Deep Learning with R Cookbook

Over 45 unique recipes to delve into neural network techniques using R 3.5.x

Swarna Gupta, Rehan Ali Ansari and Dipayan Sarkar

www.packt.com

Deep Learning with R Cookbook

Over 45 unique recipes to delve into neural network techniques using R 3.5.x

Swarna Gupta
Rehan Ali Ansari
Dipayan Sarkar

BIRMINGHAM - MUMBAI

Deep Learning with R Cookbook

Commissioning Editor: Sunith Shetty
Acquisition Editor: Yogesh Deokar
Content Development Editor: Nathanya Dias
Senior Editor: Ayaan Hoda
Technical Editor: Joseph Sunil
Copy Editor: Safis Editing
Project Coordinator: Aishwarya Mohan
Proofreader: Safis Editing
Indexer: Priyanka Dhadke
Production Designer: Jyoti Chauhan

First published: February 2020

Production reference: 1210220

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham
B3 2PB, UK.

ISBN 978-1-78980-567-3

`www.packt.com`

This book is dedicated to my mother, Mrs. Purnima Gupta; my father, Mr. Nandkumar Gupta; my sister, Rashmi Gupta; my brother, Rajat Gupta; and my husband, Rehan Ali Ansari. None of this would have been possible without their eternal support and motivation.

-Swarna Gupta

This book is dedicated to my mother, Mrs. Shama Rukhsana Parveen; my father, Mr. Yunus Ali Ansari; my sisters, Rubina and Shamina; my wife, Swarna; and the joy of my life, my nieces, Alvina and Fatima. Words are not sufficient to express my gratitude to all of them for cheering me and raising my spirit.

-Rehan Ali Ansari

This book is dedicated to Debomitra Kodak, Felicia Leow, Pravalika Aitipamula, and Rangarajan Narayanan who have been my support and inspiration throughout this journey. Without them, this book would not have been possible. A special thanks to Neelima Jauhari and Pravalika Aitipamula.

-Dipayan Sarkar

Foreword

Data and AI give the best hope to the toughest problems that the world faces today. Am I making a sweeping statement? Not really—it's a modest statement of fact. From robotics to self-driving cars, farming that alleviates world hunger, to finding a solution to early diagnostics to critical illness—deep learning is one of the most enthralling areas of discovery and disruption. It has also fuelled the transformation of numerous businesses such as media and entertainment, insurance, healthcare, retail, education, and information technology.

This book is the perfect material for every data science enthusiast who wants to understand the concepts of deep learning: with R codes explained comprehensibly, it is the best place to start. The authors have maintained a perfect balance between theoretical and practical aspects of deep learning algorithms and applications. It turned out to be a great read—thanks to the easy flow of various sections such as Getting Ready, How to Do it, and How it Works.
After starting with some good insights on how to set up a deep learning environment in a local system, the authors address how the reader can leverage various cloud platforms such as AWS, Microsoft Azure, and Google Cloud to scale deep learning applications. If you are looking for some quick thoughts on any topic, you can read any chapter individually without getting bogged about the sequence.

An interesting fact about this book is that it not only covers the generic topics of deep learning such as CNN, RNN, GAN, Autoencoders but also throws light on specific state-of-the-art techniques such as transfer learning and reinforcement learning. I like the practical examples in the chapters: Working with Convolutional Networks, Deep Generative models, Working with Text and Audio and NLP. They are bound to kindle some thought-starters on what can be done using image and text data. The data sets are very aptly chosen for the examples provided.

Overall, this book is an engaging and inspiring read. I congratulate the writers of the book-Swarna, Rehan, and Dipayan for their contribution to this field of study and I look forward to more such works from them.

Pradeep Jayaraman

Head of Analytics, Adani Ports & SEZ

Contributors

About the authors

Swarna Gupta holds a BE in computer science and has 6 years' experience in data science. She is currently working with Rolls Royce as a data scientist. Her work revolves around leveraging deep learning and machine learning to create value for the business. She has extensively worked on IoT-based projects in the vehicle telematics and solar manufacturing industries. During her current association with Rolls Royce, she implemented various deep learning techniques to build advanced analytics capabilities in the aerospace domain. Swarna also manages to find the time in her busy schedule to be a regular pro-bono contributor to social organizations, helping them to solve specific business problems with the help of data science and machine learning.

Rehan Ali Ansari has a BE in electrical and electronics engineering with 5 years' experience in data science. He is currently associated with digital competency at AP Moller Maersk Group in the capacity of a data scientist. Rehan has a diverse background of working across multiple domains including fashion and retail, IoT, the renewable energy sector, trade finance, and supply chain management. Rehan is a firm believer in the agile method for developing AI products and solutions. He holds a strong insight into the latest technologies in the field of data science. Outside of his busy schedule, Rehan manages to explore new areas in the field of robotics and AI.

Dipayan Sarkar holds an Masters in economics and has over 17 years' experience. He has won international challenges in predictive modeling and takes a keen interest in the mathematics behind machine learning techniques. Before opting to become an independent consultant and mentor in the data science and machine learning space with various organizations and educational institutions, he served as a senior data scientist with Fortune 500 companies in the U.S. and Europe. He is currently associated with the Great Lakes Institute of Management as a visiting faculty (analytics), and BML Munjal University as an adjunct faculty (analytics and machine learning). He has co-authored a book *Ensemble Machine Learning with Python*, available from Packt Publishing.

About the reviewer

Sray Agarwal has been working as a data scientist for the last 12 years and has gained experience in various domains, including BFSI, e-commerce, retail, telecommunications, hospitality, travel, education, real estate, and entertainment, among many other sectors. He is currently working for Publicis Sapient as a data scientist and is based out of London. His expertise lies in predictive modeling, forecasting, and advanced machine learning. He possesses a deep understanding of algorithms and advanced statistics. He has a background in management and economics and has attained an MSc-equivalent qualification in data science and analytics. He is also a SAS-Certified Predictive Modeler. His current area of interest is fair and explainable machine learning.

Packt is searching for authors like you

If you're interested in becoming an author for Packt, please visit `authors.packtpub.com` and apply today. We have worked with thousands of developers and tech professionals, just like you, to help them share their insight with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Table of Contents

Preface

Deep learning has taken a huge step in recent years with developments including generative adversarial networks (GANs), variational autoencoders, and deep reinforcement learning. This book serves as a reference guide in R 3.x that will help you implement deep learning techniques.

This book walks you through various deep learning techniques that you can implement in your applications using R 3.x. A unique set of recipes will help you solve regression, binomial classification, and multinomial classification problems, and explores hyper-parameter optimization in detail. You will also go through recipes that implement **convolutional neural networks** (**CNNs**), **recurrent neural networks** (**RNNs**), **long short-term memory** (**LSTM**) networks, sequence-to-sequence models, GANs, and reinforcement learning. You will learn about high-performance computation involving large datasets that utilize GPUs, along with parallel computation capabilities in R, and you will also get familiar with libraries such as MXNet, which is designed for efficient GPU computing and state-of-the-art deep learning. You will also learn how to solve common and not-so-common problems in NLP, such as object detection and action identification, and you will leverage pre-trained models in deep learning applications.

By the end of the book, you will have a logical understanding of deep learning and different deep learning packages and will be able to build the most appropriate solutions to your problems.

Who this book is for

This book is for data scientists, machine learning practitioners, deep learning researchers, and AI enthusiasts who want to learn key tasks in the deep learning domain using a recipe-based approach. You will implement deep learning techniques and algorithms in common and not-so-common challenges faced in research work or projects. A strong understanding of machine learning and a working knowledge of R is mandatory.

What this book covers

`Chapter 1`, *Understanding Neural Networks and Deep Neural Networks*, will show us how to set up a deep learning environment to train models. The readers are then introduced to neural networks, starting from how neural networks work, what hidden layers are, what backpropagation is, and what activation functions are. This chapter uses the `keras` library to demonstrate the recipes.

`Chapter 2`, *Working with Convolutional Neural Networks*, will show us CNNs and will explain how they can be used to train models for image recognition and natural language processing based tasks. This chapter also covers various hyperparameters and optimizers used with CNNs.

`Chapter 3`, *Recurrent Neural Networks in Action*, will show us the fundamentals of RNNs with real-life implementation examples. We will also introduce LSTMs and gated recurrent units (GRUs), an extension of RNNs, and take a detailed walk-through of LSTM hyper-parameters. In addition to this, readers will learn how to build a bi-directional RNN model using Keras.

`Chapter 4`, *Implementing Autoencoders with Keras*, will introduce the implementation of various types of autoencoders using the `keras` library as the backend. Readers will also learn about various applications of autoencoders, such as dimensionality reduction and image coloring.

`Chapter 5`, *Deep Generative Models*, will show us the architecture of another method of deep neural networks, **generative adversarial networks** (**GANs**). We will demonstrate how to train a GAN model comprising of two pitting nets—a generator and a discriminator. This chapter also covers the practical implementation of variational autoencoders and compares them with GANs.

`Chapter 6`, *Handling Big Data Using Large-Scale Deep Learning*, contains case studies on high-performance computation involving large datasets utilizing GPUs. Readers will also be introduced to the parallel computation capabilities in R and libraries such as MXNet, which is designed for efficient GPU computing and state-of-the-art deep learning.

`Chapter 7`, *Working with Text and Audio for NLP*, contains case studies on various topics involving sequence data, including natural language processing (NLP) and speech recognition. The readers will implement end-to-end deep learning algorithms using various deep learning libraries.

`Chapter 8`, *Deep Learning for Computer Vision*, will provide end-to-end case studies on object detection and face identification.

`Chapter 9`, *Implementing Reinforcement Learning*, will walk us through the concepts of reinforcement learning step by step. Readers will learn about various methods, such as Markov Decision Processes, Q-Learning, and experience replay, and implement these methods in R using examples. Readers will also implement an end-to-end reinforcement learning example using R packages such as `MDPtoolbox` and `Reinforcementlearning`.

To get the most out of this book

A good understanding of machine learning and strong knowledge of R is necessary for this book.

Download the example code files

You can download the example code files for this book from your account at `www.packt.com`. If you purchased this book elsewhere, you can visit `www.packtpub.com/support` and register to have the files emailed directly to you.

You can download the code files by following these steps:

1. Log in or register at `www.packt.com`.
2. Select the **Support** tab.
3. Click on **Code Downloads**.
4. Enter the name of the book in the **Search** box and follow the onscreen instructions.

Once the file is downloaded, please make sure that you unzip or extract the folder using the latest version of:

- WinRAR/7-Zip for Windows
- Zipeg/iZip/UnRarX for Mac
- 7-Zip/PeaZip for Linux

The code bundle for the book is also hosted on GitHub at `https://github.com/PacktPublishing/Deep-Learning-with-R-Cookbook`. In case there's an update to the code, it will be updated on the existing GitHub repository.

We also have other code bundles from our rich catalog of books and videos available at `https://github.com/PacktPublishing/`. Check them out!

Download the color images

We also provide a PDF file that has color images of the screenshots/diagrams used in this book. You can download it here: `http://www.packtpub.com/sites/default/files/downloads/9781789805673_ColorImages.pdf`.

Conventions used

There are a number of text conventions used throughout this book.

`CodeInText`: Indicates code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles. Here is an example: "In step 1, we imported the fashion MNIST data using the `dataset_fashion_mnist()` function and checked the dimensions of its training and testing partitions."

A block of code is set as follows:

```
fashion <- dataset_fashion_mnist()
x_train <- fashion$train$x
y_train <- fashion$train$y
x_test <- fashion$test$x
y_test <- fashion$test$y
```

Bold: Indicates a new term, an important word, or words that you see onscreen. For example, words in menus or dialog boxes appear in the text like this. Here is an example: "Go to **Anaconda Navigator** from the **Start** menu."

Warnings or important notes appear like this.

Tips and tricks appear like this.

Sections

In this book, you will find several headings that appear frequently (*Getting ready*, *How to do it...*, *How it works...*, *There's more...*, and *See also*).

To give clear instructions on how to complete a recipe, use these sections as follows:

Getting ready

This section tells you what to expect in the recipe and describes how to set up any software or any preliminary settings required for the recipe.

How to do it…

This section contains the steps required to follow the recipe.

How it works…

This section usually consists of a detailed explanation of what happened in the previous section.

There's more…

This section consists of additional information about the recipe in order to make you more knowledgeable about the recipe.

See also

This section provides helpful links to other useful information for the recipe.

Get in touch

Feedback from our readers is always welcome.

General feedback: If you have questions about any aspect of this book, mention the book title in the subject of your message and email us at `customercare@packtpub.com`.

Errata: Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you have found a mistake in this book, we would be grateful if you would report this to us. Please visit `www.packtpub.com/support/errata`, selecting your book, clicking on the Errata Submission Form link, and entering the details.

Piracy: If you come across any illegal copies of our works in any form on the Internet, we would be grateful if you would provide us with the location address or website name. Please contact us at `copyright@packt.com` with a link to the material.

If you are interested in becoming an author: If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, please visit `authors.packtpub.com`.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions, we at Packt can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about Packt, please visit `packt.com`.

1
Understanding Neural Networks and Deep Neural Networks

Deep learning has transformed many traditional businesses, such as web search, advertising, and many more. A major challenge with the traditional machine learning approaches is that we need to spend a considerable amount of time choosing the most appropriate feature selection process before modeling. Besides this, these traditional techniques operate with some level of human intervention and guidance. However, with deep learning algorithms, we can get rid of the overhead of explicit feature selection since it is taken care of by the models themselves. These deep learning algorithms are capable of modeling complex and non-linear relationships within the data. In this book, we'll introduce you to how to set up a deep learning ecosystem in R. Deep neural networks use sophisticated mathematical modeling techniques to process data in complex ways. In this book, we'll showcase the use of various deep learning libraries, such as `keras` and `MXNet`, so that you can utilize their enriched set of functions and capabilities in order to build and execute deep learning models, although we'll primarily focus on working with the `keras` library. These libraries come with CPU and GPU support and are user-friendly so that you can prototype deep learning models quickly.

In this chapter, we will demonstrate how to set up a deep learning environment in R. You will also get familiar with various TensorFlow APIs and how to implement a neural network using them. You will also learn how to tune the various parameters of a neural network and also gain an understanding of various activation functions and their usage for different types of problem statements.

In this chapter, we will cover the following recipes:

- Setting up the environment
- Implementing neural networks with Keras
- TensorFlow Estimator API
- TensorFlow Core API
- Implementing a single-layer neural network
- Training your first deep neural network

Setting up the environment

Before implementing a deep neural network, we need to set up our system and configure it so that we can apply a variety of deep learning techniques. This recipe assumes that you have the Anaconda distribution installed on your system.

Getting ready

Let's configure our system for deep learning. It is recommended that you create a deep learning environment in Anaconda. If you have an older version of R in the conda environment, you need to update your R version to 3.5.x or above.

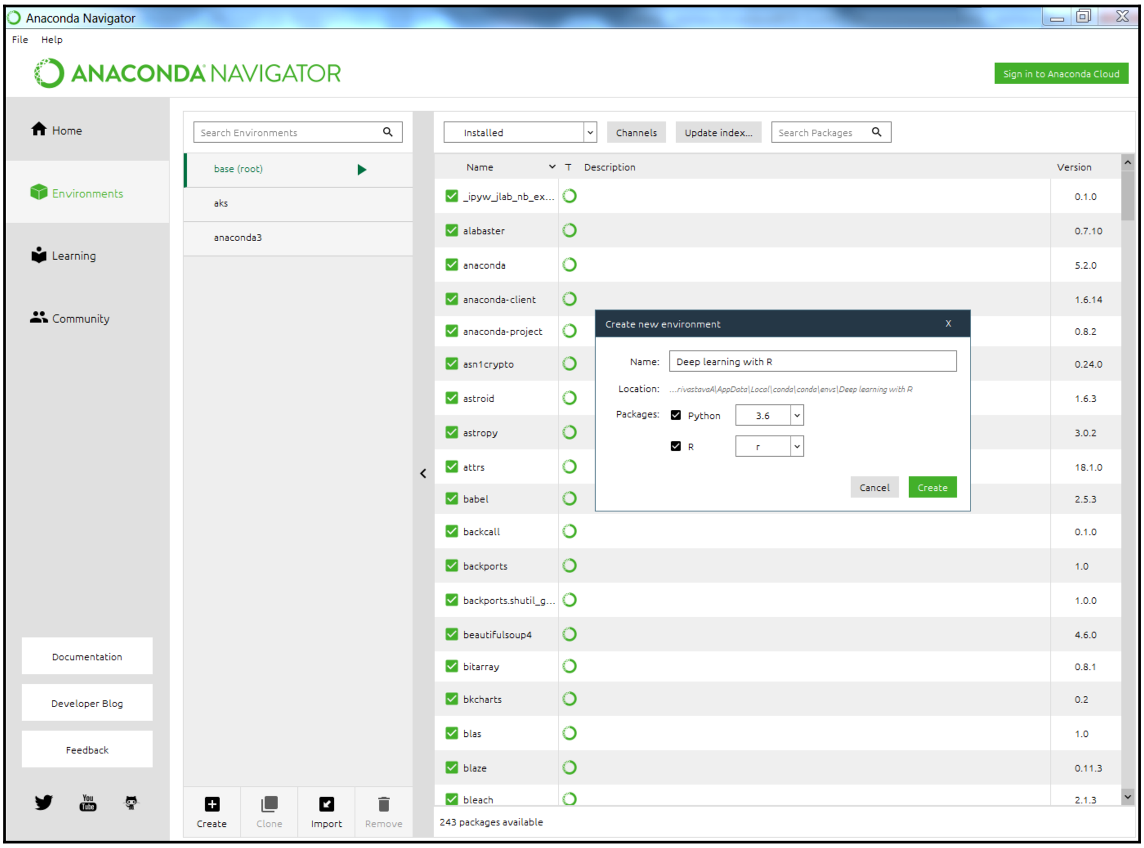
You also need to install the CUDA and cuDNN libraries for GPU support. You can read more about the prerequisites at `https://tensorflow.rstudio.com/tools/local_gpu.html#prerequisties`.

Please note that if your system does not have NVIDIA graphics support, then GPU processing cannot be done.

How to do it...

Let's create an environment in Anaconda (ensure that you have R and Python installed):

1. Go to **Anaconda Navigator** from the Start menu.
2. Click on **Environments.**
3. Create a new environment and name it. Make sure that both the Python and R options are selected, as shown in the following screenshot:



4. Install the `keras` library in R using the following command in RStudio or by using the Terminal of the conda environment created in the previous step:

```
install.packages("keras")
```

5. Install `keras` with the `tensorflow` backend.

The `keras` library supports TensorFlow as the default backend. Theano and CNTK are other alternative backends that can be used instead of TensorFlow.

To install the CPU version, please refer to the following code:

```
install_keras(method = c("auto", "virtualenv", "conda"),  conda =
"auto", version = "default", tensorflow = "default",
extra_packages = c("tensorflow-hub"))
```

For more details about this function, please go to `https://keras.rstudio.com/reference/install_keras.html`.

To install the GPU version, please refer to the following steps:

1. Ensure that you have met all the installation prerequisites, including installing the CUDA and cuDNN libraries.
2. Set the `tensorflow` argument's value to `gpu` in the `install_keras()` function:

```
install_keras(tensorflow = "gpu")
```

The preceding command will install the GPU version of `keras` in R.

How it works...

Keras and TensorFlow programs can be executed on both CPUs and GPUs, though these programs usually run faster on GPUs. If your system does not support an NVIDIA GPU, you only need to install the CPU version. However, if your system has an NVIDIA GPU that meets all the prerequisites and you need to run performance-critical applications, you should install the GPU version. To run the GPU version of TensorFlow, we need an NVIDIA GPU, and then we need to install a variety of software components (CUDA Toolkit v9.0, NVIDIA drivers, and cuDNN v7.0) on the system.

In *steps 1* to *3*, we created a new `conda` environment with both the R and Python kernels installed. In *steps 4* and *5*, we installed the `keras` library in the environment we created.

There's more...

The only supported installation method on Windows is `conda`. Therefore, you should install Anaconda 3.x for Windows before installing `keras`. The `keras` package uses the TensorFlow backend by default. If you want to switch to Theano or CNTK, call the `use_backend()` function after loading the `keras` library.

For the Theano backend, use the following command:

```
library(keras)
use_backend("theano")
```

For the CNTK backend, use the following command:

```
library(keras)
use_backend("cntk")
```

Now, your system is ready to train deep learning models.

See also

You can find out more about the GPU version installation of `keras` and its prerequisites here: `https://tensorflow.rstudio.com/tools/local_gpu.html`.

Implementing neural networks with Keras

TensorFlow is an open source software library developed by Google for numerical computation using data flow graphs. The R interface for TensorFlow is developed by RStudio, which provides an interface for three TensorFlow APIs:

- Keras
- Estimator
- Core

The `keras`, `tfestimators`, and `tensorflow` packages provide R interfaces to the aforementioned APIs, respectively. Keras and Estimator are high-level APIs, while Core is a low-level API that offers full access to the core of TensorFlow. In this recipe, we will demonstrate how we can build and train deep learning models using Keras.

Keras is a high-level neural network API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. The R interface for Keras uses TensorFlow as its default backend engine. The `keras` package provides an R interface for the TensorFlow Keras API. It lets you build deep learning models in two ways, sequential and functional, both of which will be described in the following sections.

Sequential API

Keras's Sequential API is straightforward to understand and implement. It lets us create a neural network linearly; that is, we can build a neural network layer-by-layer where we initialize a sequential model and then stack a series of hidden and output layers on it.

Getting ready

Before creating a neural network using the Sequential API, let's load the `keras` library into our environment and generate some dummy data:

```
library(keras)
```

Now, let's simulate some dummy data for this exercise:

```
x_data <- matrix(rnorm(1000*784), nrow = 1000, ncol = 784)
y_data <- matrix(rnorm(1000), nrow = 1000, ncol = 1)
```

We can check the dimension of the x and y data by executing the following commands:

```
dim(x_data)
dim(y_data)
```

The dimension of the `x_data` data is 1,000×784, whereas the dimension of the `y_data` data is 1,000×1.

How to do it...

Now, we can build our first sequential `keras` model and train it:

1. Let's start by defining a sequential model:

```
model_sequential <- keras_model_sequential()
```

2. We need to add layers to the model we defined in the preceding code block:

```
model_sequential %>%
layer_dense(units = 16,batch_size = ,input_shape = c(784)) %>%
layer_activation('relu') %>%
layer_dense(units = 1)
```