

从本专栏开始，作者正式开始研究Python深度学习、神经网络及人工智能相关知识。前一篇详细讲解了Tensorflow+Opencv实现CNN自定义图像分类案例，它能解决我们现实论文或实践中的图像分类问题，并与机器学习的图像分类算法进行对比实验。这篇文章将讲解TensorFlow如何保存变量和神经网络参数，通过Saver保存神经网络，再通过Restore调用训练好的神经网络。

本专栏主要结合作者之前的博客、AI经验和相关文章及论文介绍，后面随着深入会讲解更多的Python人工智能案例及应用。基础性文章，希望对您有所帮助，如果文章中存在错误或不足之处，还请海涵~作者作为人工智能的菜鸟，希望大家能与我在这一笔一划的博客中成长起来。写了这么多年博客，尝试第一个付费专栏，但更多博客尤其基础性文章，还是会继续免费分享，但该专栏也会用心撰写，望对得起读者，共勉！

代码下载地址：<https://github.com/eastmountyxz/AI-for-TensorFlow>

## 文章目录

- 一.保存变量
- 二.保存神经网络
- 三.总结

同时推荐前面作者另外三个Python系列文章。从2014年开始，作者主要写了三个Python系列文章，分别是基础知识、网络爬虫和数据分析。2018年陆续增加了Python图像识别和Python人工智能专栏。

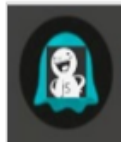
- Python基础知识系列：Python基础知识学习与提升
- Python网络爬虫系列：Python爬虫之Selenium+Phantomjs+CasperJS
- Python数据分析系列：知识图谱、web数据挖掘及NLP
- Python图像识别系列：Python图像处理及图像识别
- Python人工智能系列：Python人工智能及知识图谱实战



Python学习系列

文章：16篇

阅读：119908



Python爬虫之Selenium+Phantomjs+CasperJS

文章：33篇

阅读：443874



知识图谱、web数据挖掘及NLP

文章：44篇

阅读：488758

前文：

[Python人工智能] 一.TensorFlow2.0环境搭建及神经网络入门

[Python人工智能] 二.TensorFlow基础及一元直线预测案例

[Python人工智能] 三.TensorFlow基础之Session、变量、传入值和激励函数

[Python人工智能] 四.TensorFlow创建回归神经网络及Optimizer优化器

[Python人工智能] 五.Tensorboard可视化基本用法及绘制整个神经网络

[Python人工智能] 六.TensorFlow实现分类学习及MNIST手写体识别案例

[Python人工智能] 七.什么是过拟合及dropout解决神经网络中的过拟合问题

[Python人工智能] 八.卷积神经网络CNN原理详解及TensorFlow编写CNN

[Python人工智能] 九.gensim词向量Word2Vec安装及《庆余年》中文短文本相似度计算

[Python人工智能] 十.Tensorflow+Opencv实现CNN自定义图像分类案例及与机器学习

KNN图像分类算法对比

---

## 一.保存变量

通过tf.Variable()定义权重和偏置变量，然后调用tf.train.Saver()存储变量，将数据保存至本地“my\_net/save\_net.ckpt”文件中。

```
# -*- coding: utf-8 -*-
"""
Created on Thu Jan  2 20:04:57 2020
@author: xiuzhang Eastmount CSDN
"""

import tensorflow as tf
import numpy as np

#----- 保存文件-----
W = tf.Variable([[1,2,3], [3,4,5]], dtype=tf.float32, name='weights') #2
b = tf.Variable([[1,2,3]], dtype=tf.float32, name='biases')

# 初始化
init = tf.initialize_all_variables()

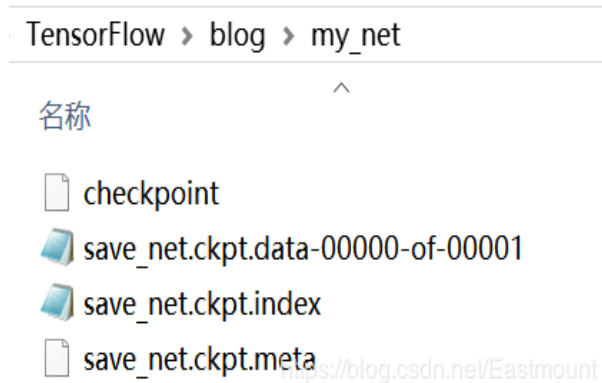
# 定义saver 存储各种变量
saver = tf.train.Saver()

# 使用Session运行初始化
with tf.Session() as sess:
    sess.run(init)
    # 保存 官方保存格式为ckpt
    save_path = saver.save(sess, "my_net/save_net.ckpt")
    print("Save to path:", save_path)
```

“Save to path: my\_net/save\_net.ckpt”保存成功如下图所示：



打开内容如下图所示：



接着定义标记变量train，通过Restore操作使用我们保存好的变量。注意，在Restore时需要定义相同的dtype和shape，不需要再定义init。最后直接通过 `saver.restore(sess, "my_net/save_net.ckpt")` 提取保存的变量并输出即可。

```
# -*- coding: utf-8 -*-
"""
Created on Thu Jan  2 20:04:57 2020
@author: xiuzhang Eastmount CSDN
"""

import tensorflow as tf
import numpy as np

# 标记变量
train = False

#-----保存文件-----
# Save
if train==True:
    # 定义变量
    W = tf.Variable([[1,2,3], [3,4,5]], dtype=tf.float32, name='weights')
    b = tf.Variable([[1,2,3]], dtype=tf.float32, name='biases')

    # 初始化
    init = tf.global_variables_initializer()

    # 定义saver 存储各种变量
```

```

saver = tf.train.Saver()

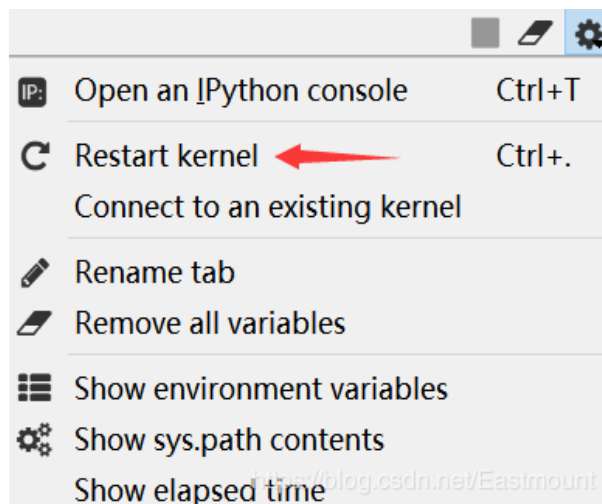
# 使用Session运行初始化
with tf.Session() as sess:
    sess.run(init)
    # 保存 官方保存格式为ckpt
    save_path = saver.save(sess, "my_net/save_net.ckpt")
    print("Save to path:", save_path)

#-----Restore变量-----
# Restore
if train==False:
    # 记住在Restore时定义相同的dtype和shape
    # redefine the same shape and same type for your variables
    W = tf.Variable(np.arange(6).reshape((2,3)), dtype=tf.float32, name=
    b = tf.Variable(np.arange(3).reshape((1,3)), dtype=tf.float32, name=

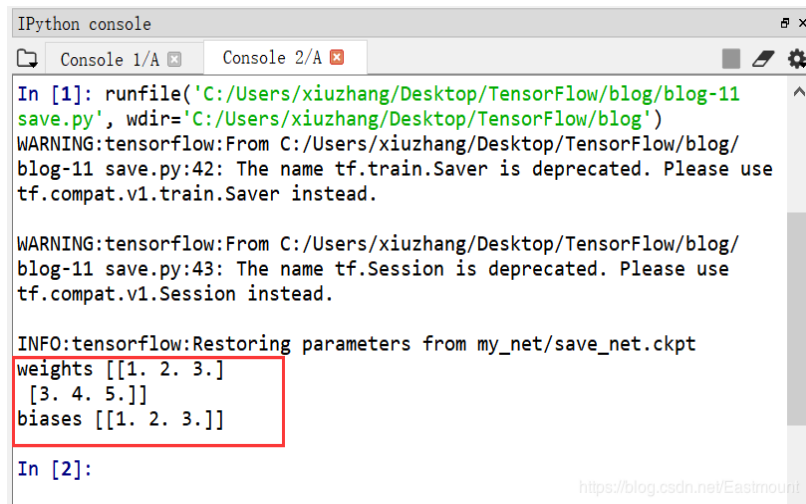
    # Restore不需要定义init
    saver = tf.train.Saver()
    with tf.Session() as sess:
        # 提取保存的变量
        saver.restore(sess, "my_net/save_net.ckpt")
        # 寻找相同名字和标识的变量并存储在W和b中
        print("weights", sess.run(W))
        print("biases", sess.run(b))

```

运行代码，如果报错“NotFoundError: Restoring from checkpoint failed. This is most likely due to a Variable name or other graph key that is missing from the checkpoint. Please ensure that you have not altered the graph expected based on the checkpoint.”，则需要重置Spyder即可。



最后输出之前所保存的变量，weights为 [[1,2,3], [3,4,5]]，偏置为 [[1,2,3]]。



```

IPython console
Console 1/A Console 2/A

In [1]: runfile('C:/Users/xiuzhang/Desktop/TensorFlow/blog/blog-11
save.py', wdir='C:/Users/xiuzhang/Desktop/TensorFlow/blog')
WARNING:tensorflow:From C:/Users/xiuzhang/Desktop/TensorFlow/blog/
blog-11 save.py:42: The name tf.train.Saver is deprecated. Please use
tf.compat.v1.train.Saver instead.

WARNING:tensorflow:From C:/Users/xiuzhang/Desktop/TensorFlow/blog/
blog-11 save.py:43: The name tf.Session is deprecated. Please use
tf.compat.v1.Session instead.

INFO:tensorflow:Restoring parameters from my_net/save_net.ckpt
weights [[1. 2. 3.]
[3. 4. 5.]]
biases [[1. 2. 3.]]

In [2]:

```

## 二.保存神经网络

那么，TensorFlow如何保存我们的神经网络框架呢？我们需要把整个网络训练好再进行保存，其方法和上面类似，完整代码如下：

```

"""
Created on Sun Dec 29 19:21:08 2019
@author: xiuzhang Eastmount CSDN
"""

import os
import glob
import cv2
import numpy as np
import tensorflow as tf

# 定义图片路径
path = 'photo/'

# ----- 第一步 读取图像 -----
def read_img(path):
    cate = [path + x for x in os.listdir(path) if os.path.isdir(path + x)]
    imgs = []
    labels = []
    fpath = []
    for idx, folder in enumerate(cate):
        # 遍历整个目录判断每个文件是不是符合
        for im in glob.glob(folder + '/*.jpg'):
            #print('reading the images:%s' % (im))
            img = cv2.imread(im) #调用opencv库读取像素点
            img = cv2.resize(img, (32, 32)) #图像像素大小一致
            imgs.append(img) #图像数据
            labels.append(idx) #图像类标
            fpath.append(path+im) #图像路径名

```

```

        #print(path+im, idx)

    return np.asarray(fpath, np.string_), np.asarray(imgs, np.float32), i

# 读取图像
fpaths, data, label = read_img(path)
print(data.shape) # (1000, 256, 256, 3)
# 计算有多少类图片
num_classes = len(set(label))
print(num_classes)

# 生成等差数列随机调整图像顺序
num_example = data.shape[0]
arr = np.arange(num_example)
np.random.shuffle(arr)
data = data[arr]
label = label[arr]
fpaths = fpaths[arr]

# 拆分训练集和测试集 80%训练集 20%测试集
ratio = 0.8
s = np.int(num_example * ratio)
x_train = data[:s]
y_train = label[:s]
fpaths_train = fpaths[:s]
x_val = data[s:]
y_val = label[s:]
fpaths_test = fpaths[s:]
print(len(x_train), len(y_train), len(x_val), len(y_val)) #800 800 200 200
print(y_val)

#-----第二步 建立神经网络-----
# 定义Placeholder
xs = tf.placeholder(tf.float32, [None, 32, 32, 3]) #每张图片32*32*3个点
ys = tf.placeholder(tf.int32, [None]) #每个样本有1个输出
# 存放DropOut参数的容器
drop = tf.placeholder(tf.float32) #训练时为0.25 测试时为0

# 定义卷积层 conv0
conv0 = tf.layers.conv2d(xs, 20, 5, activation=tf.nn.relu) #20个卷积核
# 定义max-pooling层 pool0
pool0 = tf.layers.max_pooling2d(conv0, [2, 2], [2, 2]) #pooling窗口
print("Layer0: \n", conv0, pool0)

# 定义卷积层 conv1
conv1 = tf.layers.conv2d(pool0, 40, 4, activation=tf.nn.relu) #40个卷积核

```

```

# 定义max-pooling层 pool1
pool1 = tf.layers.max_pooling2d(conv1, [2, 2], [2, 2])          #pooling窗口
print("Layer1: \n", conv1, pool1)

# 将3维特征转换为1维向量
flatten = tf.layers.flatten(pool1)

# 全连接层 转换为长度为400的特征向量
fc = tf.layers.dense(flatten, 400, activation=tf.nn.relu)
print("Layer2: \n", fc)

# 加上DropOut防止过拟合
dropout_fc = tf.layers.dropout(fc, drop)

# 未激活的输出层
logits = tf.layers.dense(dropout_fc, num_classes)
print("Output: \n", logits)

# 定义输出结果
predicted_labels = tf.argmax(logits, 1)

#-----第三步 定义损失函数和优化器-----

# 利用交叉熵定义损失
losses = tf.nn.softmax_cross_entropy_with_logits(
    labels = tf.one_hot(ys, num_classes),          #将input转化为one-hot
    logits = logits)

# 平均损失
mean_loss = tf.reduce_mean(losses)

# 定义优化器 学习效率设置为0.0001
optimizer = tf.train.AdamOptimizer(learning_rate=1e-4).minimize(losses)

#-----第四步 模型训练和预测-----

# 用于保存和载入模型
saver = tf.train.Saver()

# 训练或预测
train = False

# 模型文件路径
model_path = "model/image_model"

with tf.Session() as sess:
    if train:
        print("训练模式")

```

```

# 训练初始化参数
sess.run(tf.global_variables_initializer())
# 定义输入和Label以填充容器 训练时dropout为0.25
train_feed_dict = {
    xs: x_train,
    ys: y_train,
    drop: 0.25
}
# 训练学习1000次
for step in range(1000):
    _, mean_loss_val = sess.run([optimizer, mean_loss], feed_dict=train_feed_dict)
    if step % 50 == 0: # 每隔50次输出一次结果
        print("step = {}\t mean loss = {}".format(step, mean_loss_val))
# 保存模型
saver.save(sess, model_path)
print("训练结束, 保存模型到{}".format(model_path))
else:
    print("测试模式")
# 测试载入参数
saver.restore(sess, model_path)
print("从{}载入模型".format(model_path))
# label 和名称的对照关系
label_name_dict = {
    0: "人类",
    1: "沙滩",
    2: "建筑",
    3: "公交",
    4: "恐龙",
    5: "大象",
    6: "花朵",
    7: "野马",
    8: "雪山",
    9: "美食"
}
# 定义输入和Label以填充容器 测试时dropout为0
test_feed_dict = {
    xs: x_val,
    ys: y_val,
    drop: 0
}

# 真实label与模型预测label
predicted_labels_val = sess.run(predicted_labels, feed_dict=test_feed_dict)
for fpath, real_label, predicted_label in zip(fpaths_test, y_val, predicted_labels_val):
    # 将label id转换为label名
    real_label_name = label_name_dict[real_label]
    predicted_label_name = label_name_dict[predicted_label]

```



```

        print("{}\t{} => {}".format(fpath, real_label_name, predicted_label))
# 评价结果
print("正确预测个数:", sum(y_val==predicted_labels_val))
print("准确度为:", 1.0*sum(y_val==predicted_labels_val) / len(y_val))

```

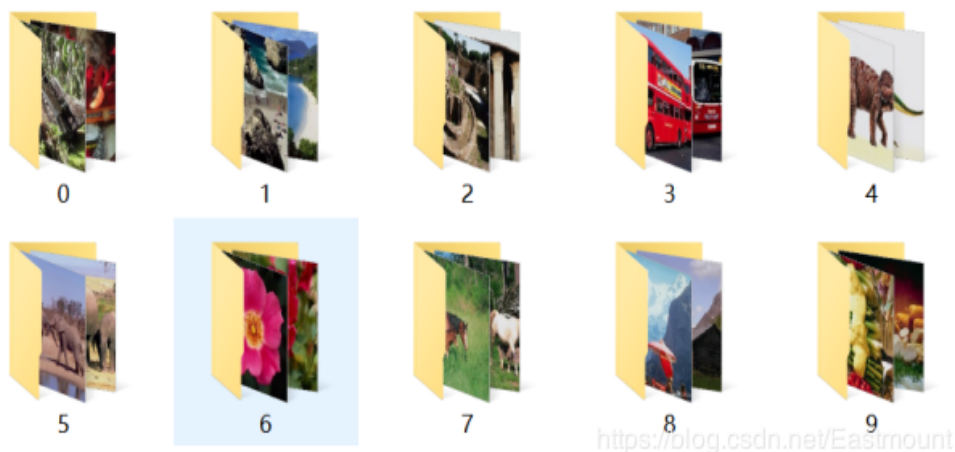
核心步骤为:

```

saver = tf.train.Saver()
model_path = "model/image_model"
with tf.Session() as sess:
    if train:
        # 保存神经网络
        sess.run(tf.global_variables_initializer())
        for step in range(1000):
            _, mean_loss_val = sess.run([optimizer, mean_loss], feed_dict=test_data)
            if step % 50 == 0:
                print("step = {}\t mean loss = {}".format(step, mean_loss_val))
            saver.save(sess, model_path)
    else:
        # 载入神经网络
        saver.restore(sess, model_path)
        predicted_labels_val = sess.run(predicted_labels, feed_dict=test_data)
        for fpath, real_label, predicted_label in zip(fpaths_test, y_val, predicted_labels_val):
            real_label_name = label_name_dict[real_label]
            predicted_label_name = label_name_dict[predicted_label]
            print("{}\t{} => {}".format(fpath, real_label_name, predicted_label))

```

预测输出结果如下图所示, 最终预测正确181张图片, 准确度为0.905。相比之前机器学习KNN的0.500有非常高的提升。



测试模式

```

INFO:tensorflow:Restoring parameters from model/image_model
从model/image_model载入模型

```

b'photo/photo/3\\335.jpg'	公交 => 公交
b'photo/photo/1\\129.jpg'	沙滩 => 沙滩
b'photo/photo/7\\740.jpg'	野马 => 野马
b'photo/photo/5\\564.jpg'	大象 => 大象
...	
b'photo/photo/9\\974.jpg'	美食 => 美食
b'photo/photo/2\\220.jpg'	建筑 => 公交
b'photo/photo/9\\912.jpg'	美食 => 美食
b'photo/photo/4\\459.jpg'	恐龙 => 恐龙
b'photo/photo/5\\525.jpg'	大象 => 大象
b'photo/photo/0\\44.jpg'	人类 => 人类

正确预测个数：181

准确度为：0.905

---

## 三.总结

写到这里，这篇文章就讲解完毕，更多TensorFlow深度学习文章会继续分享，接下来我们会分享RNN、LSTM、文本识别等内容。如果读者有什么想学习的，也可以私聊我，我去学习并应用到你的领域。

最后，希望这篇基础性文章对您有所帮助，如果文章中存在错误或不足之处，还请海涵~作为人工智能的菜鸟，我希望自己能不断进步并深入，后续将它应用于图像识别、网络安全、对抗样本等领域，指导大家撰写简单的学术论文，一起加油！

CSDN20周年快乐，感恩有你，一路同行。也非常高兴这周女神来武汉看望我，未来的人生路一起走。2020年第一篇文章，新年快乐喔！



PS: 这是作者的第一个付费专栏，会非常用心的去撰写，希望能对得起读者的9块钱。本来只想设置1快的，但CSDN固定了价格。写了八年的免费文章，这也算知识付费的一个简单尝试吧！毕竟读博也不易，写文章也花费时间和精力，但作者更多的文章会免费分享。如果您购买了该专栏，有Python数据分析、图像处理、人工智能、网络安全的问题，我们都可以深入探讨，尤其是做研究的同学，共同进步~

(By:Eastmount 2020-01-02 晚上10点夜于珞珈山 <http://blog.csdn.net/eastmount/> )

---

### 作者theano人工智能系列:

- [Python人工智能] 一.神经网络入门及theano基础代码讲解
- [Python人工智能] 二.theano实现回归神经网络分析
- [Python人工智能] 三.theano实现分类神经网络及机器学习基础
- [Python人工智能] 四.神经网络和深度学习入门知识
- [Python人工智能] 五.theano实现神经网络正规化Regularization处理
- [Python人工智能] 六.神经网络的评价指标、特征标准化和特征选择
- [Python人工智能] 七.加速神经网络、激励函数和过拟合

### 参考文献:

- [1] 冈萨雷斯著. 数字图像处理 (第3版) [M]. 北京: 电子工业出版社, 2013.
- [2] 杨秀璋, 颜娜. Python网络数据爬取及分析从入门到精通 (分析篇) [M]. 北京: 北京航空航天大学出版社, 2018.
- [3] 罗子江等. Python中的图像处理[M]. 科学出版社, 2020.
- [4] <https://study.163.com/course/courseLearn.htm?courseId=1003209007>
- [5] TensorFlow【极简】CNN - Yellow\_python大神

[6] 基于深度神经网络的定向激活功能开发相位信息的声源定位 - 章子睢Kevin

[7] [https://github.com/siucan/CNN\\_MNIST](https://github.com/siucan/CNN_MNIST)

[8] <https://github.com/eastmountyxz/Al-for-TensorFlow>

---