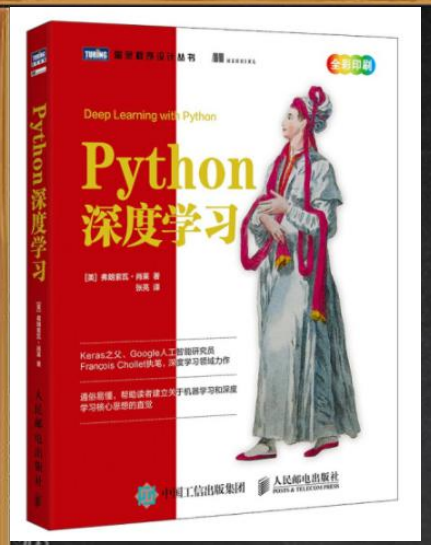


初识神经网络

祥水村东头@AI识堂

《Python 深度学习》读书笔记系列课程(对应2.1节 pp20-23)

本次胶片内容、及涉及相关代码均可移步至Github进行下载

我的代码 Github 地址：
https://github.com/david-cai/Reading-Note-for-Chollet-of-Deep-Learning-with-Python

目录

01

从0到1搭建神经网络

- 五步法
- 备数据--->搭模型--->译模型--->训模型--->评模型

关于神经网络学习方法的探讨

- 先理论，后实践

“要深入理解深度学习，需要熟悉很多数学概念，包括张量、张量运算、微分、梯度下降等”

- 先实践，后理论

“要快速了解深度学习，可以用更形象化的表达替代‘令人反感的数学符号’，在初学阶段避免因为繁复的数学表达而停滞不前”p20

神经网络工作流——“三阶段＋五步＋三定”

What？
（确定什么样的网络）

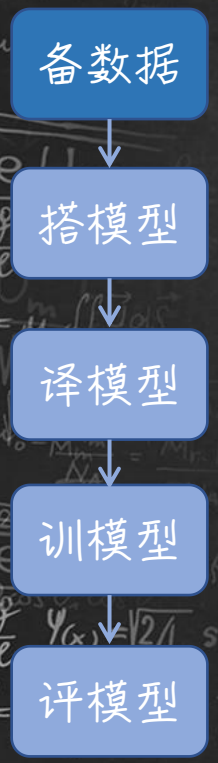
how？
（确定网络如何学习）

Why？
（确定学习后的为什么好或坏）

一备数据：获取训练集/（验证集）/测试集 定样本、定标签、定数据处理方式	mnist.load_data()
二搭模型：初始化模型-->按需添加层 定输入输出张量、定层、定神经节点及激活函数	models.Sequential() model.add()
三译模型：编译模型（制定神经网络学习规则，详见2.4节） 定损失函数、定优化器、定监控指标	model.compile()
四训模型：对训练集样本数据进行训练 定输入、定输出、定训练规模	model.fit()
五评模型：评估模型表现，分析过拟合、欠拟合等问题 定模型在训练集/验证集/测试集上的表现	model.evaluate()

神经网络工作流 之 一：备数据

备数据，即提前准备好构建/训练/预测神经网络过程中所需要的相关数据。



➢ 定样本(sample)

- 确定自变量集合

-训练样本集：事前，用于模型确定最佳参数，必须

-验证样本集：事中，用于监测模型在训练过程不同阶段的表现，非必须

-测试样本集：事后，用于评估训练后模型在其他数据集上的表现，考察泛化能力，必须

➢ 定标签（label）

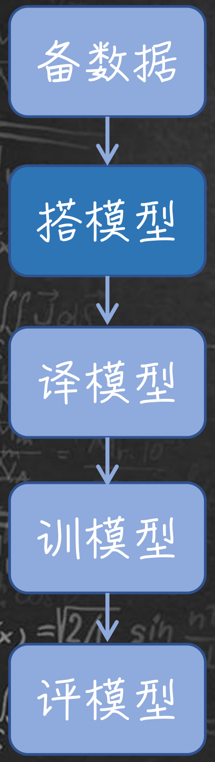
- 确定因变量，简单来说取决于不同的任务场景

如对红酒等级进行分类，其标签为离散型变量（A/B/C/D）；如对未来一周气温进行预测，其标签为连续性变量（23.5度，24度）

➢ 定数据预处理方式

- 将原始数据转换为更加合理、可靠的数据，如归一化处理等

神经网络工作流 之 二：搭模型（2.3节会重点讲解）



搭模型，即构建神经网络模型的内部结构、输入输出。

➢ 定输入输出张量

- 基于样本确定模型的输入张量规格，如3维输入张量 a x b x c
- 根据任务目标确定模型的输出张量规格

例如猫狗分类任务中输出为1个通道（二分类问题），该通道值表示特定输入是猫的概率（A），而（1-A）表示另一个类别的概率；再如单个数字识别任务重输出为10个通道（多分类问题，0-9，共10个类别），即特定输入在10个分类上的各自概率。

➢ 定层

- 确定模型由几层神经网络构成，每层网络实现什么样的功能。

例如，全连接层（Dense层）常用于解决分类问题，卷积层（Convlution层）常用于提取图像的特征

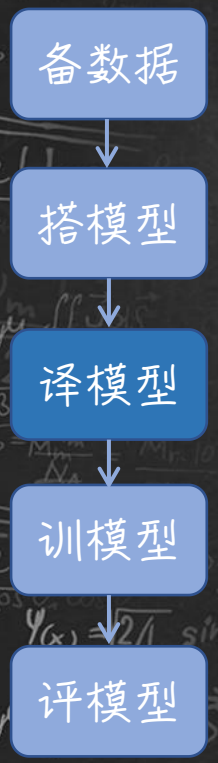
➢ 定神经节点及激活函数

- 确定每层神经网络由多少个节点构成、采用什么样的激活函数。

例如二分类问题通常采用Sigmoid作为激活函数；多分类问题通常采用Softmax作为激活函数。

神经网络工作流 之 三：译模型（2.4节会重点讲解）

译模型，即编译模型，制定神经网络中的参数学习（更新）的有关规则。



➢ 定损失函数（loss function）

- 我们需要掌握真实值（labels,y）与样本（samples）在当前网络参数下所得输出值（预测值,y'）之间的差异程度
- 我们的目标是缩小这种差异程度(即缩小y与y'之间的差异)，衡量这种差异程度的数学表达式称为“损失函数”。

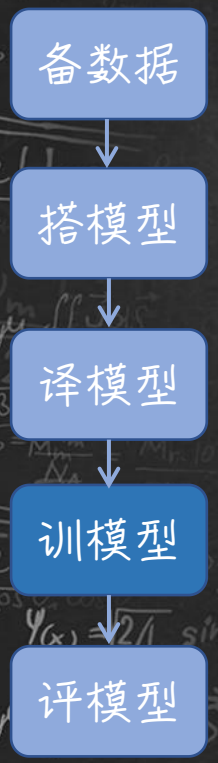
➢ 定优化器(optimizer)

- 根据这种差异程度，确定网络参数更新规则，更新规则的数学表达式称为优化函数（优化器）
- 使得差异程度逐步缩小，进而使损失函数找到最优解

➢ 定监控指标(metric)

- 确定损失函数、优化函数后，我们需要监控每一轮（训练集全部输入模型后）模型在训练集上的表现，这种表现的数学表达式称为监控指标
- 可用于观察模型表现性能随着训练轮数（epochs）增加的变化趋势，更进一步分析过拟合/欠拟合等问题。

神经网络工作流 之 四：训模型



训模型，即按照第2步创建好的模型架构、第3步制定好的模型学习规则，将训练集中的样本（Samples）和标签（Labels）灌入模型，模型根据输入开始自主学习。

➢ 定输入

- 对训练集中的样本进行预处理，以适配模型对输入张量的数据规格要求
- 常见的预处理方式包括数据维度转换等；此步可与第1步“备数据”中的数据预处理合并操作。

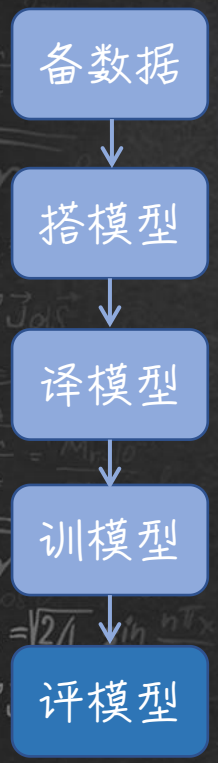
➢ 定输出

对训练集中的标签进行预处理，以适配模型对输出张量的数据规格要求

➢ 定训练次数

- 确定训练迭代轮数（epochs），即全量训练集的样本需要重复输入模型多少次
- 在每一轮训练过程中，全量训练集的样本全部输入到模型需要分n批，则每一批所包含的样本数（batch_size）

神经网络工作流 之 五：评模型（第三章会详细讲解）



评模型，即评估训练后的模型表现。

➢ 定评估对象

- 可观察模型在训练集、验证集、测试集上的表现。

➢ 定评估标准

- 确定模型在指定数据集上的表现，如误差率
- 通常是将测试集的样本输入到训练后的模型，得到预测标签结果，并衡量预测标签与真实标签之间的误差程度。

02

实践案例——MNIST数字图像分类任务

实践案例，keras中的“hello world”（pp21）：

- ✓ 通过构建一个简单的模型来识别手写数字：将一个(28 x 28)的灰度图像划分为10个类别，即0-9
- ✓ mnist是keras自带的手写数字数据集，该数据集包括60,000张训练图集、10,000张测试图集，来源于美国国家标准与技术研究院

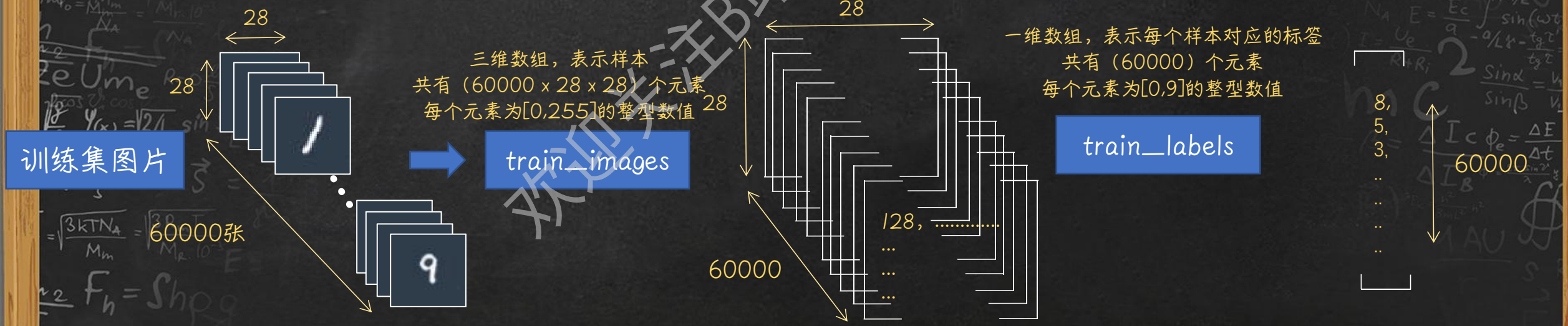
（同5.4节任务，只不过实现方式不同）

https://keras.io/api/datasets/mnist/

step 1：备数据（1/2）

```
#加载keras中预定义好与mnist相关的函数库，可以帮助我们高效处理数据
from keras.datasets import mnist

#原书数据集会下载到.../.keras/datasets/mnist.npz
#从原始数据集中通过函数load_data()获取训练集与测试集的样本和标签
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

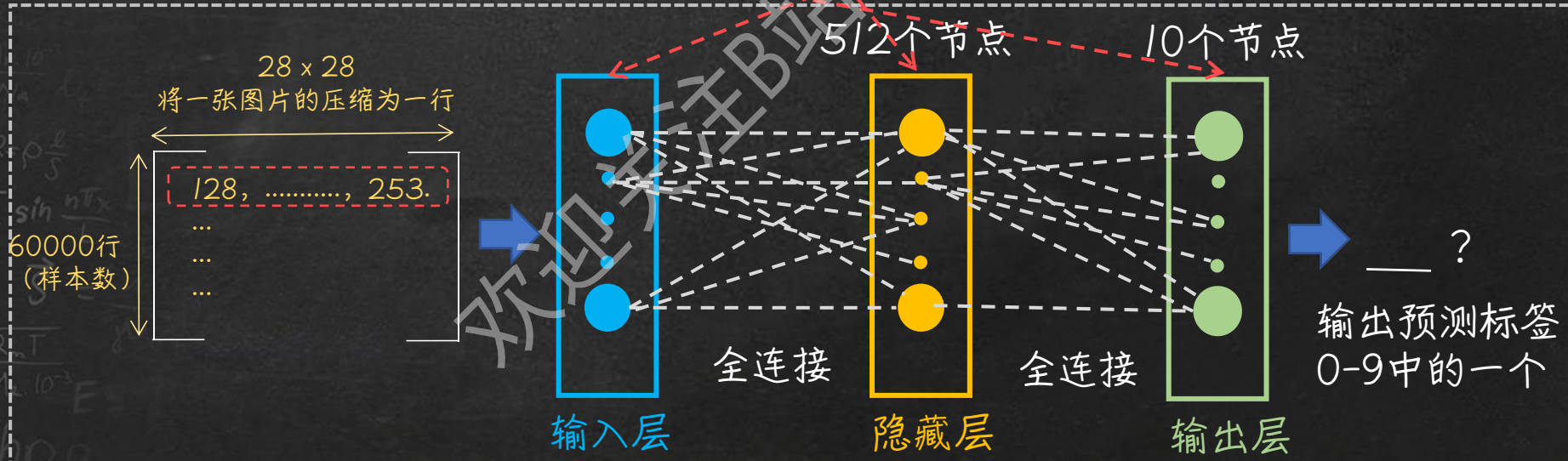


step 2：搭模型（2.3节还会重点讲解）

```
#初始化神经网络模型
network = models.Sequential()

#叠加首层——全连接层,首层还需要指定input_shape
network.add(layers.Dense(512,activation = 'relu', input_shape = (28 * 28 , )))

#叠加第二层——全连接层，注意是躲分类问题，所以激活函数应使用softmax
network.add(layers.Dense(10, activation = 'softmax'))
```



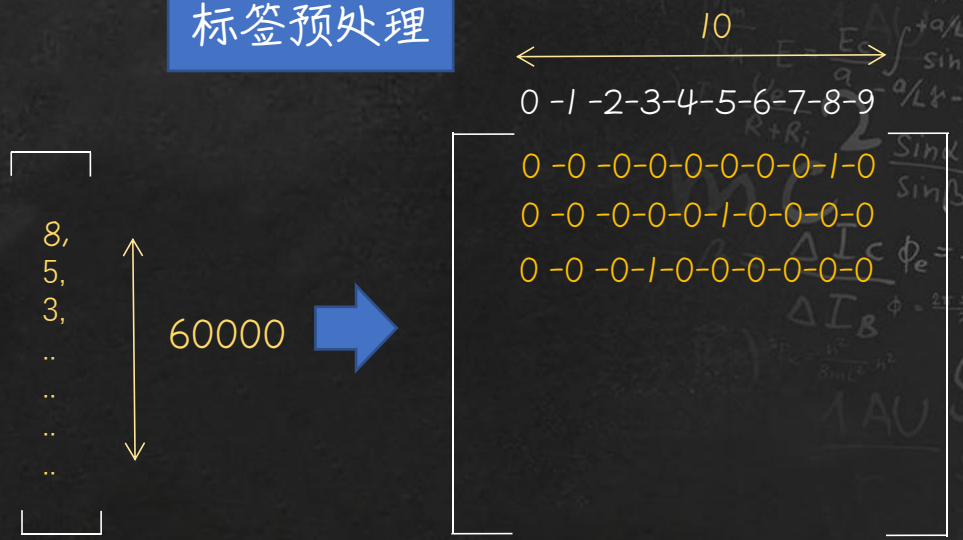
step 1：备数据（2/2）

```
#降维，(60000,28,28) ---> (60000, 28x28)，三维数组变二维数组
processed_train_images =
train_images.reshape((60000,28*28))
#转换数据类型,并归一化
processed_train_images =
processed_train_images.astype('float32')/255
```

```
#独热编码
from keras.utils import to_categorical
processed_train_labels = to_categorical(train_labels)
```


样本预处理

标签预处理



step 3：译模型（2.4节会再次进行详细讲解）

```
#编译模型，预设优化器、损失函数、监控指标
network.compile(optimizer = 'rmsprop',
loss = 'categorical_crossentropy',
metrics = ['accuracy'])
```

计算预测标签值与真实标签值之间的损失

朝着减少损失的方向更新神经网络参数

监测每一轮训练后模型预测的准确程度

-常见的损失函数（目标函数）
mse(均方误差)
categorical_crossentropy(多分类场景)
binary_crossentropy(二分类场景)
-不同的损失函数只是具体的算法不同
-本质是衡量预测值与真实值之间的差异程度

-常见的优化器
rmsprop/adam/momentum...
-不同优化器只是具体算法不同
-本质上是对目标函数求导（求梯度），找到极值点

-常见的监测指标
accuracy（分类正确率）
sparse_accuracy...
-不同监测只是具体算法不同
-本质上是观察每一轮训练后的模型在训练集的预测准确程度的变化过程

step 4：训模型

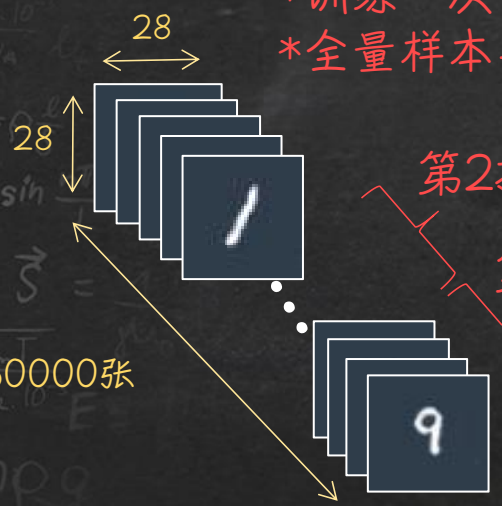
```
network.fit( processed_train_images, processed_train_labels, batch_size = 32, epochs = 10)
```

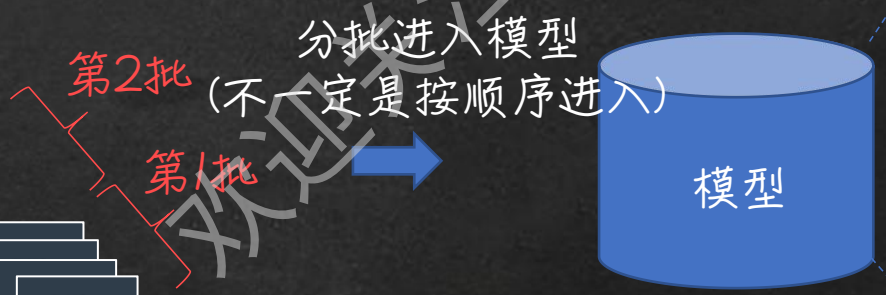
samples、batch_size、epochs之间的关系

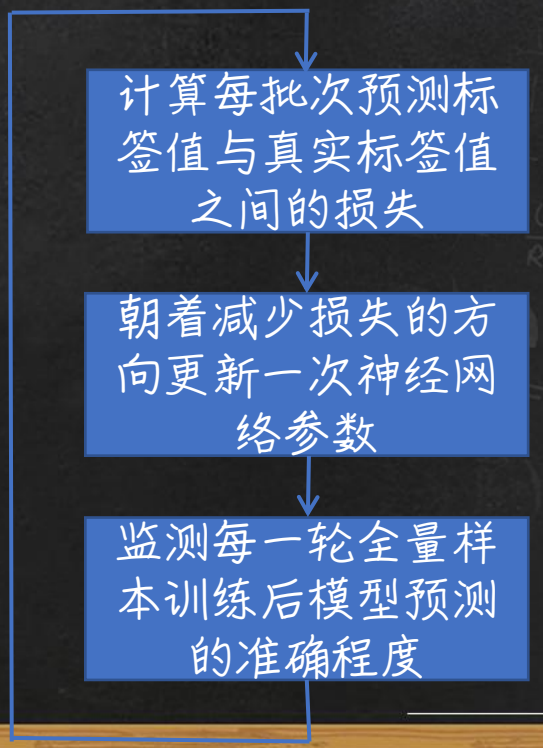
假设训练集总共有m个样本（samples），全量样本分n批次进入（batches），则每批次一起进入模型的样本有m/n个

全量样本需要反复进入模型k次（epochs）

*每批次包含32个样本（缺省值）
*训练一次全量样本需要1875个批次
*全量样本要反复进入模型10轮







step 5：评模型

```
#考察训练后的模型在测试集上的表现，导入测试集样本、标签
test_loss, test_acc = network.evaluate(processed_test_images, processed_test_labels)
```

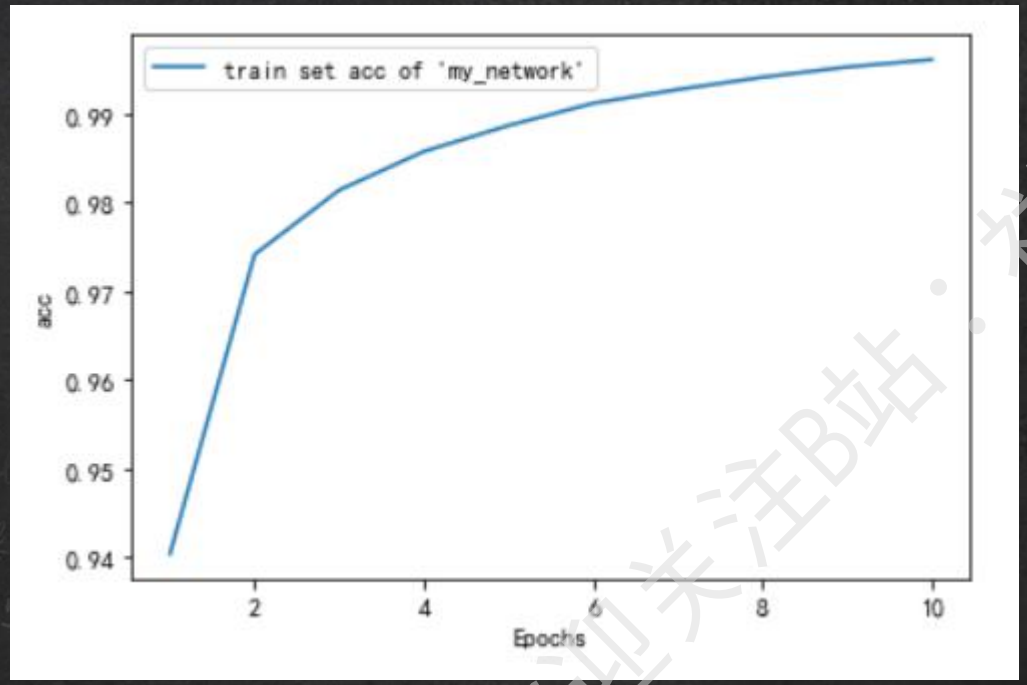
模型在测试集上的分类正确率

测试集样本

测试集真实标签值

- 全连接层模型（包含2个dense层）：97.99%
- 卷积模型（CNN，5.4节会涉及到）：99%

评估模型在训练集上的表现



- 假设训练了10轮
- 观察模型在每一轮中的训练集分类准确率
- 随着轮数不断增加，分类准确率会趋近于某一个极限值
- 通常，我们会对比模型在训练集和验证集上的分类准确率变化曲线，以此分析模型是否出现过拟合/欠拟合

“人生之旅历途甚长，所争决不在一年半日，
万不可因此着急失望，招精神上之萎靡”

——梁启超致梁思成家书
献给正在逐梦路上努力奔跑的你我他

感谢聆听

THANK YOU

本次胶片内容、及涉及相关代码均可移步至Github进行下载
感谢您的投币三连！

我的代码 Github 地址：
https://github.com/david-cai/Reading-Note-for-Chollet-of-Deep-Learning-with-Python