Udacity Logo

项目

Dog Breed Classifier

此部分属于 Deep Learning Nanodegree Program

项目审阅

注释

与大家分享你取得的成绩！

Requires Changes

还需满足 2 个要求 变化

很不错的工作和尝试！ 你离通过这个项目只差一点啦~
仔细阅读我的一些注解，按照要求完成内容，针对建议的地方进行一些模型优化，我相信你下次提交可以顺利通过的！
此外，提一点建议：希望你的学习过程能够不局限于我们提供出的示范性代码，去看一些论文、博客，大胆地做各种各样的尝试，并且将你做过的各种尝试、效果、分析记录在notebook里（事实上这也是我们让你自己动手完成project的初衷 ）。这样你会收获更多哦~加油！

推荐你阅读以下材料来加深对 CNN和Transfer Learning的理解:

- CS231n: Convolutional Neural Networks for Visual Recognition
- Building an Image Classifier
- Tips/Tricks in CNN
- Transfer Learning using Keras
- Transfer Learning in TensorFlow on the Kaggle Rainforest competition
- Transfer Learning and Fine-tuning

相关论文:

- [VGG16] VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION
- [Inception-v1] Going deeper with convolutions
- [Inception-v3] Rethinking the Inception Architecture for Computer Vision
- [Inception-v4] Inception-ResNet and the Impact of Residual Connections on Learning
- [ResNet] Deep Residual Learning for Image Recognition
- [Xception] Deep Learning with Depthwise Separable Convolutions

提交文件

本次提交包含所有必需的文件。

提交了要求中的所有文件。

第一步：检测人类

提交包含狗狗与人脸数据集前 100 张图片中，检测出的人脸的百分比。

Well done!
人脸图片中人脸识别率为: 99.00%
狗狗图片中人脸识别率为: 11.00%

你可以通过以下书写方式来简化代码：

```
def detect(detector, files):
    return sum([1 if detector(f) else 0 for f in files]) / len(files)
print('human: {:.2f}%'.format(detect(face_detector, human_files_short) * 100))
print('dog: {:.2f}%'.format(detect(face_detector, dog_files_short) * 100))
```

表明 Haar 级联检测是否是一种合适的人脸检测技术。

很不错的分析！
进一步提升人脸识别的准确度，可以尝试HOG(Histograms of Oriented Gradients)或一些基于深度学习的算法，如YOLO(Real-Time Object Detection algorithm)、FaceNet等。此外，你可以使用[imgaug]来对训练集进行增强、扩充，以增加训练集中的多样性。

补充阅读材料:

- Face Detection using OpenCV
- Haar caascade classifiers
- How can I understand Haar-like feature for face detection?
- 这个知乎专栏介绍了目前主流的基于深度学习的人脸识别算法。

第二步：检测狗狗

提交包含狗狗与人脸数据集前 100 张图片中，检测出的狗狗的百分比。

Perfect！
人脸图片中狗的识别率为: 1.00%
狗狗图片中狗的识别率为: 100.00%

你可以通过以下书写方式来简化代码：

```
print('human: {:.2f}%'.format(detect(dog_detector, human_files_short) * 100))
print('dog: {:.2f}%'.format(detect(dog_detector, dog_files_short) * 100))
```

第三步：建立一个CNN区分狗狗的种类（从头开始）

提交指明了一个 CNN 模型架构。

实现了模型并对模型进行了很棒的分析！□
使用 `GlobalAveragePooling2D` 是个明智的选择，相比 `Flatten` ，`GlobalAveragePooling2D` 可以大量减少模型参数，降低过拟合的风险，同时显著降低计算成本，这也是现在主流的一些CNN架构的做法。

我建议你增加一些Dropout [Paper]层来避免模型过拟合，或添加BatchNormalization [Paper]层来降低Covariate Shift并加速运算过程，这也是主流CNN架构中的常见做法。你可以在每个 `Conv2D` 或 `Dense` 层后、 `Activation` 前添加 `BatchNormalization` 层。这个视频演示了 `BatchNormalization` 是如何工作的。

如果你决定使用 `BatchNormalization` 层，可以参考以下代码：

```
model.add(Conv2D(16, (3, 3), strides=(1, 1), padding='valid'))
model.add(MaxPooling2D((2, 2)))
model.add(BatchNormalization())
model.add(Activation('relu'))
```

将 `MaxPooling2D` 提至 `BatchNormalization` 和 `Activation` 前和放在它们后面是等价的，但是放在前面可以减少模型运算量。

更进一步，你可以尝试不同的模型结构，如更多的卷积层和全连接层、更多的节点数、使用不同类型的正则化层（ `Dropout` 、 `BatchNormalization` 等）、使用不同的权值初始化方案（ `truncated_normal` 、 `xavier` 等）、使用不同的激活函数（ `LeakyReLU` 、 `eLU` 等）、抉择使用 `Flatten` 还是 `GlobalAveragePooling2D` 等。
在实际应用中，你需要根据场景的不同来设计不同的模型架构、使用不同的超参数。对比各类结构和超参数给模型带来的影响，有助于你更好的理解模型的结构。

Changes Required: 请在输出层使用正确的激活函数

输出层的激活函数应该是 `softmax` 而不是 `relu` ，因为我们的任务是多分类的，通过 `softmax` 可以预测出不同种类的概率。

训练过的模型在测试数据集上至少达到 1% 的准确度。

Changes Required: 模型未得到正确训练

由于你的输出层激活函数设置错误，模型几乎没有得到训练，得到1%左右的概率基本上是"猜测"得出的。请在输出层使用`softmax`激活函数。

提交指出了训练算法所用的 epochs 数。

作为实验，你可以适当提升epochs数来对模型进行更进一步的训练，直到验证集误差不再有所提升为止，看看你设计的这个模型的极限能到多少吧~

如果你想让算法自动选择epoch参数，并且避免epoch过多造成过拟合，我推荐你使用Keras中提供的early stopping callback（提前结束）方法。early stopping可以基于一些指定的规则自动结束训练过程，比如说连续指定次数epoch验证集准确率或误差都没有进步等。你可以参照[Keras' callback]官方文档来了解更多。

第五步：建立一个CNN区分狗狗的种类

提交下载了对应于 Keras 与训练模型（VGG-19, ResNet-50, Inception, or Xception）的关键特征。

选择了ResNet，很不错！
更进一步，你可以尝试不同的特征提取模型，如VGG19, InceptionV3或 Xception，然后对比这些模型的性能与优劣，并分析这些模型的性能为什么不同。因为代码模板都是一致的，这并不会占用你多少时间。
除此之外，我更推荐你尝试Xception，它在众多图像识别领域中拔得头筹。在本项目的预测任务中，它能够轻松达到85%以上的测试集合准确率。

提交指明了一种模型架构

Good job! 你按照要求实现了模型！

提交通过说明代价函数与优化方式编译结构。

Great work! 很好的定义了损失函数和优化器。
你选择了`categorical_crossentropy`作为损失函数、`rmsprop`作为优化器，干的不错！
但是，我更推荐你使用Adam [Paper] 作为优化器，这也是目前最常使用的优化器算法。想要了解更多的话， An overview of gradient descent optimization algorithms这篇文章介绍了当前流行的一些优化器算法的优劣比较，Usage of optimizers in Keras这篇文章介绍了Keras中各类优化器的使用方法。

提交应指出为什么这一架构会在分类任务中成功，以及为什么早期的尝试不成功。

不错的解释！

更多阅读材料：

- ImageNet: VGGNet, ResNet, Inception, and Xception with Keras
- ResNet, AlexNet, VGGNet, Inception: Understanding various architectures of Convolutional Networks
- Systematic evaluation of CNN advances on the ImageNet

提交使用模型的检查点训练模型，并将拥有最佳交叉验证损失的模型权重保存下来。

训练过程中，注意到第4次epoch之后验证误差就几乎没有提升了，同时因为你保存了最优模型，意味着你后面的训练都是在浪费计算资源；同时也观察到，第20次epoch时，验证误差在0.823左右，而训练误差已经降至0.006左右，这说明模型出现了过拟合。思考并尝试尽量减轻这种过拟合现象吧~

提示：

- 添加dropout层可以很有效的避免模型过拟合；
- 添加batch normalization层可以降低Covariate Shift并加速运算过程，也能带来一些降低过拟合的效果；
- 数据增强（data augmentation）也可以增加模型的鲁棒性和泛化能力。

你可以用可视化的形式将训练过程中的loss曲线输出到notebook中，具体参考Display Deep Learning Model Training History in Keras这篇文章，这样可以让训练过程更为直观，你可以更方便地判断模型是否出现了欠拟合或过拟合。

提交读取了模型获得最小交叉验证损失的模型权重。

干得不错！你保存并读取了验证误差最小的模型！

测试数据集上的准确度达到了 60% 或更多。

很棒！测试集准确度达到了81.1005%。
你可以继续优化你的网络架构和参数选择或者尝试不同的bottleneck features，根据我的经验，ResNet最好可以优化到85%以上的准确率！而Xception可以优化到88%以上准确率！

提交包含一个满足如下要求的函数，它以文件路径作为输入，并返回由 CNN 预测的狗品种。

第六步：写出你自己的算法

提交使用第五步建立的 CNN 模型检测狗的种类。提交应对不同种类的输入图片有着不同的输出结果，并且提供实际（或最接近的）狗的种类。

出色地实现了算法！
将dog_detector放在face_detector之前进行判断是一个明智的做法，因为前者的准确率更高。

你可以参考以下更为简洁的写法。

```
def predict(img_path):
    img = cv2.imread(img_path)
    cv_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    plt.imshow(cv_rgb)
    plt.show()
    if dog_detector(img_path):
        print('Hi, dog! I guess you are a {}!'.format(Resnet50_predict_breed(img_path)))
    elif face_detector(img_path):
        print('Hello, human! You look like a {}!'.format(Resnet50_predict_breed(img_pat
h)))
    else:
        print("Error!")
```

第七步：测试你的算法

提交应至少检测了 6 张图片，其中至少包含了 2 张人脸图片与 2 张狗狗图片。

尝试了许多不同的图片，很有趣~□
对提升模型性能提出了很好的建议。干得不错！□

以下是我对改进模型提出的建议，希望对你有帮助：

1. **模型融合（Model Ensembling）**
   通过利用一些模型融合的技术，如voting、bagging、blending以及staking等，可以显著提高模型的准确率与鲁棒性，且几乎没有风险。
2. **更多的数据**
   对于深度学习（机器学习）任务来说，更多的数据意味着更为丰富的输入空间，可以带来更好的训练效果。我们可以通过数据增强（Data Augmentation）、对抗生成网络（Generative Adversarial Networks）等方式来对数据集进行扩充，同时这种方式也能提升模型的鲁棒性。
3. **更换人脸检测算法**
   尽管OpenCV工具包非常方便并且高效，Haar级联检测也是一个可以直接使用的强力算法，但是这些算法仍然不能获得很高的准确率，并且需要用户提供正面照片，这带来的一定的不便。所以如果想要获得更好的用户体验和准确率，我们可以尝试一些新的人脸识别算法，如基于深度学习的一些算法。
4. **多目标监测**
   更进一步，我们可以通过一些先进的目标识别算法，如RCNN、Fast-RCNN、Faster-RCNN或Masked-RCNN等，来完成一张照片中同时出现多个目标的检测任务。

学员 FAQ