studyer_domi
码龄4年 暂无认证
267
原创
5657
积分
私信

搜博主文章

热门文章
linux查看操作系统版本、内存信息 132852
matlab模糊控制工具箱使用和模糊控制pid实例参考 36322
matlab找出二维矩阵中最大值的位置或者最小值的位置 27147
python通过pip安装包，提示 pip 不是内部或外部命令 14043
matlab 绘制三维图并标注每个点的坐标 12285

分类专栏

最新评论
2020-12-14 Python PyCharm新建项目自...
water___Wang: 文章层次很清晰！优秀的文章~
2021-01-27 CentOS系统将UTC时间修改...
Shanfenglan7:
2021-01-27 CentOS系统将UTC时间修改...
Shanfenglan7:
2021-01-27 CentOS系统将UTC时间修改...
Shanfenglan7:
Python Tensorflow神经网络实现股票预测
m0_49334468: 大佬你好，我编译的时候 “wb1 = tf.matmul(x, w1) + b1” 总是提示这...

最新文章
2021-01-28 粒子群优化算法-Python版本和Matlab函数 particleswarm 调用
2021-01-27 计算机-进程与线程区别
2021-01-27 CentOS系统将UTC时间修改为CST时间方法

2021年 25篇 2020年 166篇
2019年 71篇 2018年 5篇

目录

2020-12-11 keras通过model.fit_generator训练模型(节省内存)

原创 studyer_domi 2020-12-11 12:06:22 47 收藏 版权
分类专栏： 深度学习 keras 模型 文章标签： keras model fit_generator 训练模型

keras通过model.fit_generator训练模型(节省内存)

前言

前段时间在训练模型的时候，发现当训练集的数量过大，并且输入的图片维度过大时，很容易就超内存了，举个简单例子，如果我们有20000个样本，输入图片的维度是224x224x3，用float32存储，那么如果我们一次性将全部数据载入内存的话，总共就需要20000x224x224x3x32bit/8=11.2GB 这么大的内存，所以如果一次性要加载全部数据集的话是需要很大内存的。

如果我们直接用keras的fit函数来训练模型的话，是需要传入全部训练数据，但是好在提供了fit_generator，可以分批次的读取数据，节省了我们的内存，我们唯一要做的就是实现一个生成器（generator）。

1.fit_generator函数简介

```
fit_generator(generator,

steps_per_epoch=None,

epochs=1,

verbose=1,

callbacks=None,

validation_data=None,

validation_steps=None,

class_weight=None,

max_queue_size=10,

workers=1,

use_multiprocessing=False,

shuffle=True,

initial_epoch=0)
```

参数：

generator：一个生成器，或者一个 Sequence (keras.utils.Sequence) 对象的实例。这是我们实现的重点，后面会着介绍生成器和sequence的两种实现方式。

steps_per_epoch：这个是我们在每个epoch中需要执行多少次生成器来生产数据，fit_generator函数没有batch_size这个参数，是通过steps_per_epoch来实现的，每次生产的数据就是一个batch，因此steps_per_epoch的值我们通过会设为（样本数/batch_size）。如果我们的generator是sequence类型，那么这个参数是可选的，默认使用len(generator)。

epochs：即我们训练的迭代次数。

verbose：0, 1 或 2。日志显示模式。 0 = 安静模式, 1 = 进度条, 2 = 每轮一行

callbacks：在训练时调用的一系列回调函数。

validation_data：和我们的generator类似，只是这个使用于验证的，不参与训练。

validation_steps：和前面的steps_per_epoch类似。

class_weight：可选的将类索引（整数）映射到权重（浮点）值的字典，用于加权损失函数（仅在训练期间）。这可以用来告诉模型「更多地关注」来自代表性不足的类的样本。（感觉这个参数用的比较少）

max_queue_size：整数。生成器队列的最大尺寸。默认为10.

workers：整数。使用的最大进程数量，如果使用基于进程的多线程。如未指定，workers 将默认为 1。如果为 0，将在主线程上执行生成器。

use_multiprocessing：布尔值。如果 True，则使用基于进程的多线程。默认为False。

shuffle：是否在每轮迭代之前打乱 batch 的顺序。只能与Sequence(keras.utils.Sequence) 实例同用。

initial_epoch: 开始训练的轮次（有助于恢复之前的训练）

2.generator实现

2.1生成器的实现方式

样例代码：

```
from keras.models import Sequential
from keras.layers import Dense
import numpy as np
from PIL import Image


def process_x(path):
    img = Image.open(path)
    img = img.resize((96, 96))
    img = img.convert('RGB')
    img = np.array(img)

    img = np.asarray(img, np.float32) / 255.0
    # 也可以进行进行一些数据数据增强的处理
    return img


def generate_arrays_from_file(x_y):
    # x_y 是我们的训练集包括标签，每一行的第一个是我们的图片路径，后面的是图片标签

    global count
    batch_size = 8
    while 1:
        batch_x = x_y[(count - 1) * batch_size:count * batch_size, 0]
        batch_y = x_y[(count - 1) * batch_size:count * batch_size, 1:]

        batch_x = np.array([process_x(img_path) for img_path in batch_x])
        batch_y = np.array(batch_y).astype(np.float32)
        print("count:" + str(count))
        count = count + 1
        yield batch_x, batch_y


model = Sequential()
model.add(Dense(units=1000, activation='relu', input_dim=2))
model.add(Dense(units=2, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])
count = 1
x_y = []
model.fit_generator(generate_arrays_from_file(x_y), steps_per_epoch=10, epochs=2, max
_queue_size=1, workers=1)
```

在理解上面代码之前我们需要首先了解yield的用法。

yield关键字：

我们先通过一个例子看一下yield的用法：

```
def foo():
    print("starting...")
    while True:
        res = yield 4
        print("res:", res)


g = foo()
print(next(g))
print("----------")
print(next(g))
```

运行结果：

```
starting...
4
----------
res: None
4
```

带yield的函数是一个生成器，而不是一个函数。因为foo函数中有yield关键字，所以foo函数并不会真的执行，而是先得到一个生成器的实例，当我们第一次调用next函数的时候，foo函数才开始行，首先先执行foo函数中的print方法，然后进入while循环，循环执行到yield时，yield其实相当于return，函数返回4，程序停止。所以我们第一次调用next(g)的输出结果是前面两行。

然后当我们再次调用next(g)时，这个时候是从上一次停止的地方继续执行，也就是要执行res的赋值操作，因为4已经在上一次执行被return了，随意赋值res为None，然后执行print(“res:”,res)打印res: None，再次循环到yield返回4，程序停止。

所以yield关键字的作用就是我们能够从上一次程序停止的地方继续执行，这样我们用作生成器的时候，就避免一次性读入数据造成内存不足的情况。

现在看到上面的示例代码：

generate_arrays_from_file函数就是我们的生成器，每次循环读取一个batch大小的数据，然后处理数据，并返回。x_y是我们的把路径和标签合并后的训练集，类似于如下形式：

```
['data/img_4092.jpg' '0' '1' '0' '0' '0' ]
```

至于格式不一定要这样，可以是自己的格式，至于怎么处理，根于自己的格式，在process_x进行处理，这里因为是存放的图片路径，所以在process_x函数的主要作用就是读取图片并进行归一化等操作，也可以在这里定义自己需要进行的操作，例如对图像进行实时数据增强。

2.2使用Sequence实现generator

示例代码：

```
class BaseSequence(Sequence):
    """
 基础的数据流生成器，每次迭代返回一个batch
 BaseSequence可直接用于fit_generator的generator参数
 fit_generator会将BaseSequence再次封装为一个多进程的数据流生成器
 而且能保证在多进程下的一个epoch中不会重复取相同的样本
 """

    def __init__(self, img_paths, labels, batch_size, img_size):
        # np.hstack在水平方向上平铺
        self.x_y = np.hstack((np.array(img_paths).reshape(len(img_paths), 1), np.arra
y(labels)))
        self.batch_size = batch_size
        self.img_size = img_size

    def __len__(self):
        # math.ceil表示向上取整
        # 调用len(BaseSequence)时返回，返回的是每个epoch我们需要读取数据的次数
        return math.ceil(len(self.x_y) / self.batch_size)

    def preprocess_img(self, img_path):
        img = Image.open(img_path)
        resize_scale = self.img_size[0] / max(img.size[:2])
        img = img.resize((self.img_size[0], self.img_size[0]))
        img = img.convert('RGB')
        img = np.array(img)

        # 数据归一化
        img = np.asarray(img, np.float32) / 255.0
        return img

    def __getitem__(self, idx):
        batch_x = self.x_y[idx * self.batch_size: (idx + 1) * self.batch_size, 0]
        batch_y = self.x_y[idx * self.batch_size: (idx + 1) * self.batch_size, 1:]
        batch_x = np.array([self.preprocess_img(img_path) for img_path in batch_x])
        batch_y = np.array(batch_y).astype(np.float32)
        print(batch_x.shape)
        return batch_x, batch_y

    # 重写的父类Sequence中的on_epoch_end方法，在每次迭代完后调用。
    def on_epoch_end(self):
        # 每次迭代后重新打乱训练集数据
        np.random.shuffle(self.x_y)
```

在上面代码中，__len__和__getitem__，是我们重写的魔法方法，__len__是当我们调用len(BaseSequence)函数时调用，这里我们返回（样本总量/batch_size），供我们传入fit_generator中的steps_per_epoch参数；__getitem__可以让对象实现迭代功能，这样在将BaseSequence的对象传入fit_generator中后，不断执行generator就可循环的读取数据了。

举个例子说明一下getitem的作用：

```
class Animal:
    def __init__(self, animal_list):
        self.animals_name = animal_list

    def __getitem__(self, index):
        return self.animals_name[index]


animals = Animal(["dog", "cat", "fish"])
for animal in animals:
    print(animal)
```

输出结果：

```
dog
cat
fish
```

并且使用Sequence类可以保证在多进程的情况下，每个epoch中的样本只会被训练一次。

参考**yield方法**：

欢迎关注公众号：算法工程师的学习日志，获取算法工作相关的学习资料。如果有技术咨询，提供有偿咨询，联系qq（1762016542）或者公众号留言

点赞 评论 分享 收藏 举报 关注 一键三连

keras fit_generator 节省内存 例子 binqiang2wang 8446

之前写keras的时候，都是直接model.fit()，后来发现这样不节省内存，尤其是在输入数据本身不大，但是内部要进…行排列组合的时候就显得特别有用，这里记录一下fit_generator的用法： fit_generator(self, generator, steps_per_epoch=None, epochs=1, verbose=1, callbacks=None, validati

优质评论可以帮助作者获得更高权重 评论

keras 两种训练模型方式fit和fit_generator(节省内存) u011311291的博客 6万+

第一种，fitimport keras from keras.models import Sequential from keras.layers import Dense import numpy as n…from sklearn.preprocessing import LabelEncoder from sklearn.preprocessing import OneHotEncoder

tf.keras中model.fit_generator()和model.fit() 追梦小狂魔的博客 2543

首先Keras中的fit()函数传入的x_train和y_train是被完整的加载进内存的,当然用起来很方便，但是如果我们数据量很大，那么是不可能将所有数据载入内存的，必将导致内存泄漏，这时候我们可以用fit_generator函数来进行训练。 fit(x=None, y=None, batch_size=None, epochs=1, verbose=1, callbacks=None, val...

keras通过model.fit_generator训练模型（节省内存） CarryLvan的博客 1908

keras通过model.fit_generator训练模型（节省内存）1.fit_generator函数简介2.generator实现2.1生成器的实现方式.2.2使用Sequence实现generator 前言 前段时间在训练模型的时候，发现当训练集的数量过大，并且输入的图片维度过大时，很容易就超内存了，举个简单例子，如果我们有20000个样本，输入图片的维度是224x224x3，用float3..

浅谈keras通过model.fit_generator训练模型(节省内存) 09-16

主要介绍了浅谈keras通过model.fit_generator训练模型(节省内存)，具有很好的参考价值，希望对大家有所帮助。…一起跟随小编过来看看吧

python模型训练 控制内存_浅谈keras通过model.fit_generator训练模型(节省内存) ..weixin_29813667的博客 134

前言前段时间在训练模型的时候，发现当训练集的数量过大，并且输入的图片维度过大时，很容易就超内存了，举…个简单例子，如果我们有20000个样本，输入图片的维度是224x224x3，用float32存储，那么如果我们一次性将全部数据载入内存的话，总共就需要20000x224x224x3x32bit/8=11.2GB 这么大的内存，所以如果一次性要加载全部数据集的话是需要很大内存的。如果我们直接用kera...

在keras中model.fit_generator()和model.fit()的区别说明 09-16

主要介绍了在keras中model.fit_generator()和model.fit()的区别说明，具有很好的参考价值，希望对大家有所帮助。.一起跟随小编过来看看吧

使用Keras 的Model.fit_generator报错StopIteration Will_Ye的博客 2251

使用Keras 的Model.fit_generator报错StopIteration 之前也遇到过这个问题，解决了之后没记下来，最近跑之前代码.又出现这个，废了时间去找答案，还是要勤劳点做学习记录才行。 报错如下，问题就是批量产生的数据没有成功一批批地导入。 Epoch 1/100 Epoch 00001: CosineAnnealingScheduler setting learning ra...

Keras实现mode.fit和model.fit_generator比较 Einstellung的博客 3418

模型部分 模型部分都一样，比如我这里使用AlexNet网络来做。我做的是一个二分类任务，所以结尾部分网络有改动.。输入图片尺寸是256*256的，所以输出图片尺寸有一点改动。 from keras.models import Sequential from keras.layers import Dense, Dropout, Activation, Flatten from keras.layer...

model.fit_generator()函数参数 qq_32951799的博客 3万+

fit_generator(self, generator, steps_per_epoch, epochs=1, verbose=1, callbacks=None, validation_data=None, …alidation_steps=None, class_weight=None, max_q_size=10, workers=1, pickle_safe=False, initi...

在keras中model.fit_generator()和model.fit()有什么区别 五味鱼头的博客 1万+

首先Keras中的fit()函数传入的x_train和y_train是被完整的加载进内存的,当然用起来很方便，但是如果我们数据量很大，那么是不可能将所有数据载入内存的，必将导致内存泄漏，这时候我们可以用fit_generator函数来进行训练。 https://keras.io/zh/models/model/ fit fit(x=None, y=None, batch_...

keras 两种训练模型方式详解fit和fit_generator(节省内存) 菜鸟教程 546

更多编程教程请到：菜鸟教程 https://www.piaodoo.com/ 友情链接： 高州阳光论坛https://www.hnthzk.com/ 人人影..视http://www.op-kg.com/ 第一种，fit import keras from keras.models import Sequential from keras.layers import Dense import numpy as np from sklearn.preprocessing import LabelEn

Keras 的Model.fit_generator 报错 StopIteration qq_40540975的博客 57

要在generator函数部分加上while 1 def gen(): while 1: data = h5py.File('D:/《桌面》/CDL-500.mat', 'r') i = 0 while..True: x_train = data['x_train'][i*20: (i+1)*20] y_train = data['y_train'][i*20: (i+1)*20]

Keras： 创建自己的generator(适用于model.fit_generator)，解决内存问题 Work hard 3896

为什么要使用model.fit_generator？ 在现实的机器学习中，训练一个model往往需要数量巨大的数据，如果使用fit进.行数据训练，很有可能导致内存不够，无法进行训练。 fit_generator的定义如下： fit_generator(generator, steps_per_epoch=None, epochs=1, verbose=1, callbacks=None, val...

keras中的model.fit和model.fit_generator — echo_hao的博客 3575

fit(self, x=None, y=None, batch_size=None, epochs=1, verbose=1, callbacks=None, validation_split=0.0, validation_data=None, shuffle=True, class_weight=None, sample_weight=None, initial_epoch=0, steps...

使用model.fit_generator方法进行训练(自己的训练集-多分类) — xfjs_net的博客 1万+

我们在使用model.fit()进行训练的时候， 在这之前你肯定会有训练集的x_img_train,y_label_train两个参数。 fit(x=None, y=None, batch_size=None, epochs=1, verbose=1, callbacks=None, validation_split=0.0, validation_data=None, shuffle=Tru...

keras中model.fit和model.fit_generator的区别 — ZhuiMengLQG的博客 57

keras中model.fit和model.fit_generator的区别前言一、model.fit()函数详解二、model.fit_generator()函数详解总结...前言 Keras中的fit()函数传入的数据x_train和y_train是被完整的加载进内存的，用起来很方便，但是如果数据量很大，电脑显存不足时，容易导致内存泄漏，那么是不可能将所有数据载入内存的，这时候我们可以用fit_generator函数来进行训练。 一、model.fit()函数详解 函数调用格式 fit(x=None,

【keras】在keras中model.fit_generator()和model.fit()有什么区别 — zkq_1986的博客 337

fit() 会将数据全部装到内存，不适合大数据量。 fit_generator() 只是转载部分数据，适合大数据量。

【Python-Keras】keras.fit()和keras.fit_generator()的解析与使用 — BetterBench的博客 73

目录1 作用与区别2 解析与使用2.1 keras.fit()（1）参数介绍（2）举例使用（3）原理讲解2.2 keras.fit_generator(...（1）参数介绍（2）举例使用（3）原理解析 1 作用与区别 作用： 用于训练神经网络模型，两者可以完成相同的任务 区别： .fit()时使用的整个训练数据集可以放入内存，并没有应用数据增强，就是.fit()无需使用Keras生成器（即无需数据参数） 当我们有一个巨大的数据集可容纳到我们的内存中或需要应用数据扩充时，将使用.fit_generator()。就