1
Introduction

1.1 Why Acceleration Matters

This book aims to accelerate the inference of deep neural networks. Let us first consider why acceleration matters. Faster inference means quicker response times. For instance, in a speech recognition engine, transcription results can be delivered faster. In a translation engine, translations can be provided more quickly. This leads to an improved user experience.

The improvement in user experience through faster response can be divided into three stages. Consider a device that translates spoken Japanese into English in real time.

Stage 1: Unacceptably slow. This is when the response time is so slow that the device becomes impractical to use. For example, if the results are returned an hour or a minute later, the device is not viable. At this stage, even a slight improvement in speed yields minimal benefits. For such a device, a response time of about 5 seconds is the acceptable threshold. The goal at this stage is to reach that threshold. This book provides guidelines on what techniques to use and the level of acceleration they can achieve. By reading this book, those working on systems in Stage 1 can gain insight into whether they can reach the practical threshold. This stage is challenging because it's uncertain whether practical performance can be achieved – if it can't, the entire system may essentially become worthless. However, succeeding here is critical, as it can deliver entirely new value to users.

Stage 2: Direct contribution to user experience. At this stage, acceleration directly enhances user experience. For example, in the case of the translation device, a response time between 0.5 and 5 seconds falls into this category. A 1-second response time is far less frustrating than a 5-second response time. Applying more acceleration techniques at this stage can yield greater benefits. Thus, it is important to explore various techniques.

https://doi.org/10.1017/9781009687119.002 Published online by Cambridge University Press

Stage 3: Sufficiently fast. At this stage, acceleration is no longer required because the system is already fast enough. For the translation device, a response time under 0.5 seconds fits this description. Whether the translation result is delivered in 0.1 seconds or 0.01 seconds, the difference is negligible to users. Although acceleration may not yield direct benefits in this stage, it can still be valuable for cost reduction, as discussed later.

From this discussion, we can see that the impact of faster response times varies. For example, the benefit of halving the inference time differs significantly between Stage 2 and Stage 3. Therefore, it is important to identify which stage the target system belongs to and to establish clear response time goals, especially for Stage 1. Generally speaking, in deep neural networks, a 5× speedup can often be achieved without performance degradation. Beyond that, further acceleration typically comes at the cost of reduced performance.

Naturally, the specific numbers depend on the task and model. This book includes many numerical examples illustrating the degree of acceleration and associated performance trade-offs. By studying these examples, you will develop a better intuitive sense of the feasible range for acceleration.

The benefits of accelerating inference extend beyond faster response times. In many cases, it can reduce system operational costs. For instance, if inference time is halved, the same number of requests can be handled with half the number of servers, effectively halving operational costs. This cost-saving effect continues until the system is scaled down to its minimum required configuration (e.g., a single server). Essentially, the faster the inference speed, the lower the operational costs.

Deep neural networks often require significant computational resources, leading to high operational costs. Even a 10% reduction in operational costs can significantly boost profits. Moreover, in models that require particularly large computational resources, such as large-scale generative models, operational costs can become a bottleneck, making the system economically unfeasible. Acceleration techniques offer proactive solutions by making such systems economically viable and preventing the need to operate them at a loss or abandon the project.

Furthermore, accelerating inference can enhance performance. Larger deep neural networks are known to deliver higher performance. Even if the speed of the current model is adequate, doubling the speed can allow the use of a model twice as large, achieving higher performance while maintaining the original response time.

https://doi.org/10.1017/9781009687119.002 Published online by Cambridge University Press

1.2 Theoretical Background of Acceleration

This book also introduces the theoretical background to understand why deep neural networks can be accelerated. The basic idea is that randomly initialized deep neural networks consume a large amount of memory but can be compressed after training. Random initialization lacks structure, making it difficult to compress. However, after training, particularly when a model successfully generalizes, patterns emerge, revealing both redundancies in the network and robustness to random perturbations.

By leveraging these patterns and eliminating unnecessary redundancies, the model can be represented in a more compact manner than its apparent size (number of parameters $\times$ 4 bytes). Such efficient representations enable acceleration. Although the various techniques for accelerating deep neural networks may appear vastly different on the surface, most are fundamentally based on this concept.

Rather than merely cataloging various techniques, this book aims to explain their shared theoretical foundation to foster a systematic understanding. Although much of the theory behind deep learning is still not well understood, this book explains the theoretical background to the extent that current knowledge allows.

Equipped with this foundational knowledge, you can avoid ad hoc implementations and make informed predictions about the required number of parameters and the extent of acceleration achievable in your tasks.

1.3 Terminology Clarification

A **deep neural network** is a category of machine learning models. There is no strict definition of what constitutes a deep neural network. Models inspired by the neural structure of the brain and their derivatives are generally referred to as neural networks, and among them, those with a large number of layers are called deep neural networks. This book does not provide an explanation of deep neural networks themselves. If you are unfamiliar with the basics of deep neural networks, you may find it helpful to consult resources such as textbooks on deep learning (e.g., Goodfellow et al., 2016) or tutorials for PyTorch or TensorFlow.

Inference refers to the process of inputting data into a machine learning model and obtaining results. Examples of inference include inputting audio data into a speech recognition model to obtain a transcript, or inputting Japanese text into a translation model to obtain a translated output.

https://doi.org/10.1017/9781009687119.002 Published online by Cambridge University Press

Training, or learning, refers to the process of determining suitable parameters for a machine learning model. Typically, this involves searching for parameters such that the model's input–output behavior fits a set of ideal input–output examples. This book assumes that a trained model is already available and focuses on inference. However, many of the acceleration techniques introduced in this book are also applicable to training. A brief overview of training-related techniques is provided in Chapter 9.

Latency refers to the time it takes to obtain results after inputting data into a machine learning model. One of the main goals of this book is to reduce latency.

Throughput refers to the number of data items a machine learning model can process per unit of time. Although throughput and latency are generally inversely related, this is not always the case. For example, by enabling parallel processing of two inputs, it is possible to double the throughput while maintaining the same latency. In such cases, the user experience remains unchanged, but operational costs can be reduced, which is an important consideration in system design. This book also introduces techniques for improving throughput.

FLOPS (floating-point operations per second) denotes the number of floating-point operations performed per second. FLOPS is mainly used to indicate hardware performance. It is often combined with SI prefixes such as G (Giga) or T (Tera) and expressed as TFLOPS (Tera FLOPS, one trillion floating-point operations per second).

FLOP (floating-point operations) refers to the number of floating-point operations. FLOP count is used to quantitatively express the computational cost of a task. While FLOPS and FLOP appear and sound similar, they represent different concepts: FLOPS corresponds to speed, while FLOP count corresponds to distance.

Landau's big-O notation is a mathematical notation used to describe the growth rate of functions. It is used to theoretically express trends in computational or memory complexity. Intuitively, $O(g(n))$ denotes growth no faster than $g(n)$, $\Omega(g(n))$ denotes growth at least as fast as $g(n)$, and $\Theta(g(n))$ denotes growth at the same rate as $g(n)$. The formal definitions are as follows.

- **Big-O notation**: If $\limsup_{x\to\infty} \frac{f(x)}{g(x)} < \infty$, we write $f(x) = O(g(x))$. For example, $f(x) = O(x^2)$ means that f grows no faster than x^2.
- **Big-Omega notation**: If $\liminf_{x\to\infty} \frac{f(x)}{g(x)} > 0$, we write $f(x) = \Omega(g(x))$. For example, $f(x) = \Omega(x^2)$ means that f grows at least as fast as x^2.
- **Theta notation**: If both $f = O(g(x))$ and $f = \Omega(g(x))$ hold, we write $f = \Theta(g(x))$. For example, $f(x) = \Theta(x^2)$ means that f grows at the same rate as x^2.

https://doi.org/10.1017/9781009687119.002 Published online by Cambridge University Press

A CPU (central processing unit) is the basic component of a computer responsible for performing calculations and control. Nearly all computers, including inexpensive servers and personal computers, are equipped with a CPU. This book primarily considers inference using CPUs.

GPUs (graphics processing units) were originally designed for image processing but are now widely used for deep neural network computations. Compared with CPUs, GPUs excel at parallel computation. This book also addresses techniques applicable to inference on GPUs. Some acceleration techniques can be used on both CPUs and GPUs, while others are tailored specifically for either. When using GPUs, it is recommended to acquire a basic understanding through resources such as whitepapers for the specific architecture used[1] to facilitate understanding.

[1] For NVIDIA A100 GPUs, see https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf.

https://doi.org/10.1017/9781009687119.002 Published online by Cambridge University Press