



# *appendix A*

## *Introduction to PyTorch*

---

This appendix is designed to equip you with the necessary skills and knowledge to put deep learning into practice and implement large language models (LLMs) from scratch. PyTorch, a popular Python-based deep learning library, will be our primary tool for this book. I will guide you through setting up a deep learning workspace armed with PyTorch and GPU support.

Then you'll learn about the essential concept of tensors and their usage in PyTorch. We will also delve into PyTorch's automatic differentiation engine, a feature that enables us to conveniently and efficiently use backpropagation, which is a crucial aspect of neural network training.

This appendix is meant as a primer for those new to deep learning in PyTorch. While it explains PyTorch from the ground up, it's not meant to be an exhaustive coverage of the PyTorch library. Instead, we'll focus on the PyTorch fundamentals we will use to implement LLMs. If you are already familiar with deep learning, you may skip this appendix and directly move on to chapter 2.

### **A.1    What is PyTorch?**

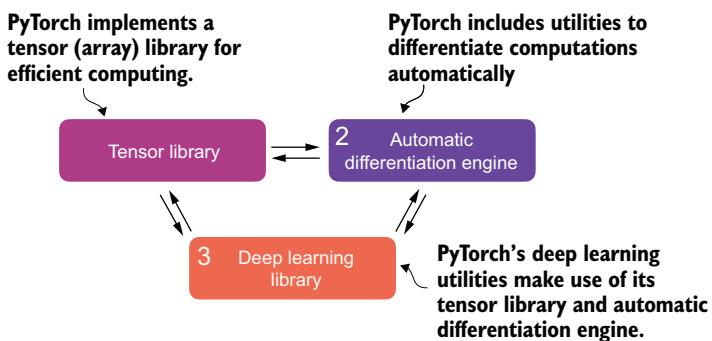
PyTorch (<https://pytorch.org/>) is an open source Python-based deep learning library. According to *Papers With Code* (<https://paperswithcode.com/trends>), a platform that tracks and analyzes research papers, PyTorch has been the most widely used deep learning library for research since 2019 by a wide margin. And, according to the *Kaggle Data Science and Machine Learning Survey 2022* (<https://www.kaggle.com/c/kaggle-survey-2022>), the number of respondents using PyTorch is approximately 40%, which grows every year.

One of the reasons PyTorch is so popular is its user-friendly interface and efficiency. Despite its accessibility, it doesn't compromise on flexibility, allowing advanced users to tweak lower-level aspects of their models for customization and

optimization. In short, for many practitioners and researchers, PyTorch offers just the right balance between usability and features.

### A.1.1 **The three core components of PyTorch**

PyTorch is a relatively comprehensive library, and one way to approach it is to focus on its three broad components, summarized in figure A.1.



**Figure A.1** PyTorch’s three main components include a tensor library as a fundamental building block for computing, automatic differentiation for model optimization, and deep learning utility functions, making it easier to implement and train deep neural network models.

First, PyTorch is a *tensor library* that extends the concept of the array-oriented programming library NumPy with the additional feature that accelerates computation on GPUs, thus providing a seamless switch between CPUs and GPUs. Second, PyTorch is an *automatic differentiation engine*, also known as autograd, that enables the automatic computation of gradients for tensor operations, simplifying backpropagation and model optimization. Finally, PyTorch is a *deep learning library*. It offers modular, flexible, and efficient building blocks, including pretrained models, loss functions, and optimizers, for designing and training a wide range of deep learning models, catering to both researchers and developers.

### A.1.2 **Defining deep learning**

In the news, LLMs are often referred to as AI models. However, LLMs are also a type of deep neural network, and PyTorch is a deep learning library. Sound confusing? Let’s take a brief moment and summarize the relationship between these terms before we proceed.

AI is fundamentally about creating computer systems capable of performing tasks that usually require human intelligence. These tasks include understanding natural language, recognizing patterns, and making decisions. (Despite significant progress, AI is still far from achieving this level of general intelligence.)