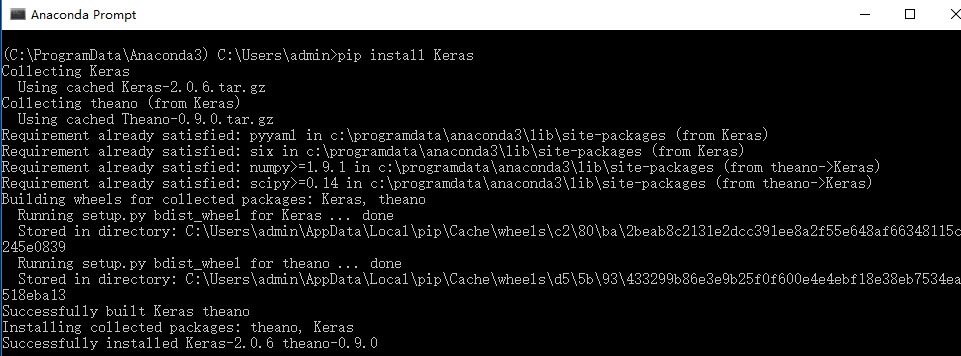
学习笔记2-CNN实践-2017-07-20

# Keras:基于Theano和TensorFlow的深度学习库

## Keras安装

前提：安装Anaconda，的版本是Anaconda3 (64-bit)

保持网络连接，从开始菜单中打开Anaconda Prompt，输入：pip install Keras



目前TensorFlow在Windows下只支持Python 3.5版本，所以我之前的2.7果断换3.5

以管理员身份打开Anaconda Prompt，

输入：pip install --upgrade --ignore-installed tensorflow

## Theano是什么？

Theano是一个Python库，可以在CPU或GPU上运行快速数值计算。这是Python深度学习中的一个关键基础库，你可以直接用它来创建深度学习模型或包装库，大大简化了程序。

Python的核心Theano是一个数学表达式的编译器。它知道如何获取你的结构，并使之成为一个使用numpy、高效本地库的非常高效的代码，如BLAS和本地代码（C++），在CPU或GPU上尽可能快地运行。它巧妙的采用一系列代码优化从硬件中攫取尽可能多的性能。

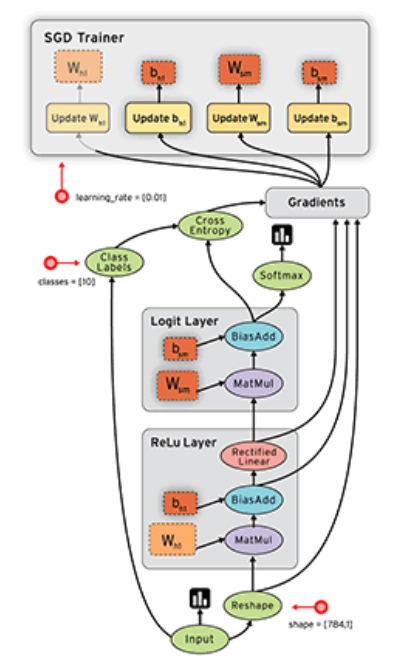
Theano表达式的实际语法是象征性的，可以推送给初学者用于一般软件开发。具体来说，表达式是在抽象的意义上定义，编译和后期是用来进行计算。它是为深度学习中处理大型神经网络算法所需的计算而专门设计的。它是这类库的首创之一（发展始于2007年），被认为是深度学习研究和开发的行业标准。

## TensorFlow是什么？

TensorFlow™ 是一个采用数据流图（data flow graphs），用于数值计算的开源软件库。节点（Nodes）在图中表示数学操作，图中的线（edges）则表示在节点间相互联系的多维数据数组，即张量（tensor）。它灵活的架构让你可以在多种平台上展开计算，例如台式计算机中的一个或多个CPU（或GPU），服务器，移动设备等等。TensorFlow 最初由Google大脑小组（隶属于Google机器智能研究机构）的研究员和工程师们开发出来，用于机器学习和深度神经网络方面的研究，但这个系统的通用性使其也可广泛用于其他计算领域。

**什么是数据流图（Data Flow Graph）?**

数据流图用“结点”（nodes）和“线”(edges)的有向图来描述数学计算。“节点” 一般用来表示施加的数学操作，但也可以表示数据输入（feed in）的起点/输出（push out）的终点，或者是读取/写入持久变量（persistent variable）的终点。“线”表示“节点”之间的输入/输出关系。这些数据“线”可以输运“size可动态调整”的多维数据数组，即“张量”（tensor）。张量从图中流过的直观图像是这个工具取名为“Tensorflow”的原因。一旦输入端的所有张量准备好，节点将被分配到各种计算设备完成异步并行地执行运算。



## Keras基本概念

### 符号主义

符号主义的计算首先定义各种变量，然后建立一个“计算图”，计算图规定了各个变量之间的计算关系。建立好的计算图需要编译以确定其内部细节，然而，此时的计算图还是一个“空壳子”，里面没有任何实际的数据，只有当你把需要运算的输入放进去后，才能在整个模型中形成数据流，从而形成输出值。

### 张量(tensor)

使用这个词汇的目的是为了表述统一，张量可以看作是向量、矩阵的自然推广，我们用张量来表示广泛的数据类型。

规模最小的张量是0阶张量，即标量，也就是一个数。

当我们把一些数有序的排列起来，就形成了1阶张量，也就是一个向量

如果我们继续把一组向量有序的排列起来，就形成了2阶张量，也就是一个矩阵

张量的阶数有时候也称为维度，或者轴，轴这个词翻译自英文axis。譬如一个矩阵[[1,2],[3,4]]，是一个2阶张量，有两个维度或轴，沿着第0个轴（为了与python的计数方式一致，本文档维度和轴从0算起）你看到的是[1,2]，[3,4]两个向量，沿着第1个轴你看到的是[1,3]，[2,4]两个向量。

要理解“沿着某个轴”是什么意思，不妨试着运行一下下面的代码：

import numpy as np

a = np.array([[1,2],[3,4]])

sum0 = np.sum(a, axis=0)

sum1 = np.sum(a, axis=1)

print sum0

print sum1

### data\_format

这是一个无可奈何的问题，在如何表示一组彩色图片的问题上，Theano和TensorFlow发生了分歧，'th'模式，也即**Theano模式**会把100张RGB三通道的16×32（高为16宽为32）彩色图表示为下面这种形式（100,3,16,32），**Caffe采取的也是这种方式**。第0个维度是样本维，代表样本的数目，第1个维度是通道维，代表颜色通道数。后面两个就是高和宽了。这种theano风格的数据组织方法，称为“channels\_first”，即通道维靠前。

而TensorFlow，的表达形式是（100,16,32,3），即把通道维放在了最后，这种数据组织方式称为“channels\_last”。

Keras默认的数据组织形式在~/.keras/keras.json中规定，可查看该文件的image\_data\_format一项查看，也可在代码中通过K.image\_data\_format()函数返回，**请在网络的训练和测试中保持维度顺序一致**。

### Batch

深度学习的优化算法，说白了就是梯度下降。每次的参数更新有两种方式。

第一种，遍历全部数据集算一次损失函数，然后算函数对各个参数的梯度，更新梯度。这种方法每更新一次参数都要把数据集里的所有样本都看一遍，计算量开销大，计算速度慢，不支持在线学习，这称为**Batch gradient descent，批梯度下降**。

另一种，每看一个数据就算一下损失函数，然后求梯度更新参数，这个称为**随机梯度下降，stochastic gradient descent**。这个方法速度比较快，但是收敛性能不太好，可能在最优点附近晃来晃去，hit不到最优点。两次参数的更新也有可能互相抵消掉，造成目标函数震荡的比较剧烈。

为了克服两种方法的缺点，现在一般采用的是一种折中手段，**mini-batch gradient decent，小批的梯度下降**，这种方法把数据分为若干个批，按批来更新参数，这样，一个批中的一组数据共同决定了本次梯度的方向，下降起来就不容易跑偏，减少了随机性。另一方面因为批的样本数与整个数据集相比小了很多，计算量也不是很大。基本上现在的梯度下降都是基于mini-batch的，所以Keras的模块中经常会出现batch\_size，就是指这个。

**顺便说一句，Keras中用的优化器SGD是stochastic gradient descent的缩写，但不代表是一个样本就更新一回，还是基于mini-batch的。**

### epochs

epochs指的就是训练过程中数据将被“轮”多少次，就这样。

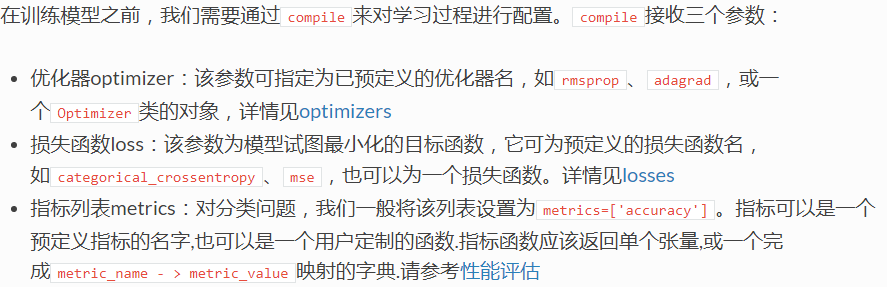
## 序贯（Sequential）模型

Keras的核心数据结构是“**模型**”，模型是一种组织网络层的方式。Keras中主要的模型是Sequential模型，**Sequential是一系列网络层按顺序构成的栈**。

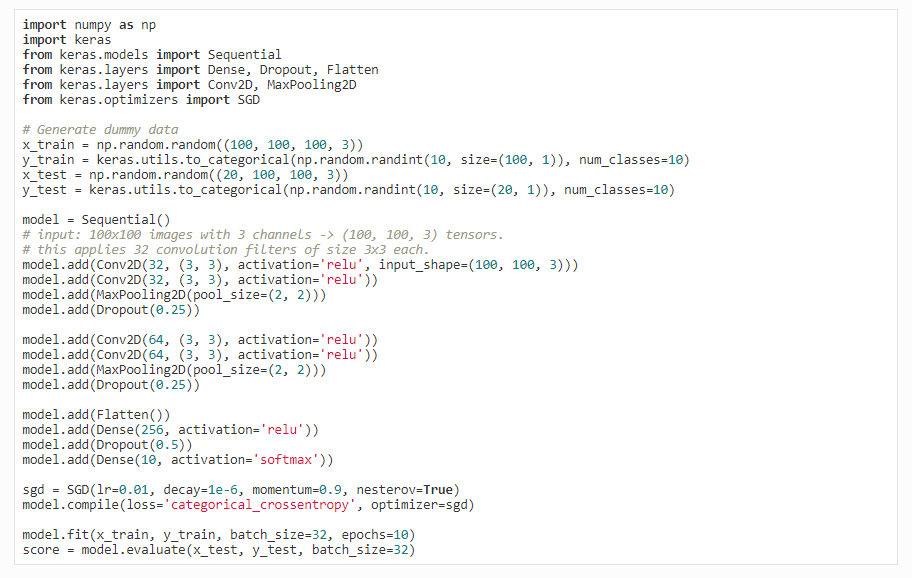
### 输入指定数据的shape



### 编译



### 例子：类似VGG的卷积神经网络



## 函数式（Functional）模型

函数式模型应用更为广泛，序贯模型是函数式模型的一种特殊情况。

from keras.models import Model

## 图片预处理



## 模型的保存与加载

你可以使用model.save(filepath)将Keras模型和权重保存在一个HDF5文件中该文件将包含：

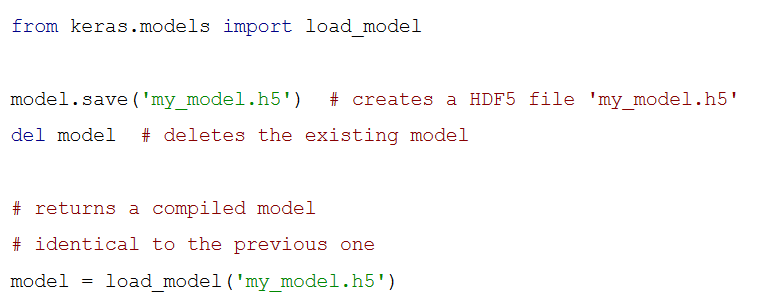
●模型的结构，以便重构该模型

●模型的权重

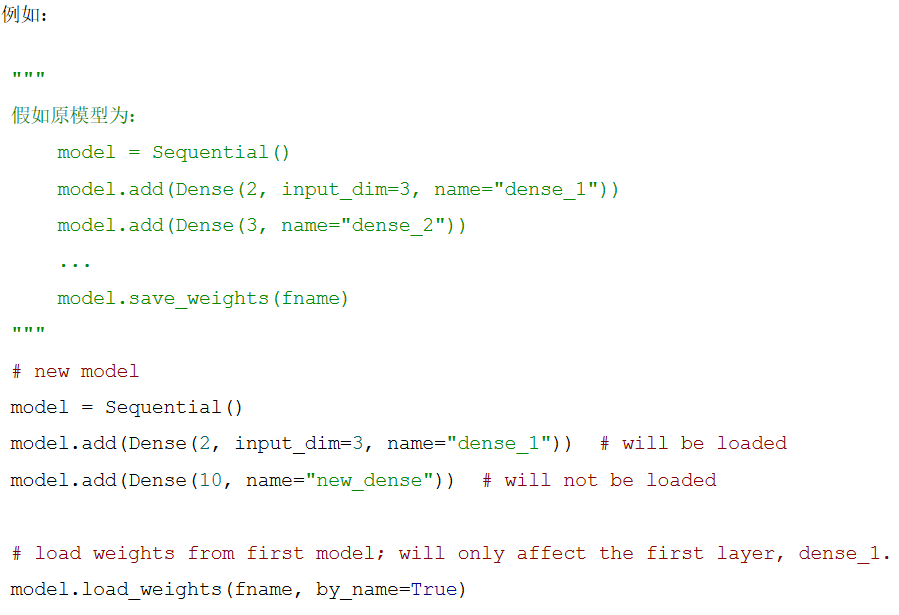
●训练配置（损失函数，优化器等）

●优化器的状态，以便于从上次训练中断的地方开始

使用keras.models.load\_model(filepath)来重新实例化你的模型，如果文件中存储了训练配置的话，该函数还会同时完成模型的编译







# 基于Region Proposal的深度学习目标检测算法

|  |  |
| --- | --- |
| 算法 | 优缺点 |
| R-CNN | 存在问题：  (1) 训练分为多个阶段，步骤繁琐: 微调网络+训练SVM+训练边框回归器  (2) 训练耗时，占用磁盘空间大：5000张图像产生几百G的特征文件  (3) 速度慢: 使用GPU, VGG16模型处理一张图像需要47s。 |
| SPP-NET | SPP-NET相比于R-CNN加快目标检测的速度，但是依然存在着很多问题：  (1) 训练分为多个阶段，步骤繁琐: 微调网络+训练SVM+训练训练边框回归器  (2) SPP-NET在微调网络的时候固定了卷积层，只对全连接层进行微调，而对于一个新的任务，有必要对卷积层也进行微调。（分类的模型提取的特征更注重高层语义，而目标检测任务除了语义信息还需要目标的位置信息） |
| Fast R-CNN | Fast R-CNN融合了R-CNN和SPP-NET的精髓，并且引入多任务损失函数，使整个网络的训练和测试变得十分方便。在Pascal VOC2007训练集上训练，在VOC2007测试的结果为66.9%(mAP)，如果使用VOC2007+2012训练集训练，在VOC2007上测试结果为70%（数据集的扩充能大幅提高目标检测性能）。使用VGG16每张图像总共需要3s左右。  缺点：region proposal的提取使用selective search，目标检测时间大多消耗在这上面（提region proposal 2~3s，而提特征分类只需0.32s），无法满足实时应用，而且并没有实现真正意义上的端到端训练测试（region proposal使用selective search先提取处来）。那么有没有可能直接使用CNN直接产生region proposal并对其分类？Faster R-CNN框架就是符合这样需要的目标检测框架。 |
| Faster R-CNN | Faster R-CNN将一直以来分离的region proposal和CNN分类融合到了一起，使用端到端的网络进行目标检测，无论在速度上还是精度上都得到了不错的提高。然而Faster R-CNN还是达不到实时的目标检测，预先获取region proposal，然后在对每个proposal分类计算量还是比较大。比较幸运的是YOLO这类目标检测方法的出现让实时性也变的成为可能。 |

除了基于Region Proposal的深度学习目标检测算法，还有基于回归方法的深度学习目标检测算法。(<https://zhuanlan.zhihu.com/p/21412911>)

## Reference:

[1] <http://keras-cn.readthedocs.io/en/latest/>

[2] <http://dataunion.org/24014.html>

[3] <https://zhuanlan.zhihu.com/p/21412911>

[4] <http://blog.csdn.net/u010159842/article/details/54407745>