An Empirical Study of Distributed Deep Learning Training on Edge (Student Abstract)

Christine Mwase[1], Albert Njoroge Kahira[2], Zhuo Zou[1]
[1]Fudan University
[2]Julich Supercomputing Center
cmwase@ieee.org, a.kahira@fz-juelich.de, zhuo@fudan.edu.cn

Abstract

Deep learning (DL), despite its success in various fields, remains expensive and inaccessible to many due to its need for powerful supercomputing and high-end GPUs. This study explores alternative computing infrastructure and methods for distributed DL on low-energy, low-cost devices. We experiment on Raspberry Pi 4 devices with ARM Cortex-A72 processors and train a ResNet-18 model on the CIFAR-10 dataset. Our findings reveal limitations and opportunities for future optimizations, paving the way for a DL toolset for low-energy edge devices.

Introduction

Deep learning (DL)'s success has been driven by large datasets and hardware technology alleviating memory and compute constraints that hindered AI growth for decades. To train larger and deeper models, DL relies on high performance computing (HPC) to leverage massive parallelism and computing power, enabling remarkable achievements such as solving biology's 50-year old challenge of predicting a protein's 3D shape from its amino-acid sequence (Callaway 2020). However, DL's need for powerful computing and high-end GPUs exacerbates inequalities in access to transformational technologies. We show that by using concepts from supercomputers and datacenters, it is possible to build computing clusters on the edge with sufficient computing power from cheap, low energy (LE) processors (Microsoft 2016; Rajapakse, Karunanayake, and Ahmed 2022).

To advance DL in LE settings, it is crucial to address challenges in compute, communication, and memory, and to redesign the tech stack for optimal balance of trade-offs such as throughput, latency, energy, and accuracy. It is also vital to assess and address DL framework limitations in LE settings. This study explores the opportunities and challenges of DL in LE settings and presents findings on the costs of distributed deep learning (DDL) at the edge. Our contributions include evaluating Raspberry Pi 4 device peak performance, analyzing DL models in LE settings, and implementing and evaluating DDL in LE settings. Ultimately, this paves the way for energy-efficient DL toolsets.

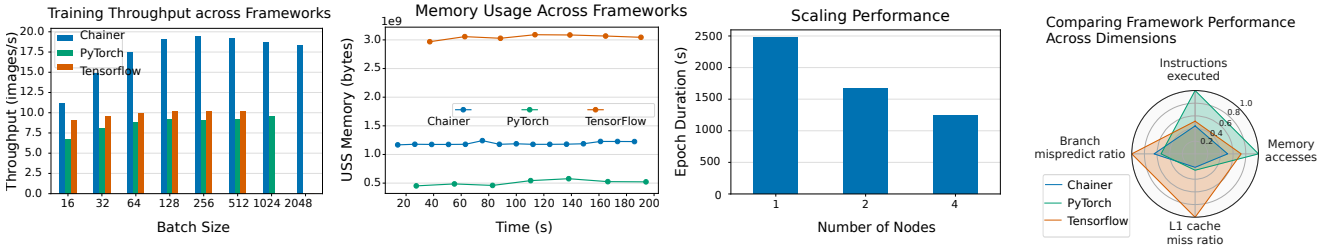
Methodology

We used the LINPACK benchmark (Dongarra, Luszczek, and Petitet 2003) to verify our setup against theoretical and previous work in literature. We then used Raspberry Pi 4 devices to train the ResNet-18 model on the CIFAR-10 dataset (Krizhevsky, Hinton et al. 2009) to evaluate compute, communication, and memory performance of DL on edge. DDL training across multiple devices then followed to evaluate scaling in clusters. Lastly, hardware-related metrics were analyzed to expose how effectively frameworks utilize low-level CPU resources.

Results and Discussion

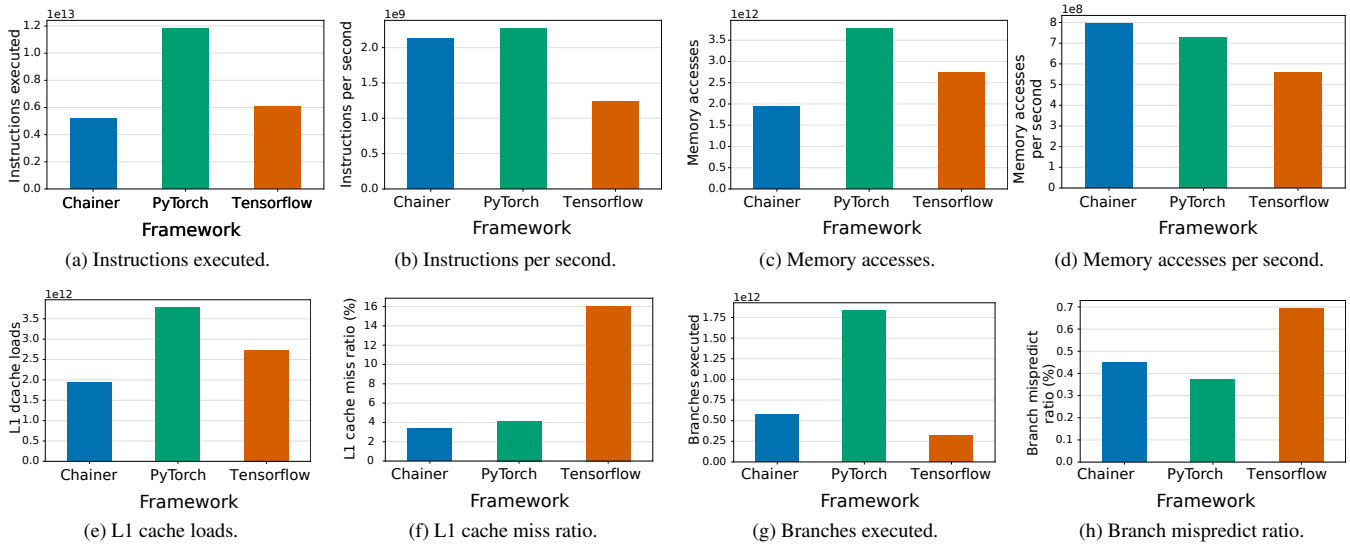
Compute, Memory, Communication and Scaling Analysis Figure 1 (a) shows throughput increasing until an optimal batch size is reached, where loading fewer batches reduces overhead. TensorFlow and PyTorch are limited to batch sizes of 512 and 1024 respectively due to memory consumption. Chainer excels, finishing in $\approx 47\%$ and 53% of the time taken by PyTorch and TensorFlow respectively at a batch size of 256. Chainer was also the easiest to install for both single- and multi-node setups without needing hardware customization. Figure 1 (b) shows Tensorflow using the most memory and PyTorch using the least, explaining the inability to use higher batch sizes for Tensorflow. This also shows the difference between different frameworks' performance due to framework optimisations. Initial results presented in Figure 1 (c) show scaling performance across 4 nodes. While it is clear that scaling is observed, this scaling is not linear and can be improved.

Hardware Based Analysis The varying performance across frameworks indicates the significance and opportunities for optimisation at the edge where resources are premium. Figures 2 (a) and 2 (c) show the impact of coding techniques and libraries on instruction count and memory access. PyTorch has the most instructions and memory accesses, but also the highest instruction throughput in Figure 2 (b), likely due to its ability to exploit parallelism and use low-level performance primitives. PyTorch also seems to effectively leverage caching and branch prediction, in Figures 2 (e) - 2 (h). In contrast, TensorFlow has fewer instructions but the lowest instruction execution rate. It also has the highest L1 cache miss ratio of $\approx 16\%$, likely explaining its



(a) Training throughput. (b) Memory consumption. (c) Scaling tests. (d) Comparing frameworks.

Figure 1: Software-based performance across different frameworks.



(a) Instructions executed. (b) Instructions per second. (c) Memory accesses. (d) Memory accesses per second.

(e) L1 cache loads. (f) L1 cache miss ratio. (g) Branches executed. (h) Branch mispredict ratio.

Figure 2: Hardware-based performance across different frameworks.

low IPC value, and the highest branch mispredict ratio for its relatively few attempted branch predictions. Figure 1 (d) summarises framework performance across multiple dimensions, highlighting strengths and weaknesses. For example, PyTorch is seen to be efficient in branch prediction, while TensorFlow can improve on that.

Conclusion

Enabling DL in LE devices expands DL applications and increases accessibility. This study evaluated DL performance on resource-limited edge devices, using Raspberry Pi 4 devices with ARM Cortex-A72 processors and 4 GB of LPDDR4 SDRAM. Our initial results show that embedded edge devices can train lightweight DL models and collaborate to train larger models, achieving 15.5 GFLOPS on one device and 47 GFLOPS on a 4-node cluster using LINPACK. Secondly, software-hardware co-design and optimization are vital for efficient edge training. DNN libraries are yet to optimize computation and communication for low-end CPUs in non-HPC settings. High- and low-level metrics showed up to $\approx 2\times$ difference in throughput, memory use, instruction count, and L1 cache miss ratio across frameworks. Thirdly, support and tools are lacking for DL training on non-GPU edge devices. Most efforts focus on optimizing inference, leaving substantial engineering efforts to researchers wishing to train on low-end CPU devices. Future work includes expanding evaluation sets, tackling communication bottlenecks, and creating tools for LE clusters, much like the TinyML initiative.

References

Callaway, E. 2020. 'It will change everything': DeepMind's AI makes gigantic leap in solving protein structures. *Nature*, 588(7837): 203–205.

Dongarra, J. J.; Luszczek, P.; and Petitet, A. 2003. The LINPACK benchmark: past, present and future. *Concurrency and Computation: practice and experience*, 15(9): 803–820.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images. Technical report, University of Toronto.

Microsoft. 2016. Democratizing AI For every person and every organization. https://news.microsoft.com/features/democratizing-ai/. Accessed: 2023-12-07.

Rajapakse, V.; Karunanayake, I.; and Ahmed, N. 2022. Intelligence at the extreme edge: a survey on reformable TinyML. *ACM Computing Surveys*.