	模型设计	接　口	部　署	性　能	架构设计	总体评分
TensorFlow	80	80	90	90	100	88
Caffe	60	60	90	80	70	72

续表

	模型设计	接　口	部　署	性　能	架构设计	总体评分
CNTK	50	50	70	100	60	66
Theano	80	70	40	50	50	58
Torch	90	70	60	70	90	76
MXNet	70	100	80	80	90	84
DeepLearning4J	60	70	80	80	70	72

本文接下来的篇幅将会重点介绍深度学习的三个框架caffe、tensorflow和keras，如果只是需要使用传统的机器学习基础算法使用scikit-learning和spark MLlib则更为合适。

三、深度学习框架比较

神经网络一般包括：训练，测试两大阶段。训练就是把训练数据和神经网络模型（AlexNet、RNN等神经网络训练框架Caffe等）用CPU或GPU提炼出模型参数的过程。测试就是把测试数据用训练好的模型（神经网络模型+模型参数）运行后查看结果。而caffe，keras，tensorflow就是把训练过程所涉及的环节数据统一抽象，形成可使用框架。

（一）Caffe

1、概念

Caffe是一个清晰而高效的深度学习框架，也是一个被广泛使用的开源深度学习框架，在Tensorflow出现之前一直是深度学习领域Github star最多的项目。主要优势为：上手容易，网络结构都是以配置文件形式定义，不需要用代码设计网络。训练速度快，组件模块化，可以方便的拓展到新的模型和学习任务上。但是Caffe最开始设计时的目标只针对于图像，没有考虑文本、语音或者时间序列的数据，因此Caffe对卷积神经网络的支持非常好，但是对于时间序列RNN，LSTM等支持的不是特别充分。Caffe工程的models文件夹中常用的网络模型比较多，比如Lenet、AlexNet、ZFNet、VGGNet、GoogleNet、ResNet等。

2、Caffe的模块结构

Caffe由低到高依次把网络中的数据抽象成Blob，各层网络抽象成Layer ，整个网络抽象成Net，网络模型的求解方法抽象成Solver。

1.Blob表示网络中的数据，包括训练数据，网络各层自身的参数，网络之间传递的数据都是通过Blob来实现的，同时Blob数据也支持在CPU与GPU上存储，能够在两者之间做同步。

2.Layer是对神经网络中各种层的抽象，包括卷积层和下采样层，还有全连接层和各种激活函数层等。同时每种Layer都实现了前向传播和反向传播，并通过Blob来传递数据。

3.Net是对整个网络的表示，由各种Layer前后连接组合而成，也是所构建的网络模型。

4.Solver 定义了针对Net网络模型的求解方法，记录网络的训练过程，保存网络模型参数，中断并恢复网络的训练过程。自定义Solver能够实现不同的网络求解方式。

3、安装方式

Caffe 需要预先安装比较多的依赖项，CUDA，snappy，leveldb，gflags，glog，szip，lmdb，OpenCV，hdf5，BLAS，boost、ProtoBuffer等；

Caffe官网：http://caffe.berkeleyvision.org/；

Caffe Github ： https://github.com/BVLC/caffe；Caffe 安装教程：

http://caffe.berkeleyvision.org/installation.html，

http://blog.csdn.net/yhaolpz/article/details/71375762；

Caffe 安装分为CPU和GPU版本，GPU版本需要显卡支持以及安装CUDA

4、使用Caffe搭建神经网络

【caffe搭建神经网络流程图】

使用流程	操作说明
1、数据格式处理	将数据处理成caffe支持格式，具体包括：LEVELDB,LMDB,内存数据，hdfs数据，图像数据，windows，dummy等。
2、编写网络结构文件	定义网络结构，如当前网络包括哪几层，每层作用是什么，使用caffe过程中最麻烦的一个操作步骤。具体编写格式可参考caffe框架自带自动识别手写体样例：caffe/examples/mnist/lenet_train_test.prototxt。
3、编写网络求解文件	定义了网络模型训练过程中需要设置的参数，比如学习率，权重衰减系数，迭代次数，使用GPU还是CPU等，一般命名方式为xx_solver.prototxt。可参考：caffe/examples/mnist/lenet_solver.prototxt。
4、训练	基于命令行的训练，如：caffe train -solver examples/mnist/lenet_solver.prototxt
5、测试	caffe test -model examples/mnist/lenet_train_test.prototxt -weights examples/mnist/lenet_iter_10000.caffemodel -gpu 0 -iterations

在上述流程中，步骤2是核心操作，也是caffe使用最让人头痛的地方，keras则对该部分做了更高层的抽象，让使用者能够快速编写出自己想要实现的模型。

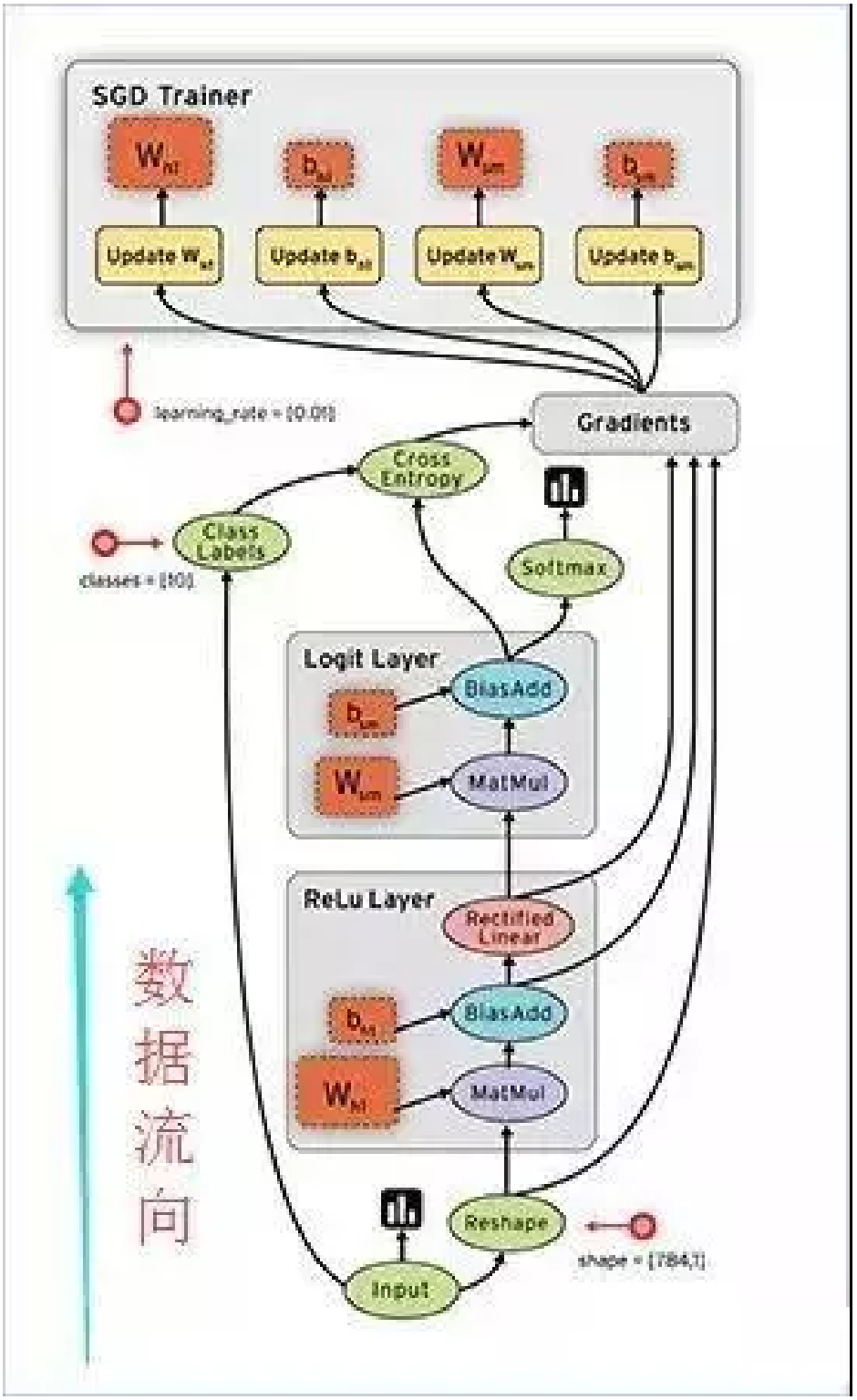
（二） Tensorflow

1、概念

TensorFlow是一个使用数据流图进行数值计算的开源软件库。图中的节点表示数学运算，而图边表示节点之间传递的多维数据阵列（又称张量）。灵活的体系结构允许使用单个API将计算部署到服务器或移动设备中的某个或多个CPU或GPU。Tensorflow涉及相关概念解释如下：

1）符号计算

符号计算首先定义各种变量，然后建立一个“计算图”，图中规定了各个变量之间的计算关系。 符号计算也叫数据流图，其过程如下图2-1所示，数据是按图中黑色带箭头的线流动的。



【2-1 数据流图示例】

数据流图用“结点”（nodes）和“线”(edges)的有向图来描述数学计算。

① “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。

② “线”表示“节点”之间的输入/输出关系。

③ 在线上流动的多维数据阵列被称作“张量”。

2）张量

张量(tensor)，可以看作是向量、矩阵的自然推广，用来表示广泛的数据类型，张量的阶数也叫维度。

0阶张量，即标量，是一个数。1阶张量，即向量，是一组有序排列的数。2阶张量，即矩阵，是一组向量有序的排列起来。3阶张量，即立方体，是一组矩阵上下排列起来。以此类推。

3）数据格式(data_format)

目前主要有两种方式来表示张量：

① th模式或channels_first模式，Theano和caffe使用此模式。

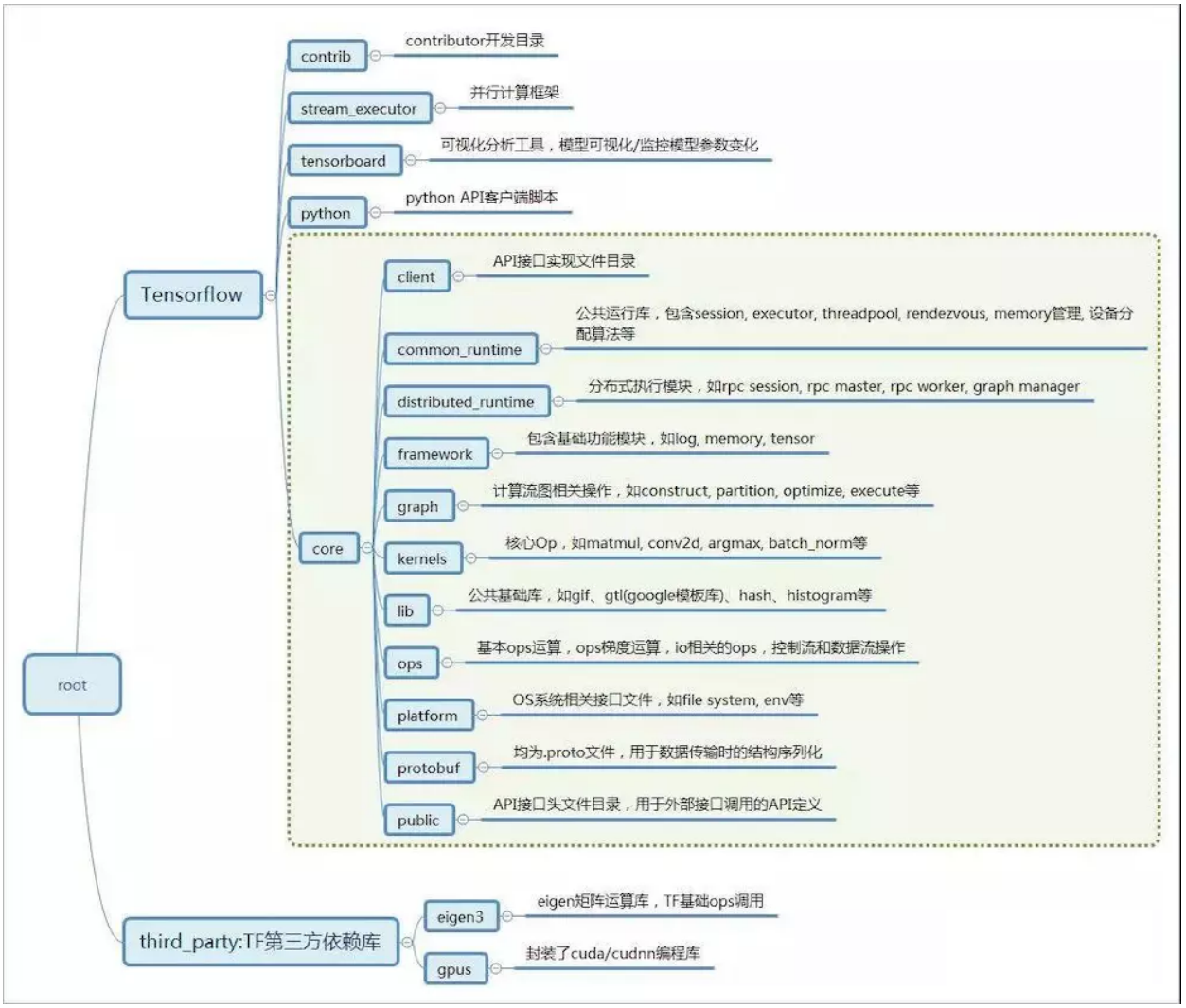
② tf模式或channels_last模式，TensorFlow使用此模式。

举例说明两种模式的区别：对于100张RGB3通道的16×32（高为16宽为32）彩色图，th表示方式：（100,3,16,32）tf表示方式：（100,16,32,3）唯一的区别就是表示通道个数3的位置不一样。

2、Tensorflow的模块结构

Tensorflow/core目录包含了TF核心模块代码，具体结构如图2-2所示：

【图 2-2 tensorflow代码模块结构】



3、安装方式

1、官网下载naconda安装：https://www.anaconda.com/download/；

2、依次在Anaconda Prompt控制台，按以下5个步骤输入指令进行安装：

1）安装py3+ cmd : conda create -n py3.6 python=3.6 anaconda；

2）激活虚拟环境 cmd : activate py3.6

3）激活TSF预安装cmd: conda create -n tensorflow python=3.6；activate tensorflow；

4）安装TSF：pip install —ignore-installed —upgrade tensorflow；pip install —ignore-installed —upgrade tensorflow-gpu；

5）退出虚拟环境cmd ：deactivate py3.6。

4、使用Tensorflow搭建神经网络

使用Tensorflow搭建神经网络主要包含以下6个步骤：

1) 定义添加神经层的函数；

2) 准备训练的数据；

3) 定义节点准备接收数据；

4) 定义神经层：隐藏层和预测层；

5) 定义loss表达式；

6) 选择optimizer使loss达到最小；

7) 对所有变量进行初始化，通过sess.run optimizer，迭代多次进行学习。

5、示例代码

Tensorflow 构建神经网络识别手写数字，具体代码如下所示：

```
importtensorflow as tfimportnumpy as np# 添加层defadd_layer(inputs, in_size, out_size, activation_function=None):    # add one more layer andreturnthe output ofthislayer    Weights= tf.Variable(tf.random_normal([in_size, out_size]))    biases = tf.Variable(tf.zeros([1, out_size]) +0.1)    Wx_plus_b = tf.matmul(inputs, Weights) + biases

ifactivation_function is None:          outputs = Wx_plus_b

else:          outputs = activation_function(Wx_plus_b)

returnoutputs#1.训练的数据# Make up some real data x_data = np.linspace(-1,1,300)[:, np.newaxis]noise = np.random.normal(0,0.05, x_data.shape)y_data = np.square(x_data) -0.5+ noise#2.定义节点准备接收数据# define placeholderforinputs to network   xs = tf.placeholder(tf.float32, [None,1])ys = tf.placeholder(tf.float32, [None,1])#3.定义神经层：隐藏层和预测层# add hidden layer 输入值是 xs，在隐藏层有10个神经元    l1 = add_layer(xs,1,10, activation_function=tf.nn.relu)# add output layer 输入值是隐藏层 l1，在预测层输出1个结果 prediction = add_layer(l1,10,1, activation_function=None)#4.定义 loss 表达式# the error between prediciton and real data      loss = tf.reduce_mean(tf.reduce_sum(tf.square(ys - prediction),                                    reduction_indices=[1]))#5.选择 optimizer 使 loss 达到最小                                  # 这一行定义了用什么方式去减少 loss，学习率是0.1

train_step = tf.train.GradientDescentOptimizer(0.1).minimize(loss)# important step 对所有变量进行初始化init = tf.initialize_all_variables()sess = tf.Session()# 上面定义的都没有运算，直到 sess.run 才会开始运算sess.run(init)# 迭代1000次学习，sess.run optimizerfori inrange(1000):# training train_step 和 loss 都是由 placeholder 定义的运算，所以这里要用 feed 传入参数
```

```
sess.run(train_step, feed_dict={xs: x_data, ys: y_data})ifi % 50==0:          #to see the step improvement

print(s

ess.run(loss, feed_dict={xs: x_data, ys: y_data}))
```

（三） Keras

1、概念

Keras由纯Python编写而成并基于Tensorflow、Theano以及CNTK后端，相当于Tensorflow、Theano、CNTK的上层接口，号称10行代码搭建神经网络，具有操作简单、上手容易、文档资料丰富、环境配置容易等优点，简化了神经网络构建代码编写的难度。目前封装有全连接网络、卷积神经网络、RNN和LSTM等算法。

Keras有两种类型的模型，序贯模型（Sequential）和函数式模型（Model），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。

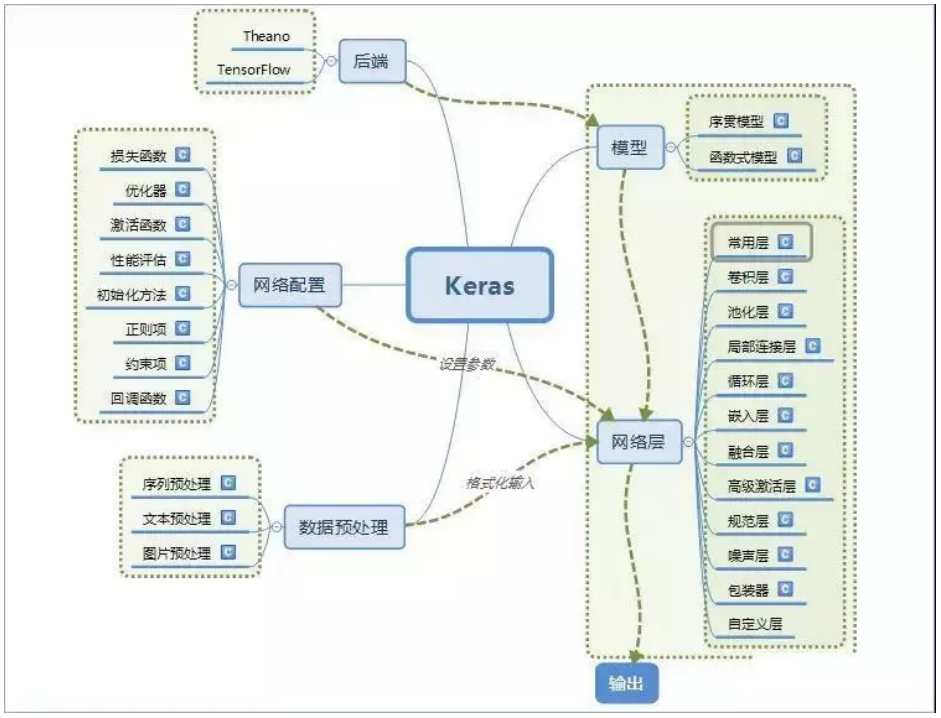
1）序贯模型（Sequential)：单输入单输出，一条路通到底，层与层之间只有相邻关系，没有跨层连接。这种模型编译速度快，操作也比较简单。

2）函数式模型（Model）：多输入多输出，层与层之间任意连接。这种模型编译速度慢。

2、Keras的模块结构

Keras主要由5大模块构成，模块之间的关系及每个模块的功能如图3-1所示：

【图 3-1 keras模块结构图】



3、安装方式

Keras的安装为以下三个步骤：

1）安装anaconda（python）；

2）用于科学计算的python发行版，支持Linux、Mac、Windows系统，提供了包管理与环境管理的功能，可以很方便的解决多版本python并存、切换以及各种第三方包安装问题；

3）利用pip或者conda安装numpy、keras、 pandas、tensorflow等库；

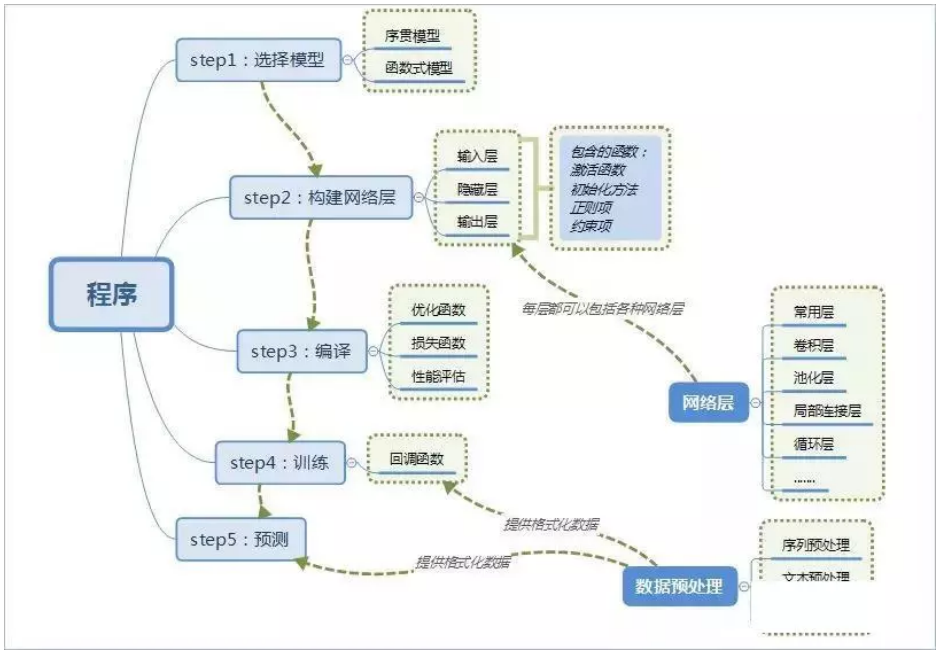
下载地址：

https://www.anaconda.com/what-is-anaconda/。

4、使用Keras搭建神经网络

使用keras搭建一个神经网络，包括5个步骤，分别为模型选择、构建网络层、编译、训练和预测。每个步骤操作过程中使用到的keras模块如图3-2所示：

【3-2 使用keras搭建神经网络步骤】



5、示例代码

Kears构建神经网络识别手写数字，具体代码如下所示：

```
from keras.models import Sequential

from keras.layers.core import Dense, Dropout, Activation

from keras.optimizers import SGD

from keras.datasets import mnist

import numpy

'''

    第一步：选择模型

'''

model = Sequential()

'''

   第二步：构建网络层

'''

model.add(Dense(500,input_shape=(784,))) # 输入层，28*28=784
```

```
model.add(Activation('tanh')) # 激活函数是tanh

model.add(Dropout(0.5)) # 采用50%的dropout

model.add(Dense(500)) # 隐藏层节点500个

model.add(Activation('tanh'))

model.add(Dropout(0.5))

model.add(Dense(10)) # 输出结果是10个类别，所以维度是10

model.add(Activation('softmax')) # 最后一层用softmax作为激活函数

'''

   第三步：编译

'''

sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True) # 优化函数，设定学习率（lr）等参数


model.compile(loss='categorical_crossentropy', optimizer=sgd, class_mode='categorical') # 使用交叉熵作为loss函数

'''

   第四步：训练

   .fit的一些参数

   batch_size：对总的样本数进行分组，每组包含的样本数量

   epochs ：训练次数

   shuffle：是否把数据随机打乱之后再进行训练

   validation_split：拿出百分之多少用来做交叉验证

   verbose：屏显模式 0：不输出    1：输出进度    2：输出每次的训练结果

'''
```

```
(X_train, y_train), (X_test, y_test) = mnist.load_data() # 使用Keras自带的mnist工具读取数据（第一次需要联网）

# 由于mist的输入数据维度是(num, 28, 28)，这里需要把后面的维度直接拼起来变成784维

X_train = X_train.reshape(X_train.shape[0], X_train.shape[1] * X_train.shape[2])

X_test = X_test.reshape(X_test.shape[0], X_test.shape[1] * X_test.shape[2])

Y_train = (numpy.arange(10) == y_train[:, None]).astype(int)

Y_test = (numpy.arange(10) == y_test[:, None]).astype(int)

model.fit(X_train,Y_train,batch_size=200,epochs=50,shuffle=True,verbose=0,validation_split=0.3)

model.evaluate(X_test, Y_test, batch_size=200, verbose=0)

'''

    第五步：输出

'''

print("test set")

scores = model.evaluate(X_test,Y_test,batch_size=200,verbose=0)

print("")

print("The test loss is %f" % scores)

result = model.predict(X_test,batch_size=200,verbose=0)

result_max = numpy.argmax(result, axis = 1)

test_max = numpy.argmax(Y_test, axis = 1)

result_bool = numpy.equal(result_max, test_max)

true_num = numpy.sum(result_bool)

print("")

print("The accuracy of the model is %f" % (true_num/len(result_bool)))
```

（四）框架优缺点对比

对比维度CaffeTensorflowKeras

上手难度1、不用写代码，只需在.prototxt文件中定义网络结构就可以完成模型训练。2、安装复杂，且在.prototxt 文件内部设计网络结构比较受限，没有在 Python中设计网络结构方便自由。3、配置文件不能用编程的方式调整超参数，对交叉验证、超参数Grid Search 等操作无法很方便的支持。1、安装简便，教学资源丰富，根据样例能快速搭建出基础模型。2、有一定的使用门槛。编程范式和数学统计，都让非机器学习或数据科学背景的使用者有上手难度。3、因为其灵活性，因此是一个相对底层的框架，使用时需要编写大量的代码，重新发明轮子。1、安装简单，旨在让用户进行最快速的原型实验，让想法变为结果的这个过程最短，非常适合最前沿的研究。2、API使用方便，用户只需要将高级的模块拼在一起，就可以设计神经网络，降低了编程和阅读别人代码时的理解开销。

框架维护GitHub项目，由伯克利视觉学中心（Berkeley Vision and Learning Center，BVLC）进行维护。被定义为最流行、最被认可的开源深度学习框架，框架结构优秀，拥有产品级的高质量代码，由 Google团队进行开发和维护，以及能力的加持。依然由google团队开发支持，API以tf.keras的形式打包在TensorFlow中；微软维护其CNTK后端；亚马逊AWS也在开发MXNet支持。其他支持的公司包括NVIDIA、Uber、苹果（通过CoreML）。

支持语言C++/CudaC++ python (Go，Java，Lua，Javascript，或者是R)Python

封装算法1、对卷积神经网络CNN的支持非常好，拥有大量训练好的经典模型（AlexNet、VGG、Inception）以及state-of-the-art（ResNet等）等模型，收藏在Model Zoo。2、对时间序列 RNN、LSTM 等支持得不是特别充分1、支持CNN与RNN， 也支持深度强化学习乃至其他计算密集的科学计算(如偏微分方程求解等)。2、计算图必须构建为静态图，这让很多计算变得难以实现，尤其是序列预测中经常使用的beam search。1、支持CNN和循环网络，支持级联的模型或任意的图结构的模型，从CPU上计算切换到GPU加速无须任何代码的改动。2、没有增强学习工具箱，自己修改实现很麻烦。封装得太高级，训练细节不能修改、penalty细节很难修改。

模型部署1、程序运行稳定，代码质量高，适合对稳定性要求严格的生产环境，第一个主流的工业级深度学习框架。2、Caffe的底层基于C++，可在各种硬件环境编译并具有良好的移植性，支持Linux、Mac和Windows，也可以编译部署到移动设备系统如Android和iOS 上。1、性能好，可以同时运行多个大规模深度学习模型，支持模型生命周期管理、算法实验，并可以高效地利用GPU资源，让训练好的模型更快捷方便地投入到实际生产环境。2、灵活的移植性，可以将同一份代码几乎不经过修改就轻松地部署到有任意数量CPU或GPU的PC、服务器或者移动设备上。1、部署简便，使用TensorFlow、CNTK、Theano作为后端，简化了编程的复杂度，节约了尝试新网络结构的时间。2、模型越复杂，收益越大，尤其是在高度依赖权值共享、多模型组合、多任务学习等模型上，表现得非常突出。

性能目前仅支持单机多GPU的训练，不支持分布式的训练。支持分布式计算，使GPU或TPU(Tensor Processing Unit)集群并行计算，共同训练出一个模型。对不同设备间的通信优化得不是很好，分布式性能还没有达到最优无法直接使用多GPU，对大规模的数据处理速度没有其他支持多 GPU和分布式的框架快。用TensorFLow backend后端时速度比纯TensorFLow下要慢很多。

欢迎工作一到五年的Java工程师朋友们加入Java进阶高级架构：855355016

本群提供免费的学习指导 架构资料 以及免费的解答

不懂得问题都可以在本群提出来 之后还会有职业生涯规划以及面试指导