

【Kaggle 竞赛】数据准备

2018 年 11 月 30 日 BY HARLEY · 0 评论

这篇文章的标题纠结了半天，因为在做深度学习的工作时，数据是非常重要的，第一步的工作也是准备数据，这中间我们需要做很多工作包括数据输入、数据预处理、数据增强等，我个人把这一步的工作命名为数据准备，当然也可以有其他命名。

前言：在我们做图像识别的问题时，碰到的数据集可能有多种多样的形式，常见的文件如 jpg、png 等还好，它可以和 tensorflow 框架无缝对接，但是如果图像文件是 tif 等 tensorflow 不支持解码的文件格式，这就给程序的编写带来一定麻烦。

环境准备

系统：Windows10/Linux 系统

软件：Python3、TensorFlow 框架、和常用的 Python 库，数据准备阶段主要是 os、cv2、numpy、skimage、csv 等。

处理流程

不同的数据集有着不同的程序设计流程，但大致都遵循以下处理流程：

- 文件名获取（主要是获取文件地址集）
- 读取文件数据（采用 Opencv3 或者 skimage 库读取图像文件，返回 ndarray 格式，或者 TensorFlow 读取图像，返回 Tensor 格式）
- 打乱数据（随机打乱数据）
- 划分 batch（根据硬件规格，可设置相应较大的 batch）

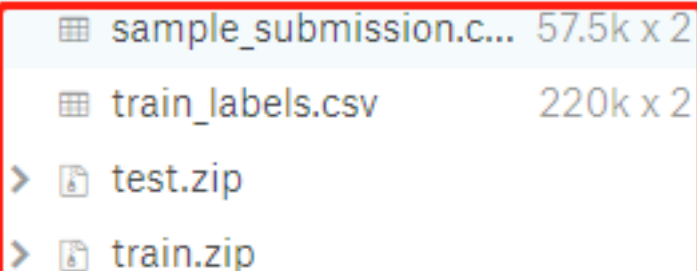
程序设计

我这里以 [Histopathologic Cancer Detection 比赛为例](#)，编写数据准备的程序，这个程序，我写了两个版本，前期的获取文件名函数都差不多，后面的打乱数据和划分 batch 部分，一个版本是采用 numpy+python 自带的功能完成的，后面一个版本是用 TensorFlow 的数据集 Dataset 框架完成打乱图像数据和划分 batch 的功能(也可采用队列形式)。

这部分，我描述的不是很好，有经验的下面的程序大致就能理解了。
数据集形式如下图所示：

Data (6 GB)

Data Sources



sample_submission.csv	57.5k x 2
train_labels.csv	220k x 2
> test.zip	
> train.zip	

第一个版本程序

纯 python 编写, 借助了 cv2、os、numpy、csv 等库

数据准备程序被我命名为 input_data.py, 里面主要是两个函数:

- get_files(获取文件名函数,从训练集标签获取)
- get_batch (读取图像数据,划分 batch)

get_files 函数如下:

```
# -----获取文件名函数,从训练集标签获取-----  
  
def get_files(label_file):  
    file_list = []  
    label_list = []  
    image_list = []  
    i = 0  
    with open(label_file, 'r') as train:  
        # 返回一个生成器对象, reader 是可迭代的  
        reader = csv.reader(train)  
        for row in reader:
```

```

        i += 1
        if i > 1:
            # print(row)
            file_path = os.path.join(train_dir,row[0]+' .tif
')    # 获取图像文件路径
            file_list.append(file_path)
            label_list.append(row[1])
            # image = cv2.imread(file_path)
# ndarray
            # print(img)
            # image = cv2.resize(image, (IMG_W, IMG_H, IMG_C))
# ndarray
            # image_list.append(image)
            # print(str(file_path))

num_files = len(file_list)                # 获取图像名列
表长度
num_labels = len(label_list)
if num_files == num_labels:
    print('num of files identify to num of labels and t
he num is %d' % num_files)
    # 打乱文件顺序
    temp = np.array([file_list,label_list])    # ndarray,
把图像名序列和标签名序列做成一个二维数组
    temp = temp.transpose()                    # ndarray,对二
维数组进行转置操作,(2,220025)-->(220026,5)
    np.random.shuffle(temp)                    # 打乱数组各行
顺序
    file_list = list(temp[:,0])                # list,获取打
乱顺序后的图像名序列
    label_list = list(temp[:,1])              # list,获取打
乱后的标签序列

```

```

        label_list = [int(i) for i in label_list]    # 把字符标签
转化为整数型标签

    return i,file_list,label_list                    # 返回文件名和
文件标签列表 list

```

get_batch()函数如下:

```

#-----读取图像数据,划分 batch-----
# 生成相同大小的批次

def get_batch(files,labels,start,batch_size):
    images = []
    start = (start+batch_size) % len(labels)
    end = start + batch_size
    # start = start+batch_size
    files = files[start:end]
    for i,file in enumerate(files):
        #print(file)
        image = cv2.imread(file)
        image = cv2.resize(image,(IMG_W,IMG_H))
        images.append(image)
    labels = labels[start:end]
    # 打乱一个 batch 的图像数据和对应标签

    temp = np.array([images,labels])                # ndarray,把图像
名序列和标签名序列做成一个二维数组

    temp = temp.transpose()                          # ndarray,对二维数
组进行转置操作,(2,220025)-->(220026,5)

    np.random.shuffle(temp)                          # 打乱数组各行顺序

    images = list(temp[:,0])                         # list,获取打乱顺
序后的图像名序列

```

```

        labels = list(temp[:,1])                # list, 获取打乱后
        的标签序列

        labels = [int(i) for i in labels]        # 把字符标签转化为
        整数型标签

        # 返回一个batch 的 images 和 labels

        return np.array(images), np.array(labels)

```

全部程序如下：

```

# coding:utf-8

# filename:input_data.py

# Environment:windows10,python3,numpy,TensorFlow1.9,glob,matplotlib,time

# Function:负责实现读取数据,生成批次 (batch)


import os
import numpy as np
from skimage import io,transform
import csv
import cv2

IMG_H = 96                # 图像高度
IMG_W = 96                # 图像宽度
IMG_C = 3                 # 图像通道
batch_size = 20           # 批次大小

label_file = 'F:/Software/Python_Project/Histopathologic-Cancer-Detection2.0/train_labels.csv'

train_dir = 'F:/Software/Python_Project/Histopathologic-Cancer-Detection/train/'

# -----获取文件名函数,从训练集标签获取-----

```

```

def get_files(label_file):
    file_list = []
    label_list = []
    image_list = []
    i = 0
    with open(label_file, 'r') as train:
        # 返回一个生成器对象, reader 是可迭代的
        reader = csv.reader(train)
        for row in reader:
            i += 1
            if i > 1:
                # print(row)
                file_path = os.path.join(train_dir, row[0] + '.tif
')    # 获取图像文件路径
                file_list.append(file_path)
                label_list.append(row[1])
                # image = cv2.imread(file_path)
# ndarray
                # print(img)
                # image = cv2.resize(image, (IMG_W, IMG_H, IMG_C))
# ndarray
                # image_list.append(image)
                # print(str(file_path))

    num_files = len(file_list)    # 获取图像名列
表长度
    num_labels = len(label_list)
    if num_files == num_labels:
        print('num of files identify to num of labels and t
he num is %d' % num_files)

```

```

        # 打乱文件顺序

        temp = np.array([file_list, label_list])      # ndarray,
        把图像名序列和标签名序列做成一个二维数组

        temp = temp.transpose()                      # ndarray, 对二
        维数组进行转置操作, (2, 220025) --> (220026, 5)

        np.random.shuffle(temp)                      # 打乱数组各行
        顺序

        file_list = list(temp[:, 0])                  # list, 获取打
        乱顺序后的图像名序列

        label_list = list(temp[:, 1])                 # list, 获取打
        乱后的标签序列

        label_list = [int(i) for i in label_list]     # 把字符标签
        转化为整数型标签

        return i, file_list, label_list               # 返回文件名和
        文件标签列表 list

#-----读取图像数据, 划分 batch-----
# 生成相同大小的批次

def get_batch(files, labels, start, batch_size):
    images = []
    start = (start+batch_size) % len(labels)
    end = start + batch_size
    # start = start+batch_size
    files = files[start:end]
    for i, file in enumerate(files):
        # print(file)                                # 仅供测试程序时
        用, 迭代训练模型时建议注释掉

        image = cv2.imread(file)

        image = cv2.resize(image, (IMG_W, IMG_H))

        images.append(image)

    labels = labels[start:end]

```

```

        # 打乱一个 batch 的图像数据和对应标签

        temp = np.array([images, labels])           # ndarray, 把
        图像名序列和标签名序列做成一个二维数组

        temp = temp.transpose()                     # ndarray, 对二
        维数组进行转置操作, (2, 220025)-->(220026, 5)

        np.random.shuffle(temp)                     # 打乱数组各行
        顺序

        images = list(temp[:, 0])                    # list, 获取打
        乱顺序后的图像名序列

        labels = list(temp[:, 1])                    # list, 获取打
        乱后的标签序列

        labels = [int(i) for i in labels]            # 把字符标签转
        化为整数型标签

        # 返回一个 batch 的 images 和 labels

        return np.array(images), np.array(labels)

#-----测试: 迭代输出一个 batch 数据-----
#-----

steps = 10
#循环迭代输出 batch 数据
for i in range(steps):
    data, label = get_batch(file_list, label_list, i, batch_si
    ze)

    print(data, label)

    print(type(data))

    print(label.dtype)

    print(len(data), len(label))

```

本程序获取文件名（文件地址集）函数，不需要列出通过训练目录下的文件，而是借助训练集标签，直接构造文件路径，实测这样速度快了很多，如果是通过 `os.listdir()+os.path.join` 的方式获取文件路径，还需要和训练集标签去一一对应相应文件名和标签，这样速度非常慢！训练集少还好点，训练集大的话，光获取文件名路径这个部分，就花很多时间了。

为了加快程序的速度，本程序的读取图像数据是按照一个批次来读取的，先随机打乱文件名数据之后，然后划分文件名 batch，再开始读取图像数据，这样就得到了一个 batch 的图像数据，shape 为 (batch,img_w,img_h,img_c)。一个 batch 一个 batch 的去读取图像，比一次性读取所有图像数据再划分 batch 要快很多。

输出结果

无图无真相，我这里设置 batch_size 的是 20。输出 data 的 shape 为 (20, 96, 96, 3)，label 的 shape 为 (20,)

```
[[162  86 180]
 [203 151 207]
 [234 223 230]
 ....
 [219 127 250]
 [205 168 204]
 [141  63 178]]] [1 0 0 1 0 1 1 0 0 0 0 0 0 1 0 1 0 0 0 0]
<class 'numpy.ndarray'>
int32
20 20
[[[191 102 195]
 [224 159 217]
 [138  61 148]
 ....
 [203 122 186]
 [228 152 237]
 [143  53 123]]

 [[211 121 224]
 [244 178 243]
 [140  64 158]
 ....
 [148  67 143]
 [187 114 200]
 [182  96 176]]

 [[186  97 213]
```

第二个版本程序

这个版本使用的是 TensorFlow 的 Dataset 框架读取处理数据，我在网上没找到使用的程序，在参考了些资料和查阅 api 之后，自己写了这个实用的程序，但是在训练的时候，出现了训练到 1000 左右 epoch 时，程序突然报错了，这让我很懵逼，目前没有找到问题。

其实正常测试读取训练集图像是没问题，主要是在训练模型的时候出了问题，还不清楚是模型训练程序还是数据准备程序的问题，所以这个版本程序仅供参考。

纯 python 编写，借助了 cv2、os、numpy、csv、TensorFlow 等库。

数据准备程序被我命名为 input_data.py，里面主要是两个函数：

- get_files(获取文件名函数,从训练集标签获取)
- read_batch_image (读取一个 batch 图像，返回图像和标签数据 ndarray)
- get_batch (生成一个 batch 的文件名地址集和标签)

程序如下：

```
# coding:utf-8

# filename:input_data.py

# Environment:windows10,python3,numpy,TensorFlow1.9,glob,matplotlib,time

# Function:负责实现读取数据,生成批次 (batch)


from skimage import io,transform
import tensorflow as tf
import numpy as np
import os
import matplotlib.pyplot as plt
import csv
import cv2


# 本地电脑训练数据读取对应地址

train_dir = "F:/Software/Python_Project/Histopathologic-Cancer-Detection/train/"

label_file = 'F:/Software/Python_Project/Histopathologic-Cancer-Detection/train_labels.csv'

# 云服务器训练数据读取对应地址

# train_dir = '/data/Histopathologic-Cancer-Detection/train/'

# label_file = '/data/Histopathologic-Cancer-Detection/train_labels.csv'
```

```

#-----图像数据及标签获取并打乱-----
# 获取数据集图像文件路径和标签
def get_files(file_dir, label_file):
    # file_dir: 文件夹路径
    # label_file: 训练数据标签文件
    # return: 乱序后的图片和标签

    # 定义存放图像数据和标签列表
    image_list = []
    label_list = []

    len_file = len(os.listdir(file_dir))          #统计指定
    文件夹中图像文件个数

    len_label = len(open(label_file).readlines())-1    # 统计
    label_file 文件有多少行

    if len_file == len_label:
        print('num of images identify to num of labels.')
        print('The number of images is %d.' % len_file)

    # csv_file = open(label_file, 'r')

    train_files = [file_dir+i for i in os.listdir(file_dir)]
    # use this for full datas

    train_files = train_files[0:300]

    # 循环列出文件夹下的所有图像文件, 获取文件路径和文件标签列表
    # for file in os.listdir(file_dir):
    for i, file in enumerate(train_files):
        file_name = file.split('/')[-1].split(sep='.')[0]

        # i += 1

```

```

    print(i, 'files load success', 'filename is', file_name)
    # print(file, 'load success')
    # image_list.append(file_dir+file)
    image_list.append(file)
    with open(label_file, 'r') as train:
        # 返回一个生成器对象, reader 是可迭代的
        reader = csv.reader(train)
        for row in reader:
            if row[0] == file_name:
                label_list.append(row[1])
                print('The label is', row[1])

    print('There are %d images\nThere are %d labels' % (len(image_list), len(label_list)))

    # 打乱文件顺序

    temp = np.array([image_list, label_list])    # ndarray, 把图像名序列和标签名序列做成一个二维数组

    temp = temp.transpose()                    # ndarray, 对二维数组进行转置操作, (2, 220025) --> (220026, 5)

    np.random.shuffle(temp)                    # 打乱数组各行顺序

    image_list = list(temp[:, 0])                # list, 获取打乱顺序后的图像名序列

    label_list = list(temp[:, 1])                # list, 获取打乱后的标签序列

    label_list = [int(i) for i in label_list]    # 把字符标签转化为整数型标签

    return image_list, label_list                # list, shape: (220025,) (220025,)

```

```
#-----读取一个 batch 图像, 返回图像和标签数据 ndarray-----
```

```
# 读取 image file name, 裁剪图像尺寸后, 返回 ndarray 格式的数据 (3*D)
```

```
def read_batch_image(file_batch, label_batch):
```

```
    # file_batch = file_batch.eval()
```

```
    # label_batch = label_batch.eval()
```

```
    image_batch = []
```

```
    # batch_label = []
```

```
    for i, file in enumerate(file_batch):
```

```
        file = str(file)
```

```
        file = file.strip('b')
```

```
        # print(file)
```

```
        image = io.imread(eval(file))
```

```
        image = transform.resize(image, (96, 96))
```

```
        image_batch.append(image)
```

```
        # image = tf.image.resize_images(image, [96, 96])
```

```
    # 返回一个 batch 图像数据, shape: (batch_size, 96, 96, 3)
```

```
    return np.array(image_batch), np.array(label_batch)
```

```
#-----生成一个 batch 的文件名地址集和标签-----
```

```
# 生成相同大小的批次
```

```
def get_batch(filenamees, labels, batch_size):
```

```
    # 此时 dataset 中的一个元素是(filename, label)
```

```
    dataset = tf.data.Dataset.from_tensor_slices((filenamees, labels))
```

```
    # 此时 dataset 中的一个元素是(file_batch, label_batch)
```

```
    dataset = dataset.shuffle(buffer_size=1000).batch(batch_size).repeat()
```

```

# 从 dataset 中实例化了一个 Iterator, 只能从头到尾读取一次元素
iterator = dataset.make_one_shot_iterator()

# file_batch, label_batch 是返回的一维张量
file_batch, label_batch = iterator.get_next()

# 返回一个 batch 的 file 和 label
return file_batch, label_batch

#-----开始从数据集读取文件名和图像数据-----
#-----

filenames, labels = get_files(train_dir, label_file)

#-----创建会话, 开始测试迭代读取图像-----
#-----

with tf.Session() as sess:

    for i in range(10):
        print('Start Iterator')
        file_batch, label_batch = get_batch(filenames, labels, 2
0)

        # 打印一个 batch 的图像数据和标签
        file_batch = sess.run(file_batch)
        label_batch = sess.run(label_batch)
        print(file_batch, label_batch)
        data, label = read_batch_image(file_batch, label_batch)
        print(data)
        print(label)

```

总结

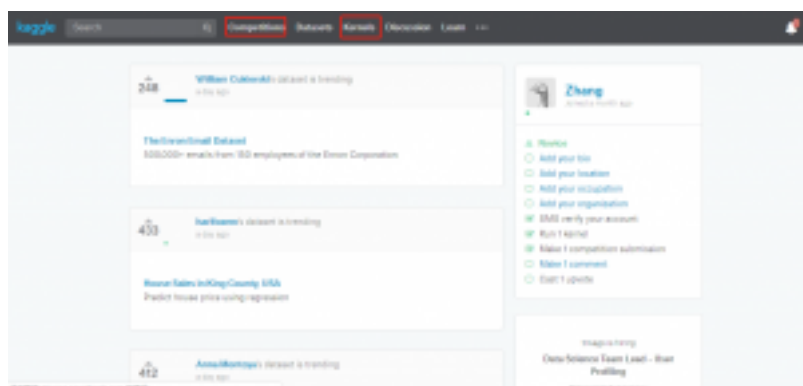
数据准备的程序真的没有模板去套，需要我们再下载分析好数据之后，设计相应的文件名获取、数据读取（打乱、划分 batch）、数据预处理、数据增强等功能函数。

【Kaggle 竞赛】Kaggle 竞赛了解

2018 年 11 月 30 日 BY HARLEY · 0 评论

关于 Kaggle 竞赛

Kaggle 是一个数据分析的竞赛平台，网址：<https://www.kaggle.com/>，网站主页面如下：



kaggle 上的竞赛主要分为 A 类赛和 B 类赛。

A 类赛主要适合用传统的机器学习算法做，偏向与文本数据处理，比如房价预测，文本分类等；

B 类赛则几乎都是用神经深度学习算法做，偏向于图像识别/目标检测等方向，比如基础的猫狗识别、cifar10 图像分类、蛋白质识别等。根据我的亲身体验，真的需要配置好的服务器做基础才行啊！

我现阶段专注于图像识别，所以我参加了三个 kaggle 竞赛都是 CV 领域的，下面是我总结的 Kaggle 的 CV 类竞赛的流程。

1. 数据准备（包括下载、分析数据后，再读取数据并做预处理，数据量过小的话做数据增强）
2. 模型设计（CNN 网络选择，基础 CNN，或者 state-of-art 模型，如 ResNet，VGGNet 等，模型）

3. 迭代训练 (迭代训练模型)
4. 模型验证 (在测试集上测试训练得到的模型)

图像识别竞赛流程

图像识别竞赛，主要是对未知图像进行分类，然后在测试集上测试后，提交结果到 Kaggle 平台，查看分数和排名。主要流程如下：

1. 数据准备
2. 模型设计
3. 迭代训练
4. 模型验证

数据准备

包括下载、分析数据后，再读取数据并做预处理，数据量过小的话做数据增强。

模型设计

图像识别的比赛，基本都是 CNN 网络，所以这里可以选择基础的 CNN 网络，或者直接上 state-of-art 模型，如 ResNet，VGGNet 等模型，模型的设计需要注意的是超参数的调节，包括基础学习率、最大迭代训练次数、Batch 批次大小等，这些都需要依靠经验和理论来去设置调节。

迭代训练

当数据准备和模型设计的工作完成以后，我们就可以对模型进行迭代训练，来获取模型最佳权重，在迭代次数完成后，记得保存模型。训练可使用 K 折交叉验证方法。

模型测试

迭代训练后的模型泛化性和效果如何，需要在测试集上测试之后才能知道，这也是 Kaggle 竞赛与网上乱七八糟的一些 demo 的不同之处，模型需要对较大的测试集进行测试，并将图像分类的测试结果写入 csv 文件提交到官网上去。为了得到好的测试结果，我们需要做验证比较多个模型、调节超参数、做数据增强、防止过拟合等工作。

总结

因为图像识别不想文本处理类的比赛所需数据量较小, 所以它前期的数据准备工作很是繁琐, 对硬件要求很高, 所以建议准备一个好的服务器平台。