Machine Learning ToolKit

Zeeshan Hooda

14th November, 2023

Contents

Your scientists were so preoccupied with whether they could, they didn't stop to think if they should.

—*Jurrasic Park*

Introduction

This document primarily serves as a collection of tools, processes, and information for real-world machine learning tasks. Plenty of good information already exists regarding the technical aspects of machine learning, so we will focus much more on the tasks that should be carried out before and after algorithm development. Such tasks include gathering requirements, considering ethical ramifications, deciding when ML should or shouldn't be used, deploying models into production, continuous monitoring, and more.

For now, this work may seem unstructured and unorganized, but we will slowly converge on an ideal structure as more items are added.

1 Solution Design

1.1 Requirements Gathering

1.2 Sourcing Data

1.3 Ethical Considerations

1.4 When to Use Machine Learning

2 Machine Learning

2.1 Supervised Learning

2.2 Unsupervised Learning

3 Deep Learning

3.1 Neural Networks

A neural network is a computational model of interconnected simplified artificial neurons which exhibit collective computational properties[1]. The emergent properties of neural networks are useful in artificial intelligence tasks with some degree of input ambiguity[2].

3.2 Hardware Acceleration

As neural networks grow in parameter count, more computational power is required to train them in a reasonable timeframe. Hardware acceleration is often needed when we start to reach the million-parameter mark. Graphics Processing Units (GPU) are the most common hardware accelerators for neural network training and inference, and NVIDIA devices are currently the industry standard. Other hardware accelerators include Tensor Processing Units (TPU) from Google, and Trainium/Inferentia accelerators from AWS.

When measuring compute requirements and accelerator performance for neural networks, we use the units of Floating-point Operations (FLOP) and Floating-point Operations per Second (FLOP/s). Both units can refer to double-, single-, half-, or quarter-precision floating-point numbers, but we assume single-precision 32-bit floats unless otherwise specified.

3.2.1 CUDA

Compute Unified Device Architecture (CUDA) is a parallel computing platform and API from NVIDIA. It allows software to use GPUs for general purpose parallel computation, such as the linear algebra tasks involved in neural network training[3]. NVIDIA GPUs contain a large number of CUDA Cores which operate in parallel. The massively parallel nature of GPU computation can decrease training times by orders of magnitudes when compared with training on CPUs.

3.2.2 XLA

Accelerated Linear Algebra (XLA) is a compiler toolchain for accelerating neural network models with minimal code changes[4]. ...

3.2.3 Neuron

AWS Neuron is an SDK for running deep learning workloads on AWS Trainium and Inferentia instances[5]. ...

3.3 Heuristics

Often when dealing with neural networks, we want rough estimates of compute requirements, training time, and CapEx + OpEx to make decisions on how best to handle model deployment. This section contains a few methods that have yielded high-quality estimations for real client projects.

3.3.1 Estimating CUDA Performance

NVIDIA states that the number of FLOP/s a CUDA device can sustain, R, is equivalent to the number of CUDA Cores, N, multiplied by the device clock rate, C, multiplied by two. The factor of two is derived from the ability to execute two operations at once using fused multiply-add (FFMA) instructions[6].

$$R = 2NC \tag{1}$$

For example, the NVIDIA Tesla P40 GPU has $N = 3,840$ CUDA Cores, and a GPU boost clock, $C = 1.531 \times 10^9$ Hz, so we can derive a reasonable estimate for peak single-precision floating point performance[7]:

$$
\begin{aligned}
R_{P40} &= (2)(3,840)(1.531 \times 10^9\,\text{Hz}) \\
&= 1.176 \times 10^{13}\,\text{FLOP/s}
\end{aligned} \tag{2}
$$

This estimate agrees with the expected value of 1.2×10^{13} FLOP/s provided by NVIDIA[7]. It should be noted this is the *maximum theoretical performance* when leveraging FFMA. We often observe a 15-25% performance decrease from overhead introduced by the communication interfaces, bus speeds, and memory tiering.

It is possible to further parallelize CUDA workloads with multiple GPU devices in a single system. This introduces some CPU overhead as workloads need to be distributed and orchestrated across multiple units. Certain NVIDIA cards designed for datacenters can directly interface with each other over NVLink, a hardware bridge between units which can avoid CPU orchestration altogether.

3.3.2 Estimating Training Compute

...

$$
C_{train} = O_f O_b PSE \tag{3}
$$

...

3.3.3 Estimating Training Time

...

$$
T_{train} = \frac{C_{train}}{R} \tag{4}
$$

...

3.3.4 Estimating TCO

Using the defined heuristics for compute and time requirements, we can develop a reasonable estimation for total cost of ownership (TCO) of a particular neural network algorithm, including upfront capital expenditures and ongoing operational expenditures. There are many sources of error that can impact estimated TCO; however, the goal is to find a reasonable range and order of magnitude for making decisions on return-on-investment potential of neural network approaches.

...

4 Solution Deployment

4.1 Model Lifecycle

4.2 Hardware and Compute

4.3 Software and Tooling

4.4 Continuous Training

4.5 Monitoring Drift

References

[1] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities." *Proc Natl Acad Sci U S A*, vol. 79, no. 8, pp. 2554–2558, Apr. 1982, Accessed: Nov. 14, 2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC346238/

[2] "Definition of Neural Network - Gartner Information Technology Glossary." Accessed: Nov. 14, 2023. [Online]. Available: https://www.gartner.com/en/information-technology/glossary/neural-net-or-neural-network

[3] H. Jang, A. Park, and K. Jung, "Neural Network Implementation Using CUDA and OpenMP," in *2008 Digital Image Computing: Techniques and Applications*, Dec. 2008, pp. 155–161. doi: 10.1109/DICTA.2008.82.

[4] "XLA: Optimizing Compiler for Machine Learning." Accessed: Nov. 14, 2023. [Online]. Available: https://www.tensorflow.org/xla

[5] "Welcome to AWS Neuron — AWS Neuron Documentation." Accessed: Nov. 14, 2023. [Online]. Available: https://awsdocs-neuron.readthedocs-hosted.com/en/latest/index.html

[6] "Achieved FLOPs." Accessed: Nov. 14, 2023. [Online]. Available: https://docs.nvidia.com/gameworks/content/developertools/desktop/analysis/report/cudaexperiments/kernellevel/achievedflops.htm

[7] "New Pascal GPUs Accelerate Inference in the Data Center." Accessed: Nov. 14, 2023. [Online]. Available: https://developer.nvidia.com/blog/new-pascal-gpus-accelerate-inference-in-the-data-center/