

Udacity DSND Capstone Project

Dog Breed classification

2020.4.8

Author: dong junlong

目录

1	项目信息	3
1.1	项目概述.....	3
1.2	问题描述.....	3
1.3	开发环境.....	4
1.4	模型优化.....	4
1.4.1	激活函数.....	4
1.4.2	梯度下降优化	5
1.5	评价指标.....	6
2	数据描述	6
3	方法步骤	6
3.1	导入数据.....	6
3.2	检测人脸.....	6
3.3	检测小狗.....	7
3.3.1	数据预处理	7
3.3.2	ResNet-50 模型.....	7
3.4	创建分类小狗品种的 CNN.....	7
3.4.1	数据预处理	7
3.4.2	模型实现.....	8
3.5	使用迁移学习分类小狗品种的 CNN	9
3.5.1	VGG-16 模型.....	9
3.5.2	Xception 模型	9
3.6	模型改进.....	10
3.7	编写算法.....	10
3.8	测试验证.....	10
4	结果分析	10
5	总结	12
6	参考文献	12

1 项目信息

1.1 项目概述

本项目是优达学城数据科学家的毕业项目，在机器学习纳米学位还是人工智能纳米学位都倍受青睐。该项目的目标是将狗狗的图片根据品种进行分类。



代码托管：

<https://github.com/dongjunlong/DSND-Capstone-Project/tree/master/dog-project>

本文实现流程源码可以在作者 [GitHub](#) 上查看。

1.2 问题描述

本项目主要任务是完成狗狗种类的自动识别和分类。这个项目会用到卷积神经网络（CNNs），在这个项目中，需要搭建一个管道模型来处理真实用户提供的图片。比如给定一张小狗的图像，算法将能够大致识别小狗的品种，但如果提供的是人脸图像，算法需要检测出这是人脸，同时代码能够识别出和人脸最相似的小狗品种。

预测图中小狗的品种是一项非常难的挑战。因为即使人类要分辨布列塔尼猎犬（Brittany）和威尔斯激飞猎犬（Welsh Springer Spaniel）也很困难。



这种不同种类，但差异很小的情况还有例如卷毛寻回犬 [Curly-Coated Retrievers](#) 和美国水猎犬 [American Water Spaniels](#)

卷毛寻回犬

美国水猎犬



还有一种复杂情况，狗狗属于同种类，但差异又很大，比如拉布拉多有黄色、巧克力色和黑色品种。基于视觉的算法需要克服这种同一类别差异很大的问题，并决定如何将所有这些不同肤色的小狗分类为相同的品种。

黄色拉布拉多

巧克力色拉布拉多

黑色拉布拉多



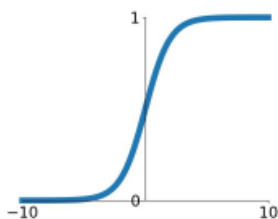
1.3 开发环境

开发环境	版本
Python	3.6.3
Keras	2.0.9
matplotlib	2.1.0
numpy	1.12.1
scikit-learn	0.19.1

1.4 模型优化

1.4.1 激活函数

1. sigmoid

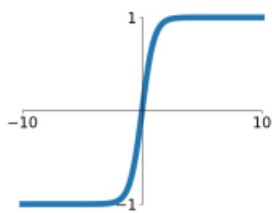


$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Sigmoid 函数的取值范围在 (0, 1) 之间，单调连续，求导容易，一般用于二分类神经网络的输出层。

Sigmoid 函数饱和区范围广，容易造成梯度消失，Sigmoid 函数包含 \exp 指数运算，运算成本也比较大。

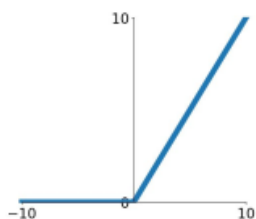
2.tanH



$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

\tanh 函数的取值范围在 $(-1, 1)$ 之间，单调连续，求导容易。收敛速度更快，但 \tanh 函数与 Sigmoid 函数一样，也存在饱和区梯度消失问题。

3.ReLU



$$\max(0, x)$$

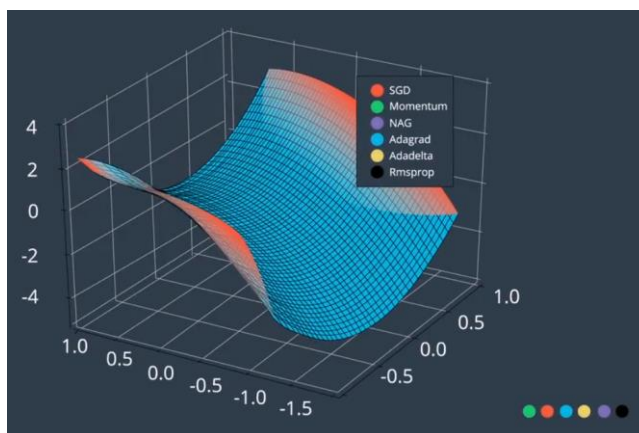
ReLU 函数是最近几年比较火热的激活函数之一。相比 Sigmoid 和 \tanh 函数，其主要优点包括以下几个方面：

- 没有饱和区，不存在梯度消失问题。
- 没有复杂的指数运算，计算简单、效率提高。
- 实际收敛速度较快，大约是 Sigmoid/ \tanh 的 6 倍。
- 比 Sigmoid 更符合生物学神经激活机制。

本项目使用的深度神经网络算法主要采用 ReLU 作为激活函数。

1.4.2 梯度下降优化

Keras 本身提供了很多优化算法，常用优化器如下：



- SGD

随机梯度下降。它使用了以下参数：

学习速率

动量（获取前几步的加权平均值，以便获得动量而不至于陷在局部最低点）。

Nesterov 动量（当最接近解决方案时，它会减缓梯度）。

- Adam

Adam (Adaptive Moment Estimation) 使用更复杂的指数衰减，不仅仅会考虑平均值（第一个动量），并且会考虑前几步的方差（第二个动量）。

- RMSProp

RMSProp (RMS 表示均方根误差) 通过除以按指数衰减的平方梯度均值来减小学习速率。

1.5 评价指标

本次实现的算法主要测试狗狗分类的准确性，所以采用 **Accuracy** 作为评价指标。

2 数据描述

本项目用于测试和训练的数据集合包括小狗图片数据以及人脸图片数据：

Data set	Number
dog categories	133
dog images	8351
training dog images	6680
validation dog images	835
test dog images	836
human images	13233

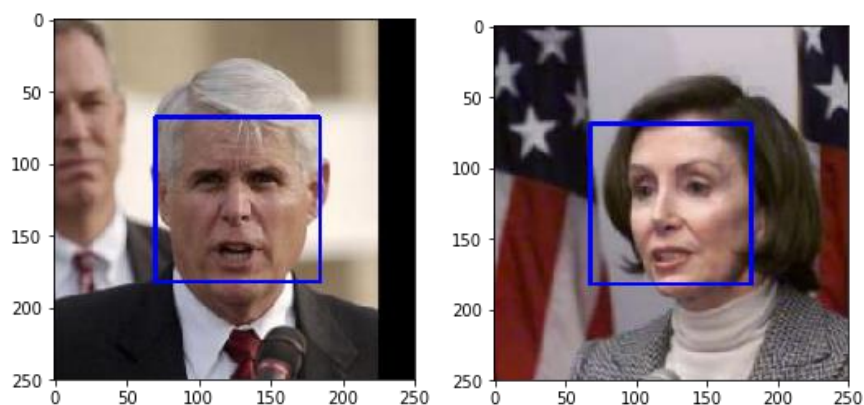
3 方法步骤

3.1 导入数据

使用 **scikit-learn** 库的 **load_files** 函数导入小狗数据，人脸数据路径存储在 **numpy** 数组。

3.2 检测人脸

使用 **OpenCV** 实现的 **Haar** 特征级联分类器（**Haar feature-based cascade classifiers**）来检测图片中的人脸。**OpenCV** 提供了很多预训练的人脸检测器，它们以 **XML** 文件的形式存储，使用此检测器从样本图像中检测人脸，检测结果：



3.3 检测小狗

3.3.1 数据预处理

Keras CNN 要求输入的是一个 4D 的 numpy 数组（我们也称之为 4 维张量），其形状为：

$(nb_samples, rows, columns, channels)$

其中 `nb_samples` 表示图片（或样本）的数量，`rows`、`columns` 和 `channels` 表示每个图片各自的行数、列数和通道数。

处理过程如下：

1. 实现函数接受一个彩色图片的文件路径字符串，返回一个适用于 Keras CNN 训练的 4 维张量。这个函数首先加载图片，然后将其重新调整为 224×224 像素的形状

图片被转成 numpy 数组，之后被整理成 4D 张量。因为我们处理的是彩色图片，每个图片有 3 个通道。还有，我们处理的是单张图片，返回的张量的形状就是： $(1, 224, 224, 3)$ 。

2. 实现函数接受一个图片路径的字符串 numpy 数组，返回一个 4 维张量。该张量的形状是：

$(nb_samples, 224, 224, 3)$

其中，`nb_samples` 是输入的图片路径中对应的样本数，或者说图片的数目。

3. 将要输入 ResNet-50 的 4 维向量准备好后，RGB 图像的通道要重排，以转换成 BGR。所有预训练的模型都需要再进行归一化，即必须将每张图片的每个像素都减去平均像素值 $[103.939, 116.779, 123.68]$ ，按 RGB 顺序排列，是用 ImageNet 中所有图像的像素值计算得到的），上述功能在导入的 `preprocess_input` 函数中已完成代码实现。

3.3.2 RESNET-50 模型

使用预训练的 ResNet-50 模型来检测图片中的狗狗。该功能通过 `predict` 方法实现。该方法会返回一个 numpy 数组。该数组的第 i 项数值表示该模型预测此图片属于第 i 个 ImageNet 分类的概率。通过对预测概率向量取 `argmax`，我们得到一个整数，该整数表示该图片所属的类别的序号。

3.4 创建分类小狗品种的 CNN

3.4.1 数据预处理

1. 按照检测小狗的预处理步骤，得到所有图像的 4D 张量

(nb_samples,224,224,3)

2.我们通过将每个图像中的每个像素除以 255 来重新缩放图像

3.4.2 模型实现

1. 模型架构

```
from keras.layers import Conv2D, MaxPooling2D, GlobalAveragePooling2D
from keras.layers import Dropout, Flatten, Dense
from keras.models import Sequential

model = Sequential()

### TODO: Define your architecture.
#20200328
model.add(Conv2D(filters=16, kernel_size=2, padding='same', activation='relu',input_shape=(224, 224, 3)))
model.add(MaxPooling2D(pool_size=2))
model.add(Conv2D(filters=32, kernel_size=2, padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Conv2D(filters=64, kernel_size=2, padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(GlobalAveragePooling2D())
model.add(Dense(133, activation='softmax'))
#end

model.summary()
```

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 224, 224, 16)	208
max_pooling2d_5 (MaxPooling2	(None, 112, 112, 16)	0
conv2d_5 (Conv2D)	(None, 112, 112, 32)	2080
max_pooling2d_6 (MaxPooling2	(None, 56, 56, 32)	0
conv2d_6 (Conv2D)	(None, 56, 56, 64)	8256
max_pooling2d_7 (MaxPooling2	(None, 28, 28, 64)	0
global_average_pooling2d_2 ((None, 64)	0
dense_2 (Dense)	(None, 133)	8645
Total params: 19,189		
Trainable params: 19,189		
Non-trainable params: 0		

- 1) 构建一个 CNN，输入层接受的是 224 x 224 x 3（对应于高 224、宽 224、深 3 的三维数组）的图片。第一层卷积层具有 16 个过滤器，每个宽和高分别为 2。在进行卷积操作时，过滤器每次跳转 1 个像素，激活函数采用 ReLU
- 2) 添加最大池化层，降低卷积层的维度，在最大池化层中使用 2x2 窗口
- 3) 依次增加卷积层和最大池化层
- 4) 添加全局平均池化层

5) 输出是 133 个狗狗种类，使用 softmax 激活函数

2. 模型编译

```
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
```

3.5 使用迁移学习分类小狗品种的 CNN

3.5.1 VGG-16 模型

这个模型使用预训练的 VGG-16 模型作为固定的特征提取器（fixed feature extractor），也就是说，VGG-16 的最后一层卷积层的输出会作为我们模型的输入。我们仅仅加入了一层全局平均池化层（global average pooling layer）和一个全连接层（fully connected layer），后者包含一个节点，该节点和所有狗狗品种的节点相连，并使用 softmax 激活函数。

```
VGG16_model = Sequential()
VGG16_model.add(GlobalAveragePooling2D(input_shape=train_VGG16.shape[1:]))
VGG16_model.add(Dense(133, activation='softmax'))

VGG16_model.summary()
```

Layer (type)	Output Shape	Param #
global_average_pooling2d_3 ((None, 512)		0
dense_3 (Dense)	(None, 133)	68229
Total params: 68,229		
Trainable params: 68,229		
Non-trainable params: 0		

模型编译参数（优化器，损失函数，评价指标）与 3.4 小结模型编译参数一致。

3.5.2 XCEPTION模型

与 VGG-16 模型思路一样，Xception 建立如下网络结构：

```

from keras.applications import Xception

Xception_model_transfer = Sequential()
Xception_model_transfer.add(GlobalAveragePooling2D(input_shape=train_Xception.shape[1:]))
Xception_model_transfer.add(Dense(133, activation='softmax'))

Xception_model_transfer.summary()

```

Layer (type)	Output Shape	Param #
global_average_pooling2d_4 ((None, 2048)		0
dense_4 (Dense)	(None, 133)	272517
Total params: 272,517		
Trainable params: 272,517		
Non-trainable params: 0		

模型编译参数（优化器，损失函数，评价指标）与 3.4 小结模型编译参数一致。

3.6 模型改进

1. 使用 early stopping 减少训练时间
2. 对采用的 Xception 模型采用不同的优化参数：rmsprop, Adam, Nadam, SGD, RMSprop

3.7 编写算法

基于以上分类模型实现识别算法，将图像的文件路径作为输入，并首先判断图像中是否包含人脸、小狗，或二者都不含。

- 如果在图像中检测到了小狗，则返回预测的品种。
- 如果在图像中检测到了人脸，则返回相似的小狗品种。
- 如果二者都没检测到，则输出错误消息。

3.8 测试验证

最后，导入几张测试图片验证算法，图片包括小狗、猫、老虎、狼、人、其他图片。

4 结果分析

1. 导入数据集：导入 133 种小狗种类图片，共计 8351 张，人类头像数据 13233 张；
2. 检测人脸：对于人脸数据集中的前 100 张图像，100 张图像检测到了人脸，对于小狗数据集中的前 100 张图像，有 11 张图像检测到了人脸
3. 检测小狗：对于人脸数据集中的图像，有 0% 图像检测到了小狗，对于小狗数据集中的图像，100% 图像检测到了小狗。

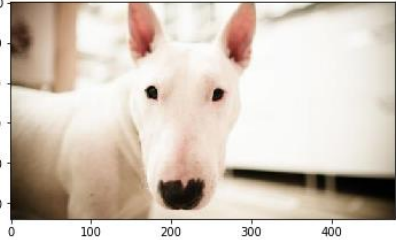
- 创建 CNN 分类模型：测试集准确率在 epochs = 10 情况下可以达到 4.3062%，略好于随机猜测，随机猜测的正确概率约为 1/133，准确率不到 1%，初步建立的模型基本满足预期效果。
- 使用 CNN 分类迁移学习模型（VGG-16）：测试集准确率在 epochs = 20 情况下可以达到 49.5215%
- 使用 CNN 分类迁移学习模型（Xception）：测试集准确率在 epochs = 20 情况下可以达到 84%以上
- Xception 模型使用不同优化策略的模型准确率对比：

CNN	Optimizers	Accuracy
Xception	rmsprop	84.5694%
	Adam	85.6459%
	Nadam	84.3301%
	SGD	85.1675%
	RMSprop	85.4067%

8. 算法验证


- 算法对狗类识别表现的很好，准确的识别出牛头梗，法国斗牛犬，北京犬等

look at this picture:




Hello, dogs!
You are a Bull_terrie.

look at this picture:



Hello, dogs!
You are a French_bulldo.


look at this picture:



Hello, dogs!
You are a Pekinges.

- 对人类头像也可以准确的辨别，并认为她和腊肠犬（Dachshund）很像，哈哈

look at this picture:

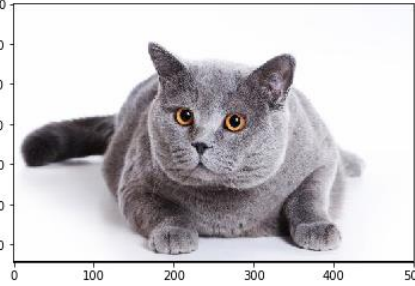


Hello, humans!
If you were a dog, you are a Dachshun.



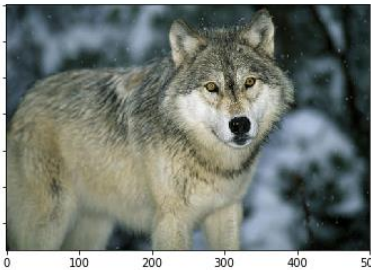
- 对猫和狼也准确的识别出不属于狗和人类

look at this picture:



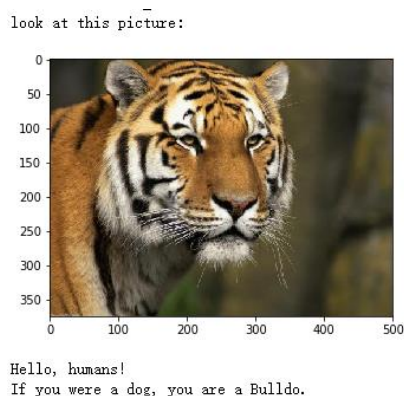
You are neither dogs nor humans.

look at this picture:



You are neither dogs nor humans.

4) 算法将老虎识别成了人脸，人脸识别算法仍然存在误差。但人们常形容一个人“虎头虎脑”，也就可以理解了。



5 总结

1. 结果比预期的要好，人脸识别存在一定误差，容易将小狗头像识别成人脸
2. Xception 在不同优化器策略下准确率表现基本一致
3. 可以尝试其他新的预训练模型，VGG19、Resnet50、InceptionV3、InceptionResNetV2、MobileNet、DenseNet 等等
4. 需要模型参数进行调整，尝试不同优化器的参数，不同损失函数、学习速率等等
5. 增加高质量的训练数据

6 参考文献

<https://keras.io/>

<https://keras.io/optimizers/>

<https://www.jianshu.com/p/0acd30a23e4e>

<https://yq.aliyun.com/articles/567420>