机器学习(进阶)纳米学位毕业项目开题报告

项目名称：猫狗大战
作者：Zax
提交日期：180618
修改 1：180619

一、项目背景

本项目名为“猫狗大战”，原为 Kaggle[1]上的一个已经结束的竞赛项目[2]。数据源来自 Kaggle 官方网站。

猫狗是日常非常常见的宠物，猫狗的图片在互联网上也非常多，所以将猫狗进行分类也是一种非常直接的日常需求，虽然简单，但很有意义。本项目重点在于将深度学习应用于图像识别。“猫狗大战”是个典型的“二分类”问题，预测结果只有“猫”与“狗”两个类别。在完成构建模型后，最终需要针对 Kaggle 所提供的 test 数据集(或图片集)进行预测，并写入一个 csv 文件，提交至 Kaggle 官方网站获取评估分数。评估分数是判断模型是否满足毕业项目要求的唯一标准。

“二分类”问题是所有分类中最为基本的问题，是解决“多分类”问题的基础。而现实生活中更多的是“多分类”问题。

二、问题描述

本项目需要构建一个监督学习(Supervised Learning)[3]的深度学习网络模型，并根据 Kaggle 提供的数据集（分为 train, test 两部分，训练只使用 train 数据集）对模型进行训练，然后使用训练完毕的模型对 test 数据集所有图片进行预测，并按 Kaggle 要求的格式（“sample_submission.csv”，在 Kaggle 官网随 Kaggle 数据集一起提供）在 Kaggle 官方网站提交一个 csv 文件，用于评估模型的分数。

根据项目要求，Kaggle 的评估分数需要达到 kaggle Public Leaderboard 前 10%，根据 Kaggle 官网“猫狗大战”已有的分数排名[4]，10%位置的 score 大约为 0.06149，也即本项目中最终评分不应低于 0.06149，需要说明的是评分数值越小分数越高。

三、输入数据

输入数据是由 Kaggle 提供下载的数据集，下载地址：https://www.kaggle.com/c/dogs-vs-cats/data。数据集分为三部分[5]：

[1] Kaggle 介绍：https://en.wikipedia.org/wiki/Kaggle
[2] 猫狗大战比赛主页：https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition
[3] 监督学习：https://en.wikipedia.org/wiki/Supervised_learning
[4] Kaggle 的猫狗大战 leaderboard:https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/leaderboard
[5] 猫狗大战数据集：https://www.kaggle.com/c/dogs-vs-cats/data

- train.zip：训练数据集，一共有 25000 张图片，其中包含猫 12500 张，狗 12500 张，所以样本分布非常均衡，该数据集主要用于训练搭建的深度学习网络模型。通过浏览图片目录知道图片的格式为：“类别”+“.”+“数字”+“.jpg”，如：“cat.1233.jpg”，可以根据这个规律提取类别标签。其中类别只有 cat 和 dog 两类。训练数据集的用途主要是将其中一部分按比例划分为训练集，另一部分划分为验证集，考虑到猫狗两类过于集中，为防止训练结果过拟合，在使用训练集和验证集前需要将训练集图片进行混洗，将其顺序打乱。通过浏览发现，训练集中还有一些异常图片，见图 1，异常图片有模糊不清的，有卡通形象的，有几乎完全被遮掩的，也有只有一个轮廓的。当然可能还有其他类型的异常图片，显然，需要对这些图片进行异常值清洗，以便使模型学到“正确的”的知识。

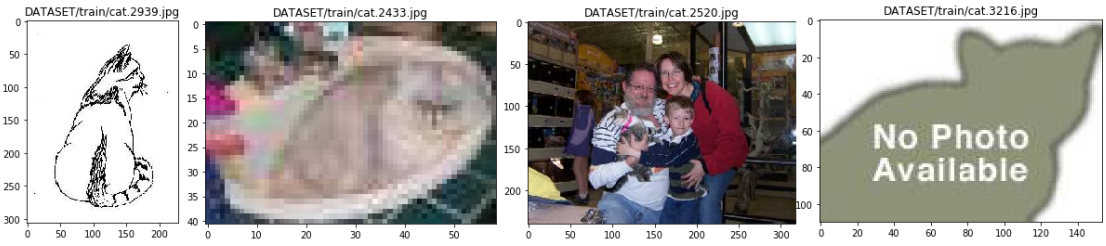

图 1

- test.zip：测试数据集，一共还有 12500 张图片，用于评估模型；
- sample_submission.csv：提交文件的模板，一共有 12500 行数据，用于向 Kaggle 提交预测结果，以便 Kaggle 对模型进行打分。

四、解决方案

本项目计划使用迁移学习和模型融合的方法生成一个预测模型。在进行模型构建之前需要对图片进行清洗，以便清除异常图片，主要的异常图片可能有：非猫非狗的图片、过于模糊的图片、同时出现猫和狗的图片、非实体的猫狗图片（卡通形象等）。

本项目计划执行的步骤：

- 图片清洗：清洗后判断为异常的图片移动到 DATASET/exception 目录下。
  拟使用多个模型对图片进行清洗，取较为平衡（尽可能多地包含异常图片，尽可能少地包含误判的图片，可能需要人工核对）的 top 值，对多个模型获取到的异常图片取并集，作为最终需要清洗掉的异常图片集。
- 搭建模型：
  1） 选择预训练模型：InceptionResNetV2[6]、DenseNet201[7]、Xception[8]。
  以上三个模型都是 Keras 自带的预训练模型[9]，属于 Keras 的应用模块。Keras 提供了带有预训练权值的深度学习模型，这些模型可以用来进行预测、特征提取和微调（fine-tune）。
  2） 训练特征向量：输入为 train 数据集，采用 1）所述的模型分别对输入数据集进

[6] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, Alex Alemi:Inception-v4, Inception-ResNet and the Impact of Residual, Connections on Learning,23 Aug 2016:https://arxiv.org/abs/1602.07261
[7] Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger:Densely Connected Convolutional Networks,28 Jan 2018:https://arxiv.org/abs/1608.06993
[8] François Chollet, Xception: Deep Learning with Depthwise Separable Convolutions, 4 Apr 2017: https://arxiv.org/abs/1610.02357
[9] Keras 提供的预训练模型：https://keras.io/zh/applications/

行训练，并将训练完成后的特征向量保存在 saved_models 文件夹中。

3） 载入特征向量：主要目的是将三个模型的特征向量融合为一个完整的向量空间，完成这一步骤后形成新的输入向量和标签，并对输入的特征进行混洗（shuffle）。

4） 编译模型：选择优化器 optimizer：'adadelta'，损失函数 loss：'binary_crossentropy'，评价指标 metrics：'accuracy'。

5） 训练模型：对融合后的新模型进行训练，并保存最佳的模型文件（命名为：weights.final_model.hdf5），位置位于 saved_model 文件夹。

6） 评价模型：使用 Epochs-Loss 曲线及 Epochs-Acc 曲线对模型进行评价，其中 Epochs 为完整训练的轮数，Loss 为评价指标的损失分数，分为 loss（对训练集划分为 train 和 valid 后，train 对应的损失分数）和 val_loss（valid 数据集上的损失分数），Acc 为评价指标的准确率，分为 acc(对应划分后的 train 数据集)和 val_acc（对应划分后的 valid 数据集）。Epochs 为横坐标，各 metrics 指标为纵坐标，观察曲线特征，比如收敛程度、loss 与 val_loss 位置比较、acc 与 val_acc 位置比较，判断模型的曲线是否收敛、是否过拟合或欠拟合。

- 预测：使用上面搭建好的模型对 test 数据集中的图片进行预测，并套用 Kaggle 提供的模板文件 sample_submission.csv 将预测结果写入一个新的 csv 文件（初步命名为 pred_submission.csv）。
- 提交预测文件，获取 Kaggle 评分，将评分结果与项目要求进行对照，如不满足要求，需要对上述 1)~6)的过程进行检查和修改，并进行迭代，直到达到项目要求为止。

五、基准模型

在本项目中可以设置一个作为参照的基准模型，比如自行搭建一个 CNN 网络。一个典型的 CNN 网络可以如图 2 所示：

```
model = Sequential()
model.add(Conv2D(16, (2, 2), activation='relu', input_shape=input_shape))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(32, (2, 2), activation='relu'))
model.add(Conv2D(32, (2, 2), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (2, 2), activation='relu'))
model.add(Conv2D(64, (2, 2), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(1, activation='sigmoid'))
model.summary()
```

图 2

基准模型隐藏节点的激活函数选用了'relu'，输出分类函数选用了'sigmoid'，中间添加了 Conv2D 和 MaxPooling2D，卷积核大小选用了 2x2，另外添加了 BatchNormalization 防止过拟合。

此外需要设置一个基准阈值，也即本项目所需要达到的目标：Kaggle 排行榜前 10%，也就是 LogLoss 分数不低于 0.06127。

六、评价指标

本项目使用的评价指标也是 Kaggle 采用的评价指标：损失函数 LogLoss。LogLoss 是一种基于交叉熵的损失函数[10]，用于评价错误分类所带来的损失，当预测结果是正确的时候，LogLoss 应该是 0，当预测结果为错误的时候，LogLoss 为-log0，默认 Kaggle 会将 log0 中的 0 处理为一个接近 0 的小数，比如 1e-15，这样当做出错误的预测时，LogLoss 值将是一个 0~+∞的正数。对分类器来说，LogLoss 越小越好。在本项目中 LogLoss 表示为 Loss 评分，作为最终提交到 Kaggle 的结果。

LogLoss 定义为：

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

其中：n为测试集图片数量，$\hat{y}_i$为测试集中图片i预测为预测值的可能性，y_i为测试集中图片i的实际结果，即本来应是猫或狗，也即标签类别，若标签为猫，预测为猫，则y_i为1，否则为0。为了显示方便，可以对预测概率进行处理，比如对预测最小值和最大值都进行限定： $y_i = y_i.\text{clip}(\text{min} = 0.005, \text{max} = 0.995)$。

七、设计大纲

第一步： 数据归类。从 Kaggle 下载数据集并解压缩，分为三个文件：train.zip, test.zip, sample_submission.csv，解压缩后在项目文件夹中生成 train 文件夹、test 文件夹。然后建立一个 DATASET 文件夹，作为分类完毕后供模型使用的数据文件夹。DATASET 文件夹中建立 train 文件夹和 validation 文件夹，在两个文件夹中又分别建立 cats 和 dogs 文件夹。异常图片也放在 DATASET 下的文件夹 exception 中，另建一个 submission 文件夹，用于存储提交的 csv 数据文件，项目中 DATASET 大致目录结构如下：

```
DATASET/
    ImageNetFullClasses.csv
    exception/
    submission/
        sample_submission.csv
    test/
        pics/
    train/
        cats/
        dogs/
```

第二步：数据清洗。主要是找出异常图片，判断为异常的图片移动到 DATASET/exception

[10] LogLoss 定义：
https://en.wikipedia.org/wiki/Loss_functions_for_classification#Cross_entropy_loss_(Log_Loss)

目录下。拟使用多个模型对图片进行清洗，取较为平衡（尽可能多地包含异常图片，尽可能少地包含误判的图片，可能需要人工核对）的 top 值，对多个模型获取到的异常图片取并集，作为最终需要清洗掉的异常图片集。

第三步：搭建模型。完成第一步和第二步后，可以用来建模的图片应该都位于 DATASET/train 下，并按 cats 和 dogs 分好类。

搭建模型的过程大致如下：

1）选择预训练模型：可选模型较多，本项目中选取几个到目前为止得分比较高的预训练模型有：InceptionResNetV2、DenseNet201、Xception。这些模型都是 Keras 提供的预训练模型（原模型来自于 ImageNet，而 ImageNet 一共有 1000 个分类），便于用来进行预测、特征提取和 fine-tune。Keras 提供的预训练模型见表 1[11]：

表 1

模型	大小	Top-1 准确率	Top-5 准确率	参数数量	深度
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.715	0.901	138,357,544	23
VGG19	549 MB	0.727	0.910	143,667,240	26
ResNet50	99 MB	0.759	0.929	25,636,712	168
InceptionV3	92 MB	0.788	0.944	23,851,784	159
InceptionResNetV2	215 MB	0.804	0.953	55,873,736	572
MobileNet	17 MB	0.665	0.871	4,253,864	88
DenseNet121	33 MB	0.745	0.918	8,062,504	121
DenseNet169	57 MB	0.759	0.928	14,307,880	169
DenseNet201	80 MB	0.770	0.933	20,242,984	201

通过表 1 观察各模型的 top 准确率，除了 Xception 和 DenseNet201 之外，还有 Inception 系列，但 InceptionV3 与 InceptionRestNetV2 属于同源，后者是以前者为基础进行了改进，所以最终在两者之中选了 InceptionRestNetV2。

2）训练特征向量：输入为 train 数据集，采用 1）所述的预训练模型分别进行训练，并将训练完成后的特征向量保存在 saved_models 文件夹中。

3）载入特征向量：主要目的是将三个模型的特征向量融合为一个完整的向量空间，完成这一步骤后形成新的输入向量和标签，并对输入的特征进行混洗（shuffle）。

4）编译模型：选择优化器 optimizer：'adadelta'，损失函数 loss：'binary_crossentropy'，评价指标 metrics：'accuracy'。

5）训练模型：对融合后的新模型进行训练，并保存最佳的模型参数文件（命名为：weights.final_model.hdf5），位置位于 saved_model 文件夹。

6）调参：可调的参数主要有 Epochs，dropout，learning-rate(即 lr)。其中主要调节 Epochs 和 dropout。lr 使用较为平衡的默认值即可(Optimizer 使用 Adadelta，建议使用优化器的

[11] 预训练模型比较：https://keras.io/zh/applications/

默认参数[12])。Epochs 可使用 Keras 提供的 EarlyStopping 方法[13]。dropout 参数常用范围为 0~1 之间的小数[14]，常用的调节范围为 0.1~0.9，可以考虑使用循环从 0.1 到 0.9 逐个训练最终模型，取最佳的 dropout 值，可以参考的经验值一般在 0.2~0.5 之间。

7）评价模型：使用 Epochs-Loss 曲线及 Epochs-Acc 曲线对模型进行评价，其中 Epochs 为完整训练的轮数，Loss 为评价指标的损失分数，分为 loss（对训练集划分为 train 和 valid 后，train 对应的损失分数）和 val_loss（valid 数据集上的损失分数），Acc 为评价指标的准确率，分为 acc(对应划分后的 train 数据集)和 val_acc（对应划分后的 valid 数据集）。Epochs 为横坐标，各 metrics 指标为纵坐标，观察曲线特征，比如收敛程度、loss 与 val_loss 位置比较、acc 与 val_acc 位置比较，判断模型的结果是否收敛、是否过拟合或欠拟合。

第四步：预测。使用上面搭建好的模型对 test 数据集中的图片进行预测，并套用 Kaggle 提供的模板文件 sample_submission.csv 将预测结果写入一个新的 csv 文件（初步命名为 pred_submission.csv）。

第五步：提交预测文件，获取 Kaggle 评分。将评分结果与项目要求进行对照，如不满足要求，需对上述步骤进行检查和修改，并进行迭代，直到达到项目要求为止。

八、引用

1. Kaggle 介绍：https://en.wikipedia.org/wiki/Kaggle
2. 猫狗大战比赛主页：https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition
3. 监督学习：https://en.wikipedia.org/wiki/Supervised_learning
4. Kaggle 的猫狗大战 leaderboard: https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition/leaderboard
5. 猫狗大战数据集：https://www.kaggle.com/c/dogs-vs-cats/data
6. Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, Alex Alemi: Inception-v4, Inception-ResNet and the Impact of Residual, Connections on Learning,23 Aug 2016:https://arxiv.org/abs/1602.07261
7. Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger: Densely Connected Convolutional Networks,28 Jan 2018:https://arxiv.org/abs/1608.06993
8. François Chollet, Xception: Deep Learning with Depthwise Separable Convolutions, 4 Apr 2017: https://arxiv.org/abs/1610.02357
9. Keras 提供的预训练模型：https://keras.io/zh/applications/
10. LogLoss 定义：https://en.wikipedia.org/wiki/Loss_functions_for_classification#Cross_entropy_loss_(Log_Loss)
11. 预训练模型比较：https://keras.io/zh/applications/
12. 优化器：https://keras.io/zh/optimizers/
13. EarlyStopping 方法：https://keras.io/callbacks/

[12] 优化器：https://keras.io/zh/optimizers/
[13] EarlyStopping 方法：https://keras.io/callbacks/
[14] Dropout 方法：https://keras.io/layers/core/#dropout

14. Dropout 方法：https://keras.io/layers/core/#dropout