，提取特征，然后进行特征分类。但是，局部特征提取一般都需要有标记的图片，所以，实际应用比较困难。整体识别除了本项目中使用的常规的 CNN 提取特征外，一般还有提取视觉词袋，然后进行分类。

1.2 问题陈述

本项目将识别 10357 张狗的图片，识别出每张图片中的狗是属于 120 个品种的哪一个品种。

本项目中，使用已经给出预训练权重的特点，主要将会先使用 YOLO V2[2]提取图片中的狗区域。然后，使用 InceptionV3 提取特征。最后，将这些特征输入一个多层神经网络训练，得到模型。[3][4][5][6][7][8][9][10]

使用 InceptionV3 作为基准模型，代表最基本的图片特征提取方法，并以此模型获得的 Score 作为基准模型的 benchmark，约为 0.39893。也可以选取其他的 Keras 提供的预训练模型。但是，因为本文进行的是对照组实验，所以使用相同的模型是必须的。

本项目 YOLO+InceptionV3 模型最终提交 Kaggle 获得 Score 是 0.35806。

1.3 评估指标

选用 Kaggle 官方的 Multi Class Log Loss 作为最终评价指标[11]。在模型训练过程中，以 keras 输出的每个迭代的历史准确度为主要的评判依据。

二、分析

2.1 数据

本项目的输入数据包括三个部分：

1） Keras 预训练模型的权重，计划使用 InceptionV3 这个模型。的预训练结果表 1 所示[12]。实际使用中，以上模型的参数从网上（https://github.com/fchollet/deep-learning-models/releases）直接下载。

模型	尺寸	Top-1 准确率	Top-5 准确率	参数	深度
InceptionV3	92 MB	0.788	0.944	23,851,784	159

表 1 Keras 预训练权重模型在 ImageNet 数据集上 Top-1 和 Top-5 准确率

2） 训练数据包括，10222 张已标记的 jpg 格式彩色图片，共 344MB。各张图片的大小并不统一，实际使用前需要调整图片大小。图片样例如图 1。

图 1 训练数据图片样例，以上图片的实际大小从左到右分别为 500x375、200x280 和 500x375

3） 测试数据包括，10357 张未标记的 jpg 格式彩色图片，共 345MB。各张图片的大小并不统一，实际使用前需要调整图片大小。图片样例如图 2。这些图片对应的分类的结果将作为输出，提交 Kaggle，验证最终算法的正确率。

图 2 测试数据图片样例，以上图片的实际大小从左到右分别为 500x375、500x347 和 375x500

2.2 算法

本项目的主要方法由文章《面部识别技术能用来识别鲸鱼》[12]启发，先定位要识别的目标，然后再进行识别。因为 YOLO V2 提供了对于狗的识别，所以，可以不用人工的标记数据，实践证明这是非常方便的。本项目中，使用 YOLO V2 裁剪图片中的狗所在的矩形区域。然后，使用预训练 InceptionV3 提取特征。最后，将这些特征输入一个多层神经网络训练，得到模

型。

2.3 开发环境

硬件环境：Thinkpad E470（NVidia 920M，i5，16GB 内存）。
操作系统：Windows 10 专业版。
Python 开发环境：Anaconda3-4.4.0、python 3.5。
主要 Python 开发包：Jupyter 1.0.1、Notebook 5.2.2、Tensorflow-gpu 1.4.0、Keras 2.1.2。

2.4 基准模型

将使用 InceptionV3 作为基准模型，代表最基本的图片特征提取方法，并以此模型获得的 Score 作为基准模型的 benchmark，约为 0.39893。InceptionV3 特征提取和全链接网络训练的结构如图 3。也可以选取其他的 Keras 提供的预训练模型。但是，因为本文进行的是对照组实验，所以使用相同的模型是必须的。Inception 的 Fine-Tune 结构如图 3。

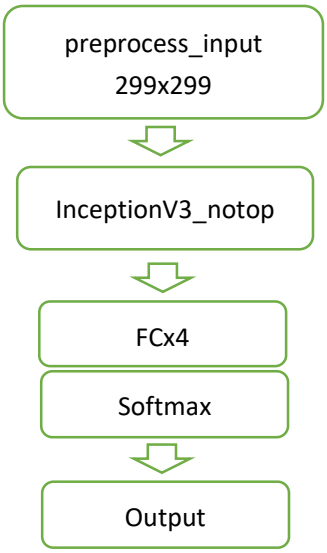

图 3 InceptionV3 特征提取和全链接网络训练结构图

三、方法

3.1 数据预处理

1）解压原始数据。Kaggle 提供的原始数据是 zip 文件，在使用之前，首先需要解压文件。主要使用以下代码：

```
def Unzip(data_path, zip_name):
    extract_name = zip_name[0:-4]
    extract_path = os.path.join(data_path, extract_name)
    zip_path = os.path.join(data_path, zip_name)
```

```
        if not (os.path.isdir(extract_path) or os.path.isfile(extract_path)):
            with zipfile.ZipFile(zip_path) as file:
                for name in file.namelist():
                    file.extract(name, data_path)
cwd = os.getcwd()
data_path = os.path.join(cwd, 'input')
Unzip(data_path, os.path.join(data_path, 'labels.csv.zip'))
Unzip(data_path, os.path.join(data_path, 'sample_submission.csv.zip'))
Unzip(data_path, os.path.join(data_path, 'test.zip'))
Unzip(data_path, os.path.join(data_path, 'train.zip'))
```

2）整理图片文件夹。在提取特征的时候，使用 Keras 的 ImageDataGenerator 获取数据。ImageDataGenerator 需要以下的文件结构（图 4）：

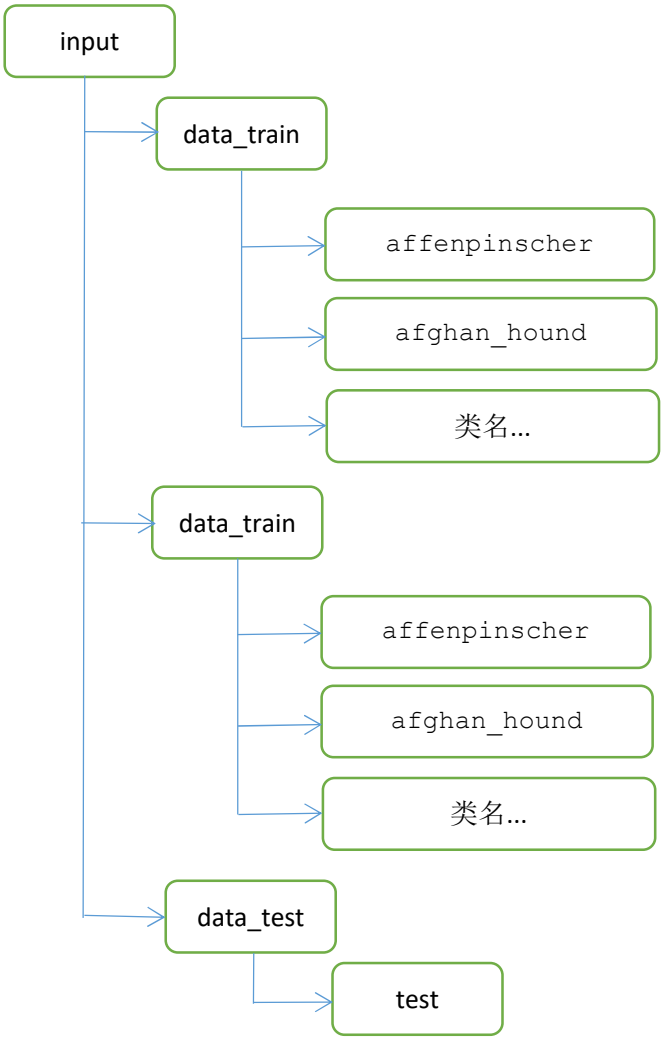

图 4 ImageDataGenerator 处理需要的文件结构

data_train 文件夹下，各个子类的类名作为子文件夹的名称，ImageDataGenerator 将会以这些子文件夹的名称作为读取数据之后的标签。

在文件夹结构创建完成之后，按照 labels.csv 中的文件名和类名的对应关系，开始从之前解压之后的文件夹中复制图片到对应类名的子文件夹中。test 数据集没有 label，所以，全都复制到 test 子文件夹下。

3）使用 YOLO 提取图片中狗所在的矩形区域，然后保存这个区域的图片到对应分类的文件夹中。一级目录对应关系是：data_train 对应 yolo_data_train，data_val 对应 yolo_data_val，data_test 对应 yolo_data_test。二级目录按照狗的分类名称文件夹对应。主要操作如下[13]：

- 从 Darknet 官网下载 model：https://pjreddie.com/darknet/yolo/。本文使用 YOLOv2 608x608。
- 将 YOLOv2 模型转换成 Keras 模型：./yad2k.py cfg/yolo.cfg yolo.weights model_data/yolo.h5。
- 提取狗所在的区域。保存 YOLO 获取到的有狗的区域：

  cropped_img = image.crop(crop_box)

  cropped_img.save(os.path.join(output_path,image_file), quality=90)

  因为 YOLO 的识别准确率并没有特别的高，所以有的图片中不能识别到狗。这种图片直接复制到对应的文件夹中。

为了节省磁盘空间，可以使用移动图片，而不是复制图片。因为，开发过程中有一个摸索 ImageDataGenerator 正确工作方式的过程，所以，复制图片减少了，操作出错之后，每次都需要重新解压文件带来的麻烦。或者使用文件链接，移动图片都可以避免了。但是，这个文件链接过程，在 Windows 环境下，需要额外的配置，增加了本项目的复现难度，所以，没有采取这个方式。

3.2 特征提取

本项目使用已经给出预训练权重的特点，主要将会使用 InceptionV3 提取特征。然后，将这些特征输入一个多层神经网络训练，得到模型。特征提取结构如图 3 上半部分。主要步骤为：

1）InceptionV3 提取特征，需要预处理层 preprocess_input；

2）创建 model（使用 notop 模型）；

3）创建 ImageDataGenerator，这里可以同时用于 data_train、data_val 和 data_test；

4）使用 model.predict_generator 输出结果，即为模型提取的特征；

5）使用 h5py 把输出的 train 特征、train 标签、val 特征、val 标签和 test 特征存到 h5 文件中。

主要函数代码如下[14]：

```
def get_features(MODEL, image_size, date_str, lambda_func=None, batch_size=1):
    print('{0} start.'.format(MODEL.__name__))
    start_time = time.time()

    width = image_size[0]
    height = image_size[1]
    input_tensor = Input((height, width, 3))
    x = input_tensor
    # 1）如果是 Xception 或者 InceptionV3，则需要处理输入的层：preprocess_input
    if lambda_func:
        print(lambda_func.__name__)
```

```
        x = Lambda(lambda_func)(x)
    # 2）创建 model（使用 notop 模型），当提取 YOLO 处理之后的图片特征时，将这里的 data_train 换成 yolo_data_train，data_val 换成 yolo_data_val，data_test 换成 yolo_data_test。
    base_model = MODEL(input_tensor=x, weights='imagenet', input_shape=(height, width, 3), include_top=False)
    model = Model(base_model.input, GlobalAveragePooling2D()(base_model.output))

    cwd = os.getcwd()
    data_train_path=os.path.join(cwd, 'input', 'data_train') #YOLO：data_train-->yolo_data_train
    data_val_path=os.path.join(cwd, 'input', 'data_val') #YOLO：data_val-->yolo_data_val
    data_test_path=os.path.join(cwd, 'input', 'data_test') #YOLO：data_test-->yolo_data_test
    # 创建 ImageDataGenerator，这里可以同时用于 data_train、data_val 和 data_test
    gen = ImageDataGenerator()
    train_generator = gen.flow_from_directory(data_train_path, image_size, shuffle=False,
                                              batch_size=batch_size)
    val_generator = gen.flow_from_directory(data_val_path, image_size, shuffle=False,
                                            batch_size=batch_size)
    test_generator  = gen.flow_from_directory(data_test_path,  image_size, shuffle=False,
                                              batch_size=batch_size)
    print(len(train_generator.filenames))
    print(len(test_generator.filenames))
    # 4）使用 model.predict_generator 输出结果，即为模型提取的特征
print('train_generator')
    train = model.predict_generator(train_generator, verbose=1, steps=len(train_generator.filenames))
    print('val_generator')
    val = model.predict_generator(val_generator, verbose=1, steps=len(val_generator.filenames))
    print('test_generator')
    test = model.predict_generator(test_generator, verbose=1, steps=len(test_generator.filenames))
    folder_path = os.path.join(cwd, 'model')
    if not os.path.exists(folder_path):
        os.mkdir(folder_path)
    file_name = os.path.join(cwd, 'model', 'feature_{0}_{1}.h5'.format(MODEL.__name__, date_str))
    print(file_name)
    if os.path.exists(file_name):
        os.remove(file_name)
    # 5）使用 h5py 把输出的 train 特征、train 标签和 test 特征存到 h5 文件中
    with h5py.File(file_name) as h:
        h.create_dataset("train", data=train)
        h.create_dataset("train_labels", data=train_generator.classes)
        h.create_dataset("val", data=val)
```

```
        h.create_dataset("val_labels", data=val_generator.classes)
        h.create_dataset("test", data=test)
```

调用方法为：get_features(InceptionV3, (299, 299), date_str, inception_v3.preprocess_input)。

3.3 全链接神经网络分类

常用的分类方法有很多，比如决策树、支持向量机、随机森林、XGBoost、全连接神经网络等。注意，由于 SVM 不能预测属于各个的概率，而本项目需要计算属于各个分类的概率，所以不能使用。

在实际调参时发现，全链接神经网络使用在训练过程中逐渐减小学习率是个很好的实践。前期，取比较大的学习率，可以加速收敛。后期，随着学习率逐渐减小，可以更好的拟合曲线。

3.4 完善过程

本项目中尝试使用 LogisticRegression 和 XGBoost 替换全链接神经网络分类器，但是效果都不及全连接神经网络分类器。

四、结果

本项目使用全链接神经网络作为分类器每次的结果都会略有不同，提交 Kaggle 的结果 0.35806，结果比较符合预期。

五、项目结论

5.1 结果可视化

本项目证明，使用 YOLO 获取狗所在的区域，然后，使用 Keras 预训练模型 InceptionV3 提取特征，最后，使用全链接网络作为分类器这种方式，与只使用使用 InceptionV3，然后使用全链接网络分类器相比，提升明显。

另外，InceptionV3 在验证数据集上获得了比 YOLO+InceptionV3 更好的结果：val_loss: 0.3861，val_acc: 0.8887，YOLO+InceptionV3 的结果为：val_loss: 0.4093，val_acc: 0.8828。但是，提交 Kaggle 的结果却是 YOLO+InceptionV3 更好（0.35806），而 InceptionV3 的 Score 为 0.39893。说明，YOLO+InceptionV3 具有更好的泛化能力。

学习曲线如图 6。由曲线可以看出，虽然 YOLO+InceptionV3 的准确率与单独 InceptionV3 相比提升较大，模型对训练数据的拟合比较好，但是泛化能力仍然不够，交叉测试的准确率最终与测试集相差还比较大，说明，模型还有较大的改进空间。

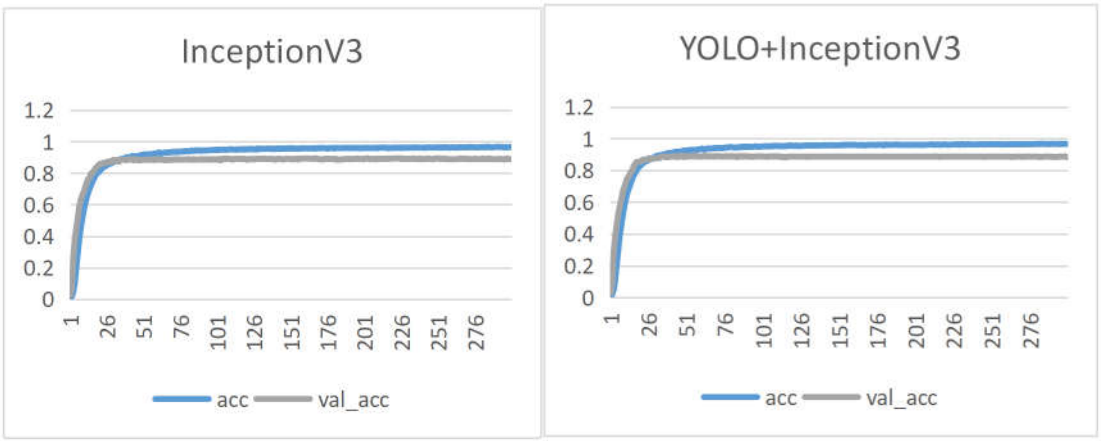

图 6 学习曲线

5.2 对项目的思考

1）项目来源。一开始选择这个项目，主要是因为 Kaggle 上面的项目会一直都在，数据也会一直都在，避免项目源代码中还不得不携带数据，或者其他人想测试本项目的代码，而数据却难以获得，也便于以后自己回溯。现在进行的 Challenger.AI 的场景分类其实也是一个不错的项目，但是数据的获取和使用不是很方便。

2）开发成本。这个项目的开发成本比较高，ucloud 的 GPU 主机花了好几百块钱。这个项目之后，花时间切换到其他厂商，即使是 Linux 的系统也可以，关键是要便宜。最终采用的方案是在本地笔记本电脑来运行数据，整个过程持续几天，上百个小时。经济成本和时间成本都是巨大的。Kaggle 算是上手了，以后会接着做，对计算能力的需求应该会更大。

3）新手难题。对于新手，独自搭建环境是个巨大的挑战，尤其是在不知道一个可以运行的环境需要哪些要素的时候。经过一两个项目，增加的不仅是整体的开发能力，还有信心。

4）各种内存溢出。本项目数据量比较大，在调试过程中遇到了各种形式的内存溢出，算是这方面积累一定的经验。

5）特征提取和特征分类分成两步，会节省很多时间。

6）结果不够好。虽然，这个结果符合预期，但是，这个结果不够好，问题是，Leaderboard 上前面的那些人用的方法，他们的方法是从哪里学到的。这个打渔的方法，还值得进一步的摸索。

5.3 需要做出的改进

本项目的结果证明，YOLO+InceptionV3 比单独 InceptionV3，会使结果更好。进一步的讲，提取出关键的信息，或者去除无用的信息，然后再提取特征会取得更好的结果。

博物学的角度分析识别特征会是更加准确，且有效的方式。对于细粒度领域分类，人类对于训练数据的标签是按照博物学的分类规则，人识别物体的多个特征，然后确定物体时属于那个类别。算法程序，可是可以按照这个思路去识别那个可以表明物体种类的特征。比如，本项目中，人标记训练数据中的狗的类别，是根据狗的毛发，体型大小等等细粒度的特征来确定狗是属于那个类别。那么，算法也可以采用类似的这种方式。

整个开发的代码比较零散。因为是新开发的流程，很多东西都是一点一点调试出来的，所以，代码封装不是很好。以后可以逐渐封装一些常用的方法，然后作为 python 包直接从

文件导入。

很多基础处理技术不是很熟。导致开发速度较慢，以后多做 Kaggle 里面的项目，熟能生巧。

参考文献

[1] https://www.kaggle.com/c/dog-breed-identification

[2] YOLO9000: Better, Faster, Stronger，https://arxiv.org/abs/1612.08242

[3] https://github.com/fchollet/deep-learning-models/releases

[4] https://www.kaggle.com/gaborfodor/use-pretrained-keras-models-lb-0-3

[5] Xception: Deep Learning with Depthwise Separable Convolutions，https://arxiv.org/abs/1610.02357

[6] Very Deep Convolutional Networks for Large-Scale Image Recognition，https://arxiv.org/abs/1409.1556

[7] Deep Residual Learning for Image Recognition，https://arxiv.org/abs/1512.03385

[8] Rethinking the Inception Architecture for Computer Vision，https://arxiv.org/abs/1512.00567

[9] MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications

[10] https://keras.io/applications/

[11] https://www.kaggle.com/wiki/MultiClassLogLoss

[12] 面部识别技术能用来识别鲸鱼，https://mp.weixin.qq.com/s?srcid=0717jR0shJpsHITF5EjsY84z&scene=1&mid=2651651403&sn=f0360e5cdcabf938fd8139a18c9d557f&idx=4&__biz=MjM5MTQzNzU2NA%3D%3D&chksm=bd4ddad88a3a53ce9e84c03dbce03674df9e7e6b36cf6221dce4ed0d0e07222205a98c42282a&mpshare=1#rd

[13] https://www.jianshu.com/p/3e77cefeb49b

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

[15] https://keras.io/