浙江大学

本科实验报告

课程名称：	人工智能
姓　　名：	王若鹏
学　　院：	信息与电子工程学院
专　　业：	电子科学与技术
学　　号：	3170105582
指导教师：	王东辉

2020 年 4 月 27 日

基于机器视觉的深度学习：垃圾分类

1. 背景

自今年 7 月 1 日起，上海市将正式实施 《上海市生活垃圾管理条例》。垃圾分类，看似是微不足道的“小事”，实则关系到 13 亿多人生活环境的改善，理应大力提倡。

垃圾识别分类数据集中包括玻璃 (glass) 、硬纸板 (cardboard) 、金属 (metal) 、纸 (paper) 、塑料 (plastic) 、一般垃圾 (trash) ，共 6 个类别。生活垃圾由于种类繁多，具体分类缺乏统一标准，大多人在实际操作时会“选择困难”，基于深度学习技术建立准确的分类模型，利用技术手段改善人居环境。

2. 实验要求

- 建立深度神经网络模型，并尽可能将其调到最佳状态。
- 绘制深度神经网络模型图、绘制并分析学习曲线。
- 用准确率等指标对模型进行评估。

3. 开发环境

- 编程语言：Python 3.5
- 开发工具：Anaconda、Python shell
- 训练框架：Keras、TensorFlow
- 开发平台：Mo 人工智能训练平台
- 操作系统：MacOS Mojave 10.14.6

4. 实验内容与算法设计

4.1 数据集简介

该数据集包含了 2307 个生活垃圾图片。数据集的创建者将垃圾分为了 6 个类别，如下表所示。物品都是放在白板上在日光/室内光源下拍摄的，压缩后的尺寸为 512 x 384。

序号	中文名	英文名	数据集大小
1	玻璃	glass	457
2	纸	paper	540
3	硬纸板	cardboard	370
4	塑料	plastic	445
5	金属	metal	380
6	一般垃圾	trash	115

导入数据集成功后路径：

data_path = "./datasets/la1ji1fe1nle4ishu4ju4ji22-momodel/dataset-resized"

4.2 图片预处理 preprocessing

图片生成器 ImageDataGenerator: keras.preprocessing.image 模块中的图片生成器，主要用以生成一个 batch 的图像数据，支持实时数据提升。训练时该函数会无限生成数据，直到达到规定的 epoch 次数为止。同时也可以在 batch 中对数据进行增强，扩充数据集大小，增强模型的泛化能力，比如进行旋转，变形，归一化等等。

图片生成器的主要方法：

- fit(x, augment=False, rounds=1)：计算依赖于数据的变换所需要的统计信息(均值方差等)。
- flow(self, X, y, batch_size=64, shuffle=True, seed=None, save_to_dir=None, save_prefix='', save_format='png')：接收 Numpy 数组和标签为参数,生成经过数据提升或标准化后的 batch 数据，并在一个无限循环中不断的返回 batch 数据。
- flow_from_directory(directory): 以文件夹路径为参数，会从路径推测 label，生成经过数据提升/归一化后的数据，在一个无限循环中无限产生 batch 数据。

核心代码如下所示：先通过 ImageDataGeneraotr 对图片进行放缩、翻转等操作，然后生成训练集 train_generator 和验证集 validation_generator。

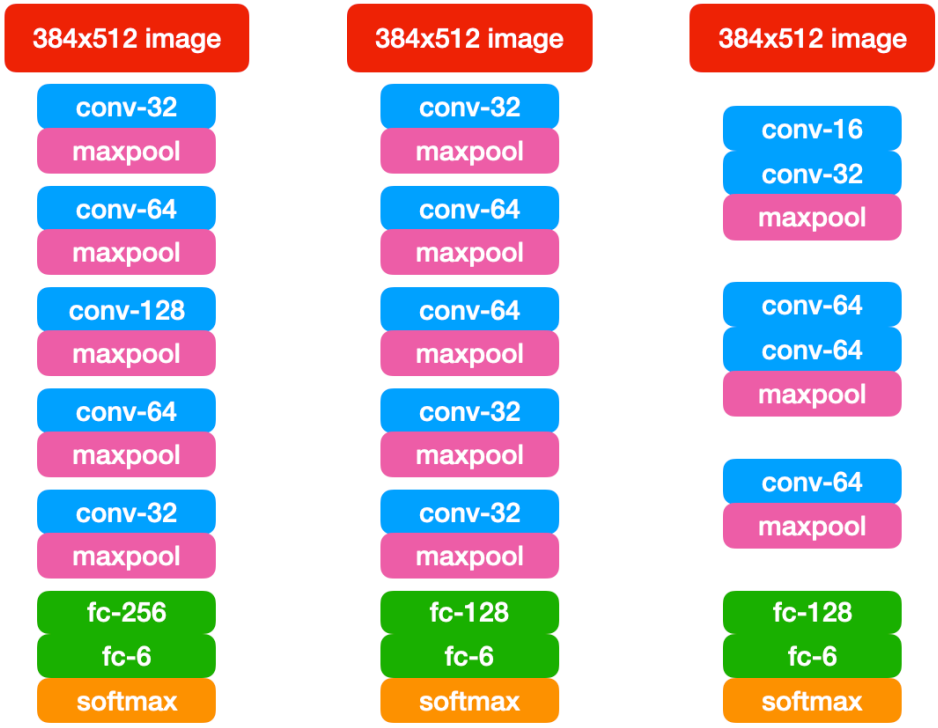
```
batch_size = 64
width, height = 384, 512
train_datagen = ImageDataGenerator(
        rescale=1./255,
        shear_range=0.2, zoom_range=0.2,
        width_shift_range=0.2, height_shift_range=0.2,
        horizontal_flip=True, vertical_flip=True,
        validation_split=0.1)

test_datagen = ImageDataGenerator(rescale=1. / 255, validation_split=0.1)

train_generator = train_datagen.flow_from_directory(
    data_path, target_size=(width, height), batch_size=batch_size,
    class_mode='categorical', subset='training', seed=0)

validation_generator = test_datagen.flow_from_directory(
    data_path, target_size=(width, height), batch_size=batch_size,
    class_mode='categorical', subset='validation', seed=0)
```

4.3 创建并训练深度学习模型 mymodel

通过 Keras 构建深度学习模型的步骤如下：

- 定义模型——创建一个模型并添加配置层
- 编译模型——指定损失函数和优化器，并调用模型的 compile() 函数，完成模型编译。
- 训练模型——通过调用模型的 fit() 函数来训练模型。
- 模型预测——调用模型的 evaluate() 或者 predict() 等函数对新数据进行预测。

Keras 的核心数据结构是 model，一种组织网络层的方式。最简单的模型是 Sequential 顺序模型，它由多个网络层线性堆叠。经过不断设计和调试，我的最好的 3 个模型结构为：



在模型的建立过程中，我以顺序模型编写代码，如下所示：

```
input_shape = (384, 512, 3)
model = Sequential()
model.add(SeparableConv2D(filters=32, kernel_size=3, activation='relu', padding='same', input_shape=input_shape))
model.add(MaxPooling2D(pool_size=2, strides=2))

model.add(SeparableConv2D(filters=64, kernel_size=3, activation='relu', padding='same'))
model.add(BatchNormalization(axis=1))
model.add(MaxPooling2D(pool_size=2, strides=2))

model.add(SeparableConv2D(filters=128, kernel_size=3, activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=2, strides=2))

model.add(SeparableConv2D(filters=64, kernel_size=3, activation='relu', padding='same'))
model.add(BatchNormalization(axis=1))
model.add(MaxPooling2D(pool_size=2, strides=2))

model.add(SeparableConv2D(filters=32, kernel_size=3, activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=2, strides=2))

model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(6, activation='softmax'))
```

对于模型的编译，由于是多分类问题，损失函数 loss 采用 categorical_crossentropy，优化器采用 Adam，准确率 accuracy 作为特定指标。对于模型的训练，epoch 选取 100，其他参数默认，如下所示：

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

tensorboard = TensorBoard(log_dir)

d = model.fit_generator(train_data,
                epochs=100,
                steps_per_epoch=len(train_data),
                validation_data=test_data,
                validation_steps=len(test_data),
                callbacks=[tensorboard])
```

4.4 评估模型 evaluate

evaluate 模块的主要功能是加载模型并对其进行评估，绘制学习曲线、测试集数据的 loss 值、准确率及混淆矩阵等评价指标。其中，准确率曲线的绘制代码如下：

```
plt.figure(2)
plt.plot(res.history['accuracy'], label='acc') # 绘制模型训练过程中的准确率曲线，标签是 acc
plt.plot(res.history['val_accuracy'], label='val_acc') # 绘制模型训练过程中的平均准确率曲线，标签是 val_acc
plt.legend(loc='upper right') # 绘制图例,展示出每个数据对应的图像名称，图例的放置位置为默认值。
plt.xlabel('epoch')
plt.ylim(0, 1)
plt.savefig('./results/evlpic/dnn20-acc.png')
plt.show() # 展示图片
plt.close(2)
```

加载模型，获取最终的 loss 和 acc，输出结果：

```
model = load_model(save_model_path) # 加载模型
loss, accuracy = model.evaluate_generator(test_data) # 获取验证集的 loss 和 accuracy
print("\nLoss: %.2f, Accuracy: %.2f%%" % (loss, accuracy * 100))
```

4.5 模型预测 predict

predict 模块的主要功能是加载模型并对输入的图片进行垃圾分类。为了提高准确率，我载入了 3 个效果最好的模型，对 3 个预测结果进行平均。因为经过测试，发现用单个模型最好能达到 10 个照片成功识别 9 个的水平，然而三者结合就能取长补短，核心代码如下：

```
# 获取图片的类别，共 'cardboard','glass','metal','paper','plastic','trash' 6 个类别
img = image.img_to_array(img) # 把图片转换成为numpy数组
img = 1.0/255 * img # 图片放缩
# expand_dims的作用是把img.shape转换成(1, img.shape[0], img.shape[1], img.shape[2])
x = np.expand_dims(img, axis=0)

# 模型预测
y1 = model1.predict(x)
y2 = model2.predict(x)
y3 = model3.predict(x)
y = (y1+y2+y3)/3

labels = {0: 'cardboard', 1: 'glass', 2: 'metal', 3: 'paper', 4: 'plastic', 5: 'trash'}
y_predict = labels[np.argmax(y)] # 获取输入图片的类别
```

4.6 main 函数

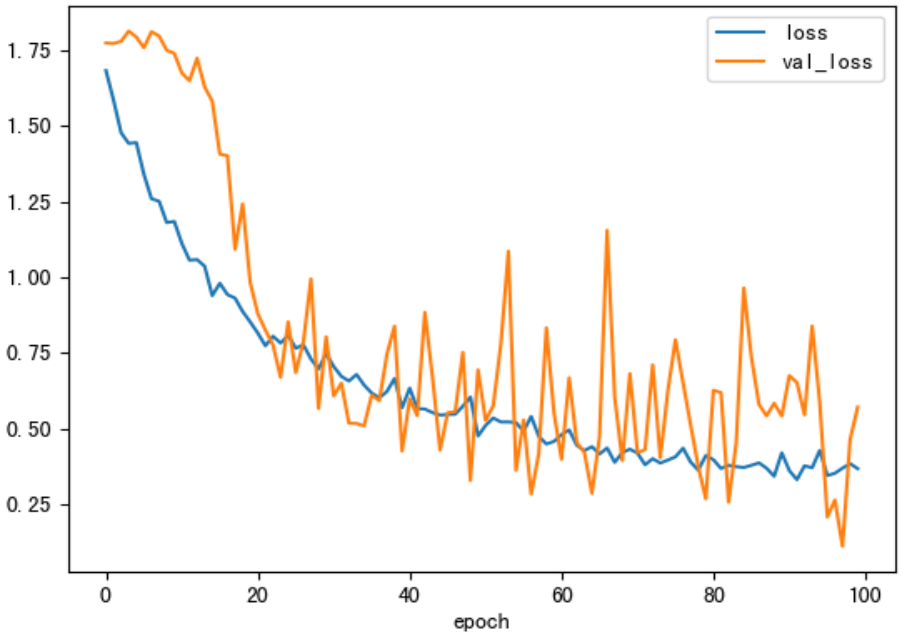
main 函数的功能是将各个模块予以整合：按照获取数据、创建训练保存模型、评估模型的顺序进行。核心代码如下：

```
data_path = "./datasets/la1ji1fe1nle4ishu4ju4ji22-momodel/dataset-resized"  # 数据集路径
save_model_path = 'results/myDNN27.h5'  # 保存模型路径和名称
log_path = './results/logs/dnn27' # 模型训练日志

# 获取数据
train_data, test_data = preprocessing(data_path)
# 创建、训练和保存模型
res, model = mymodel(train_data, test_data, save_model_path, log_path)
# 评估模型
evaluate_mode(test_data, save_model_path, res)
```

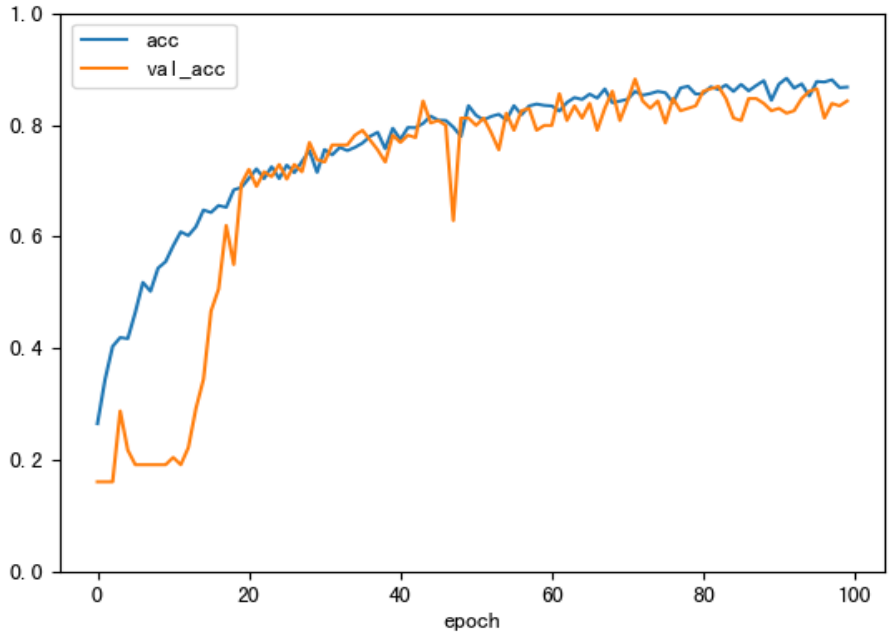
5. 实现结果

5.1 学习曲线

以 myDNN26.h5 模型为例，对训练过程中的各 epoch，训练集的 loss 和验证集的 loss 变化如下图所示。可见训练集的 loss 下降较为稳定，而验证集不太稳定。



对训练过程中的各 epoch，训练集的 accuracy 和验证集的 accuracy 变化如下图所示。可见训练集的准确率上升较为稳定，最终维持在 87%左右，验证集不太稳定，但两者的趋势大体上是一致的。



5.2 测试结果

对给的 test.jpg 进行测试，输出结果为 metal，答案正确。

```
2020-05-03T08:49:29.346196641Z Loss: 0.34, Accuracy: 86.46%
2020-05-03T08:49:46.128328276Z 照片里的垃圾为：metal
2020-05-03T08:49:47.471264795Z SYSTEM: Finishing...
2020-05-03T08:49:49.667507602Z SYSTEM: Done!
```

最后把训练好的三个表现最好模型的 24、26、27 代入进行测试，能够正确识别 10 个图片中的 10 个，正确率 100%：

测试详情

测试点	状态	时长	结果
在 10 张照片上测试模型	✓	36s	正确数，10/10

6. 评价

- 优点：以较少的参数、不太复杂的模型，实现了相对较高的识别准确率。
- 缺点：准确率的提升还有很大空间，可通过模型预训练、加深通道数等方式改进。虽然系统测试中 10 个全部识别正确，但在比赛中的准确率只有 75%.

7. 心得体会

这是我第一次完成深度学习相关的项目，整个过程让我学到了很多。垃圾分类是当下的热点，能用所学的人工智能知识解决生活中的问题是一件非常快乐的事情。

在课堂上学习过相关内容后就开始写本次大作业，由于是第一次写神经网络，几乎所有地方都是陌生的，需要现学。好在 notebook 给的非常详细，再结合 Keras 的官方指南，能写出一个网络不是一件困难的事。然而麻烦在于，如何设计一个性能优异的网络。经过了大量的实践，调整了各种参数，学习网络上多分类问题的一些经验性的做法，经过大半个月的时间，终于训练出了性能还算可以的网络。虽然不及排行榜前几名 99%的准确率，但我也很满足了。希望自己以后继续钻研相关的理论，从而在建立模型训练模型上面得到更好的效果。

8. 参考资料

Keras 官方教程：https://keras-cn.readthedocs.io/

以及 GitHub、CSDN 的各种关于深度神经网络的学习博客

One more thing …

我把自己的照片输入训练好的深度学习网络进行识别，得到的结果是 paper。希望以后我能写出不是学术“垃圾”的 paper！