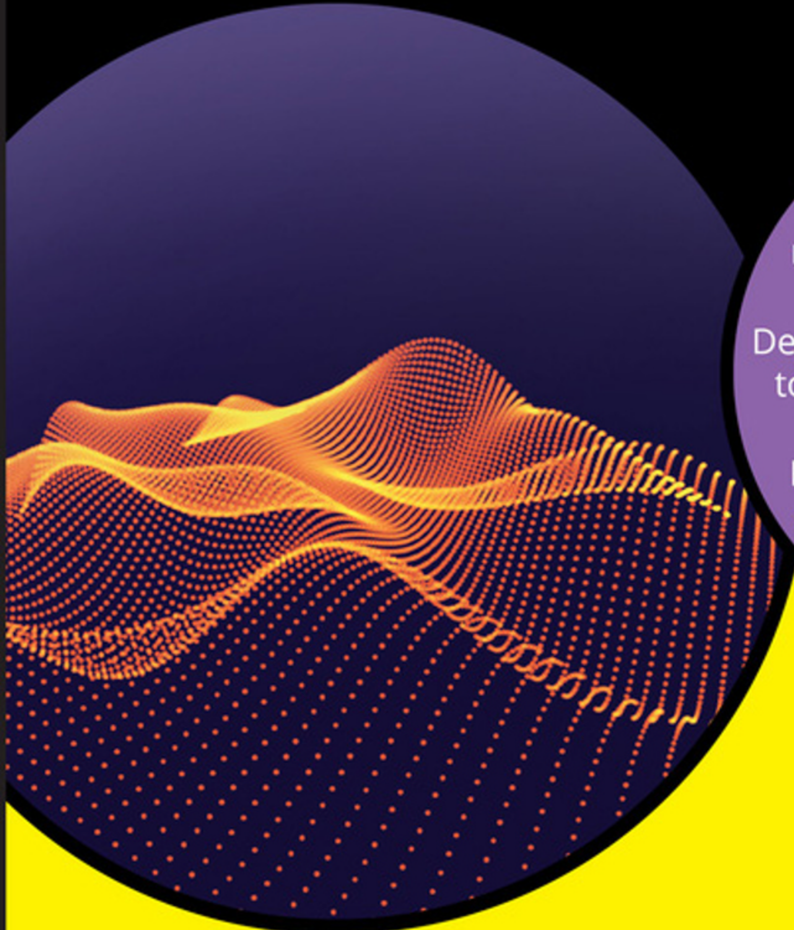
LEARNING MADE EASY

TensorFlow® for dummies®

A Wiley Brand



Explore the underlying machine learning concepts

Deploy TensorFlow applications to the Google Cloud Platform

Learn TensorFlow modules and create a neural network

Matthew Scarpino

TensorFlow®

by Matthew Scarpino

for dummies®
A Wiley Brand

TensorFlow® For Dummies®

Published by: **John Wiley & Sons, Inc.,** 111 River Street, Hoboken, NJ 07030-5774, www.wiley.com

Library of Congress Control Number: 2018933981

ISBN 978-1-119-46621-5 (pbk); ISBN 978-1-119-46619-2 (ePub); 978-1-119-46620-8 (ePDF)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

Contents at a Glance

Table of Contents

Introduction

Machine learning is one of the most fascinating and most important fields in modern technology. As I write this book, NASA has discovered faraway planets by using machine learning to analyze telescope images. After only three days of training, Google's AlphaGo program learned the complex game of Go and defeated the world's foremost master.

Despite the power of machine learning, few programmers know how to take advantage of it. Part of the problem is that writing machine learning applications requires a different mindset than regular programming. The goal isn't to solve a specific problem, but to write a general application capable of solving many unknown problems.

Machine learning draws from many different branches of mathematics, including statistics, calculus, linear algebra, and optimization theory. Unfortunately, the real world doesn't feel any obligation to behave mathematically. Even if you use the best mathematical models, you can still end up with lousy results. I've encountered this frustration on many occasions, and I've referred to neural networks more than once as "high-tech snake oil."

TensorFlow won't give you the ideal model for analyzing a system, but it will reduce the time and frustration involved in machine learning development. Instead of coding activation functions and normalization routines from scratch, you can access the many built-in features of the framework. *TensorFlow For Dummies* explains how to access these features and put them to use.

About This Book

TensorFlow is a difficult subject to write about. Not only does the toolset contain thousands of classes, but many of them perform similar roles. Furthermore, some classes are deprecated, while others are simply "not recommended for use."

Despite the vast number of classes, there are three classes that every TensorFlow developer should be familiar with: `Tensor`, `Graph`, and `Session`. The chapters in the first part of this book discuss these classes in detail and present many examples of their usage.

The chapters in Part 2 explain how you can use TensorFlow in practical machine learning tasks. I start with statistical methods, including linear regression, polynomial regression, and logistic regression. Then I delve into the fascinating topic of neural networks. I explore the operation of basic neural networks, and then I present convolutional neural networks (CNNs) and recurrent neural networks (RNNs).

The chapters in Part 3 present high-level TensorFlow classes that you can use to simplify and accelerate your applications. Of the many topics discussed, the most important is the Estimator API, which allows you to implement powerful machine learning algorithms with minimal code. I explain how to code estimators and execute them at high speed using the Google Cloud Platform (GCP).

Foolish Assumptions

In essence, this book covers two topics: the theory of machine learning and the implementation of the theory using TensorFlow. With regard to theory, I make few assumptions. I expect you to know the basics of linear algebra, but I don't expect you to know anything about machine learning. I also don't expect you to know about statistical regression or neural networks, so I provide a thorough introduction to these and other concepts.

With regard to TensorFlow development, I made assumptions related to your programming background. TensorFlow supports a handful of programming languages, but the central language is Python. For this reason, this book is Python-centric, and I provide all of the example code in Python modules. I explain how to install TensorFlow and access its modules and classes, but I don't explain what modules and classes are.

Icons Used in this Book

To help you navigate through the text, I inserted icons in the book's margin. Here's what they mean:

TIP

This icon indicates that the text contains suggestions for developing machine learning applications.

TECHNICAL STUFF

This icon precedes content that delves into the technical theory of machine learning. Many readers may find this theory helpful, but you don't need to know all the gritty details.

REMEMBER

As much as I love TensorFlow, I admit that it isn't simple to use or understand. There are many critical points to be familiar with, and in many cases, I use this icon to emphasize concepts that are particularly important.

Beyond the Book

This book covers a great deal of the TensorFlow API, but there's still a lot more to learn. The first place to look is the official documentation, which you can find at `www.tensorflow.org`. If you're interested in TensorFlow's functions and data structures, the best place to look is `www.tensorflow.org/api_docs`.

If you have a problem that you can't solve using this book or the official documentation, a great resource is StackOverflow. This site enables programmers to present questions and receive answers, and in my career, I've provided plenty of both. For TensorFlow-specific questions, I recommend visiting `www.stackoverflow.com/questions/tagged/tensorflow`.

In addition to what you're reading right now, this product also comes with a free access-anywhere Cheat Sheet that gives you some pointers on using TensorFlow. To get this Cheat Sheet, simply go to `www.dummies.com` and search for "TensorFlow For Dummies Cheat Sheet" in the Search box.

I also provide a great deal of example code that demonstrates how to put the theory into practice. Here's how to download the `tfbook.zip` file for this book.

1. On `www.dummies.com`, search for *TensorFlow For Dummies* or the book's ISBN.
2. When the book comes up, click on the More about this book link.

   You are taken to the book's product page, and the code should be on the Downloads tab.

After decompressing the archive, you'll find a series of folders named after chapters of this book. The example code for Chapter 3 is in the `ch3` folder, the code for Chapter 6 is in `ch6`, and so on.

Where to Go from Here

The material in this book proceeds from the simple to the complex and from the general to the recondite. If you're already a TensorFlow expert, feel free to skip any chapters you're already familiar with. But if you're new to the toolset, I strongly recommend starting with Chapter 1 and proceeding linearly through Chapters 2, 3, 4, and so on.

I've certainly enjoyed writing this book, and I hope you enjoy the journey of discovery. *Bon voyage!*

1

Getting to Know TensorFlow

IN THIS PART . . .

Explore the fascinating field of machine learning and discover why TensorFlow is so vital to machine learning development.

Download the TensorFlow package to your computer and install the complete toolkit.

Discover the fundamental data types of TensorFlow and the many operations that you can perform on tensors.

Understand how tensors and operations are stored in graphs and how graphs can be executed in sessions.

Investigate the process of TensorFlow training, which minimizes the disparity between a mathematical model and a real-world system.

IN THIS CHAPTER

» Looking at machine learning over time

» Exploring machine learning frameworks

Chapter 1
Introducing Machine Learning with TensorFlow

TensorFlow is Google's powerful framework for developing applications that perform machine learning. Much of this book delves into the gritty details of coding TensorFlow modules, but this chapter provides a gentle introduction. I provide an overview of the subject and then discuss the developments that led to the creation of TensorFlow and similar machine learning frameworks.

Understanding Machine Learning

Like most normal, well-adjusted people, I consider *The Terminator* to be one of the finest films ever made. I first saw it at a birthday party when I was 13, and though most of the story went over my head, one scene affected me deeply: The heroine calls her mother and thinks she's having a warm conversation, but she's really talking to an evil robot from the future!

The robot wasn't programmed in advance with the mother's voice or the right sequence of phrases. It had to figure these things out on its own. That is, it had to analyze the voice of the real mother, examine the rules of English grammar, and generate acceptable sentences for the conversation. When a computer obtains information from data without receiving precise instructions, it's performing *machine learning*.

The Terminator served as my first exposure to machine learning, but it wouldn't be my last. As I write this book, machine learning is everywhere. My email provider knows that messages involving an "online pharmacy" are spam, but messages about "cheap mescaline" are important. Google Maps always provides the best route to my local Elvis cult, and Amazon.com always knows when I need a new horse head mask. Is it magic? No, it's machine learning!

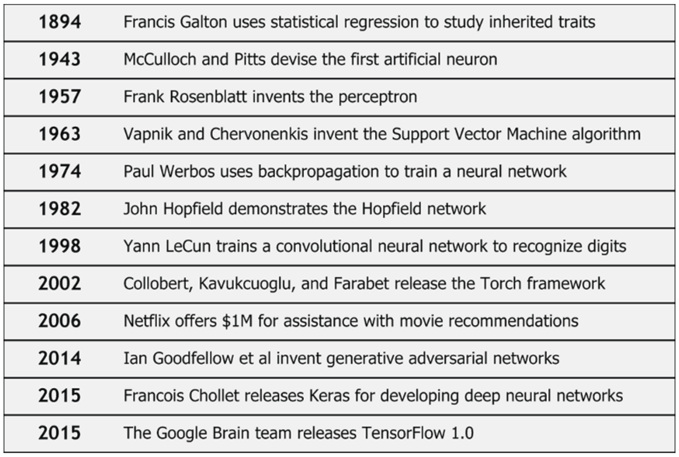
Machine learning applications achieve this power by discovering patterns in vast amounts of data. Unlike regular programs, machine learning applications deal with uncertainties and probabilities. It should come as no surprise that the process of coding a machine learning application is completely different than that of coding a regular application. Developers need to be familiar with an entirely new set of concepts and data structures.

Thankfully, many frameworks have been developed to simplify development. At the time of this writing, the most popular is TensorFlow, an open-source toolset released by Google. In writing this book, my goal is to show you how to harness TensorFlow to develop your own machine learning applications.

Although this book doesn't cover the topic of ethics, I feel compelled to remind readers that programming evil robots is wrong. Yes, you'll impress your professor, and it will look great on a resume. But society frowns on such behavior, and your friends will shun you. Still, if you absolutely have to program an evil robot, TensorFlow is the framework to use.

The Development of Machine Learning

In my opinion, machine learning is the most exciting topic in modern software development, and TensorFlow is the best framework to use. To convince you of TensorFlow's greatness, I'd like to present some of the developments that led to its creation. Figure 1-1 presents an abbreviated timeline of machine learning and related software development.



1894	Francis Galton uses statistical regression to study inherited traits
1943	McCulloch and Pitts devise the first artificial neuron
1957	Frank Rosenblatt invents the perceptron
1963	Vapnik and Chervonenkis invent the Support Vector Machine algorithm
1974	Paul Werbos uses backpropagation to train a neural network
1982	John Hopfield demonstrates the Hopfield network
1998	Yann LeCun trains a convolutional neural network to recognize digits
2002	Collobert, Kavukcuoglu, and Farabet release the Torch framework
2006	Netflix offers $1M for assistance with movie recommendations
2014	Ian Goodfellow et al invent generative adversarial networks
2015	Francois Chollet releases Keras for developing deep neural networks
2015	The Google Brain team releases TensorFlow 1.0

FIGURE 1-1: Developments in machine learning extend from academia to corporations.

Once you understand why researchers and corporations have spent so much time developing the technology, you'll better appreciate why studying TensorFlow is worth your own time.

Statistical regression

Just as petroleum companies drill into the ground to obtain oil, machine learning applications analyze data to obtain information and insight. The formal term for this process is *statistical inference*, and its first historical record comes from ancient Greece. But for this purpose, the story begins with a nineteenth-century scientist named Francis Galton. Though his primary interest was anthropology, he devised many of the concepts and tools used by modern statisticians and machine learning applications.

Galton was obsessed with inherited traits, and while studying dogs, he noticed that the offspring of exceptional dogs tend to acquire average characteristics over time. He referred to this as the *regression to mediocrity.* Galton observed this phenomenon in humans and sweet peas, and while analyzing his data, he employed modern statistical concepts like the normal curve, correlation, variance, and standard deviation.

To illustrate the relationship between a child's height and the average height of the parents, Galton developed a method for determining which line best fits a series of data points. Figure 1-2 shows what this looks like. (Galton's data is provided by the University of Alabama.)

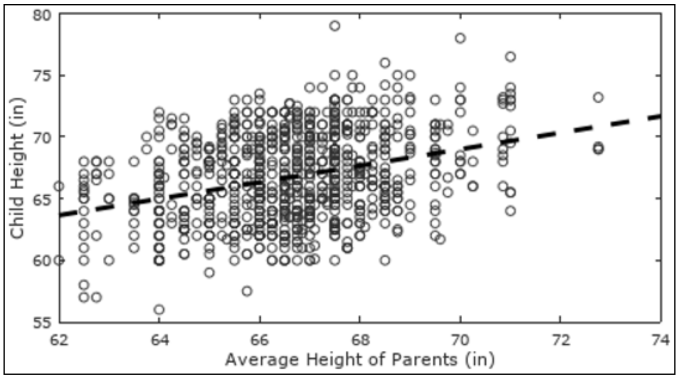


FIGURE 1-2: Linear regression identifies a clear trend amidst unclear data points.

Galton's technique for fitting lines to data became known as *linear regression*, and the term *regression* has come to be used for a variety of statistical methods. Regression plays a critical role in machine learning, and Chapter 6 discusses the topic in detail.

Reverse engineering the brain

In 1905, Ramón y Cajal examined tissue from a chicken's brain and studied the interconnections between the cells, later called *neurons.* Cajal's findings fascinated scientists throughout the world, and in 1943, Warren McCulloch and Walter Pitts devised a mathematical model for the neuron. They demonstrated that their artificial neurons could implement the common Boolean `AND` and `OR` operations.

While researching statistics, a psychologist named Frank Rosenblatt developed another model for a neuron that expanded on the work of McCulloch and Pitts. He called his model the *perceptron,* and by connecting perceptrons into layers, he created a circuit capable of recognizing images. These interconnections of perceptrons became known as *neural networks.*

Rosenblatt followed his demonstrations with grand predictions about the future of perceptron computing. His predictions deeply influenced the Office of Naval Research, which funded the development of a custom computer based on perceptrons. This computer was called the Mark 1 Perceptron, and Figure 1-3 shows what it looks like.

The future of perceptron-based computing seemed bright, but in 1969, calamity struck. Marvin Minsky and Seymour Papert presented a deeply critical view of Rosenblatt's technology in their book, *Perceptrons* (MIT Press). They mathematically proved many limitations of two-layer feed-forward neural networks, such as the inability to learn nonlinear functions or implement the Boolean Exclusive OR (XOR) operation.

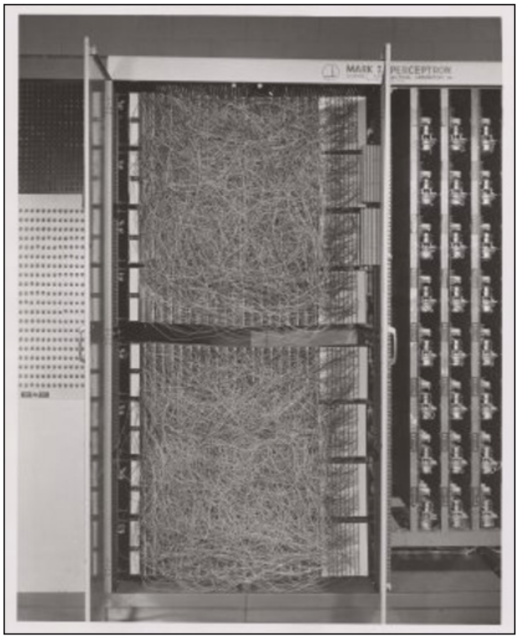


FIGURE 1-3: The Mark 1 Perceptron was the first computer created for machine learning.

Credit: Cornell Aeronautical Laboratory.

Neural networks have progressed dramatically since the 1960s, and in hindsight, modern readers can see how narrow-minded Minsky and Papert were in their research. But at the time, their findings caused many, including the Navy and other large organizations, to lose interest in neural networks.

Steady progress

Despite the loss of popular acclaim, researchers and academics continued to investigate machine learning. Their work led to many crucial developments, including the following:

- In 1965, Ivakhnenko and Lapa demonstrated multilayer perceptrons with nonlinear activation functions.
- In 1974, Paul Werbos used backpropagation to train a neural network.
- In 1980, Kunihiko Fukushima proposed the neocognitron, a multilayer neural network for image recognition.
- In 1982, John Hopfield developed a type of recurrent neural network known as the Hopfield network.
- In 1986, Sejnowski and Rosenberg developed NETtalk, a neural network that learned how to pronounce words.

These developments expanded the breadth and capabilities of machine learning, but none of them excited the world's imagination. The problem was that computers lacked the speed and memory needed to perform real-world machine learning in a reasonable amount of time. That was about to change.

The computing revolution

As the 1980s progressed into the 1990s, improved semiconductor designs led to dramatic leaps in computing power. Researchers harnessed this new power to execute machine learning routines. Finally, machine learning could tackle real-world problems instead of simple proofs of concept.

As the Cold War intensified, military experts grew interested in recognizing targets automatically. Inspired by Fukushima's neocognitron, researchers focused on neural networks specially designed for image recognition, called *convolutional neural networks* (CNNs). One major step forward took place in 1994, when Yann LeCun successfully demonstrated handwriting recognition with his CNN-based LeNet5 architecture.

But there was a problem. Researchers used similar theories in their applications, but they wrote all their code from scratch. This meant researchers couldn't reproduce the results of their peers, and they couldn't re-use one another's code. If a researcher's funding ran out, it was likely that the entire codebase would vanish.

In the late 1990s, my job involved programming convolutional neural networks to recognize faces. I loved the theory behind neural networks, but I found them deeply frustrating in practice. Machine learning applications require careful tuning and tweaking to get acceptable results. But each change to the code required a new training run, and training a CNN could take *days*. Even then, I still didn't have enough training data to ensure accurate recognition.

One problem facing me and other researchers was that, while machine learning theory was mature, the process of software development was still in its infancy. Programmers needed frameworks and standard libraries so that they weren't coding everything by themselves. Also, despite Intel's best efforts, practical machine learning still required faster processors that could access larger amounts of data.

The rise of big data and deep learning

As the 21st century dawned, the Internet's popularity skyrocketed, and the price of data storage plummeted. Large corporations could now access terabytes of data about potential consumers. These corporations developed improved tools for analyzing their data, and this revolution in data storage and analysis has become known as the *big data revolution*.

Now CEOs were faced with a difficult question: How could they use their wealth of data to create wealth for their corporations? One major priority was advertising — companies make more money if they know which advertisements to show to their customers. But there were no clear rules for associating customers with products.

Many corporations launched in-house research initiatives to determine how best to analyze their data. But in 2006, Netflix tried something different. They released a large part of their database online and offered one million dollars to whoever developed the best recommendation engine. The winner, BellKor's Pragmatic Chaos, combined a number of machine learning algorithms to improve Netflix's algorithm by 10 percent.

Netflix wasn't the only high-profile corporation using machine learning. Google's AdSense used machine learning to determine which advertisements to display on its search engine. Google and Tesla demonstrated self-driving cars that used machine learning to follow roads and join traffic.

Across the world, large organizations sat up and paid notice. Machine learning had left the realm of wooly-headed science fiction and had become a practical business tool. Entrepreneurs continue to wonder what other benefits can be gained by applying machine learning to big data.

Researchers paid notice as well. A major priority involved distinguishing modern machine learning, with its high complexity and vast data processing, from earlier machine learning, which was simple and rarely effective. They agreed on the term *deep learning* for this new machine learning paradigm. Chapter 7 goes into greater detail regarding the technical meaning of deep learning.

Machine Learning Frameworks

One of the most important advances in practical machine learning involved the creation of frameworks. *Frameworks* automate many aspects of developing machine learning applications, and they allow developers to re-use code and take advantage of best practices. This discussion introduces five of the most popular frameworks: Torch, Theano, Caffe, Keras, and TensorFlow.

Torch

Torch is the first machine learning framework to attract a significant following. Originally released in 2002 by Ronan Collobert, it began as a toolset for numeric computing. Torch's computations involve multidimensional arrays called *tensors*, which can be processed with regular vector/matrix operations. Over time, Torch acquired routines for building, training, and evaluating neural networks.

Torch garnered a great deal of interest from academics and corporations like IBM and Facebook. But its adoption has been limited by its reliance on Lua as its interface language. The other frameworks in this discussion —Theano, Caffe, Keras, and TensorFlow — can be interfaced through Python, which has emerged as the language of choice in the machine learning domain.

Theano

In 2010, a machine learning group at the University of Montreal released Theano, a library for numeric computation. Like NumPy, Theano provides a wide range of Python routines for operating on multidimensional arrays. Unlike NumPy, Theano stores operations in a data structure called a *graph*, which it compiles into high-performance code. Theano also supports *symbolic differentiation*, which makes it possible to find derivatives of functions automatically.

Because of its high performance and symbolic differentiation, many machine learning developers have adopted Theano as their numeric computation toolset of choice. Developers particularly appreciate Theano's ability to execute graphs on graphics processing units (GPUs) as well as central processing units (CPUs).

Caffe

As part of his PhD dissertation at UC Berkeley, Yangqing Jia created Caffe, a framework for developing image recognition applications. As others joined in the development, Caffe expanded to support other machine learning algorithms and many different types of neural networks.

Caffe is written in C++, and like Theano, it supports GPU acceleration. This emphasis on performance has endeared Caffe to many academic and corporate developers. Facebook has become particularly interested in Caffe, and in 2007 it released a reworked version called Caffe2. This version improves Caffe's performance and makes executing applications on smartphones possible.

Keras

While other offerings focus on performance and breadth of capabilities, Keras is concerned with modularity and simplicity of development. François Chollet created Keras as an interface to other machine learning frameworks, and many developers access Theano through Keras to combine Keras's simplicity with Theano's performance.

Keras's simplicity stems from its small API and intuitive set of functions. These functions focus on accomplishing standard tasks in machine learning, which makes Keras ideal for newcomers to the field but of limited value for those who want to customize their operations.

François Chollet released Keras under the MIT License, and Google has incorporated his interface into TensorFlow. For this reason, many TensorFlow developers prefer to code their neural networks using Keras.

TensorFlow

As the title implies, this book centers on TensorFlow, Google's gift to the world of machine learning. The Google Brain team released TensorFlow 1.0 in 2015, and as of the time of this writing, the current version is 1.4. It's provided under the Apache 2.0 open source license, which means you're free to use it, modify it, and distribute your modifications.

TensorFlow's primary interface is Python, but like Caffe, its core functionality is written in C++ for improved performance. Like Theano, TensorFlow stores operations in a graph that can be deployed to a GPU, a remote system, or a network of remote systems. In addition, TensorFlow provides a utility called TensorBoard, which makes visualizing graphs and their operations possible.

Like other frameworks, TensorFlow supports execution on CPUs and GPUs. In addition, TensorFlow applications can be executed on the Google Cloud Platform (GCP). The GCP provides world-class processing power at relatively low cost, and in my opinion, GCP processing is TensorFlow's most important advantage. Chapter 13 discusses this important topic in detail.