CHAPTER 1

Introduction

furor est profecto, furor egredi ex eo et, tamquam interna eius cuncta plane iam nota sint, ita scrutari extera, quasi vero mensuram ullius rei possit agere qui sui nesciat, aut mens hominis videre quae mundus ipse non capiat.

It is madness, perfect madness, to go out of this world and to search for what is beyond it, as if one who is ignorant of his own dimensions could ascertain the measure of anything else, or as if the human mind could see what the world itself cannot contain.

Pliny the Elder, Natural History 2.1

1.1 WHAT THIS BOOK IS ABOUT

This book is about deep learning or, as it is more popularly known, *artificial intelligence (AI)*, and its application to problems in traditional quantitative finance. It grew from a desire on my own part to learn as much as possible about the subject. Before 2017, I had extensive experience in quantitative finance and in the numerical algorithms typically employed in pricing models, particularly in the context of XVA, the subject of my first book *XVA: Credit, Funding and Capital Valuation Adjustments* (Green, 2015). XVA typically involves high-performance computing (HPC) and, in my particular case, experience with GPUs. I also had exposure to algorithmic differentiation. While I had limited previous experience in machine learning beyond simple regression models, deep learning, using GPUs and gradient-based learning, seemed like a natural fit for quantitative finance.

The first concrete application of deep learning in quantitative finance I explored was replicating a basket option as described in Ferguson and Green (2018). This basket option was chosen because it was of moderate dimension and, simultaneously, challenging because the replicated underlying model used a Monte Carlo simulation. The work aimed to demonstrate that an accurate replicating model could be trained to handle noisy input and that the performance during inference would exceed the original model once trained. It also allowed the impact of hyperparameter tuning to be presented. The experiment was successful; however, simple replication of a pricing function has proved to be a small subset of the capabilities of deep learning in quantitative finance.

What category of method is deep learning? Quantitative analysts have historically used numerical methods to construct models, such as partial differential equation (PDE) solvers and Monte Carlo simulations. Deep learning and, more broadly, machine learning are no different. Deep neural networks are approximation models with weights that are *learned*. A better description would be that the weights are optimised using gradient-based optimisers that are variants of stochastic gradient descent. Neural networks should be seen as another numerical method in the arsenal of such methods available to quantitative analysts and the broader community of scientists. Neural networks have strengths and weaknesses like any other method.

Deep Learning in Quantitative Finance, First Edition. Andrew Green.

However, deep learning is not without controversy in quantitative finance as elsewhere. Like any new approach, it has its sceptics and supporters. The accompanying AI revolution and media hype have not been conducive to the wider adoption of deep learning in quantitative finance. Deep learning is connected with the search for *artificial general intelligence (AGI)* and the timeless quest for humans to create an intelligence like themselves. Media hype there has been, but the last decade has genuinely seen a revolution with the advent of tools like *large language models (LLM)* that are clearly transformative in many fields.

Deep learning is an *empirical* and practical discipline. While mathematical, it is usually limited to linear algebra and hence is relatively straightforward. Training a neural network model, however, is not always straightforward, and making advances requires practical experience. Neural networks are often criticised as 'black boxes' and this opacity can be extended to both the training stage and inference time. It is not always clear why results from a particular network configuration perform poorly or well. Hence, this book contains many examples coded in Python and presented in Jupyter Notebooks. The examples are presented 'warts and all'; many work well, but some do not. Working with deep neural networks means developing experience and intuition to make things work.

The book has two main blocks of chapters. The first block explores deep learning techniques in their original content. This means covering many topics that are not related to quantitative finance at all, such as image classification and image generation. This is justified as it is how I have personally explored the deep learning discipline and provides a firm foundation for using neural network models in any context. The second block of chapters explores deep learning in quantitative finance, building on the foundation of the earlier general chapters. Hence, the book aims to serve two purposes: firstly, to provide a practical introduction to a wide variety of neural network models and, secondly, to explore the application of those techniques in traditional quantitative finance. The selection of topics is personal, reflecting the areas I am interested in. I have made it reasonably comprehensive, but there are inevitably omissions. This is in no small part due to the explosive growth of AI research in the last decade and during the writing of this text.

1.2 THE RISE OF AI

Figure 1.1 presents a timeline of the main developments in AI since 1900. What is notable in this image is the fact that half the entries are from developments in the last 25 years, reflecting the rise of AI in this period. Quantitative finance has explored the use of neural networks since the 1980s. However, practical applications have only arisen since the re-emergence of deep learning in recent years, with a significant increase in interest after 2016. The key challenge this has presented in writing a book has been the pace of change. This has inevitably meant the contents of the book's conclusion are different than they were at the project's inception. For example, when starting to write about generative models, generative adversarial networks (GANs) were the pre-eminent models, but around 2022, diffusion-based models became the leaders and GANs were quickly left behind. This has necessitated an almost constant updating of the text at specific points to reflect ongoing developments.

1.2.1 LLMs

One development that happened part way through this project was the arrival of highly capable LLMs like GPT-4 from OpenAI and Claude 3 from Anthropic. Writing assistants like Grammarly now embed generative AI tools within their workflow. I have used LLMs and Grammarly as writing assistants and as a coding copilot to accelerate the preparation of examples since the release of these tools in 2023. LLMs are helpful as productivity tools, a subject I will return to later in Chapter 20.

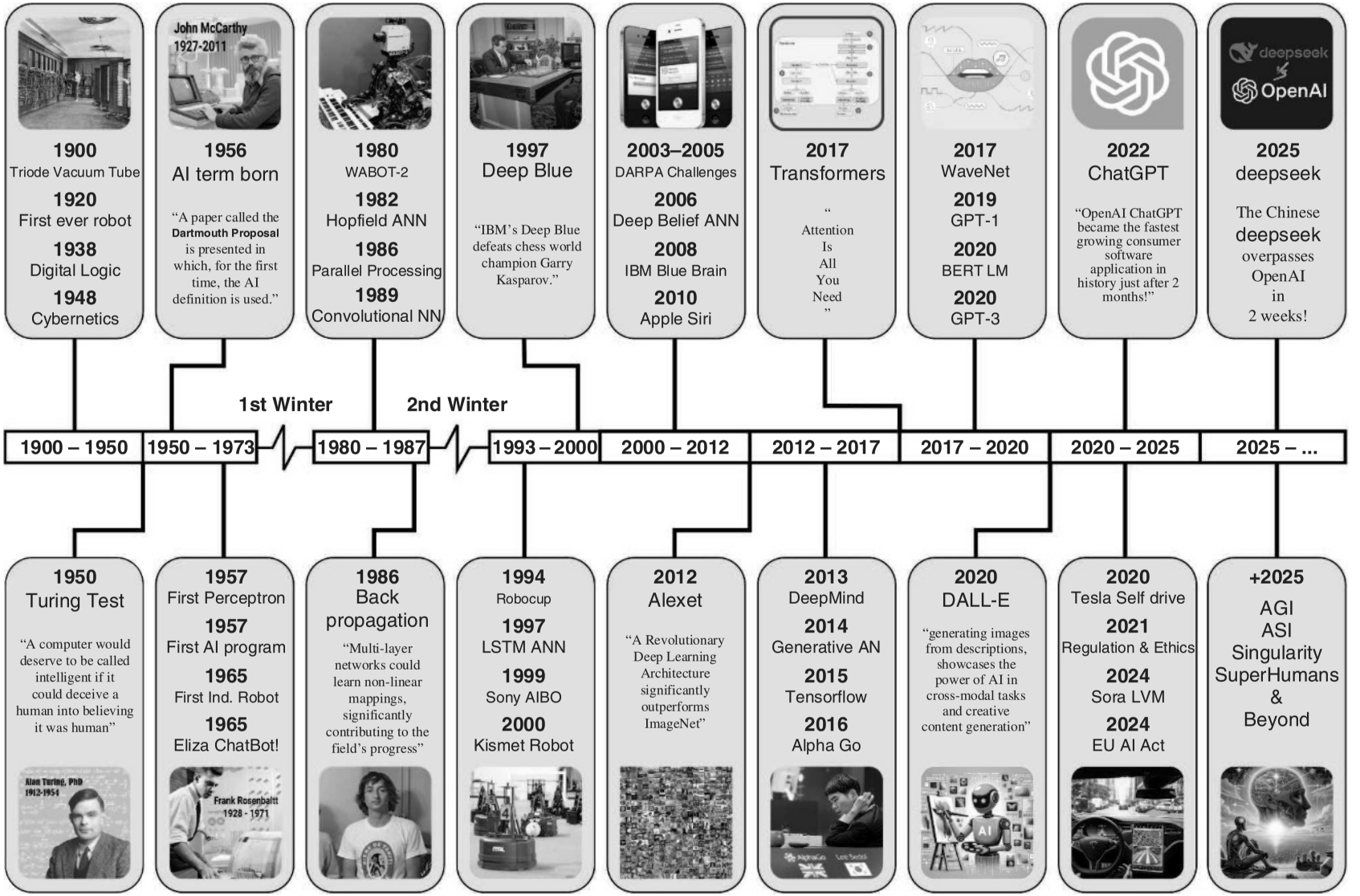

FIGURE 1.1 A timeline of the history of AI. *Source*: Tarjomyar on Wikimedia Commons. Licensed under CC BY-SA 4.0 https://creativecommons.org/licenses/by-sa/4.0/deed.en. Printed versions have been converted to greyscale.

1.3 THE PROMISE OF AI IN QUANTITATIVE FINANCE

One driver for this book is the *promise* of neural networks in traditional quantitative finance. While many and varied examples of deep learning are covered in the later chapters, that promise is not yet fulfilled. There is limited evidence of many deep learning models in production applications at the time of writing in 2025. This reflects partly on the inertia of financial market participants, where more traditional techniques still hold sway. However, quants are steadily finding new ways to use neural networks, and initial scepticism is slowly being replaced by practical steps forward. There is more to do; hence, this work also advocates using deep learning models. Deep learning will never replace traditional methods, but it does *complement* them well.

1.4 PRACTICALITIES

As noted earlier, deep neural networks are often seen as 'black boxes' that cannot be understood. In reality, they are, at best, 'grey boxes' that can sometimes be characterised through exploring the features they learn, as can be seen from the various methods to visualise features in computer vision applications as described in Chapter 6. This is only possible with small models; it is impossible for LLMs with trillions of parameters. While there are essential theoretical results such as the *universal function approximation theorem* (Hornik et al., 1989, 1990; Cybenko, 1989; Hornik, 1991), they are limited in number. Advances in deep learning have typically come through experiment and data science. Experience gained through practice is invaluable in guiding research.

1.4.1 The Examples

The examples are available on the GitHub repo:

```
https://github.com/greandrew/DeepLearningBookExamples
```

Each chapter with examples has its subdirectory, and the Python packages are grouped by chapter. Using Python virtual environments to manage the dependencies is possible, but I recommend using Docker containers. All examples are presented in Jupyter Notebooks or through Jupyter Lab. A small number of Python files are sometimes used to manage utilities. Some examples are computationally demanding, and a reasonably powerful consumer GPU card with NVIDIA® CUDA™ capability is strongly recommended (NVIDIA Corporation, 2007).[1] Most models have been trained using an NVIDIA RTX™3060 or NVIDIA RTX™4090.[2]

Note that all code examples either presented in the book or the accompanying code repository are subject to the following disclaimer:

> THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM 'AS IS' WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

[1] NVIDIA® CUDA™ is a trademark of the NVIDIA Corporation in the United States and/or other countries.
[2] NVIDIA RTX™ is a trademark of the NVIDIA Corporation in the United States and/or other countries.

1.4.2 Python and PyTorch

Python is the language of choice for AI model development and training, which has been true for the last decade. For deep learning model development, I have adopted PyTorch

```
https://pytorch.org/
```

exclusively. Python is a powerful and succinct language which facilitates rapid development. It is normally interpreted and hence is not as fast as compiled languages like C++. However, given that most of the operations used in deep learning happen inside dedicated AI frameworks like PyTorch, this is not generally a problem. Python is among the best-supported languages, with many libraries and tools available.

All languages have weaknesses of one type or another, and Python is no exception. Python's Achilles heel is compatibility, and while there are many tools available, not all of these are compatible with each other and often depend on specific versions of Python or dependent packages. Resolving these dependencies can be complex, and while installation tools like `pip` do an excellent job of finding compatible packages, they can be slow and sometimes fail. Managing a self-consistent base version of Python and all the required packages is impossible; hence, virtual environments are essential. However, even virtual environments can prove problematic if different versions of CUDA and the associated packages are also needed. This has led me to switch to containers for Python development and Docker.

1.4.3 Docker

Docker provides *containerisation*, that is, it provides a service that allows applications to be isolated from each other but use fewer resources than a full virtual machine:

```
https://www.docker.com/
```

Docker containers make managing software environments straightforward. In particular, it allows different versions of libraries, like those that form part of CUDA, to coexist on the same physical workstation or server without conflict. They can run applications without human intervention, but they can also be run interactively and support an external web browser. The ability to run a container with a browser output means it can be used interactively with Jupyter Notebook and Jupyter Lab, as with these examples.

Each chapter with examples provides two or three files:

- `docker-compose.yml` – A YAML configuration file.
- `Dockerfile` – A command file that tells Docker how to build the container.
- `requirements.txt` – Some chapters have a separate requirements file that is used to install Python packages with `pip`.

The `docker-compose.yml` file should be modified to specify the location of the chapter example directly on the host machine. The `volumes` parameter tells Docker where to map the internal container directory. So, for example:

```
version: '3'
services:
jupyter:
runtime: nvidia
build:
context: .
dockerfile: Dockerfile
working_dir: /home/notebooks
```

```
shm_size: 32gb
ports:
- "8888:8888"
deploy:
resources:
reservations:
devices:
- capabilities: [gpu]
volumes:
- PUT_YOUR_PATH_HERE:/home/notebooks
- .:/app
```

In this example, the default size is 32 GB, which may need to be modified if you have less RAM.

To build the docker container:

```
$ docker-compose build
```

To start the docker container:

```
$ docker-compose up
```

Shutting down the Jupyter Notebook or Lab will shut down the container.

1.5 READING THIS BOOK

The book is organised into four parts: Introducing Deep Neural Networks, Foundations, Applications and The Future.

The first part contains this introduction. Part II comprehensively reviews deep learning methodology. Chapter 2 introduces *feedforward neural networks* or *multilayer perceptrons*, the most basic type of deep network. Training neural networks by stochastic gradient descent is the subject of Chapter 3, while Chapter 4 introduces various regularisation techniques. Choosing good values for hyperparameters is essential for successful deep learning and is discussed in depth in Chapter 5, including several different methodologies for hyperparameter search. Computer vision tasks have been dominated by *convolutional neural networks (CNNs)*, and this is explored in Chapter 6 in their original context of images. However, CNNs are also a pre-requisite for *generative* models and find application elsewhere, such as in time-series generation and market prediction. Sequence models are another primary type of neural network model, discussed in Chapter 7. Sequence models include recurrent neural networks but now include *attention* mechanisms and *transformers*. Like CNNs and images, sequence models are most naturally introduced in the context of *natural language processing (NLP)*. Autoencoders, neural networks that reproduce their input, are introduced in Chapter 8; the *variational* type of autoencoder is applied in generative modelling. The block of chapters from 2 through 8 provides the foundation upon which all subsequent chapters rely.

The last two chapters in Part II discuss generative models (Chapter 9) and deep reinforcement learning (Chapter 10), respectively. Generative models are introduced in their original context of image generation. However, this chapter provides the key background for the later application of generative models to market data generation in Chapter 18. Reinforcement learning (RL) is a subdiscipline of machine learning in its own right, and deep RL is the subset that uses neural networks for function approximation. Chapter 10 provides a basic introduction to RL and deep RL and is essential background for Chapter 19 on *deep hedging*.

Part III presents a series of chapters, each focused on a key application of deep neural networks to traditional quantitative finance. Chapter 11 trains deep neural networks to replicate derivative valuation functions. PDEs and backward stochastic differential equations can be solved by deep learning, which is the subject of Chapter 12, while Monte Carlo methods can also be accelerated by neural networks, which are discussed in Chapter 13.

Chapter 13 also includes *deep optimal stopping* and the application of neural networks to the pricing of Bermudan and American-style options. *Static replication* models have a long history in quantitative finance, and the neural network variant of these models is explored in Chapter 14. Chapters 15 and 16 discuss the closely related applications of volatility surface modelling and model calibration. Chapter 17 reviews the applications of neural networks to XVA that are not discussed elsewhere in the book. Derivative valuation and deep optimal stopping are directly applicable to XVA as well. Generating realistic synthetic market data is the key application of generative models in quantitative finance and is reviewed alongside classical techniques in Chapter 18. Similarly, deep hedging is the critical application of deep RL in quantitative finance and is explored in detail in Chapter 19.

The book concludes with Part IV and the last chapter, Chapter 20. Here, the future of quantitative finance is discussed, and the expanding influence of AI is considered.