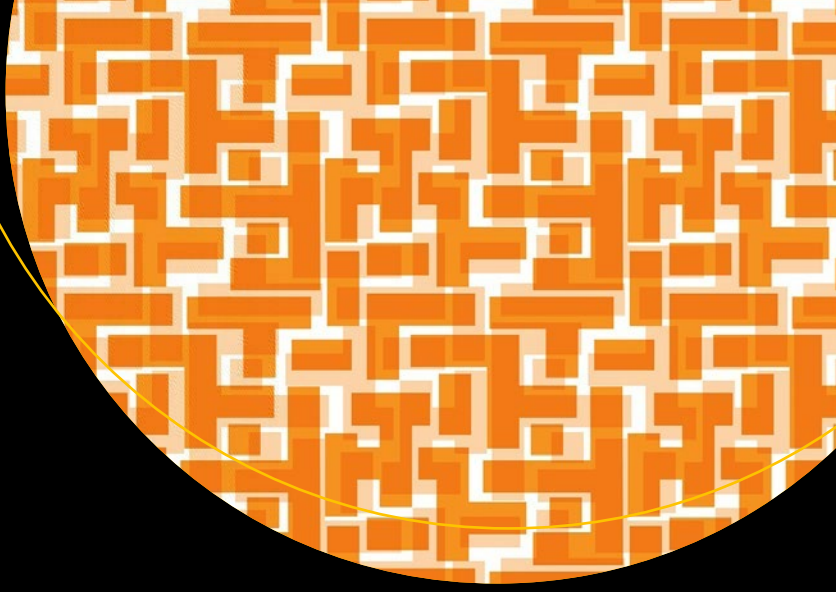

Applied Neural Networks with TensorFlow 2

API Oriented Deep Learning with Python

Orhan Gazi Yalçın

Apress®

Applied Neural Networks with TensorFlow 2

API Oriented Deep Learning with Python

Orhan Gazi Yalçın

Apress®

Applied Neural Networks with TensorFlow 2: API Oriented Deep Learning with Python

Orhan Gazi Yalçın
Istanbul, Turkey

ISBN-13 (pbk): 978-1-4842-6512-3 ISBN-13 (electronic): 978-1-4842-6513-0
https://doi.org/10.1007/978-1-4842-6513-0

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Aaron Black
Development Editor: James Markham
Coordinating Editor: Jessica Vakili

Distributed to the book trade worldwide by Springer Science+Business Media New York,1 NY Plazar, New York, NY 10014. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/978-1-4842-6512-3. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

I dedicate this book to my overcurious dad, Lutfi – who kept sneaking into the study room to see how far I was into the book – and to my mom, Ayşe, for always supporting and encouraging me.

I would also like to thank my friend, Enes, for encouraging me to write this book in the first place.

Finally, I would like to thank my sister and brother, Merve and Kürşat, and all my friends who supported me throughout the whole process – all the way – till the last word.

Table of Contents

About the Author

Orhan Gazi Yalçın is a joint PhD candidate at the University of Bologna and the Polytechnic University of Madrid. After completing his double major in business and law, he began his career in Istanbul, working for a city law firm, Allen & Overy, and a global entrepreneurship network, Endeavor. During his academic and professional career, he taught himself programming and excelled in machine learning. He currently conducts research on hotly debated law and AI topics such as explainable artificial intelligence and the right to explanation by combining his technical and legal skills. In his spare time, he enjoys free diving, swimming, exercising, as well as discovering new countries, cultures, and cuisines.

- You can visit Orhan's personal web page at

  `www.orhangaziyalcin.com`

- Also feel free to connect with Orhan on Linkedin at

  `www.linkedin.com/in/orhangaziyalcin`

About the Technical Reviewer

Vishwesh Ravi Shrimali graduated from BITS Pilani in 2018, where he studied mechanical engineering. Since then, he has worked with BigVision LLC on deep learning and computer vision and was involved in creating official OpenCV AI courses. Currently, he is working at Mercedes Benz Research and Development India Pvt. Ltd. He has a keen interest in programming and AI and has applied that interest in mechanical engineering projects. He has also written multiple blogs on OpenCV and deep learning on LearnOpenCV, a leading blog on computer vision. He has also coauthored *Machine Learning for OpenCV4* (second edition) by Packt. When he is not writing blogs or working on projects, he likes to go on long walks or play his acoustic guitar.

Acknowledgments

This book was written during a global lockdown due to the Covid-19 pandemic, which created a new normal that I have never experienced before. Writing a book in the middle of a global crisis was a very intense experience, and I was uncertain about taking this responsibility for a long time. Thanks to my family and friends, I was able to complete the book even earlier than scheduled. Now I am glad that I accepted Aaron's invitation, who guided me throughout the whole process. Thank you very much for reaching out to me in the first place and making it possible to have this book written.

I would like to thank Jessica Vakili for coordinating the entire project and for being there whenever I needed. I would also like to thank Vishwesh Ravi Shrimali for reviewing every single line of the book and providing me with all the valuable comments, which helped to improve the quality of the book tremendously.

Being surrounded with people who all have a positive attitude made this experience very fruitful, and I am looking forward to working with them in the future. Thank you all very much!

CHAPTER 1

Introduction

In this book, we dive into the realms of deep learning (DL) and cover several deep learning concepts along with several case studies. These case studies range from image recognition to recommender systems, from art generation to object clustering. Deep learning is part of a broader family of machine learning (ML) methods based on **artificial neural networks (ANNs)** with representation learning. These neural networks mimic the human brain cells, or neurons, for algorithmic learning, and their learning speed is much faster than human learning speed. Several deep learning methods offer solutions to different types of machine learning problems: (i) supervised learning, (ii) unsupervised learning, (iii) semi-supervised learning, and (iv) reinforcement learning.

This book is structured in a way to also include an introduction to the discipline of machine learning so that the reader may be acquainted with the general rules and concepts of machine learning. Then, a detailed introduction to deep learning is provided to familiarize the reader with the sub-discipline of deep learning.

After covering the fundamentals of deep learning, the book covers different types of artificial neural networks with their potential real-life applications (i.e., case studies). Therefore, at each chapter, this book (i) introduces the concept of a particular neural network architecture with details on its components and then (ii) provides a tutorial on how to apply this network structure to solve a particular artificial intelligence (AI) problem.

O. G. Yalçın, *Applied Neural Networks with TensorFlow 2*,
https://doi.org/10.1007/978-1-4842-6513-0_1

Since the goal of this book is to provide case studies for deep learning applications, the competency in several technologies and libraries is sought for a satisfactory learning experience.

Before diving into machine learning and deep learning, we start with the introduction to the technologies used in this book. This introduction includes the latest developments and the reasoning as to why these technologies are selected. Finally, this chapter also covers how to install these technologies and prepare your environment with a minimum amount of hassle. The technologies that are in the center of this book are as follows:

- Our Selected Programming Language: **Python 3.x**
- Our Selected Deep Learning Framework: **TensorFlow 2.x**
- Our Development Environment: **Google Colab** (*with Jupyter Notebook alternative*)

Note A TensorFlow Pipeline Guide showing how to use TensorFlow can be found in Chapter 5, whereas the relevant libraries used with TensorFlow are covered in Chapter 4.

Please note that this book assumes that you use Google Colab, which requires almost no environment setup. The chapter also includes a local Jupyter Notebook installation guide if you prefer a local environment. You may skip the Jupyter Notebook installation section if you decide to use Google Colab.

Note When learning a new programming discipline or technology, one of the most demoralizing tasks is the environment setup process. Therefore, it is important to simplify this process as much as possible. Therefore, this chapter is designed with this principle in mind.

Python as Programming Language

Python is a programming language created by Guido van Rossum as a side project and was initially released in 1991. Python supports object-oriented programming (OOP), a paradigm based on the concept of objects, which can contain data, in the form of fields. Python prioritizes the programmer's experience. Therefore, programmers can write clear and logical code for both small and large projects. It also contains support for functional programming. Python is dynamically typed and garbage collected.

Python is also considered as an interpreted language because it goes through an interpreter, which turns code you write into the language understood by your computer's processor. An interpreter executes the statements of code "one by one." On the other hand, in compiled languages, a compiler executes the code entirely and lists all possible errors at a time. The compiled code is more efficient than the interpreted code in terms of speed and performance. However, scripted languages such as Python show only one error message even though your code has multiple errors. This feature helps the programmer to clear errors quickly, and it increases the development speed.

Timeline of Python

Let's take a look at the timeline of Python:

- In the late 1980s, Python was conceived as a successor to the ABC language.
- In December 1989, Guido van Rossum started Python's implementation.
- In January 1994, Python version 1.0 was released. The major new features included were the functional programming tools lambda, map, filter, and reduce.

- October 2000, Python 2.0 was released with major new features, including a cycle-detecting garbage collector and support for Unicode.
- Python 3.0 was released on December 3, 2008. It was a major revision of the language that is only partially backward compatible. Many of its major features were backported to Python 2.6.x and 2.7.x version series. Releases of Python 3 include the 2 to 3 utility, which automates (at least partially) the translation of Python 2 code to Python 3.
- As of January 1, 2020, no new bug reports, fixes, or changes are made to Python 2, and **Python 2 is no longer supported**.

Python 2 vs. Python 3

One of the common questions a new deep learning programmer might have is whether to use Python 2.x or Python 3.x since there are many outdated blog posts and web articles comparing two major versions. As of 2020, it is safe to claim that these comparisons are not relevant. As you may see in the preceding timeline, the delayed deprecation of Python 2.x finally took place as of January 1, 2020. Therefore, programmers may not find official support for Python 2.x versions anymore.

One of the essential skills for a programmer is to be up to date with the latest technology, and therefore, this book only utilizes the use of Python 3.x versions. For the readers who are only familiar with Python 2.x versions, this preference should not pose a problem since the differences between the syntax used in this book for Python 2.x and Python 3.x are not significant. Therefore, Python 2.x programmers may immediately familiarize themselves with the source code in this book.

Why Python?

Compared to other programming languages, there are several reasons for Python's popularity among data scientists and machine learning engineers. 2019 Kaggle Machine Learning and Data Science Survey revealed that Python is by far the most popular programming language for data science and machine learning; see Figure 1-1.

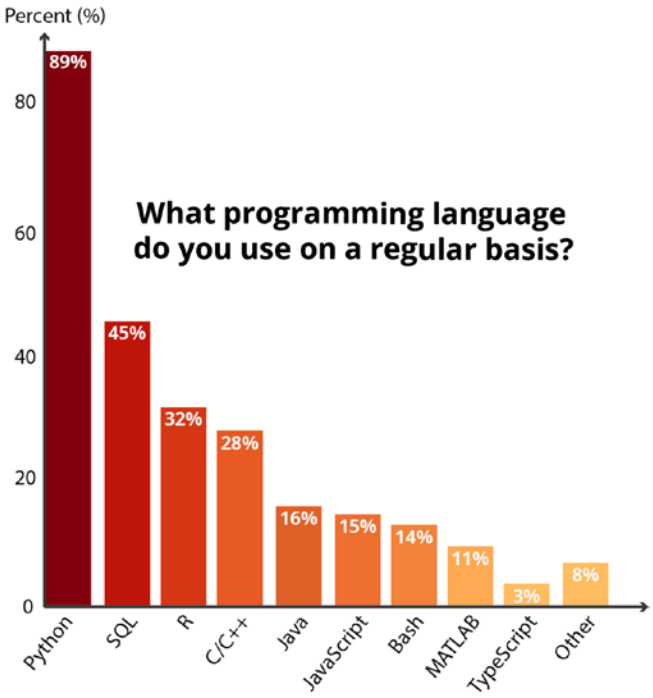

Figure 1-1. *2019 Kaggle Machine Learning and Data Science Survey*

There are several reasons for Python's popularity compared to other languages. A non-exhaustive list of benefits of Python may be the following.

Ease of Learning

One of the main reasons for newcomers to choose Python as their primary programming language is its ease of learning. When compared to other programming languages, Python offers a shorter learning curve so that programmers can achieve a good level of competency in a short amount of time. Python's syntax is easier to learn, and the code is more readable compared to other popular programming languages. A common example to show this is the amount of code required by different programming languages to print out "Hello, World!". For instance, to be able to print out Hello, World! in Java, you need the following code:

Hello, World! In Java

```
public class Main {
   public static void main(String[] args) {
       System.out.println("Hello, World!");
   }
}
```

The same result may be achieved with a single line of code in Python:

```
Hello, World! in Python
print("Hello, World!")
```

A Variety of Available Data Science Libraries

Another powerful characteristic of Python compared to other programming languages is its wide variety of data science libraries. The data science libraries such as Pandas, NumPy, SciPy, and scikit-learn reduce the time to prepare the data for model training with their standardized functions and modules for logical and mathematical operations. Furthermore, thanks to the vibrant community of Python developers, as soon as the developers detect a common problem, a new library is immediately designed and released to address this problem.

Community Support

The powerful community support is another advantage of Python over other programming languages. More and more volunteers are releasing Python libraries, and this practice made Python the language with modern and powerful libraries. Besides, a high number of seasoned Python programmers are always ready to help other programmers with their problems on online community channels such as Stack Overflow.

Visualization Options

Data visualization is an important discipline to extract insights from raw data, and Python offers several useful visualization options. The good old Matplotlib is always there with the most customizable options. In addition, Seaborn and Pandas Plot API are powerful libraries that streamline the most common visualization tasks used by data scientists. Additionally, libraries like Plotly and Dash allow users to create interactive plots and sophisticated dashboards to be served on the Web. With these libraries, data scientists may easily create charts, draw graphical plots, and facilitate feature extraction.

Now that we covered why favorite language of data scientists is Python, we can move on to why we use TensorFlow as our machine learning framework.

TensorFlow As Deep Learning Framework

TensorFlow is an open source machine learning platform with a particular focus on neural networks, developed by the Google Brain team. Despite initially being used for internal purposes,

Google released the library under the Apache License 2.0 in November 2015, which made it an open source library.[1] Although the use cases of TensorFlow are not limited to machine learning applications, machine learning is the field where we see TensorFlow's strength.

The two programming languages with stable and official TensorFlow APIs are Python and C. Also, C++, Java, JavaScript, Go, and Swift are other programming languages where developers may find limited-to-extensive TensorFlow compatibility. Finally, there are third-party TensorFlow APIs for C#, Haskell, Julia, MATLAB, R, Scala, Rust, OCaml, and Crystal.

Timeline of TensorFlow

Although this book focuses on TensorFlow 2.x with Python API, there are several complementary TensorFlow libraries released by Google. Understanding the development of the TensorFlow platform is essential to see the full picture. The timeline of the milestones achieved by Google as part of the TensorFlow project may be summarized as follows:

- In 2011, Google Brain built a machine learning system called **DistBelief** using deep learning neural networks.
- November 2015, Google released the TensorFlow library under the Apache License 2.0 and made it **open source** to accelerate the advancements in artificial intelligence.

[1] Google Just Open Sourced TensorFlow, Its Artificial Intelligence Engine | WIRED, www.wired.com/2015/11/google-open-sources-its-artificial-intelligence-engine/ (last visited Jun 5, 2020)

- In May 2016, Google announced an application-specific integrated circuit (an ASIC) built for machine learning and tailored for TensorFlow, called **Tensor Processing Unit (TPU)**.
- In February 2017, Google released **TensorFlow 1.0.0.**
- In May 2017, Google announced **TensorFlow Lite,** a library for machine learning development in mobile devices.
- In December 2017, Google introduced **Kubeflow**, which allows operation and deployment of TensorFlow on Kubernetes.
- In March 2018, Google announced **TensorFlow.js** version 1.0 for machine learning with JavaScript.
- In July 2018, Google announced the **Edge TPU**. Edge TPU is Google's purpose-built ASIC chip designed to run TensorFlow Lite machine learning (ML) models on smartphones.
- In January 2019, Google announced **TensorFlow 2.0** to be officially available in September 2019.
- In May 2019, Google announced **TensorFlow Graphics** for deep learning in computer graphics.
- In September 2019, TensorFlow Team released **TensorFlow 2.0**, a new major version of the library.

This timeline shows that the TensorFlow platform is maturing. Especially with the release of TensorFlow 2.0, Google has improved the user-friendliness of TensorFlow APIs significantly. Besides, the TensorFlow team announced that they don't intend to introduce any other significant changes. Therefore, it is safe to assume that the methods and syntax included in this book are to keep their relevance for a long time.

Why TensorFlow?

There are more than two dozens of deep learning libraries developed by tech giants, tech foundations, and academic institutions that are available to the public. While each framework has its advantage in a particular sub-discipline of deep learning, this book focuses on TensorFlow with Keras API. The main reason for choosing TensorFlow over other deep learning frameworks is its popularity. On the other hand, this statement does not indicate that the other frameworks are better – yet, less popular – than TensorFlow. Especially with the introduction of version 2.0, TensorFlow strengthened its power by addressing the issues raised by the deep learning community. Today, TensorFlow may be seen as the most popular deep learning framework, which is very powerful and easy to use and has excellent community support.

What's New in TensorFlow 2.x

Since its introduction in 2015, TensorFlow has grown into one of the most advanced machine learning platforms in the market. Researchers, developers, and companies widely adopted the technologies introduced by the TensorFlow team. Around its 4th birthday, TensorFlow 2.0 was released in September 2019. The TensorFlow team put a lot of effort into simplifying the APIs by cleaning up deprecated APIs and reducing duplication. The TensorFlow team introduced several updates to achieve simplicity and ease of use in TensorFlow 2.0. These updates may be listed as follows:

1. Easy model building with Keras and eager execution
2. Robust model deployment in production level on any platform
3. Robust experimentation for research
4. Simplified API thanks to cleanups and duplication reduction

Easy Model Building with Keras and Eager Execution

The TensorFlow team further streamlined the model building experience to respond to expectations with the new or improved modules such as `tf.data`, `tf.keras`, and `tf.estimators` and the `Distribution Strategy API`.

Load Your Data Using tf.data

In TensorFlow 2.0, training data is read using input pipelines created with the `tf.data` module. `tf.feature_column` module is used to define feature characteristics. What is useful for newcomers is the new DataSets module. TensorFlow 2.0 offers a separate DataSets module which offers a range of popular datasets and allows developers to experiment with these datasets.

Build, Train, and Validate Your Model with tf.keras, or Use Premade Estimators

In TensorFlow 1.x, developers could use the previous versions of `tf.contrib`, `tf.layers`, `tf.keras`, and `tf.estimators` to build models. Offering four different options to the same problem confused the newcomers and drove some of them away, especially to PyTorch. TensorFlow 2.0 simplified the model building by limiting the options to two improved modules: `tf.keras` (TensorFlow Keras API) and `tf.estimators` (Estimator API). TensorFlow Keras API offers a high-level interface that makes model building easy, which is especially useful for *proof of concepts (POC)*. On the other hand, Estimator API is better suited for production-level models that require scaled serving and increased customization capability.

Run and Debug with Eager Execution, Then Use AutoGraph API for the Benefits of Graphs

TensorFlow 1.x versions were prioritizing TensorFlow graphs, which is not friendly to newcomers. Even though this complicated methodology was kept in TensorFlow 2.0, eager execution – the contrast concept – was made default. Google explained the initial reasoning for this change with the following statement:

> *Eager execution is an imperative, define-by-run interface where operations are executed immediately as they are called from Python. This makes it easier to get started with TensorFlow, and can make research and development more intuitive.*[2]

Eager execution makes the model building easier. It offers fast debugging capability with immediate runtime errors and integration with Python tools, which makes TensorFlow more beginner friendly. On the other hand, graph execution has advantages for distributed training, performance optimizations, and production deployment. To fill this gap, TensorFlow introduced AutoGraph API called via `tf.function` decorator. This book prioritizes eager execution over graph execution to achieve a steep learning curve for the reader.

Use Distribution Strategies for Distributed Training

Model training with large datasets necessitates distributed training with multiple processors such as CPU, GPU, or TPU. Even though TensorFlow 1.x has support for distributed training, Distribution Strategy API optimizes and streamlines the distributed training across multiple GPUs, multiple

[2] Google AI Blog: Eager Execution: An imperative, define-by-run interface to TensorFlow, https://ai.googleblog.com/2017/10/eager-execution-imperative-define-by.html (last visited Jun 8, 2020)

machines, or TPUs. TensorFlow also provides templates to deploy training on Kubernetes clusters in on-prem or cloud environments, which makes the training more cost-effective.

Export to SavedModel

After training a model, developers may export to SavedModel. `tf.saved_model` API may be used to build a complete TensorFlow program with weights and computations. This standardized SavedModel can be used interchangeably across different TensorFlow deployment libraries such as (i) TensorFlow Serving, (ii) TensorFlow Lite, (iii) TensorFlow.js, and (iv) TensorFlow Hub.

Robust Model Deployment in Production on Any Platform

TensorFlow has always made efforts to provide a direct path to production on different devices. There are already several libraries which may be used to serve the trained models on dedicated environments.

TensorFlow Serving

TensorFlow Serving is a flexible and high-performance TensorFlow library that allows models to be served over HTTP/REST or gRPC/Protocol Buffers. This platform is platform and language-neutral as you may make an HTTP call using any programming language.

TensorFlow Lite

TensorFlow Lite is a lightweight deep learning framework to deploy models to mobile devices (iOS and Android) or embedded devices (Raspberry Pi or Edge TPUs). Developers may pick a trained model, convert the model into a compressed fat buffer, and deploy to a mobile or embedded device with TensorFlow Lite.

TensorFlow.js

TensorFlow.js enables developers to deploy their models to web browsers or Node.js environments. Developers can also build and train models in JavaScript in the browser using a Keras-like API.

With TensorFlow 2.0, the capability and parity across platforms and components are greatly improved with standardized exchange formats and aligning APIs. The new simplified architecture of TensorFlow 2.0 is shown by the TensorFlow team in Figure 1-2.

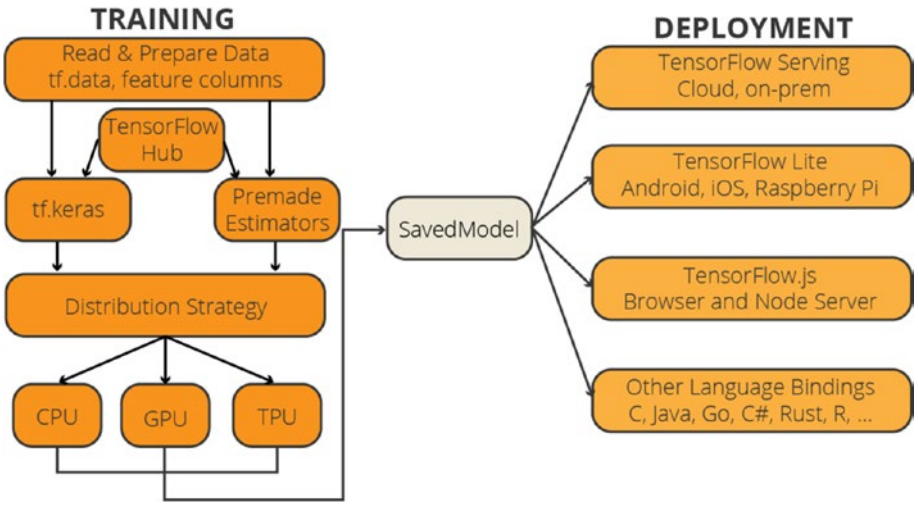

Figure 1-2. *A Simplified Diagram for the TensorFlow 2.0 Architecture*[3]

Improved Experimentation Experience for Researchers

Researchers often need an easy-to-use tool to take their research ideas from concept to code. A proof of concept may only be achieved after several iterations and the concept may be published after several

[3]What's coming in TensorFlow 2.0 - TensorFlow - Medium, https://medium.com/tensorflow/whats-coming-in-tensorflow-2-0-d3663832e9b8 (last visited Jun 8, 2020)