机器学习(进阶)纳米学位

毕业项目

项目名称：猫狗大战
作者：Zax
提交日期：180624

I.问题的定义

1.1 项目概述

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]，其目的是建立一个能够区分“猫”和“狗”的模型。本项目数据源来自 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项目要求的唯一标准（要求 kaggle Public Leaderboard 前 10%）。

“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。

本项目使用了近几年来在 Keras 中排名比较靠前的预训练模型，这些模型在报告后面会有更进一步的介绍。利用这些预训练模型，通过迁移学习的方法对本项目的数据集进行训练，最终取得了比较不错的识别效果。

1.2 问题陈述

本项目目标是构建了一个监督学习(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

---

[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

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。

1.3 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[5]，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 是 0，当预测结果为错误的时候，LogLoss 为-log0，LogLoss 值将是一个 0~+∞之间的正数，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，就能计算出一个具体的而又不是很夸张的正数。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 Kaggle 返回的成绩评分。

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.clip(min = 0.005, max = 0.995)。

II.分析

2.1 数据探索

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[6]：

- 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。
- test.zip：测试数据集，一共还有 12500 张图片，用于评估模型，图片格式为：“数字”+“.jpg”。对测试数据集中的图片进行预测，每一张图片将生成一个预测为“猫”或“狗”的概率。
- sample_submission.csv：提交文件的模板，一共有 12500 行数据，一共有两列，分别为“id”和“label”，与 test 测试数据集中的图片数字序号和预测概率一一对应，该文件用于向 Kaggle 提交预测结果，以便 Kaggle 对模型进行打分。

[5] LogLoss 定义：https://en.wikipedia.org/wiki/Loss_functions_for_classification#Cross_entropy_loss_(Log_Loss)
[6] 猫狗大战数据集：https://www.kaggle.com/c/dogs-vs-cats/data

2.2 探索性可视化

训练数据集中随机抽取 5 张图片显示如图 1 所示：

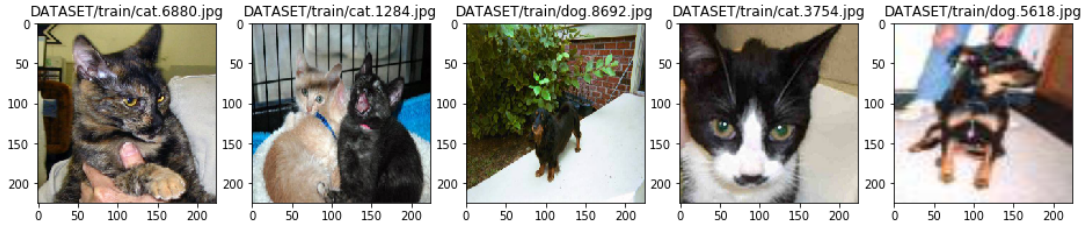

图 1

测试数据集中随机抽取 5 张图片显示如图 2 所示：

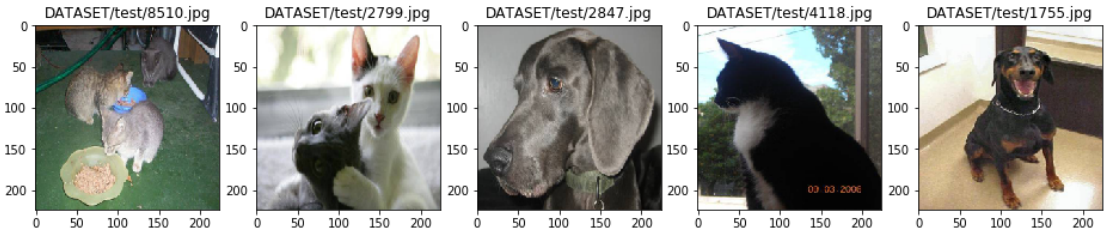

图 2

通过浏览发现，训练数据集中存在一些异常图片，随便找了几张异常图片显示如图 3 所示：

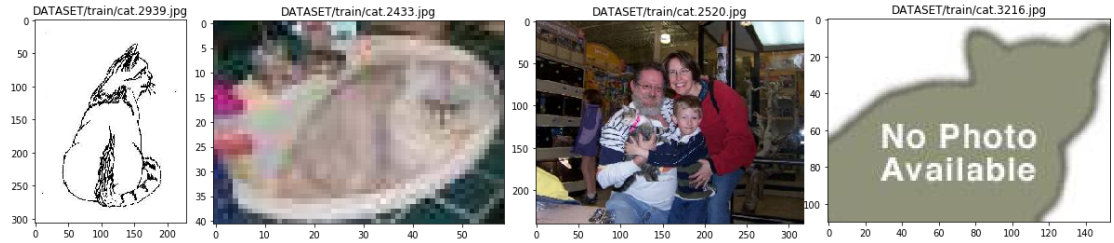

图 3

异常图片主要有模糊不清的，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。

2.3 算法与技术

本项目以深度学习为基础，使用了卷积神经网络、正则化等技巧，通过将 Keras 自带的预训练模型 InceptionResNetV2[7]、DenseNet201[8]、Xception[9]进行迁移学习，最后使用模型融合

[7] 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

[8] Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger:Densely Connected Convolutional Networks,28 Jan 2018:https://arxiv.org/abs/1608.06993

[9] François Chollet. Xception: Deep Learning with Depthwise Separable Convolutions, 4 Apr 2017: https://arxiv.org/abs/1610.02357

的方法，将这三个模型融合为一个可训练的模型，并对该模型进行微调（fine-tune），从而生成一个最终用来预测的实用模型。

2.3.1 深度学习

一般说来，深度学习是人工神经网络的一种，也是机器学习的一个分支。截至到目前机器学习已经经过了三次发展浪潮[10]：20 世纪 40 年代到 60 年代被称为控制论，20 世纪 80 年代到 90 年代被称为连接机制，而第三次浪潮起始于 2006 年，以深度学习闻名，被称为机器学习或人工智能的复兴，目前仍处于人工智能的第三次浪潮。

在人工智能领域的发展初期，一些早期的算法通过对生物学习的模拟，产生了人工神经网络（Artificial Neural Network）。在 20 世纪 50 年代，感知机成为第一个能根据每一个类别输入的样例对权重进行学习的模型。同期出现的还有可以用来调节权重的随机梯度下降算法（Stochastic Gradient Descent，SGD），也是目前深度学习的一个重要算法。

深度学习使用图构成的网络来尝试对数据进行高层次的抽象，并专注于学习数据的特征，从而达到可以使用抽象后的信息来处理新信息的目的。深度学习超越了目前机器学习模型的神经科学观点，诉诸于学习多层次组合这一更普遍的原理，而这一原理也可以用于那些并非受神经科学启发的机器学习框架。第三次浪潮已开始注重于新的无监督学习技术和深度模型在小数据集上的泛化能力。

2.3.2 卷积神经网络

卷积神经网络（Convolutional Neural Network，CNN），是深度学习中常用的一种重要的网络结构，专门用来处理具有类似网格结构的数据。“卷积神经网络”一词表明该网络是用了卷积的数学运算。CNN 是神经科学原理影响深度学习的典型代表。

在深度学习的背景下，常见的卷积神经网络动辄达到几十层甚至上百层，这样的卷积神经网络也被称为深度卷积神经网络。

CNN 是第一个解决重要商业应用的神经网络，并且在当今深度学习商业应用的前沿仍非常活跃。CNN 也是第一批能使用反向传播并有效训练的深度网络之一。

在 CNN 中，卷积的第一个参数通常被称为输入（Input），第二个参数被称为核函数（Kernel function），输出也被称为特征映射（feature map）。

CNN 中的卷积运算有三个重要的思想：稀疏交互（sparse interactions）、参数共享（parameter sharing）、等变表示（equivariant representations）。通过这三种思想，CNN 可以实现自主学习，如一张金毛巡回犬，并不需要告诉 CNN 去寻找任何直线、曲线、鼻子、毛发等等，CNN 就可以从训练集中学习金毛巡回犬的特征值。一张典型的 CNN 图片分解如图 4 所示：

[10] Ian Goodfellow, Yoshua Bengio, Aaron Courville. Deep Learning[M].北京：人民邮电出版社,2017,（8-10）

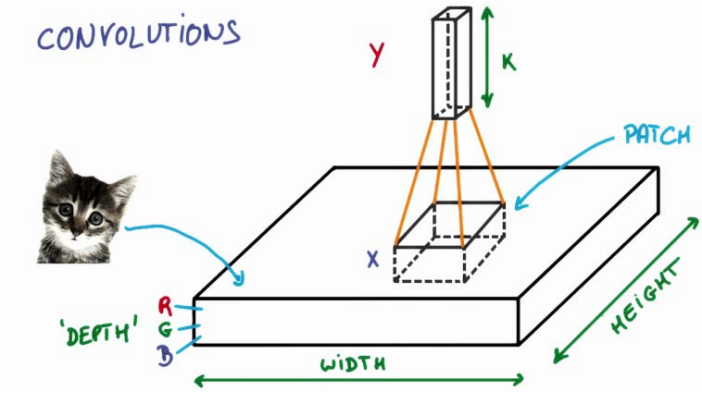

图 4

2.3.3 预训练模型

本项目使用的深度学习框架是 Keras，Kera 的应用模块 Application 提供了带有预训练权重的 Keras 模型，这些模型可以用来进行预测、特征提取和微调（fine-tune）。预训练模型也是迁移学习的基础。

Keras 中可以选择的预训练模型较多，本项目选取了几个到目前为止得分比较高的预训练模型：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。

本项目中使用了三个预训练模型：InceptionResNetV2、DenseNet201、Xception ，其

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

中 InceptionRestNetV2 模型是 Google 发布的一种 CNN 网络，是早期 Inception V3 模型变化而来，从微软的残差网络（ResNet）论文中得到了一些灵感。InceptionRestNetV2 是 Inception V4 加 ResNet，计算量和 Inception v4 相当，属于较大的模型，准确率也更高。网络结构见图 5：

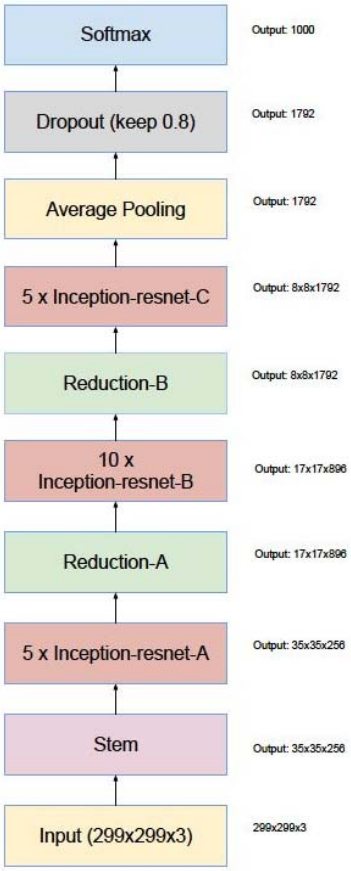

图 5

InceptionRestNet 在 InceptionV4 的基础上引入 residual connection 连接，把 Inception 和 ResNet 结合起来，让网络又宽又深，residual connection 并不会明显提升模型精度，而是会加快训练收敛速度。由于引入了 residual connection，网络太深了会不稳定，不太好训练，故又引入了 scale。scale 的引入不仅解决了上面的问题，而且能够在不降低模型精度的前提下使得网络更加稳定（见脚注 7 中的论文）。

DenseNet201 模型：DenseNet201 是 DenseNet 模型中的一种，201 表示层数。DenseNet 是 RestNet 的扩展。DenseNet 模型一个典型网络图见图 5：

Architecture
Input, $m = 3$
3×3 Convolution, $m = 48$
DB (4 layers) + TD, $m = 112$
DB (5 layers) + TD, $m = 192$
DB (7 layers) + TD, $m = 304$
DB (10 layers) + TD, $m = 464$
DB (12 layers) + TD, $m = 656$
DB (15 layers), $m = 880$
TU + DB (12 layers), $m = 1072$
TU + DB (10 layers), $m = 800$
TU + DB (7 layers), $m = 560$
TU + DB (5 layers), $m = 368$
TU + DB (4 layers), $m = 256$
1×1 Convolution, $m = c$
Softmax

图 6

受 GooLeNet 的启发，DenseNet 各层通过串联的方式结合，并进行分块，每个块被称为 DenseBlock，每个 DenseBlock 的之间层称为 transition layers，由于每个层的输入是所有之前层输出的连接，因此每个层的输出不需要像传统网络一样多。除了在 DenseBlock 内部减少特征图的数量，还可以在 transition layers 中来进一步 Compression。在取得相同分类精度的情况下，DenseNet 比 ResNet 少了 2/3 的参数，而 ResNet 虽然可以收敛到更小的 loss 值，但是最终的 test error 与 DenseNet 相差无几，以上都说明了 DenseNet 参数效率（Parameter Efficiency）很高（见脚注 8 论文）。

Xception 模型：是 google 继 Inception 后提出的对 Inception v3 的另一种改进，主要是采用 depthwise separableconvolution 来替换原来 Inception v3 中的卷积操作。Xception 的网络结构见图 7。

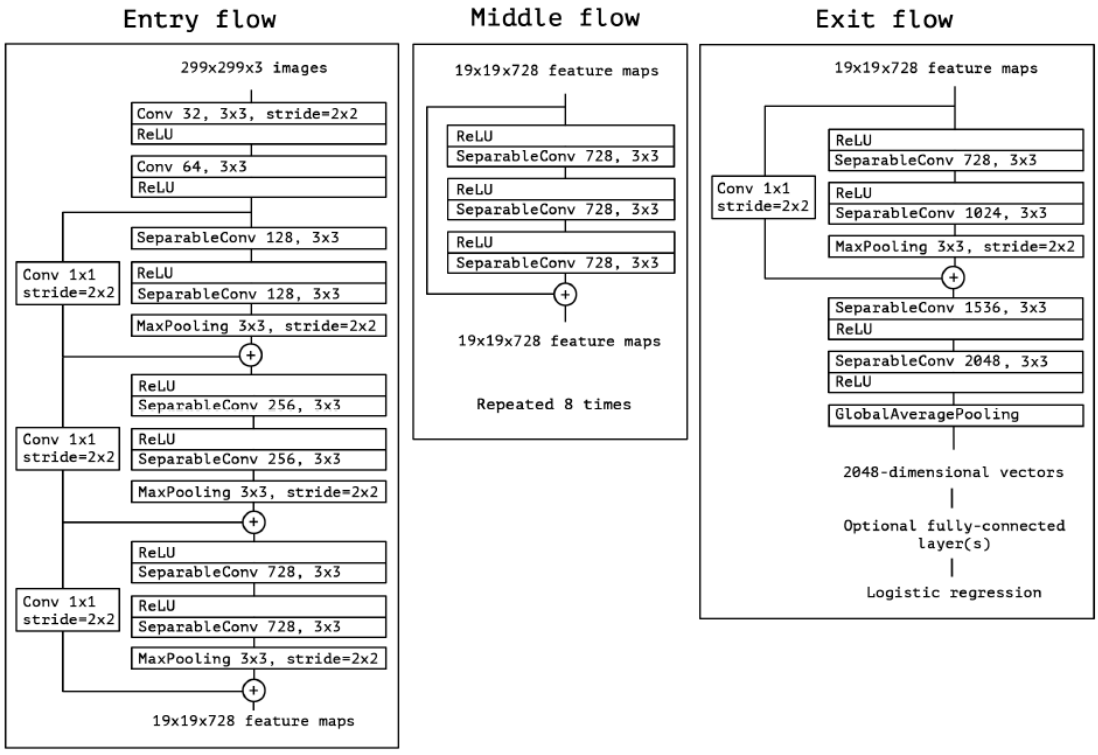

图 7

Xception 作为 Inception v3 的改进，主要是在 Inception v3 的基础上引入了 depthwise separable convolution，在基本不增加网络复杂度的前提下提高了模型的效果。图 8 是 depthwise separable convolution 的示意图：

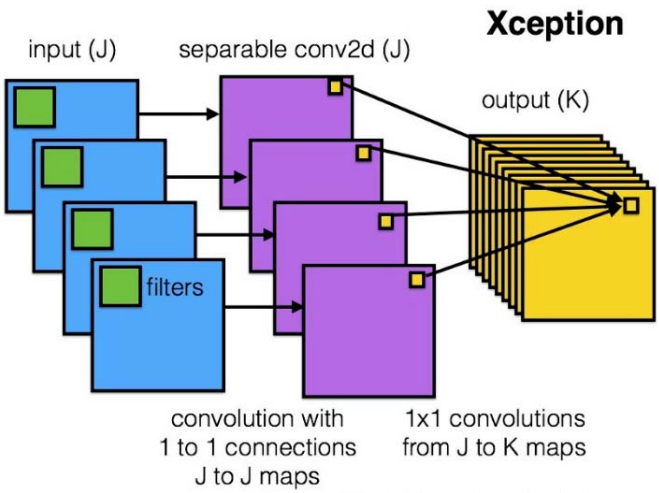

图 8

以上是将传统的卷积操作分成两步，假设原来是 3*3 的卷积，那么 depthwise separable convolution 就是先用 J 个 3*3 卷积核一对一卷积输入的 J 个 feature map，不求和，生成 K 个结果，然后用 J 个 1*1 的卷积核正常卷积前面生成的 K 个结果，求和，最后生成 K 个结果。

2.3.4 迁移学习

迁移学习（Transfer Learning）兴起于 1995 年，有很多别名，如 knowledge transfer, inductive transfer 等等。与迁移学习密切相关的学习技术是多任务学习框架，这个框架尝试同时学习多个任务，即使它们是不同的。多任务学习的一个典型方法是发现能使每个任务受益的共同（潜在）特征。多任务学习的一个典型方法是发现能使每个任务受益的共同（潜在）特征。迁移学习的目的是从一个或多个源任务中提取知识，并将知识应用于目标任务。与多任务学习相比，迁移学习最关心的是目标任务，而不是同时学习所有的源任务和目标任务。在迁移学习中，源和目标任务的作用不再对称。迁移学习允许源空间、任务空间在测试集和训练集中的分布是不同的。图 9 展示了传统机器学习与迁移学习之间的区别[12]：

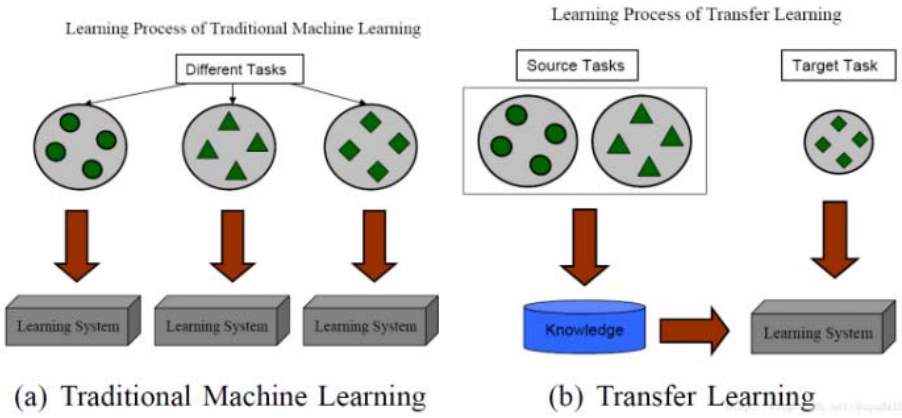

图 9

本项目主要使用了 Keras 提供的三个预训练模型，通过对预训练模型分别进行训练并提取特征向量、载入特征向量，最后融合为新的特征向量。

2.3.5 模型融合

模型融合是 Kaggle 等比赛中经常使用到的一个利器，它通常可以在各种不同的机器学习任务中使结果获得提升[13]。

模型融合是综合考虑不同模型的情况，并将它们的结果融合到一起。一般而言，模型融合主要实现方法：

1） 从提交结果文件中融合：

最简单便捷的方式大概是从竞赛提交结果文件中进行融合，因为这样做并不需要重新训练模型，只需要不同模型的测试结果，通过某种措施融合成一个结果就行。常用的方法有：多数表决法、加权表决法、平均法。

2） stacking：

stacking（也称堆叠）的基本思想是用一些基分类器进行分类，然后使用一个分类器对结果进行整合。堆叠模型因为它的平滑性和突出每个基本模型在其中执行得最好的能力，并且降低执行不佳的基本模型所占的比重，所以将优于每个单个模型。因此，当基本模型显著不同时，堆叠是最有效的。

---

[12] Maxime Oquab, Eon Bottou, Ivan Laptev. Learning and Transferring Mid-level Image Representations Using Convolutional Neural Networks.2014.

[13] KAGGLE ENSEMBLING GUIDE: https://mlwave.com/kaggle-ensembling-guide/,2015.

3） blending：

blending（也称交叉融合法）与 stacking 类似，区别在于 blending 对训练集不是通过 k-fold 的 CV 策略来获取预测值从而生成第二阶段模型的特征，而是建立一个 holdout 集，例如 10%的训练数据，第二阶段的 stacking 模型就基于第一阶段模型对这 10%训练数据的预测值进行拟合，也即 blending 将 stacking 中的 k-fold CV 改成了 holdout CV。

本项目中使用的模型融合方法采用了一种类似结果融合平均的方法，将三个预训练模型的特征抽取后先进性合并，然后加入全连接层，正则化 Dropout 后使用 sigmoid 激活输出。

2.4 模型基准

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，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。

III.方法

3.1 数据预处理

3.1.1 异常图片检测

在前面已有提及，在此进一步说明本项目中实际使用的异常图片检测流程：

1）从网上找到 ImageNet 分类的 1000 个类别列表，并放在文件夹“DATASET”下，将其转换为字典并存储为“ImageNet_full_classes_dict”，然后获取其中为“猫”或“狗”对应的类别编码，代码：

```
# 获取猫和狗的具体类别,输入：字典 dic,主要是前面已经获取到的 ImageNet_full_classes_dict，输出：只含猫和狗的具体类别 code 列表
def get_classes(dic):
    cat_dog_classes_code = []
    if dic:
        for k in dic:
            if dic[k] == "猫" or dic[k] == "狗":
                cat_dog_classes_code.append(k)
    else:
        print("ImageNet_full_classes_dict is empty, please check!")
    return cat_dog_classes_code
# print(ImageNet_full_classes_dict)
valid_classes = get_classes(ImageNet_full_classes_dict)
```

生成的类别编码格式：['n02085620', 'n02085782',…]

2）选择一个预训练模型并使用模型对图片进行预测，关键代码：

```
# 获取预测结果
def get_preds(model, X, top = 40):
    results = []
    model = model
    for x in tqdm(X):
        pred = model.predict(x)
        pred_decode = decode_predictions(pred, top = top)[0]
        results.append(pred_decode)
    return results
```

3）选择预训练模型，本项目中选择了 3 个预训练模型：Xception，DenseNet201，InceptionResNetV2：

```
# model: Xception
model_xception = Xception(weights='imagenet')
# model: DenseNet201
model_DenseNet201 = DenseNet201(weights='imagenet')
# model: InceptionResNetV2
model_InceptionResNetV2 = InceptionResNetV2(weights='imagenet')
```

4）找出训练集中模型预测非猫非狗的图片，相关说明见代码中的注释，需要说明的是选择合适的 top 值很重要，这里的处理方法是选择其中一个模型，对不同 top 值时的异常图片预测效果进行观察，原则：

- 观察 top=3,10,30,40,50,60 时异常图片误判的情况，发现取 top=40 的能尽量多地包含异常图片且误判较少。
- 取以上三种模型预测结果(except_index_InceptionResNetV2_40, except_index_DenseNet201_40, except_index_xception_40)的并集，尽量避免单模型的不足之处。

```
# 定义函数:找出非猫非狗的图片，返回 except_index，格式：[0, 0, 1, 1, 1]，假设有 5 张图片，0 表示正常，1 表示异常
def get_except_index(preds):
    except_index = []
    right_index = []
    right_list_num = []
    for pred in preds:
        right_list_num = [item[0] for item in pred]
        right_list_num = [1 for item in right_list_num if item in valid_classes]
        right_index.append(sum(right_list_num))
    for i in right_index:
        if i == 0:
            except_index.append(1)
        else:
            except_index.append(0)
    return except_index
def exception_info(model, top = 50, shape=img_shape):
```

```
    shape_224 = (224,224,3)
    if list(shape)==list(shape_224):
        inpt = X2
    else:
        inpt = X
    preds = get_preds(model, inpt, top = top)
    except_index_name = get_except_index(preds)
    except_index_sum = sum(except_index_name)
    # 异常图片数量
    print("图片总数：{}, top = {}时，异常图片的数量：{}".format(len(except_index_name),
top, except_index_sum))
    return except_index_name
# to show exception pics preds when top = 40
except_index_xception_40 = exception_info(model_xception, top = 40)
#使用 InceptionV3 筛选 top = 40 时的异常图片，返回 except_index
except_index_DenseNet201_40 = exception_info(model_DenseNet201, top = 40,
shape=img_shape2)
#使用 InceptionResNetV2 筛选 top = 40 时的异常图片，返回 except_index
except_index_InceptionResNetV2_40 = exception_info(model_InceptionResNetV2, top
= 40)
```

最后一共找到异常图片 115 张。通过人工核查这些异常图片，准确率能达到 90%以上。

3.1.2 图片分类

将所有训练集中的图片按名称进行归类，在提取图片名称的同时也获取该图片对应的分类，保证每张图片与其分类一一对应。对应的分类图片目录：

```
DATASET/
    test/
        pics/
    train/
        cats/
        dogs/
```

3.1.3 数据归一化处理

主要是针对输入数据进行归一化处理，也就是 Keras 中的“preprocess_input”，其原理是使用训练数据集中的平均通道值对图像数据进行零值处理，使图像所有的点的和为 0，即零平均处理（zero-mean）。经过处理后，数据会处于-1 与+1 之间，而通道的顺序为 RGB。这一步比较重要，对预测效果影响较大。

3.1.4 分类数据转换

主要是针对图片的分类进行数据转换，由于训练使用的指标是交叉熵（cross-entropy），故需要将图片分类的数据转换成二进制的分类矩阵。

3.1.5 混洗

为了尽可能地避免连续性同类的数据特征导致过拟合，本项目中将清洗后的训练图片进行混洗（shuffle）。这里使用的混洗来自 sklearn.utils 中的 shuffle 方法[14]：
X_train, y_train = shuffle(X_train, y_train)。

3.2 执行过程

3.2.1 构建环境

项目执行过程中使用的机器学习库是 Keras，对应的 CNN 框架是 Google 开源的 Tensorflow。由于使用 CNN 处理图片所需的计算量非常大，故本项目使用了 GPU 对计算进行加速，计算平台使用了亚马逊的 AWS 云平台的 p3.2xlarge，显卡为 Tesla P100。在安装 Tensorflow 和 Keras 的同时还安装了 CUDA 和 cuDNN，从而能够支持 ubuntu 系统使用 GPU 进行计算。

3.2.2 数据准备

数据准备主要分为以下几个步骤：

1）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。

2）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 目录下以备后面检查使用。使用的模型为 Keras 上评分较高的三个预训练模型：InceptionResNetV2、DenseNet201、Xception。数据清洗详细过程可以参见本报告 3.1.1 异常图片清洗。

3.2.3 搭建模型

完成第 3.2.1 和 3.2.2 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：

[14] Shuffle 方法：http://scikit-learn.org/stable/modules/generated/sklearn.utils.shuffle.html

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。

2） 通过预训练模型搭建最终模型：将预训练模型的特征向量文件合并成一个独立的特征向量空间，然后添加全连接 Dense 层，通过二分类激活函数 sigmoid（activation='sigmoid'）将输入压平，之后添加 Dropout 层，输出预测结果。最终模型的结构如图 10 所示：

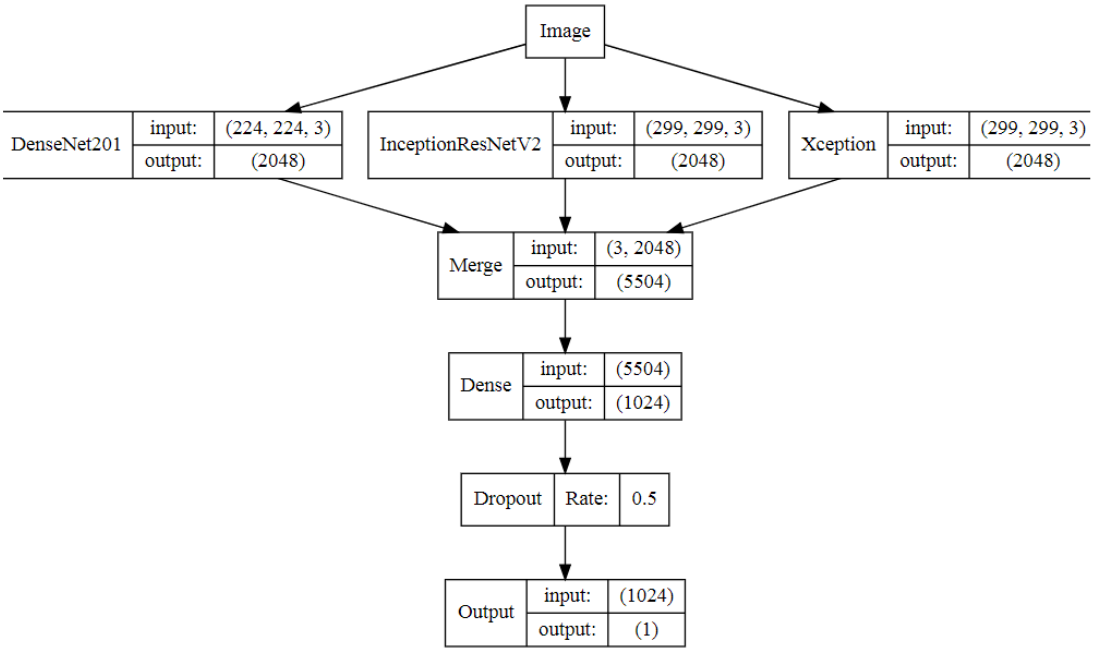

图 10

3） 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。

4） 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）。

3.2.4 编译和训练

1） 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。

3.2.5 调参

可调的参数主要有 Epochs 和 Dropout，lr（learning-rate）使用较为平衡的默认值即可(Optimizer 使用 Adadelta，根据 Keras 文档中的建议采用了优化器的默认参数[15])。Epochs 可使用 Keras 提供的 ModelCheckPoint 方法中的 save_best_only 参数，保存 val_loss 最小时的模

[15] 优化器：https://keras.io/zh/optimizers/

型参数。Dropout 参数常用范围为 0~1 之间的小数[16]，常用的调节范围为 0.1~0.9，因可以尝试的数值较少，故本项目中没采用 for 循环，而是手动从 Dropout 值为 0.1 到 0.9 逐个检查 Epochs-Loss 曲线，取最佳的 Dropout 值，最终采用的 Dropout 值为 0.85。

3.2.6 评价模型

主要使用 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。各曲线见图 11：

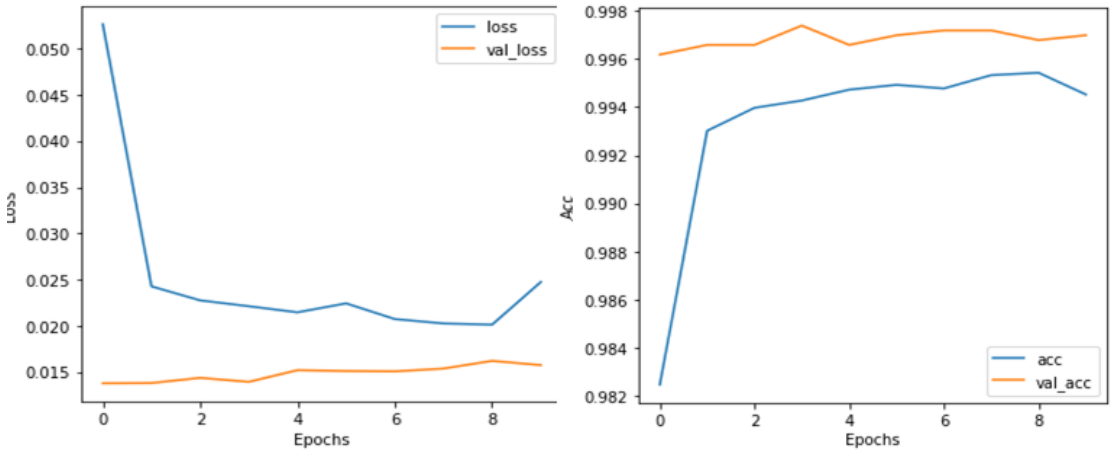

图 12

3.2.7 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（命名为 pred_submission.csv）。

3.2.8 提交预测文件

提交预测文件，获取 Kaggle 评分，本项目 Kaggle 评分为 0.03678，在 Public Leaderboard 中位于第 7 名。远高于项目要求的 10%位置（132 名）的 score：0.06149。Kaggle 评分见图 13：

[16] Dropout 方法：https://keras.io/layers/core/#dropout

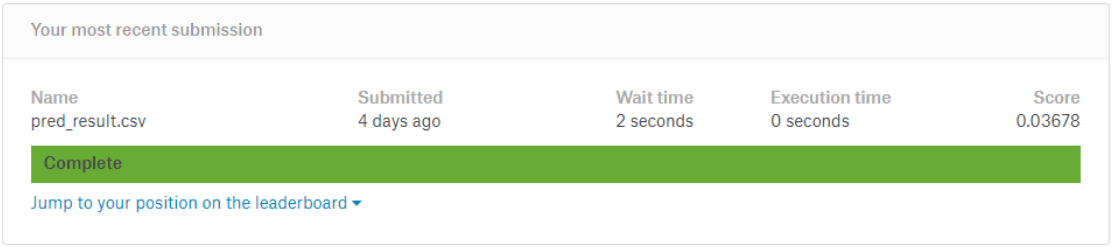

图 14

IV.结果

4.1 模型的评估与验证

从 3.2 节执行过程中可以看出，实际使用的预测模型 Epochs-Loss 曲线及 Epochs-Acc 曲线主要特征：所有 Epochs 中，验证集 Val_loss 始终都小于训练集 Loss，而验证集 Val_acc 始终都大于训练集 acc。整体上曲线较为稳定，没有出现剧烈的震荡。从 Kaggle 返回的评分来看，对排名第七的成绩还是较为满意的，说明模型的预测效果满足项目要求。

4.2 合理性分析

本项目针对模型融合的方法做了一次有益的尝试，可以发现，模型能够收敛并能很好的拟合数据，在最后也获得了不错的 LogLoss 分数，达到了 Kaggle top10 的项目要求。

V.项目结论

5.1 结果可视化

Epochs-Loss 曲线和 Epochs-Acc 曲线见 3.2.6 中的图 7。为了更直观得查看模型在测试图片集上的预测效果，对测试图片集前 10 个图片的预测结果见图 15：

	id	label
0	1	0.995
1	2	0.995
2	3	0.995
3	4	0.995
4	5	0.005
5	6	0.005
6	7	0.005
7	8	0.005
8	9	0.005
9	10	0.005

图 16

而测试图片集前 10 张图片显示见图 17：

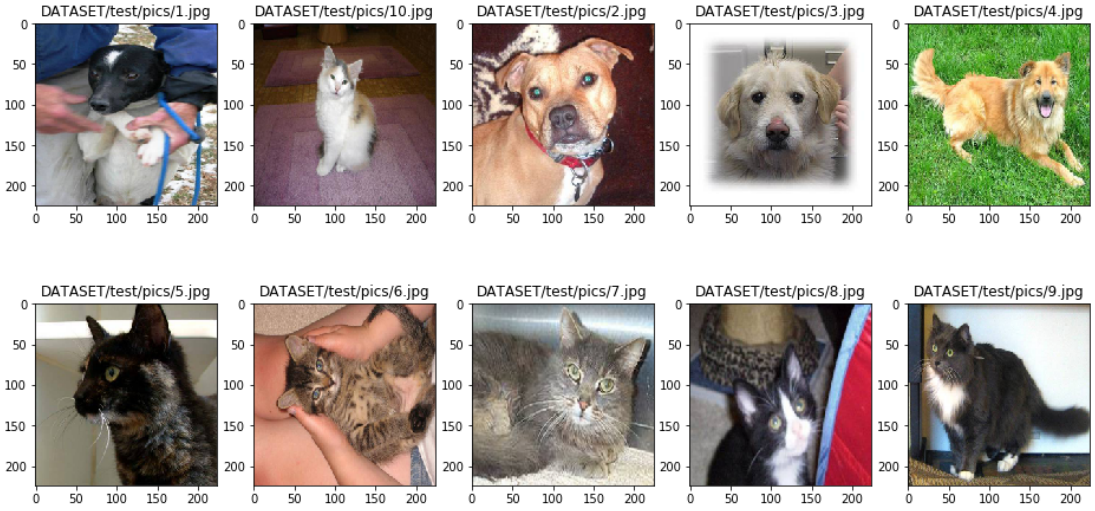

图 18

需要说明的是，当结果接近 1 的时候，为确定为狗的概率，结果接近 0 的概率是确定为猫的概率。比较图片与预测结果，可以知道图片序号 1，2，3，4 为狗，而其余为猫，预测与图片实际分类完全一致。

5.2 对项目的思考

本项目只是采用了一种比较流行的迁移学习加模型融合的方法，对于本项目，第七的排名说明还有进步的余地，可以继续尝试一些其他的方法，比如单模型调优、数据增强、其他的模型融合方法，只有不断尝试各种方法才能形成感官上和理性上的认识，在选择建模方法时也会更加方便、快捷。

本项目只是比较基础的二分类，通过本项目掌握好图片分类的基本技巧，这些技巧也可以很好地迁移到其他图片分类项目中，满足日常业务实际项目的需求。

本项目的一个重点是数据清洗，也即异常图片判断，怎样自动而准确地判断出异常图片又不误判是一个非常重要的问题，在本报告中提出了一个解决思路（见 5.3 中的 1））。

5.3 需要作出的改进

可以做出的改进主要有三个方面：

1） 本项目清洗后的 105 张异常图片量并不大，完全还可以通过迭代方法对图片进行再次清晰，将误清理的图片移动到 train 文件夹下，使异常图片清洗更为精准，训练模型也可以学会更多的特征，理论上是有可能提高预测的准确率的。

其中迭代的过程可以这样进行：模型生成后提交 Kaggle 打分，若分数尚可，可以将该模型用作清洗模型，将比预训练模型更为精准。清洗后也可以在此检查，并重复模型融合和生成训练模型的过程，重新对测试图片进行预测，通过 LogLoss 曲线观察清洗效果。

2） 可以考虑使用数据增强的方法；

3） 可以尝试融合更多的模型，观测效果，也有可能提高预测准确率及 Kaggle 评分。

VI.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. LogLoss 定义：https://en.wikipedia.org/wiki/Loss_functions_for_classification#Cross_entropy_loss_(Log_Loss)
6. 猫狗大战数据集：https://www.kaggle.com/c/dogs-vs-cats/data
7. 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
8. Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger: Densely Connected Convolutional Networks,28 Jan 2018:https://arxiv.org/abs/1608.06993
9. François Chollet, Xception: Deep Learning with Depthwise Separable Convolutions, 4 Apr 2017: https://arxiv.org/abs/1610.02357
10. Ian Goodfellow, Yoshua Bengio, Aaron Courville. Deep Learning[M].北京：人民邮电出版社,2017,（8-10）
11. 预训练模型比较：https://keras.io/zh/applications/
12. Maxime Oquab,Eon Bottou,Ivan Laptev.Learning and Transferring Mid-level Image Representations Using Convolutional Neural Networks.2014.
13. KAGGLE ENSEMBLING GUIDE: https://mlwave.com/kaggle-ensembling-guide/,2015.
14. Shuffle 方法：http://scikit-learn.org/stable/modules/generated/sklearn.utils.shuffle.html
15. 优化器：https://keras.io/zh/optimizers/
16. Dropout 方法：https://keras.io/layers/core/#dropout