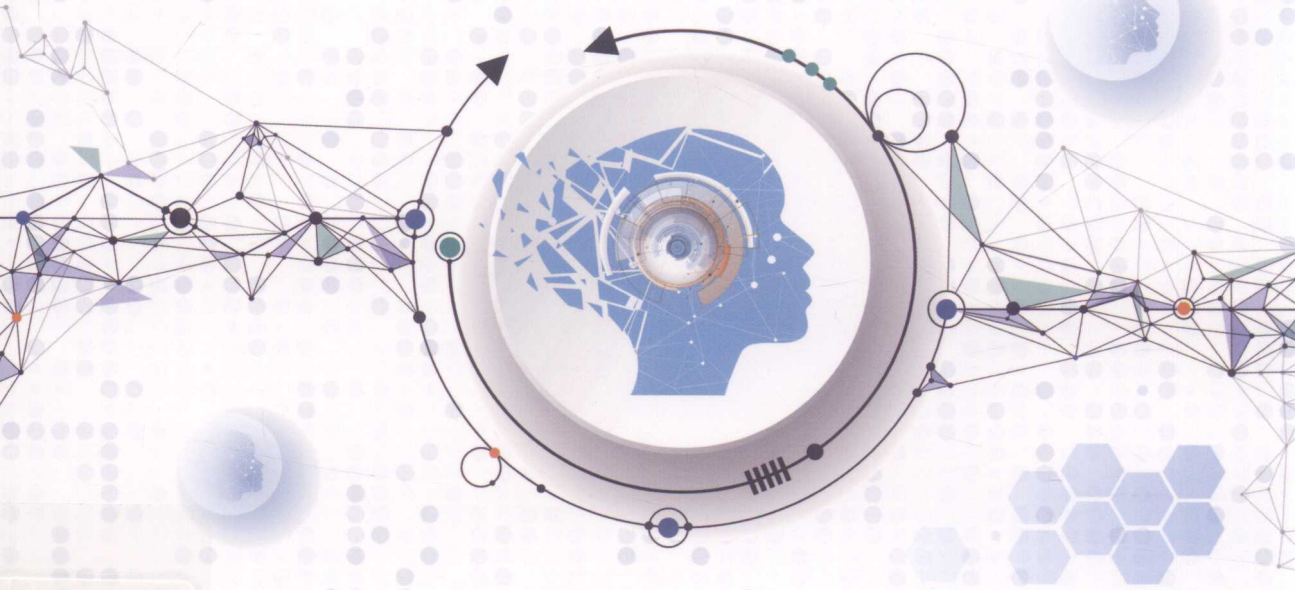
HZ BOOKS 华章IT
[PACKT] PUBLISHING
智能系统与技术丛书

Practical Convolutional Neural Networks
Implement advanced deep learning models using Python

实用卷积神经网络

运用Python实现高级深度学习模型

[印度] 莫希特·赛瓦克（Mohit Sewak）
[孟加拉] 穆罕默德·礼萨·卡里姆（Md. Rezaul Karim）著
[美] 普拉蒂普·普贾里（Pradeep Pujari）
王彩霞 译



机械工业出版社
China Machine Press

智能系统与技术丛书

Practical Convolutional Neural Networks
Implement advanced deep learning models using Python

实用卷积神经网络

运用Python实现高级深度学习模型

[印度] 莫希特·赛瓦克（Mohit Sewak）
[孟加拉] 穆罕默德·礼萨·卡里姆（Md. Rezaul Karim）著
[美] 普拉蒂普·普贾里（Pradeep Pujari）
王彩霞 译

机械工业出版社
China Machine Press

人工智能爱好者。获取使用极大数据集和不同 CNN 架构的实践经验，从而构建高效、智能的卷积网络模型。本书读者最好对深度学习基本概念和 Python 编程语言基础知识已经有所了解。

各章概览

第 1 章对深度神经网络的科学原理和实现这种网络的不同框架以及框架背后的数学机制提供一个快速回顾。

第 2 章向读者介绍卷积神经网络，并展示如何利用深度学习从图像中提取信息。

第 3 章从零开始针对图像分类问题构建一个简单的 CNN，并阐明如何调整参数、优化训练时间以及 CNN 的性能，以分别提高效率和准确率。

第 4 章介绍几种经典的（在竞赛中胜出的）CNN 架构的优势和运作机制，以及它们之间的差异和如何使用这些架构。

第 5 章讲授如何使用预先训练好的网络，并使其适用于新的且不同的数据集。在实际应用中也有一种自定义分类问题，它使用的技术称为转移学习。

第 6 章介绍一种称为自编码器的无监督学习技术，同时介绍了 CNN 自编码器的不同应用，比如图像压缩。

第 7 章讲授目标检测、实例分割和图像分类的区别。然后介绍多种使用 CNN 进行目标检测和实例分割的技术。

第 8 章探究生成式 CNN 网络，然后将其与我们学习得到的有识别力的 CNN 网络相结合，用 CNN/GAN 创造新的图像。

第 9 章讲授深度学习中注意力背后的思想，并学习如何使用基于注意力的模型来实现一些高级解决方案（图像捕捉和 RAM）。我们还将了解不同类型的注意力以及强

化学习在硬注意力机制中的作用。

充分利用本书

本书主要用 Python 语言构建 CNN。我们使用 Python 2.7（2x）来构建各种应用程序，并且基于 Python、Spyder、Anaconda、PyCharm 构建开源的企业级专业软件。许多示例也能兼容 Python 3x。作为一种好的实践，我们鼓励用户使用 Python 虚拟环境来实现这些代码。

本书主要关注如何以最佳的方式利用各种 Python 库和深度学习库（如 Keras、TensorFlow 和 Caffe）来构建真实世界的应用程序。本着这种精神，我们尽量保持所有代码的友好性和可读性，以便使读者能够更容易地理解代码，并能在不同的场景中复用这些代码。

下载示例代码及彩色图像

本书的示例代码及所有截图和样图，可以从 http://www.packtpub.com 通过个人账号下载，也可以访问华章图书官网 http://www.hzbook.com，通过注册并登录个人账号下载。

排 版 约 定

代码块设置如下：

```
import tensorflow as tf

#Creating TensorFlow object
hello_constant = tf.constant('Hello World!', name = 'hello_constant')
#Creating a session object for execution of the computational graph
with tf.Session() as sess:
```

当要强调代码块的特定部分时，相关代码行或者单词会设置为粗体：

```
x = tf.subtract(1, 2,name=None) # -1
y = tf.multiply(2, 5,name=None) # 10
```

警告或重要提示的标记。

提示或技巧的标记。

ABOUT THE AUTHORS

关于作者

Mohit Sewak 是 IBM 的高级认知数据科学家，也是比尔拉技术与科学学院的人工智能和计算机科学博士。他在人工智能、深度学习和机器学习方面拥有多项专利和著作。他曾是一些非常成功的人工智能 / 机器学习软件和行业解决方案的首席数据科学家，并在早期就参与了沃森认知商业产品线的解决方案研究。他在 TensorFlow、Torch、Caffe、Theano、Keras、Watson 等架构的设计和解决方案方面有 14 年的丰富经验。

Md. Rezaul Karim 是德国 Fraunhofer FIT 的研究科学家。他也是德国亚琛工业大学的博士研究生。在加入 FIT 之前，他曾在爱尔兰的 Insight 数据分析中心担任研究员。他也曾是韩国三星电子的首席工程师。

他在 C++、Java、R、Scala 和 Python 方面有 9 年的研发经验，也在生物信息学、大数据和深度学习方面发表过研究论文。此外在 Spark、Zeppelin、Hadoop、Keras、Scikit-Learn、TensorFlow、Deeplearning4j、MXNet、H2O 等方面都有实际的工作经验。

Pradeep Pujari 是沃尔玛实验室的机器学习工程师，也是 ACM 的杰出成员。他的核心专业领域是信息检索、机器学习和自然语言处理。在空闲的时候，他喜欢研究人工智能技术、阅读和辅导。

ABOUT THE REVIEWER

关于审阅者

Sumit Pal 是 Apress 出版社的作家。他在软件领域（从初创企业到成熟企业）有超过 22 年的工作经验，他从事大数据、数据可视化和数据科学方面的工作，是一位独立的顾问。他构建了端到端的数据驱动分析系统。

他曾为微软（SQLServer）、甲骨文（OLAP Kernel）和威瑞森（Verizon）工作过。他为客户提供数据架构方面的建议，并用 Spark 和 Scala 构建解决方案。他曾在北美和欧洲的许多会议上发表演讲，并为 Experfy 培养大数据分析师。他有计算机科学硕士和学士学位。

CONTENTS

目　录

01 CHAPTER

第1章 深度神经网络概述

在过去几年，我们已经见证了人工智能（深度学习）领域的显著进步。如今，深度学习是许多先进技术应用的基础，从自动驾驶汽车到自动创作艺术作品和音乐。科学家的目标是要让计算机不仅能理解语音，而且还能讲自然语言。深度学习是一种基于学习数据表示的机器学习方法，而不是特定的算法。深度学习能够使计算机从更简单和更小的概念构建复杂的概念。例如，深度学习系统通过将较低的标记边缘和角结合起来，并以分层的方式将它们组合成身体的各部分来识别一个人的形象。在不远的将来，深度学习将扩展到能让机器独立思考的相关应用。

本章将讨论以下主题：

- 创建神经网络块。
- TensorFlow 介绍。
- Keras 介绍。
- 反向传播。

1.1 创建神经网络块

神经网络由很多人工神经元组成。它是大脑的一种表征还是某种知识的数学表征？这里将试图讲解如何在实践中使用神经网络。卷积神经网络（CNN）是一种非常

特殊的多层神经网络。CNN 设计的目的是，以最小的数据处理代价直接从图像中识别出视觉模式。此网络的图形表示如图 1-1 所示。神经网络领域最初是受生物神经系统建模目标所启发的，但是从那时起，它就向不同的方向发展，成为一个工程问题，并在机器学习任务中取得良好效果。

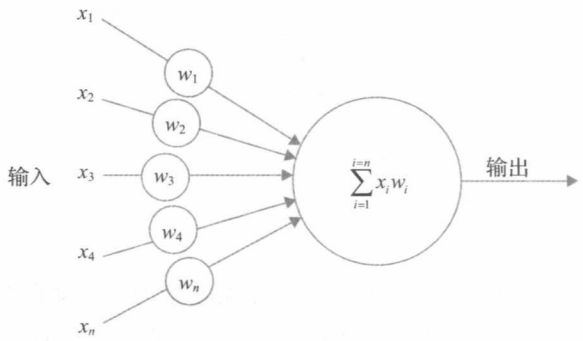


图 1-1　CNN 的图形表示

人工神经元是一个接收输入并产生输出的函数。使用神经元的个数取决于要解决的问题。可能少至两个，也可能多至几千。有很多种将人工神经元连接在一起构建 CNN 的方法。一种常用的拓扑结构是前馈网络。

每个神经元接收来自其他神经元的输入。每个输入项对神经元的影响由权重控制，权重可以是正的也可以是负的。整个神经网络通过理解这种范式进而执行有效的计算来识别对象。现在我们把这些神经元连接成一个网络，称之为前馈网络。这意味着每一层的神经元将其输出向前传递到下一层，直到得到最终输出。相应公式如下：

$$w_1x_1 + w_2x_2 + \cdots + w_nx_n$$

$$\sum_{i=1}^{i=n} w_i x_i = w_1x_1 + w_2x_2 + \cdots + w_nx_n$$

前向传播神经元的实现如下：

```
import numpy as np
import math
```

```
class Neuron(object):
    def __init__(self):
        self.weights = np.array([1.0, 2.0])
        self.bias = 0.0
    def forward(self, inputs):
        """ Assuming that inputs and weights are 1-D numpy arrays and the
bias is a number """
        a_cell_sum = np.sum(inputs * self.weights) + self.bias
        result = 1.0 / (1.0 + math.exp(-a_cell_sum)) # This is the sigmoid
activation function
        return result
neuron = Neuron()
output = neuron.forward(np.array([1,1]))
print(output)
```

1.2　TensorFlow 介绍

TensorFlow 是基于计算图实现的。思考下面的数学表达式：

$$c = (a + b),\ d = b + 5$$

$$e = c * d$$

图 1-2 是 TensorFlow 的一个计算图。因为计算都是并行完成的，所以这很强大。

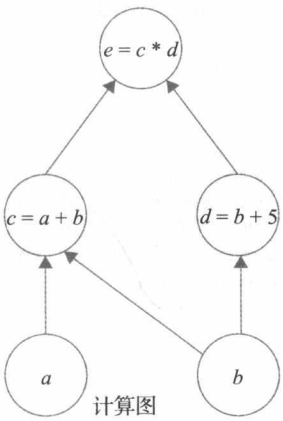


图 1-2　TensorFlow 计算图

TensorFlow 安装

有两种简单的 TensorFlow 安装方法：

- 使用虚拟环境（推荐并在此说明）。
- 使用 Docker 镜像。

1. 针对 macOS X/Linux 不同版本

下面代码片段用于创建一个 Python 虚拟环境，并在该环境安装 TensorFlow。运行此代码之前，需要先安装 Anaconda：

```
#Creates a virtual environment named "tensorflow_env" assuming that python
3.7 version is already installed.
conda create -n tensorflow_env python=3.7
#Activate points to the environment named "tensorflow"
source activate tensorflow_env
conda install pandas matplotlib jupyter notebook scipy scikit-learn
#installs latest tensorflow version into environment tensorflow_env
pip3 install tensorflow
```

请在 TensorFlow 官方页面查看最新版本。

尝试在你的 Python 控制台运行以下代码来验证 TensorFlow 的安装是否成功。如果安装成功，控制台应该打印出“Hello World!”。

```
import tensorflow as tf

#Creating TensorFlow object
hello_constant = tf.constant('Hello World!', name = 'hello_constant')
#Creating a session object for execution of the computational graph
with tf.Session() as sess:
    #Implementing the tf.constant operation in the session
    output = sess.run(hello_constant)
    print(output)
```

2. TensorFlow 基础

在 TensorFlow 中，数据不是以整型、浮点型、字符串或其他原始数据类型存储的，而是用一个称为张量的数据结构来表示这些数据。它是一个由原始值组成的任意维的数组。张量的维数称为阶。在前面的例子中，hello_constant 是一个 0 阶的常量字符串张量。一些常量张量的例子如下：

```
# A is an int32 tensor with rank = 0
A = tf.constant(123)
# B is an int32 tensor with dimension of 1 ( rank = 1 )
B = tf.constant([123,456,789])
# C is an int32 2- dimensional tensor
C = tf.constant([ [123,456,789], [222,333,444] ])
```

TensorFlow 程序的核心是计算图。计算图是由以下两部分组成的有向图：

- 创建计算图。
- 运行计算图。

计算图在会话中执行。会话是计算图的运行时环境，它负责分配 CPU、GPU 并管理 TensorFlow 运行时状态。下面的代码使用 tf.Session 创建一个名为 sess 的会话实例。然后运行 sess.run() 函数进行张量的运算，并将返回结果存储在 output 变量中。最终打印出“Hello World!”

```
with tf.Session() as sess:
    # Run the tf.constant operation in the session
    output = sess.run(hello_constant)
    print(output)
```

可以使用 TensorBoard 可视化图形。运行 TensorBoard 的命令如下：

```
tensorboard --logdir=path/to/log-directory
```

如下所示，让我们创建一段简单的加法代码。首先创建一个整型常量 x，赋值 5；然后创建一个新的变量 y，赋值为 x 加 5，打印出 y 的值：

```
constant_x = tf.constant(5, name='constant_x')
variable_y = tf.Variable(x + 5, name='variable_y')
print (variable_y)
```

不同的是，没有像在 Python 代码中那样将 x+5 的结果直接赋给变量 variable_y，而是写成了一个方程。这意味着当计算变量 y 时，取 x 在那个时间点的值然后加上 5。variable_y 值的计算在前面的代码中从未实际执行过。上面的这段代码实际上属于典型的 TensorFlow 程序的计算图创建部分。运行这段代码后，你将得到类似

于 <tensorflow.python.ops.variables.Variable object at 0x7f074bfd9ef0> 的内容，而不是 variable_y 的实际值 10。为了解决这个问题，必须执行计算图的代码部分，如下所示：

```
#initialize all variables
init = tf.global_variables_initializer()
# All variables are now initialized

with tf.Session() as sess:
    sess.run(init)
    print(sess.run(variable_y))
```

这里执行一些基本的数学函数，如张量间的加、减、乘、除。

3. TensorFlow 中的数学基础

tf.add() 函数取 2 个数字，2 个张量，或 1 个数字和 1 个张量，然后以张量的形式返回它们的和：

```
Addition
x = tf.add(1, 2, name=None) # 3
```

减法和乘法的例子如下：

```
x = tf.subtract(1, 2,name=None) # -1
y = tf.multiply(2, 5,name=None) # 10
```

如果想用非常量该怎么处理呢？如何向 TensorFlow 提供输入数据集？为此，TensorFlow 提供了一个 API——tf.placeholder()，并运用 feed_dict 类型。

placeholder 是后续会在 tf.session.run() 函数中赋予值的变量，基于此，我们可以在暂时没有数据的情况下创建我们的操作和计算图。然后，在 tf.session.run() 函数中使用 feed_dict 参数值设置 placeholder 张量，通过这些占位符（placeholder）将数据输入到计算图中。如下示例，在会话运行之前，将字符串“Hello World”设置给张量 x：

```
x = tf.placeholder(tf.string)

with tf.Session() as sess:
    output = sess.run(x, feed_dict={x: 'Hello World'})
```

也可以使用 feed_dict 参数值设置多个张量，如下所示：

```
x = tf.placeholder(tf.string)
y = tf.placeholder(tf.int32, None)
z = tf.placeholder(tf.float32, None)

with tf.Session() as sess:
    output = sess.run(x, feed_dict={x: 'Welcome to CNN', y: 123, z:
123.45})
```

占位符也可以在多维情况下存储数组，请参见如下示例：

```
import tensorflow as tf

x = tf.placeholder("float", [None, 3])
y = x * 2

with tf.Session() as session:
    input_data = [[1, 2, 3],
                  [4, 5, 6],]
    result = session.run(y, feed_dict={x: input_data})
    print(result)
```

当传递给 feed_dict 参数的数据与张量类型不匹配或者不能强制转换为张量类型时，将抛出错误“ValueError: invalid literal for...”。

tf.truncated_normal() 函数返回一个具有正态分布随机值的张量。此函数主要用于网络权重初始化：

```
n_features = 5
n_labels = 2
weights = tf.truncated_normal((n_features, n_labels))
with tf.Session() as sess:
  print(sess.run(weights))
```

4. TensorFlow 中的 softmax 函数

softmax 函数将其输入（被称为 logit 或者 logit score）转换为 0 到 1 之间的值，