

Benchmarking Neural Networks on Heterogeneous Hardware Resources

Christopher Noel Hesse^{a,b}, Holger Eichelberger^b

^a*Aptiv Services Deutschland GmbH, Hildesheim, Germany*

^b*Universität Hildesheim, 31141 Hildesheim, Germany*

Abstract

In recent years, artificial intelligence (AI) became a key enabling technology for many domains. To achieve best performance, modern AI methods have high resource demands, e.g., GPU servers for the training of neural networks. With the advent of further processor technologies, such as tensor processors or re-wirable processors, AI methods may be executed in shorter time while even saving energy. For many application domains such as autonomous driving or unmanned aerial vehicles, real-time constraints mandate low end-to-end latencies in AI processing.

In this paper, we present a combined micro- and macro-benchmarking approach to analyze the performance as well as the power demands of modern processor architectures using convolutional neural networks as workload. We discuss tradeoffs among the different processor types and indicate issues and challenges that arise when performing such benchmarks on heterogeneous hardware resources.

We show that FPGAs allow for an increase of 7x up to 45x in performance over high-end GPUs while using only 10% of the power. In the consumer space, novel architectures such as the Apple M1 are able to offer 3-5x better performance at 10-20% the power draw of current x86 CPU or GPU hardware.

Keywords

Latency, power, benchmarking, Artificial Intelligence, neural networks, CNN, GPU, TPU, FPGA

1. Introduction

Artificial intelligence (AI) is a key enabling technology to address challenging problems, like self-driving cars. The increasing capabilities of AI also require more powerful compute resources, e.g., for training of neural networks often graphical processing units (GPU) are utilized. In contrast, a typical expectation is that prediction using an already trained AI model (commonly referred to as inference) can also be performed on less powerful resources.

In recent time, also further hardware architectures such as tensor processing units (TPU) or re-wirable processors like field-programmable gate arrays (FPGA) are applied. While GPUs and TPUs can be programmed through specific software libraries, FPGAs typically require deep hardware knowledge as well as a different development approach. Nowadays, some AI frameworks started to fill this gap and offer support for write-once-run-everywhere code. Examples include OpenCL (for GPU, some FPGA and some CPU backends) or Intel's OneAPI.

SSP'21: Symposium on Software Performance, November 09–10, 2021, Leipzig, Germany


✉ christopher.hesse@aptiv.de (C. N. Hesse); eichelberger@sse.uni-hildesheim.de (H. Eichelberger)

🌐 <https://aptiv.com/> (C. N. Hesse); <http://www.sse.uni-hildesheim.de> (H. Eichelberger)

🆔 0000-0002-2584-5558 (H. Eichelberger)



© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

For developing AI-enabled hardware-accelerated applications, which in the extreme case even rely on multiple hardware architectures, the question arises, which of these architectures shall be applied to which AI method for the best performance. Besides performance measures such as throughput, latency or memory utilization, also acquisition costs or energy consumption are relevant, e.g., to trade-off AI benefits and AI usage with the impact on the environment.

This work was conducted in the context of HAISEM-Lab¹, a BMBF-funded AI-lab, in which the Universities of Hannover and Hildesheim collaborate on AI, software engineering and hardware acceleration. We report on an evaluation of the performance and the power consumption of specialized hardware architectures for AI workloads. Our research question is *whether neural networks can be used as micro- or macro-benchmarks to compare such systems*. We aim at benchmarking CPU-, GPU-, TPU- and FPGA-based systems as well as at the identification of bottlenecks and problems. Related evaluations [1, 2] or benchmarks like MLPerf², cnn-benchmark³ or mixbench⁴ usually focus on other workloads or hardware architectures.

Our results show that FPGA hardware can lead to an increase of 7x up to 45x in performance over a high-end GPU system while using only 10% of the power. Novel consumer space architectures such as the Apple M1 are able to offer 3-5x better performance at 10-20% the power draw compared to current x86 hardware.

Structure of the paper: In Section 2, we discuss related work. We present our benchmarking approach in Section 3 and discuss results in Section 4. We conclude the paper in Section 5.

2. Related Work

In heterogeneous computing, many works also on benchmarking are published, e.g., [3] introducing the the Rodinia benchmark suite. Although similar hardware resources or frameworks are used, most works differ to ours as they do not take AI workloads into account.

Some publications target the analysis or benchmarking of AI methods on different types of processors. Among them, Qasaimeh et al. [1] report on a performance and energy evaluation of AI computer vision kernels: For simple kernels, GPUs provide the best performance while for more complex kernels FPGAs outperform the GPUs. In [2], Karki et al. present a benchmark for deep neural network applications using CUDA and OpenCL on a server GPU, a mobile GPU, and a mobile FPGA. They show that, e.g., the FPGA is more power efficient than the mobile GPU platform. Moreover, there are several (industrial) macro-benchmarking suites such as MLPerf², cnn-benchmark³, or Yolo⁵ as well as the micro-benchmark mixbench⁴.

However, the works discussed above do not cover all hardware resources that we aim for or they impose limitations, e.g., through the programming approach or the workload type.

¹Hardware-optimized AI Applications using modern Software Engineering Methods, <http://haisem-lab.de/>

²<https://mlperf.org/>

³<https://github.com/jcjohnson/cnn-benchmarks>

⁴<https://github.com/ekondis/mixbench>

⁵<https://github.com/pjreddie/darknet>

3. Approach

To perform a systematic comparison for the major types of AI accelerators (ASICs, i.e., application-specific chips are out of our scope), we developed a combined micro- and macro-benchmarking approach for different hardware architectures. As the target hardware is installed on the premises of HAISEM-Lab partners, also power measurements using energy meters are possible.

We selected Artificial Neural Networks (ANN) as workload due to the widespread interest in this AI method. As there are various types of ANNs, we focus on Convolutional Neural Networks (CNN), which are designed for image analysis, e.g., to realize driver support systems. CNNs are intrinsically parallel so that we can yield stress on the highly-parallelizable compute cores of GPUs, TPUs or FPGAs. Moreover, CNNs are reasonably understood and implementation approaches are mature, i.e., frameworks such as TensorFlow or OpenCV even support CNNs for different processor types. A CNN is typically structured in three layers:

1. Dimensionality reduction through **convolution** matrices, also called kernels or filters. In this layer, filters of fixed size, e.g., 3x3 or 5x5, are used to reduce neighbored input pixel values to a single value. Filters can be trained, e.g., to detect edges.
2. Introduction of non-linearity through **activation** functions, which operate on individual pixels. Examples are Sigmoid or rectified linear unit activation (ReLU) function.
3. Reduction of spatial dimensions by **pooling** (subsampling). This layer reduces the computational complexity while preserving as much information as possible. Example techniques are maximizing or averaging sliding two-dimensional windows of fixed size.

Figure 1 illustrates our synthetic CNN pipeline involving these three layers. The custom CNN used for the macrobenchmark looks like the following: convolution → pooling → convolution → pooling → convolution → flatten → dense → dense. In total, it features 122,570 trainable parameters. Please note that real CNNs may employ a larger number of layers or other kinds of processing steps. In our approach (cf. [4] for details), each CNN layer is made up of multiple invocations of the same function (convolution, pooling, activation) and each function forms a **micro-benchmark workload**. As an additional stress test, we apply a 2D depthwise convolution, where each input channel of a colour input is processed individually. An entire execution of a CNN represents a **macro-benchmark** targeting also interactions among the layers. As **input** for the convolution, we use artificial single-channel gray-scale and multi-channel (RGB)

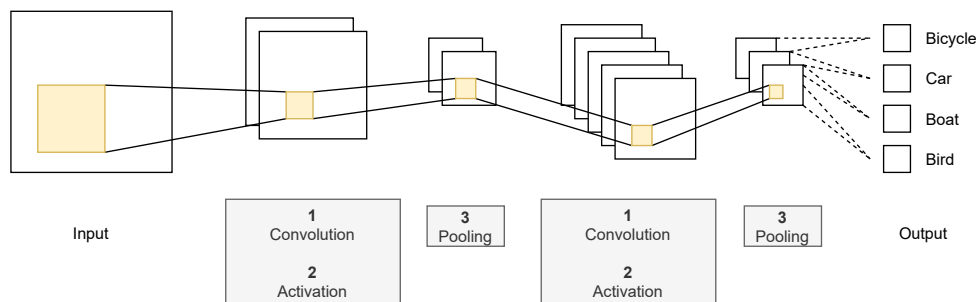


Figure 1: Basic CNN pipeline used as micro- and macro-benchmarks.

Id	Type	CPU	RAM	Accelerator
DGX-1	GPU	2 Intel Xeon E5-2698	512 GB	8 Nvidia Tesla V100
P100	GPU	2 Intel Xeon E5-2697	256 GB	2 Nvidia Tesla P100
V100	GPU	2 Intel Xeon E5-2697	256 GB	2 Nvidia Tesla V100
V100-SXM2	GPU	2 Intel Xeon Gold 6148	384 GB	2 Nvidia Tesla V100-SXM2
A100	GPU	2 AMD Epyc 7662	1 TB	4 Nvidia Tesla A100-SXM4
Arria	FPGA	2 Intel Xeon Gold 6248	384 GB	4 Intel Arria 10 GX PAC
Omen	GPU	1 Intel Core i7 9750H	16 GB	1 NVidia GeForce RTX 2070 Max-Q
M1	GPU	1 Apple M1	16 G	1 Apple M1
Jetson	GPU	1 ARM A57 SoC	4 GB	1 Nvidia Maxwell GPU

Table 1

Utilized hardware used in the experiment (as provided by the HAISEM-Lab partners).

images of sizes 100x100, 1000x1000 and 3000x3000 with usual convolution kernels of size 3x3 or 5x5. For analyzing the pooling as well as the ReLU activation we utilize the single-channel input. For the macro-benchmarks, we make use of the CIFAR-10 dataset⁶ for training.

To enable comparability among different hardware resources, we base the **implementation** of the benchmarks on TensorFlow 2.3.0 wherever feasible. For FPGAs, which are not directly supported by TensorFlow, we use Intel OpenVino⁷ to convert the models to the FPGAs. For the macro-benchmarks, we use two similar CNNs, one developed from scratch in TensorFlow and, to exploit the power of state-of-the-art frameworks, one in YOLOv3⁵ with model weights converted to TensorFlow. During a benchmark, we **collect** one data point every 10 seconds containing CPU and RAM usage (perfstat) and GPU usage (Nvidia-smi). However, Nvidia-smi does not run on embedded devices, so we monitor also the system-level power consumption via an external Voltcraft Energy Logger EL4000. The EL4000 records one data point per second.

We use an intentionally wide range of **hardware subjects** as shown in Table 1, which include server (DGX-1, P100, V100, V100-SXM2, A100, Arria), desktop (M1), laptop (Omen) and embedded (Jetson) resources. Most subjects run Ubuntu Linux (version 18.04 or 20.04), Omen is installed with Fedora Workstation 33 and the Mac Mini M1 with macOS 11.2. Initial plans to also utilize TPUs did not work out due to software incompatibilities. As alternative, we aimed for TPUs in Google Cloud, which were not available to us due to license/region restrictions.

A single benchmark consists of setup (including startup of the monitoring), execution and teardown to save the results and to prepare the next iteration. Each iteration runs for 10 minutes so that potential warm-up phases can be excluded during the analysis and that sufficient information can be recorded by the external energy logger(s).

4. Results and Discussion

We summarize now the results of conducting the micro- and macro-benchmarks⁸ (cf. [4] for more insights). The data gathered hints at CNNs lending themselves very well to parallelization.

⁶<https://www.cs.toronto.edu/~kriz/cifar.html>

⁷<https://docs.openvino toolkit.org/>

⁸Replication package <https://doi.org/10.5281/zenodo.5572610>

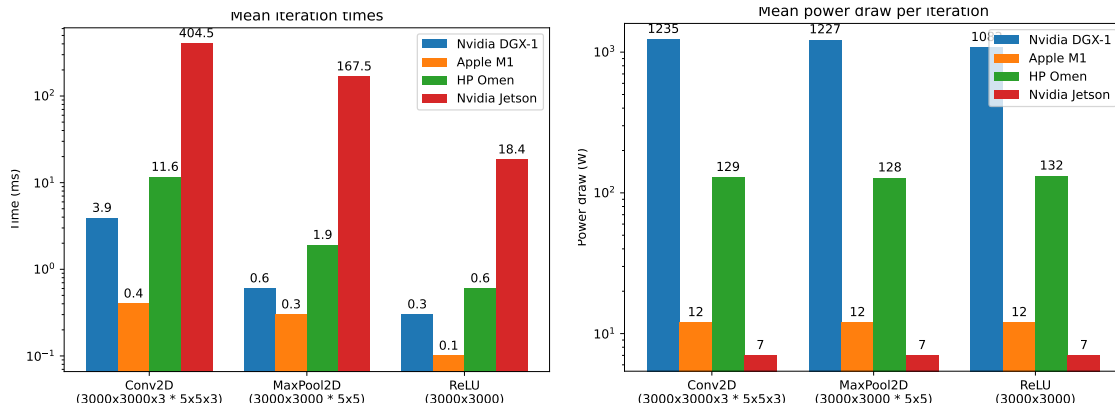


Figure 2: Micro-benchmark iteration times (left) and power draw (right) for image size 3000x3000.

At the same, a large number of low-complexity cores (such as those used in contemporary GPU models) win over a smaller number of high-complexity ones (such as modern CPUs). This is hardly surprising, as CPU cores are optimized for serialized computing. In most of our micro-benchmarks, sufficiently large input sizes such as RGB images containing 3000x3000 pixels are able to fully saturate even high-end GPUs. This is not the case with e.g. the ReLU benchmark, as it can hardly be parallelized. Here, the subjects were very close in performance.

Figure 2 illustrates some **micro-benchmarking** results for inputs with 3000x3000 pixels. For the power draw, we show the watts as reported by our energy logger (at a sample rate of 1Hz). Since each benchmark was executed for 60 seconds, we can calculate the approximate energy consumption as mean power draw times the duration, e.g. for Conv2D and DGX-1: $1235 \text{ W} * 60 \text{ s} = 74,100 \text{ Ws} = 74,100 \text{ Joule}$. The Apple M1 SoC is able to deliver impressive levels of performance, being composed of only CPU and GPU cores. Compared to traditional x86 platforms with dedicated accelerator chips, we find the unified memory architecture to be victorious. On the other hand, much of the support libraries and frameworks such as TensorFlow are still in beta for Apple silicon: We encountered here issues where the performance was invariant with regards to the input size which makes little sense. No other tested platform exhibited such issues. The various servers equipped with Nvidia GPUs performed in line with our expectations and their specs. For that reason, we only display the DGX-1 system results in the result diagrams. For all collected benchmark data, the variances were generally low given a sufficient input size. For example, the DGX-1 system showed standard deviations around 0.001 for a mean value of 0.09 ms at an input size of 100x100x3. Given an input of 3000x3000x3 pixels, the mean was 3.9 ms and the standard deviation was 0.03 for the 2D convolution benchmark.

The **macro-benchmark** results can be summarized as follows. Using the CIFAR-10 dataset to train our own network resulted in low resource usage in most of the server class systems (for both, training and inference). This can most likely be attributed to the small input size of 32x32x3 pixels. During the YOLOv3 inference benchmarks, the FPGA system outperformed all other tested systems by a large margin with a mean forward pass time of just 0.9 milliseconds. Curiously, the Apple M1 performed rather poor here, with a mean time of 401.6 ms when using the CPU only and 1214.9 ms when using the GPU cores. The Nvidia GPU servers yielded

numbers in the range of 35 to 55 ms for a full forward pass. Interestingly, the system equipped with the V100 GPUs outperformed the one with the A100 GPUs.

5. Conclusions

Artificial Intelligence is a key enabling technology in many domains, but the success of AI depends on supporting hardware architectures. In this paper, we provided experimental insights into performance and power consumption of recent hardware architectures through micro- and macro-benchmarks, showing that CNNs can be utilized for system comparisons.

From our results, it is evident that generic hardware architectures such as FPGAs outmatch the performance of other architectures such as GPUs and CPUs. At the same time, they consume less power and, thus, allow for higher efficiency. Specialized ASICs such as TPUs, which were unfortunately not available to us for technical and license reasons, may perform even better. A major problem with ASIC and even FPGA programming remains in the complexity of the software stack. Programming can be very complex and requires highly specialized domain experts. Heterogeneous computing stacks such as OpenCL aim to solve this issue, but the actual quality of the implementation and proper mapping to hardware vary greatly between vendors.

One curious observation is that novel architectures such as the Apple M1 are able to leverage traditional components (CPU and GPU) but achieve much higher efficiency and performance through unified memory architectures. This avoids costly buffer copies and reduces the latency by a great deal, as can be seen in our micro-benchmark results. Overall, the Apple M1 ends up outperforming all other platforms when it comes to performance per dollar and even performance per watt, with FPGA systems being a possible exception.

Acknowledgments

HAISEM-Lab is partially funded by the Federal Ministry of Education and Research (BMBF) under grant 01|S19063B.

References

- [1] M. Qasaimeh, K. Denolfy, J. Loy, K. Vissersy, J. Zambreno, P. H. Jones, Comparing Energy Efficiency of CPU, GPU and FPGA Implementations for Vision Kernels, in: Intl. Conference on Embedded Software and Systems (ICCESS'19), 2019, pp. 1–8.
- [2] A. Karki, C. P. Keshava, S. M. Shivakumar, J. Skow, G. M. Hegde, H. Jeon, Tango: A Deep Neural Network Benchmark Suite for Various Accelerators, CoRR abs/1901.04987 (2019).
- [3] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, K. Skadron, Rodinia: A Benchmark Suite for Heterogeneous Computing, in: Intl. Symposium on Workload Characterization (IISWC'09), 2009, pp. 44–54.
- [4] C. N. Hesse, Analysis and Comparison of Performance and Power Consumption of Neural Networks on CPU, GPU, TPU and FPGA, 2021. URL: <https://sse.uni-hildesheim.de/studium-lehre/beispiele-guter-ausarbeitungen/>.