The University of Dublin, Trinity College

Benchmarking Neural Networks on Heterogeneous Hardware

Author:
Michaela Blott

Supervisor:
Prof. Linda E. Doyle
Prof. Miriam Leeser

This thesis is submitted for the degree of
Doctor of Philosophy

20th May 2021

I would like to dedicate this thesis to my loving family, my colleagues and friends at Xilinx, and the ones who motivated me during its preparation.

Declaration

I declare that this thesis has not been submitted as an exercise for a degree at this or any other university and is entirely my own work.

I agree to deposit this thesis in the University's open access institutional repository or allow the Library to do so on my behalf, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement.

I consent to the examiner retaining a copy of the thesis beyond the examining period, should they so wish (EU GDPR May 2018).

Signed:

Michaela Blott

Acknowledgements

Firstly, I would like to express my deepest gratitude to my supervisor Prof. Linda Doyle and my co-supervisor Prof. Miriam Leeser. Both of them have provided excellent guidance, patience, encouragement, enthusiasm and support. Their insights, feedback and their way of conducting research greatly impacted my work and improved my research skills during the last four years.

Secondly, I would like to thank everyone in my team in Xilinx. I never stop learning in discussions with Yaman Umuroglu, Nicholas Fraser, Giulio Gambardella, Lucian Petrica, Ken O'Brien, and Alessandro Pappalardo. In particular, I would like to thank three of my interns, namely Lisa Halder, Alina Vasilciuc and Johannes Kath, who helped with many experimental measurements under my supervision.

I would like to thank Kevin Cooney, who was the one who actually pushed me into pursuing my PhD degree, my brother Joerg, Kees Vissers for guidance and collaborating over 10 years at Xilinx, and Ivo Bolsens for his continuous encouragement when it comes to self-development.

Finally, most thanks go to my husband Paul and my whole family for their love and support throughout the entire effort.

Contents

Abstract

Neural Networks have become one of the most successful machine learning algorithms and are playing a key role in enabling machine vision and speech recognition. Their computational complexity and memory demands are challenging which limits deployment in particular within energy-constrained, embedded environments. To address these challenges, a broad spectrum of customized and heterogeneous hardware architectures have emerged, often accompanied with co-designed algorithms to extract maximum benefit out of the hardware. Furthermore, numerous optimization techniques are being explored to reduce compute and memory requirements while maintaining accuracy. This results in an abundance of algorithmic and architectural choices, some of which fit specific use cases better than others and it is not obvious which approach benefits from which optimization and to what degree. Finally, there is a vast amount of published numbers that were measured under different deployment settings such as power and operating modes, batch sizes, thread counts, and stream sizes, and not always using the same measurement methodologies, which obfuscates this already complex design space even further.

For system-level designers and computer architects, there is currently no good way to systematically compare the variety of hardware, algorithm and optimization options. While a number of benchmarking efforts have emerged in this field, they don't address the particular demands of heterogeneous hardware architectures and cover only subsections of the embedded design space. None of the existing benchmarks support essential algorithmic optimizations such as quantization inherently. We propose a novel benchmark suite that addresses this need. **QuTiBench** is a novel multi-tiered benchmarking methodology, including microbenchmarks and theoretical baselines, that supports algorithmic optimizations and helps system developers understand the benefits and limitations of these novel compute architectures. The theoretical level of the benchmark is unique: It can predict performance and track compute efficiency. Finally, QuTiBench is systematic with a clear measurement methodology. As such we hope it can help form a basis to drive future innovation in this field.

We evaluate our benchmarking methodology systematically, initially in the context of inference, with different types of CNN topologies leveraging both pruning and quantization as the most promising optimization techniques. We test across a spectrum of FPGA implementations, GPUs, TPU and VLIW processor, for a selection of systematically pruned and quantized neural networks (including ResNet50, GoogleNetv1, MobileNetv1, a VGG derivative, and a multilayer perceptron). We take the full design space into account including batch sizes, thread counts, stream sizes and operating modes, and considering power, latency, and throughput at a specific accuracy as figures of merit.

These results validate our approach. We show that the benchmark adequately represents the potential of this broad spectrum of solutions, and provides sufficient coverage to drive clarity within the complexity of this design space. The theoretical analysis was shown to be highly effective in predicting performance and optimal design

solutions. As a result it has the potential save significant experimentation time. The microbenchmarks provided interesting system-level insights although we encountered many practical constraints which limited the amount of experimentation that can be conducted. Additionally, the systematic measurements exposed typical behaviour for the different types of hardware architectures. Finally, we've provided experimental proof that the measurement methodology with the distinction between system and compute level performance illustrates the individual data movement characteristics of the various hardware platforms.

There is a critical need for community support as well as truly open data access to generate meaningful research impact. As such we have put significant effort into a web portal which supports third part contributions and offers downloadable and indexed access to all measured and theoretical data points. We expect that through this web portal, our benchmarking efforts can contribute to collective research insights within the community. Alternatively, some of the novel concepts, such as the theoretical baselines as well as the systematic measurement aspects, could be potentially adopted in other large scale benchmarking efforts which already have wider industry support.

1 Introduction

1.1 Motivation

The Rising Popularity of Convolutional Neural Networks. Over the last several years, Convolutional Neural Networks (CNNs)[1] have become incredibly successful. Fueled by the availability of large scale training datasets such as ImageNet as well as the significantly increased floating point compute power made available by the latest generations of Graphics Processing Units (GPUs), training of sufficiently large networks to interesting accuracy levels has become possible. The accuracy, typically given in prediction error percentage. has become comparable to human levels for a diverse set of tasks. A huge variety of neural networks are increasingly deployed in conjunction with robotics, Advanced Driver Assistance System (ADAS), security monitors and many other applications. Furthermore, as they have the theoretical property of being a universal approximator which requires zero domain expertise, they are increasingly applied to previously unsolved problems, and sometimes to replace existing algorithms, unless of course the original algorithm is much lower complexity. In essence, CNNs are going to be present in most parts of technology and our lives in the future.

The challenge of deploying these networks lies foremost in their compute and memory intensity, which poses the largest barrier to adoption particularly within the embedded space where chip area, power, memory and compute availability are at a premium. According to researchers at Baidu, inference requires often billions of operations and training for modern algorithms involves tens of single-precision exaflops to converge and has tens of millions of parameters [1]. Given current hardware capabilities, this clearly limits their deployment in particular within the embedded space where energy is at a premium. Further, this comes at a most inopportune moment in time, as the semiconductor industry is challenged by the end of Moore's law, where performance scaling with every subsequent technology node is limited. Additionally the cost per transistor is on the rise since VLSI manufacturing features have gone below 28nm. This disincentivizes semiconductor companies to increase a chip's complexity. As a result, there will be no immediate relief with upcoming technology nodes and it will be extremely challenging to address the rising needs of CNNs.

[1]We are using the term neural network and model synonymously throughout this thesis.

Enabling CNN Deployment. To enable the deployment of CNNs in particular within energy-constrained compute environments, a rise in **algorithmic** and **architectural innovation** has been spawned. We will introduce these here briefly and discuss in much greater detail in later chapters.

Algorithmic optimizations include topological transformations with pruning and compression schemes. For example, pruning removes synaptic connections between neurons in a CNN. For the example in Table 1.1, pruning can significantly reduce the compute requirements (given in the table in giga operations (GOPs)), down to 30% to be precise. The resulting implementation shows roughly twice the throughput in frames per second (fps) and half the latency in milliseconds (msec) with similar power consumption in Watts (Ws). This is a significant improvement over the baseline implementation prior to pruning and similar benefits can be observed for many different CNNs. The only downside of this technique is a small degradation in level of accuracy. The accuracy in this example is given as top5 which is defined as the models probability to predict the 5 best matching answers.

Table 1.1: Benefits of pruning on ResNet50, Ultra96 using Xilinx DPU: In this example, pruning reduces the compute requirements to a third. The resulting implementation shows roughly twice the throughput and half the latency at almost the same level of accuracy.

ResNet50 Ultra96	GOPs	Latency [msec]	Throughput [fps]	Power [W]	top5 acc [%]
baseline (100%)	7.72	78.27	25.5	8.35	91.26
pruned to 30%	2.45	43.3	46.16	8.25	90.16
Summary	0.32x	0.55x	1.81x	0.988x	-1.10%

In addition, the general trend towards transprecision computing [2, 3] can be nicely exploited within this particular application context. The idea is simply to adjust the precision to the minimum required by the application, through quantization. Let's consider the example of CNNs: Neural networks using floating point operations are the initial choice from the machine learning community, but trained parameters can contain a lot of redundant information [4]. It has been demonstrated that moving from floating-point arithmetic to low-precision integer arithmetic only impacts the network accuracy lightly, especially if the network is retrained [5].

The quantization of neural networks for inference down to 8-bit integers has been widely adopted and is well supported by designated and optimized software libraries, such as gemmlowp [6] and the ARM Compute Library [7]. Even further reduced precisions all the way down to the extreme case of Binary Neural Networks (BNNs) with binary representations for synapse weights, input and output activations have been shown to be highly effective. The key computational advantages obtained by using quantized arithmetic in inference are threefold: Firstly, the quantized weights and activations have a significantly lower memory footprint and the working set of some Quantized Neural Networkss (QNNs) may entirely fit into on-chip memory. Fewer off-chip memory accesses also mean orders of magnitude less energy consumption. Furthermore, as on-chip memory can deliver much higher bandwidth, the utilization of the compute resources is increased for a higher performance. For example, binarization reduces the model size of VGG-16 [8] from 4.4 Gbit to 138 Mbit and that of YOLOv2 [9] from 1.6 Gbit to 50 Mbit. Secondly, replacing floating-point with fixed-point representations inherently reduces processing

power by orders of magnitude [10]. While the reported study is for 45nm ASICs, Field Programmable Gate Arrays (FPGAs) follow similar general trends. Finally, the hardware resource cost of quantized operators is significantly smaller than that of floating-point ones. This allows a much higher compute density with the same amount of resources and thereby an easier performance scaling, as the key computation in neural network inference is repeated multiply-accumulate (MAC) operations. Hardware complexity of an integer MAC basically grows proportional to the bitwidths of both factors [11]. In summary, extreme reduced precision neural networks, which can optimize datatypes down to ternary or even binary representations can bring significant hardware cost savings, scale performance by orders of magnitude, improve energy efficiency, and all that at minimal accuracy loss. For example, a ResNet50 [12] reduced precision variant achieves 76.7% top1 and 93.3% top5 accuracy compared to floating point accuracy of 76.9 (93.4%). Some recent data points are given in Table 1.2 for both top5 and top1 accuracy. Top5 accuracy is as defined above. top1 refers to the probability of the model predicting the correct answer. As can be seen, reduced precision variants can achieve very interesting accuracy results compared to the floating point baselines.

Table 1.2: Accuracy of QNNs: This table shows the accuracy of some QNNs compared to their floating point variants.

Network	**float top1(top5)**	**RPNN top1(top5)**
GoogLeNet [13]	72.9% (91.3%)	68.2% (88.1%)
VGG-like [13]	72.0% (90.5%)	68.8% (88.1%)
ResNet50 [12]	76.9 (93.4%)	76.7% (93.3%)
DenseNet-121 [13]	75.3 (92.5%)	69.6% (89.1%)

The overall potential design trade-offs between accuracy and hardware cost are illustrated in Figure 1.1[14]. The chart visualizes compromises of application-level accuracy and corresponding hardware cost. Hardware cost is measured on an FPGA as a function of programmable resource usage. Floating point CNNs (shown in orange squares) are by far the most expensive to implement, however with the lowest error rate, while reduced precision CNNs can offer competitive low error for a significantly reduced hardware cost. We show different variations of reduced precision CNNs in the figure. The smaller the precision, the greater the hardware cost savings. Optimal solutions are positioned along the pareto frontier. We refer to these charts as shown in Figure 1.1 as **pareto graphs** and use these throughout the thesis.

Architectural innovation is showcased by a broad range of companies, for example Google's TPU [15], numerous start-up companies such as Habana, Nervana, Graphcore, GROC, Wave Computing and Cerebras, as well as a spectrum of reconfigurable accelerators leveraging FPGAs, for example Microsoft's Brainwave [16]. These ASICs offer interesting implementation alternatives in addition to common CPUs, and SIMD-based vector processors such as GPUs. These latest generations of novel chips, typically referred to as Deep Learning Processing Unit (DPU), are not only very different regarding their architectures, they are also increasingly complex leveraging heterogeneous compute fabrics.

For DPUs, we distinguish between tensor processors which leverage a matrix of processing engines (MPE) and spatial architectures which can be further specialized

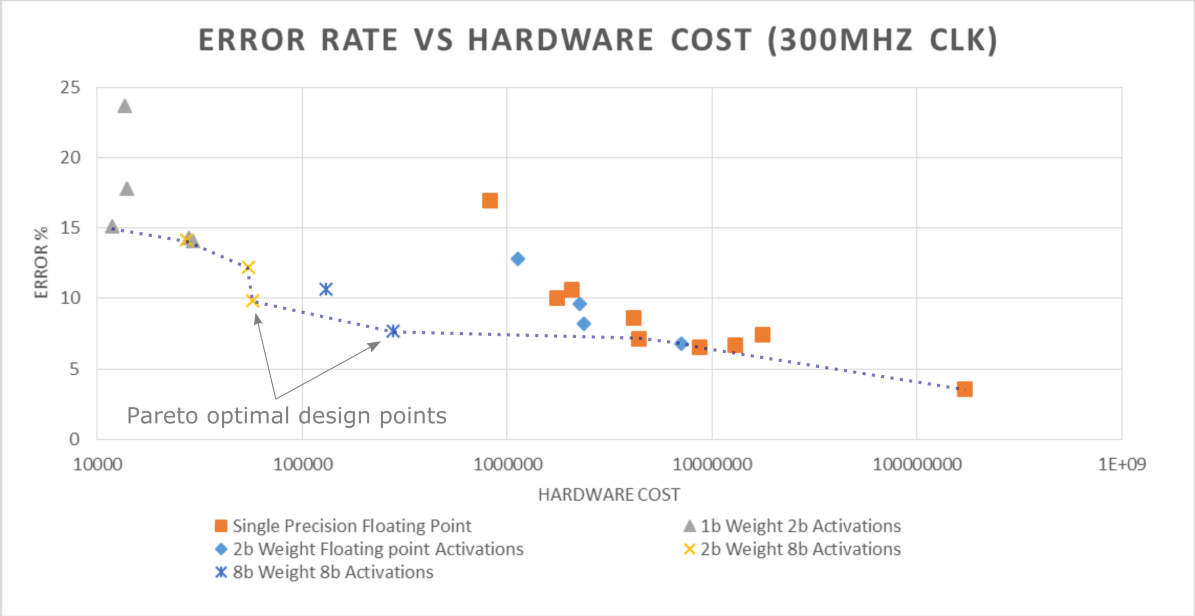


Figure 1.1: Accuracy-hardware cost trade-offs: The chart visualizes compromises of application-level accuracy and corresponding hardware cost. Floating point CNNs are by far the most expensive to implement, while reduced precision CNNs can offer competitive low error for a significantly reduced hardware cost. Optimal design points are positioned along the pareto frontier, which is represented through the dotted line.

for specific topologies using FPGAs. CPUs are the most general solution but high in power. GPUs and DPUs offer highest performance, whereby GPU are more expensive in regards to energy cost. Spatial DPU architectures excel at latency and provide highest compute efficiency through maximized customization. CPUs, GPUs and DPUs (MPE flavour) use a sequential layer by layer compute model whereas spatial DPUs execute all layers of the network concurrently. Hardened topologies in form of ASICs, CPU and GPU offer a fixed set of native dataypes, whereas FPGAs can adopt any precision and numerical representation, which provides utmost flexibility and leverages optimization with quantization to the maximum, whereas hardened approaches need to default to the next higher supported precision. However the programmability in the FPGA fabric also comes at speed and energy cost.

There is no doubt that each of these architectures brings their own inherent benefits. To unleash the benefits associated with the specialized features of the various hardware architectures, the algorithms need to be co-designed. For example, to take advantage of a reduced precision 2bit integer algorithmic logic unit (ALU), the algorithm needs to describe the associated datatypes and operations as 2bit integer. Memory locations might have to be coded explicitly, and similarly, SIMD operations or pipeline architectures often needs explicit instrumentation of the code. The result is, that in order to achieve representative performance on each hardware architecture, a code rewrite is needed.

The Complexity of the Design Space. The resulting design space with specialized heterogeneous hardware architectures and co-designed solutions is highly complex, and

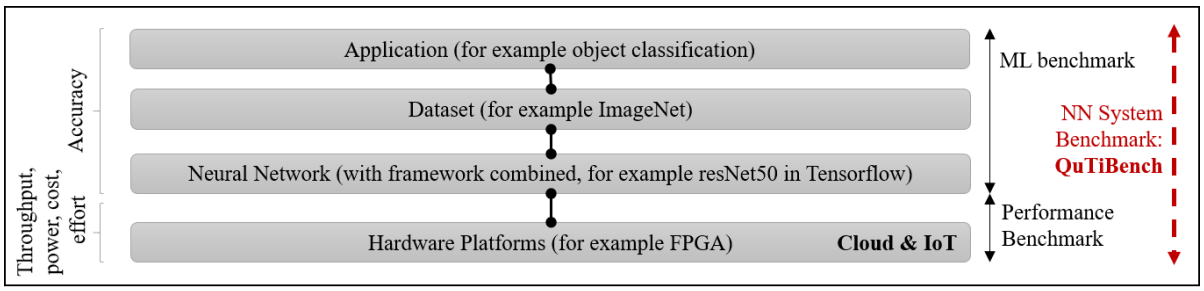


Figure 1.2: The ML design space is complex and consists of different applications, training datasets, neural networks and hardware platforms. Performance benchmarks only focus on hardware platforms. ML benchmarks only represent the upper three layers. A new form of benchmark is needed - so called NN systems benchmark - that capture all dimensions of the design space.

it is becoming increasingly difficult to predict which optimization technique combined with which architecture will deliver what performance for which particular compressed variant of a neural network. This is something I personally observe in my own line of work as a researcher in a big semiconductor company which aims to build chips for acceleration of machine learning algorithms. A thorough and systematic understanding of what algorithm suits which architecture as well as the competitive landscape is essential.

Figure 1.2 visualizes the total complexity which includes further dimensions to the problem. First of all, there are numerous **machine learning applications**, and each of these can be trained with different **datasets** and different neural network **models** and variations, and depending on these factors (as well as potential optimizations with numerical representations, pruning or similar), as well as learning techniques and other hyperparameter settings, the system can produce different results, the key figure of merit being test error rate or conversely, **accuracy**. In addition, there are numerous choices with different **hardware platforms** within the cloud and IoT spaces and everywhere in between. Furthermore, there are different alternatives with regards to the implementation itself. For example within an FPGA, there is a choice between a data parallel and a pipelined implementation. Also, the implementation can be run under different deployment settings such as power modes. All combinations of choices so far result in a different implementation and each of the implementation alternatives will deliver different **performance throughput**, **response time**, **power consumption**, **cost** and required **development effort**. All terminology will be explained in greater detail in the background Chapter 2.

The Need for Benchmarking. As technology developers, it becomes increasingly difficult to predict which architecture will deliver what performance (or other figure of merit) for which particular neural network under which conditions given the complexity of the design space. A benchmarking methodology which sheds light onto which architecture works well for which specific algorithms, given a specific set of figures of merit. Benchmarks at their core encompass a suite of tests for evaluating performance or level of quality and as such are extremely important to avoid poor system choices. When done well, as for example with TPC in the context of data systems [17], benchmarking creates clarity by establishing fair baselines and providing representative comparisons

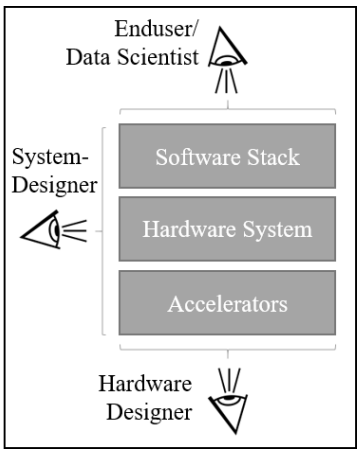


Figure 1.3: The different perspectives of hardware designers, system designers and end user. everyone is interested in a different part of the system. Benchmarking can help bring clarity from all angles.

between different platforms and compute fabrics. They act as the antidote to product marketing and provide system designers a toolbox to avoid making poor choices where end systems fail to meet requirements such as throughput, power or cost, and delay product launch. But the benefits of a good benchmarking suite go beyond this and provide insights from all perspectives (see Figure 1.3). Benchmarks can be of of equally high benefit to hardware designers as well as end users. Benchmarks drive optimizations for semiconductor companies who are customizing compute fabrics for deep learning applications, and for end users, standardized tests help drive optimal purchasing choices. Finally, for newcomers to the domain, benchmarking suites can offer objective summaries that introduce key figures of merit and basic choices as well as setting expectations of the state of the art.

Within this space there are two main types of benchmarks: **Machine Learning (ML) benchmarks** and **performance benchmarks**, indicated in the right part of Figure 1.2. ML benchmarks are typically aimed at achieving low test error, independent of the expensive hardware implications, therefore being of limited efficiency. Resulting algorithms cannot necessarily be deployed in an energy-constrained compute environment. Examples are the ILSVRC ImageNet competition, as well as more sophisticated efforts such as MLBench [18].

On the other side, performance benchmarks are agnostic of the target application, measuring performance characteristics such as throughput and power for characteristic compute patterns. Even when tailored towards characteristic ML workloads, they do not capture the fact that for different hardware architectures, different compute patterns should be used. A system designer in particular for IoT applications would typically leverage co-design and adjust the algorithms to the specifics of the hardware. Support for algorithmic optimizations is essential in this space and poses a particular challenge. As these performance benchmarks do not correlate their results regarding algorithmic optimization back to the application level target, which is accuracy, they cannot provide the necessary algorithmic freedom and scope for algorithmic optimizations, which is an essential ingredient to extracting performance out of heterogeneous computing systems.

A new type of benchmark is needed, we refer to it as **NN system benchmark**, shown in the right part of Figure 1.2. This is the focus of this effort. A NN system benchmark spans the full design space and correlates application performance with system performance. At the start of this thesis, this was a novel concept and didn't exist yet. However, in parallel to our own efforts, a number of alternative approaches have emerged, although they come with significant differences. We will review, discuss and compare these in detail in later chapters.

1.2 Objective

In summary, AI will penetrate most pieces of technology affecting many aspects of our lives. Technology developers are challenged in finding the best solutions for specific deployment scenarios due to the complexity of the design space. The objective of this thesis is to develop a **NN system benchmarking methodology** that enables a fair comparison, a benchmark that sheds light onto which hardware architecture works well for which specific algorithms, given a well-chosen set of figures of merit with the aim of creating clarity within this complex design space of algorithms and emerging hardware architectures for Neural Networks. We aim our benchmark to be representative of common workloads and to be supportive of algorithmic modification. We intend the benchmark to be objective and work on a broad spectrum of heterogeneous hardware systems. Further, we also design for providing fast system insights, perhaps at the expense of prediction accuracy and as such offer complexity vs accuracy trade-offs in results. Finally, this thesis intends to create true research impact for the community.

Research Question: *How can we design a benchmark for neural networks on heterogeneous hardware architectures considering the algorithmic and architectural breadth of the design space.*

1.3 Approach

The purpose of this thesis is to propose and evaluate **QuTiBench**, a benchmarking suite that lies at the intersection of the machine learning and hardware communities and spans the full design space. It is a methodology and not a specific piece of software. QuTiBench couples neural network performance with hardware performance and as such can provide insights as to what is the best possible combination within this design space for specific use cases initially constrained to the embedded space. Although there are a number of efforts emerging simultaneously in this space, such as DeepBench and most notably MLPerf, there is currently no comprehensive benchmarking suite in existence that addresses the scope of what is needed, and in particular targets embedded systems. QuTiBench is

unique in the way it supports algorithmic optimizations and in particular quantization (**Qu**) which is one of the most popular optimization technique for neural networks and leveraged by many specialized hardware architectures. Furthermore, QuTiBench provides multiple tiers of tests (**Ti**) which can provide deep insights for the composition of complex systems and provide trade-offs between speed and accuracy across a broad range of systems. In particular, QuTiBench supports theoretical results as its lowest tier. The theoretical performance predictions can act as a measuring stick and give rough guidance of performance. The middle tier benchmarks different computational and data movement patterns, while on the highest tier we measure full applications for addressing the end user design space. It supports algorithmic optimizations by correlating everything to the application level's figures of merit. Additionally, special emphasis is given to clear measuring methodology. In order to maximize our true impact on the research community, we adhere to FAIR guiding principles [19]. This is required to address findability, accessibility, interoperability, and reusability for all experimental and theoretical datapoints through creating of a persistent identifier, a repository, a web portal [20], and creation of metadata. QuTiBench is open to community contributions, by opening access for contributions through the web portal. Finally, we aim to create clarity in the design space by adding focus on specific data visualizations, whereby we focus all experimentation on CNN inference only.

Compared to the most significant alternative approach, MLPerf, our key differentiators are the proposed multi-tiered concepts with theoretical analysis and microbenchmarks, as well as the systematic testing across all deployment settings rather than MLPerf's focus on a selection of currently 5 very specific application domains only reporting on one figure of merit. As such, QuTiBench can really provide much more in-depth system insights that are interesting to hardware and system engineers, rather than just report for one specific application the achieved performance. Finally, although MLPerf offers some scope for optimizations in its open section, QuTiBench offers more algorithmic freedom and offers visualization of the full design space. In the evaluation chapter, we will show that QuTiBench can create insights that other benchmarks cannot offer and these are highly relevant. For example in my current line of work, we are exploring novel hardware architectures for next generation semiconductors aimed at CNN inference. Through QuTiBench we can create in-depth understanding of possible hardware-software design choices.

1.4 Contributions

The key contributions of this thesis are as follows:

Design of a Benchmark. The benchmark offers the following unique features:

- **Comparisons within a multi-dimensional design space** The challenge in this application domain is that many end-solutions provide optimal results in different dimensions. For example using a single criteria such as best inference response

time for a single image from ImageNet dataset, with a specific minimum accuracy, provides limited insights. It could be that another application might need less accuracy but requires a lower response time. As will be explained later on, we solve this by always tying back all performance figures of merit with application-level performance. Further, we leverage the aforementioned pareto graphs, which highlight the optimal design points. These provide fair and complete comparisons between the various solution endpoints across all dimensions of the design space, and as such offer a broad overview of all the advantages and disadvantages of the various solutions.

- **Clear definition of figure of merit and measurement methodology,** Due to the different nature of relevant hardware platforms, for each platform we are faced with many different deployment options such as power modes and batch sizes. It is essential to establish clear and transparent measurement methodology to help clarify the complex and obfuscated design space.

- **Multi-tiered approach** We use a multi-tiered approach which includes microbenchmarks and a theoretical basis. The microbenchmarks run subsets of compute and data movement patterns on the various hardware platforms to provide insights into system bottlenecks and offer different levels of quality in performance predictions. We leverage a theoretical baseline analysis using roofline models [21], to provide fast performance estimations without having to run any experimentation. Furthermore, this theoretical baseline is highly useful to track compute efficiency on all levels of the benchmark and highlight difficult compute and data movement patterns. Finally, it can be used to measure achieved compute efficiency.

- **Systematic evaluation** QuTiBench includes a highly systematic evaluation approach that measures all figures of merit, for all ML tasks, inference only for the embedded space, for all topologies with all available optimizations, for all deployment settings on all hardware platforms. This is highly beneficial when it comes to creating system-level insights, such as how does batch size impact latency and throughput, or what performance-power compromises can be achieved through different power modes.

In addition to the actual benchmark, there are the following contributions worth noting:

Data Visualization Techniques. We have researched different data visualization routines that help create a better understanding of the multi-dimensional design space. This includes the previously mentioned pareto graphs, heatmaps, box-whisker charts and so-on.

Large Experimental Evaluation and Comparison to State of the Art. We have conducted close to a thousand experiments for three different ML tasks, over 43 different Neural Network (NN) models trained and deployed on 10 different hardware platforms, to get a thorough understanding on how well the benchmark performs in regards to

its unique features, whereby we constraint ourselves to CNN inference for embedded devices. In addition, this brought interesting insights into pros and cons of the different hardware architectures, different optimization schemes, and the various compromises that can be achieved with different deployment options. Furthermore, we compare it with other benchmarking efforts, in particular with MLPerf, which has evolved in parallel to QuTiBench during the development of the thesis. Finally we propose how the benchmarking efforts could potentially be combined.

Web Portal and open and FAIR Data. We understand the critical need for a community to support this effort as well as open and FAIR [19] data to generate meaningful research impact. As such we have put significant effort into a web portal located here: `https://rcl-lab.github.io/QutibenchWeb`. In addition to providing downloadable access to all measured and theoretical data points and including all data analysis and visualizations that were derived within the thesis, this web portal also supports third party contributions. This is essential given the scope of the benchmarking effort which is required within the space. We hope that this web portal can help pull together the research community such that we collectively can have scientific impact.

1.5 Thesis Overview

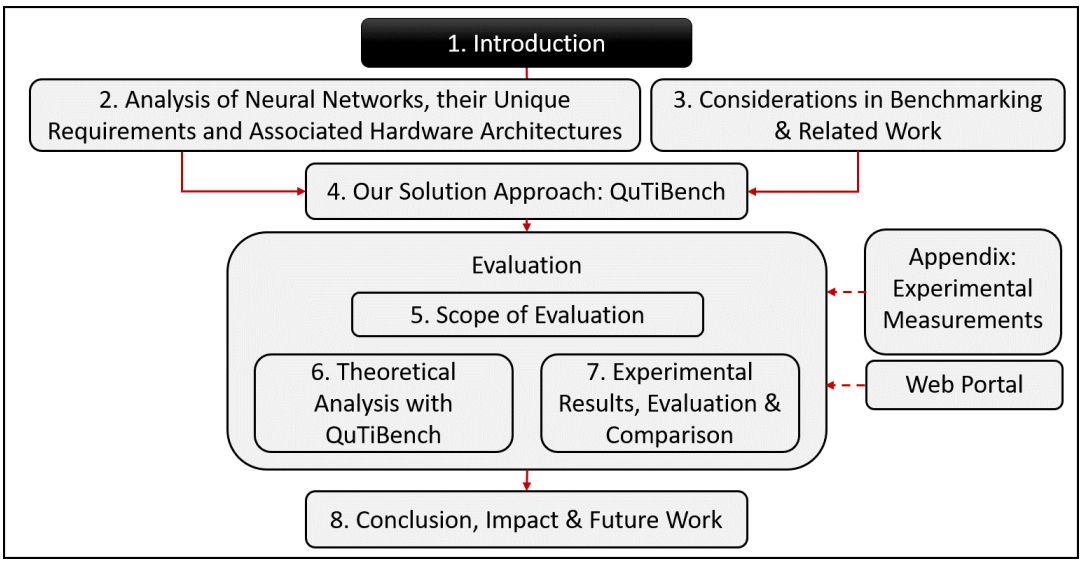


Figure 1.4: Thesis overview: This figure shows an overview of the chapters contained within the thesis and the general flow.

As is visualized in Figure 1.4, the thesis adopts the following flow: Chapter 2 provides a thorough analysis of neural networks and popular hardware architectures used for deep learning. We investigate challenges and characteristics in regards to benchmarking in Chapter 3, which also surveys related work. Chapter 4 explains in detail our proposal towards benchmarking which we then evaluate in the subsequent chapters. Chapter 5

details the experimental setup and scope of the evaluation. Chapter 6 provides the results of the theoretical analysis. The full experimental analysis of the measurements and evaluation of the benchmarking approach is contained within Chapter 7, including a comparison to current and most prominent benchmarking solutions. Finally, Chapter 8 concludes the thesis. It offers insights on lessons learnt, includes insights into the characteristics of the different hardware platforms and a discussion of maximizing our impact through FAIR guiding principles and the web portal and provides an outlook on future work. Supplement material can be found in the appendix as well as the web portal which provides full interactive and open access to all data, theoretical and experimental as well as all code.

1.6 Publications

As part of this thesis, I published the following conference papers and articles, three of which received a best paper award:

Publications as a first author:

- [22] M. Blott, A. Vasilciuc, M. Leeser, and L. Doyle, "Evaluating theoretical baselines for ml benchmarking across different accelerators," *IEEE Design Test*, pp. 1–1, 2021
- [23] M. Blott, N. Fraser, G. Gambardella, L. Halder, J. Kath, Z. Neveu, Y. Umuroglu, A. Vasilciuc, M. Leeser, and L. Doyle, "Evaluation of optimized CNNs on heterogeneous accelerators using a novel benchmarking approach," *IEEE Transactions on Computers*, pp. 1–1, 2020
- [24] M. Blott, L. Halder, M. Leeser, and L. Doyle, "QuTiBench: Benchmarking neural networks on heterogeneous hardware," *ACM Journal of Emerging Technologies in Computing Systems (JETC) Special Issue*, 2018
- [11] M. Blott, T. B. Preußer, N. J. Fraser, G. Gambardella, K. O'Brien, Y. Umuroglu, M. Leeser, and K. Vissers, "FINN-R: An end-to-end deep-learning framework for fast exploration of quantized neural networks," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 2018. This won the best article of the year award.
- [14] M. Blott, T. B. Preußer, N. Fraser, G. Gambardella, K. O'Brien, Y. Umuroglu, and M. Leeser, "Scaling neural network performance through customized hardware architectures on reconfigurable logic," in *ICCD 2017*. IEEE, 2017, pp. 419–422
- [25] M. Blott, J. Kath, L. Halder, Y. Umuroglu, N. Fraser, G. Gambardella, M. Leeser, and L. Doyle, "Evaluation of optimized CNNs on FPGA and non-FPGA based accelerators using a novel benchmarking approach," pp. 317–317, 2020

Co-authored Publications:

- [26] Y. Umuroglu, Y. Akhauri, N. J. Fraser, and M. Blott, "High-throughput DNN inference with LogicNets," in *2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 2020, pp. 238–238. This received another best paper award.

- [27] M. Qasaimeh, K. Denolf, A. Khodamoradi, M. Blott, J. Lo, L. Halder, K. Vissers, J. Zambreno, and P. H. Jones, "Benchmarking vision kernels and neural network inference accelerators on embedded platforms," *Journal of Systems Architecture*, p. 101896, 2020

- [28] E. Giacoumidis, Y. Lin, M. Blott, and L. P. Barry, "Real-time machine learning based fiber-induced nonlinearity compensation in energy-efficient coherent optical networks," *APL Photonics*, vol. 5, no. 4, p. 041301, 2020

- [29] M. Kroes, L. Petrica, S. Cotofana, and M. Blott, "Evolutionary bin packing for memory-efficient dataflow inference acceleration on FPGA," *arXiv preprint arXiv:2003.12449*, 2020

- [30] G. Gambardella, J. Kappauf, M. Blott, C. Doehring, M. Kumm, P. Zipf, and K. Vissers, "Efficient error-tolerant quantized neural network accelerators," in *2019 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*. IEEE, 2019, pp. 1–6. This won the best paper award at DFT'2019.

- [31] Y. Umuroglu, D. Conficconi, L. Rasnayake, T. B. Preusser, and M. Själander, "Optimizing bit-serial matrix multiplication for reconfigurable computing," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 12, no. 3, pp. 1–24, 2019

- [32] Y. Yang, Q. Huang, B. Wu, T. Zhang, L. Ma, G. Gambardella, M. Blott, L. Lavagno, K. Vissers, J. Wawrzynek *et al.*, "Synetgy: Algorithm-hardware co-design for ConvNet accelerators on embedded FPGAs," in *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2019, pp. 23–32

- [33] V. Rybalkin, A. Pappalardo, M. G. Mohsin, G. Gambardella, N. Wehn, and M. Blott, "FINN-L: Library extensions and design trade-off analysis for variable precision LSTM networks on FPGA," *arXiv preprint https://arxiv.org/abs/1807.04093*, 2018

- [34] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers, "FINN: A framework for fast, scalable binarized neural network inference," in *ISFPGA 2017*. ACM, 2017, pp. 65–74

- [35] J. Faraone, G. Gambardella, N. Fraser, M. Blott, P. Leong, and D. Boland, "Customizing low-precision deep neural networks for FPGAs," in *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2018, pp. 97–973

- [36] J. Su, N. J. Fraser, G. Gambardella, M. Blott, G. Durelli, D. B. Thomas, P. H. Leong, and P. Y. Cheung, "Accuracy to throughput trade-offs for reduced precision neural networks on reconfigurable logic," in *Applied Reconfigurable Computing. Architectures, Tools, and Applications: 14th International Symposium, ARC 2018, Santorini, Greece, May 2-4, 2018, Proceedings*, vol. 10824. Springer, 2018, p. 29

- [37] J. Faraone, N. Fraser, M. Blott, and P. H. Leong, "Syq: Learning symmetric quantization for efficient deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4300–4309

- [38] T. B. Preußer, G. Gambardella, N. Fraser, and M. Blott, "Inference of quantized neural networks on heterogeneous all-programmable devices," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2018*. IEEE, 2018, pp. 833–838

- [39] J. Faraone, N. Fraser, G. Gambardella, M. Blott, and P. H. Leong, "Compressing low precision deep neural networks using sparsity-induced regularization in ternary networks," in *International Conference on Neural Information Processing*. Springer, 2017, pp. 393–404

- [40] N. J. Fraser, Y. Umuroglu, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers, "Scaling binarized neural networks on reconfigurable logic," in *PARMA DITAM2017*, 2017, pp. 25–30

- [41] Y. Umuroglu, N. J. Fraser, G. Gambardella, and M. Blott, "A C++ library for rapid exploration of binary neural networks on reconfigurable logic," *H2RC Archive*, 2016

Finally, I have co-authored two commercial whitepapers on compute architectures for inference acceleration, published on the website of my current employer and presented numerous tutorials in this space, most importantly a tutorial on hardware architectures for deep learning at HotChips in 2017:

- [42] Xilinx Whitepaper: The Anatomy of an Embedded Machine Learning Accelerator. Available: https://www.xilinx.com/support/documentation/white_paperswp514-emergingdnn.pdf

- [43] Xilinx Whitepaper: FPGAs in the Emerging DNN Inference Landscape. Available: https://www.xilinx.com/support/documentation/white_paperswp515mlaccelera

- [44] HotChips'2018 (HC30T2): Architectures for Accelerating Deep Neural Nets. Available: https://www.youtube.com/watch?v=ydsZ7A0FF0I&feature=youtu.be

2 Analysis of Neural Networks, their Unique Requirements & Associated Hardware Architectures

2.1 Introduction

This chapter provides the background to the thesis and includes details on the application space, as well as the evolution of associated hardware architectures that cater for it. It is essential to have a good understanding of both parts as in essence they formulate the design objectives of the benchmark. The benchmark needs to be representative of both the application and possible implementations. As such, there are two distinct parts to this chapter: In the first part, we provide an in-depth analysis of Convolutional Neural Networks (CNNs). In particular, it is important to understand the following aspects: neural networks, their evolving topologies, typical characteristics, compute and memory requirements, and popular optimization techniques. The latter are essential to enable adoption of these algorithms within the embedded space and extract maximum performance from the specialized hardware architectures. In the second part of this chapter, we provide a taxonomy of emerging hardware architectures to address the computational needs of CNNs. In order to provide an understanding of the characteristics that a benchmark should cover, we describe the different choices and design trade-offs that this broad spectrum of approaches take, and explain possible deployment options and their impact on performance metrics. We restrict this part to a minimum on what's necessary to understand the thesis. For the interested reader, there are excellent books and tutorials available on this matter, for example [45] and [44].

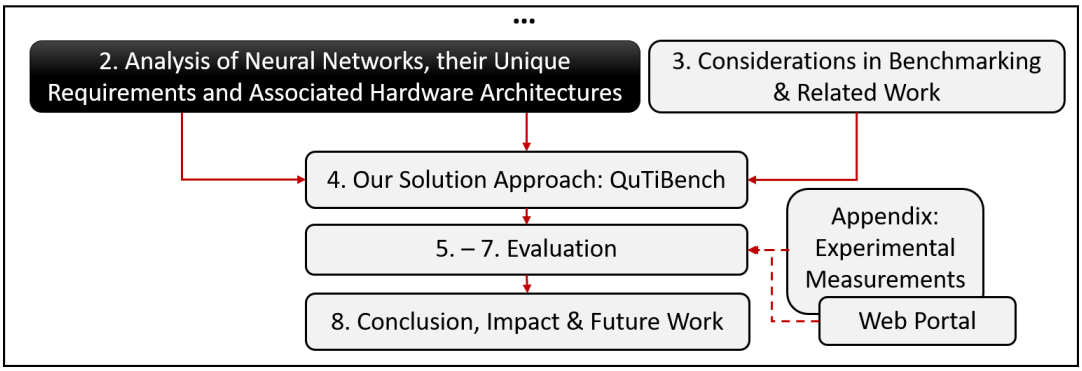


Figure 2.1: Location within the thesis

2.2 Neural Networks

2.2.1 Background and Evolution

Our effort focuses on CNNs, a particular class of Machine Learning (ML) algorithms that forms a subclass of artificial intelligence (see Figure 2.2). ML algorithms differentiate themselves in that they can learn the function, rather than being explicitly programmed, using for example the supervised learning paradigm. CNNs are now widely used for numerous applications such as image processing, natural language processing and speech recognition and their popularity is still increasing because of a number of factors. First of all, with its theoretical property of being a universal approximator [46], neural networks increasingly outperform and replace existing algorithms unless a simpler algorithm exists already. This was demonstrated for example with the original AlexNet in the context of ImageNet classification [47]. Secondly, neural networks can help provide solutions for previously unsolved applications, where no algorithms exist yet. Finally, no domain level expertise is required, instead just sufficiently large datasets together with a sufficiently large topology is required for the network to train for a given accuracy target. All of these factors contribute to their immense and growing popularity.

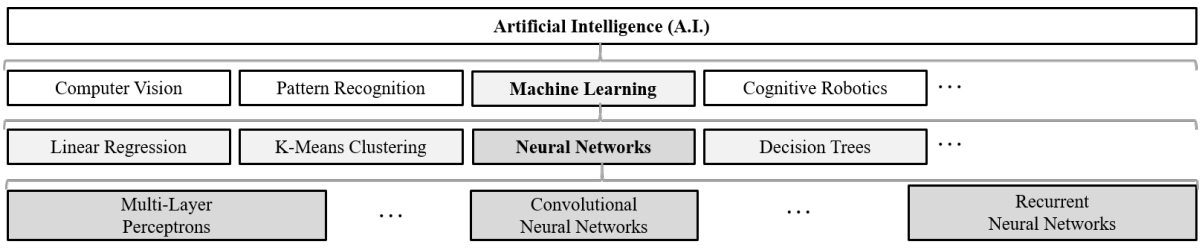


Figure 2.2: Artificial Intelligence - Machine Learning - Neural Networks

Table 2.1: Breadth of popular ML tasks and NN types is vast, categorized here by learning technique, application domain and task. Each task can be carried out by different NN types. We indicate typical compute types in the last column.

Learning Technique	Application Domain	Task	NN Types Models	Compute Type
Supervised	**Vision**	Image Classification	MLPs, **ResNet**, VGG, AlexNet, InceptionV3	FC, CNV
		Object Detection	Faster R-CNN, Yolo9000, Yolov2	FC, CNV
		Semantic Segmentation	Mask R-CNN, SSD	FC, CNV
	NLP	Machine Translation	Transformer, Seq2Seq	FC, CNV, recurrent
		Speech Recognition	DeepSpeech2	FC, CNV, recurrent
		Sentiment Analysis	Seq-CNN	FC, CNV, recurrent
		Language Modeling	Memory Networks	memory network
	Recommendation	Movies	NCF	...
Unsupervised	**Vision**	Feature Extraction	Autoencoder	FC
Generative Adversarial Learning	**Vision**	Image Generation/Modification	WGAN	...
Deep Reinforcement Learning	**Game**	Go	MiniGo	...
		Atari ALE	DeepQ, A3C	...

2.2.2 Design Space and its Associated Complexity

We have visualized the large application space for neural networks in Table 2.1 with domains ranging from vision to Natural Language Processing (NLP) to gaming and recommendation systems. Please consider this as a snapshot in time as new application domains are emerging continuously. In each domain, there are numerous tasks which are amenable for neural networks; for example, within the vision processing context: image classification, object detection, and semantic segmentation. Furthermore, these models can be trained using different training techniques. The machine learning task combined with a training dataset forms the first 2 dimensions of the design space as discussed further below, see Figure 2.3, and the neural network is the third dimension. Note that it is not easy to define clear categories as terms overlap. For example, deep reinforcement learning techniques can be applied to any network. Seq2Seq networks is a full family of networks, while ResNet50, VGG, and InceptionV3 refer to specific topologies.

The design space, as shown in Figure 2.3 is highly complex. For every ML application, there are many different types of neural networks, and new algorithms are still evolving. Furthermore, different types of datasets can be used to train them on specific tasks. The resulting combination can achieve different accuracy targets, and is accompanied with different compute requirements. Hyperparameter selection, such as learning rate and chosen initialization of weights, might impact the accuracy targets too. Furthermore, a neural network model is always paired with a particular framework in which it was trained, as the framework can have impact on the accuracy. These are just the potential combinations on the application side, which are then to be combined with all possible hardware combinations.

The key figure of merit for ML applications is **accuracy** and depending on whether it is image classification, object detection or speech recognition, it might be measured in a different way. For the chosen ML tasks in this thesis, we measure accuracy in top1 and top5 accuracy. Top1 accuracy is the conventional accuracy, and measures the probability of the model answer (the one with the highest probability) matching the expected answer for the test or validation dataset. Top5 accuracy measures the probability of the expected answer to match one of models 5 highest scoring predictions.

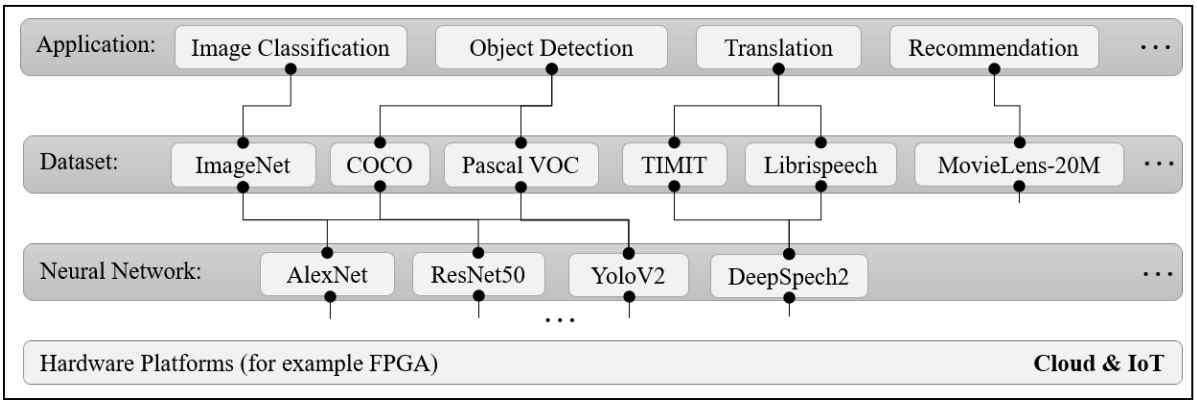


Figure 2.3: Design space is complex and consists of different applications, training datasets, neural networks and hardware platforms

2.2.3 Basics

Great tutorials are available to explain the basics of Neural Networks (NNs) such as [45] as well as my own [44]. As mentioned before, we would like to refer the reader to these. In this thesis, we will cover only the minimum concepts that form the necessary background for this thesis: The goal for NNs is to find a function, $g(x_i)$, which approximates a mapping $x_i \rightarrow y_i \; \forall \; i$, where $\{x_i, y_i\}$ is an input/output pair known as a training example. The class of algorithms used for $g(x_i)$ is on a very high level inspired by the human brain and consist of computational elements named neurons which are arranged in a number of layers. This is depicted in Figure 2.4 and 2.5.

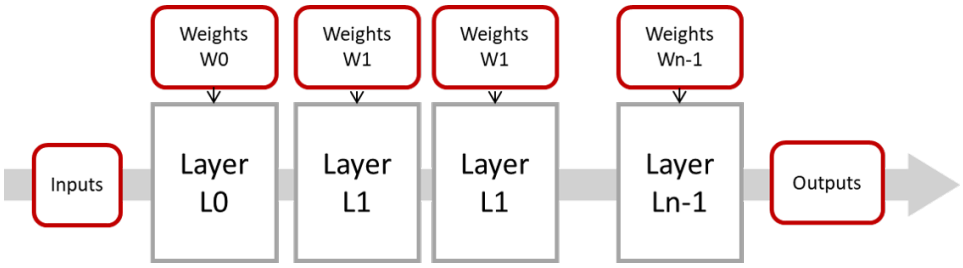


Figure 2.4: Basic structure of a CNN, which consists of a sequence of layers of different types.

In its most basic form, the output of a neuron (n_i) is called an activation and is computed through an activation function which operates on the weighted sum of its inputs plus a bias. An output is only generated when the sum exceeds a given threshold.

Figure 2.5: A layer consists of a number of neurons which are computed with the activation function.

During the training process, which is part of the supervised learning paradigm, the parameters of the NN, the function $g(x_i)$, are adjusted while iterating repeatedly over full

training sets of training examples, such that the generated error, which is the difference between the generated and expected result, is minimized. There are different optimization strategies leveraged, one of the most popular one is Stochastic Gradient Descent (SGD). This iterative training process is visualized in Figure 2.6.

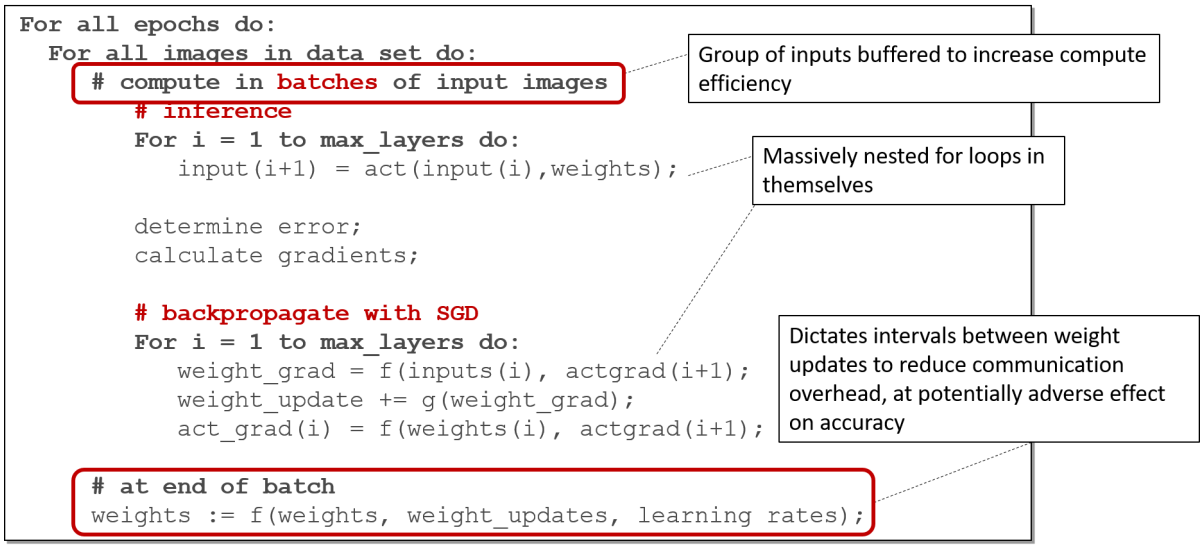


Figure 2.6: Training a NN is an iterative process with massively nested loops. For all epochs over all training inputs, we run both inference and backpropagation, whereby both of those compute over all layers.

There are many common layer types being used. The most basic and popular compute layers are **fully connected, convolutional, pooling, normalization** and **recurrent layers**. These come with very different compute and memory requirements and are briefly introduced here such that the reader has an understanding of how we can derive memory and compute requirements for NNs.

Fully connected layers compute the full cross product between input tensors (for example) and a vector of parameters (also referred to as weights); the latter are determined during the training process. Summed to a bias, this is then fed into an **activation function** as previously introduced. Mathematically, the output, $a_{l,n}$, for the n^{th} neuron in the l^{th} layer of a fully connected network is calculated as follows, where $w_{l,n,s}$ is the weight of the s^{th} synapse connected to the input of the n^{th} neuron in the l^{th} layer, $b_{l,n}$ is a bias term, f_{act} is the activation function:

$$a_{l,n} = f_{act}(\sum_{s=0}^{S_l} w_{l,n,s} a_{l-1,s} + b_{l,n}) \quad , \tag{2.1}$$

Popular activation functions include the hyperbolic tangent function, sigmoid, as well as rectified linear unit (ReLU) and leaky ReLu. Many more are emerging such as for example Mish, which was recently introduced as a first non-monotonic activation function [48]. Fully connected layers are challenging for hardware implementations because of their memory intensive nature and their strong connectivity between inputs and outputs. To be more specific, equation 2.2 provides the calculations for weight

memory W_i, activation memory T_i and compute requirements O_i, whereby IN_i, OUT_i represents the number of inputs and outputs of layer i, and $bits_i$ the given bit precision of the datatype and *batch* represents the number of inputs processed in parallel.

$$\begin{aligned} W_i &= IN_i * OUT_i * bits_i; \\ T_i &= batch * IN_i * bits_i; \\ O_i &= 2 * batch * IN_i * OUT_i; \end{aligned} \tag{2.2}$$

In convolutional layers, the output receives inputs from a small **receptive field** of the previous layer. This approach greatly reduces the number of parameters (or weights) involved and allows local features (e.g. edges, corners) to be found [49]. As such the connectivity issue associated with fully connected layers and the memory footprint are greatly reduced. A basic 2D convolutional layer is similar to a fully connected layer except that: a) each neuron receives an image as input and produces an image as its output (instead of a scalar); b) each synapse learns a small array of weights which is the size of the convolutional window; and c) each pixel in the output image is created by the sum of the convolutions between all synapse weights and the corresponding images. The output of the l^{th} convolutional layer, which takes as input S_l images of dimension $R_l \times C_l$, the pixel, $p_{l,n,r,c}$, at location (r, c) of the n^{th} output image is calculated as follows where $J_l \times K_l$ are the dimensions of the convolution window:

$$p_{l,n,r,c} = f_{act}(\sum_{s=0}^{S_l} \sum_{j=0}^{J_l} \sum_{k=0}^{K_l} w_{l,n,s,j,k} p_{l,n,r+j,c+k}) \tag{2.3}$$

The associated compute and memory requirements can be calculated as follows, whereby IN_CH_i and OUT_CH_i represent the numbers of input and output channels, and F_DIM_i the filter dimensions:

$$\begin{aligned} W_i &= IN\_CH_i * OUT\_CH_i * F\_DIM_i^2 * bits_i; \\ T_i &= batch * F\_DIM_i^2 IN\_CH_i * bits_i; \\ O_i &= 2 * batch * F\_DIM_i^2 * F\_DIM_i^2 I_i * IN\_CH_i * OUT\_CH_i; \end{aligned} \tag{2.4}$$

Pooling layers are introduced to downsample images and thereby reduce the compute in the subsequent layers. Typically a maximum or an average over a receptive field is used for the down-sampling process.

Batch normalization layers [50] help with normalizing the statistics of activation values across layers and with that significantly reduce training times of networks, improve accuracy and renders them less sensitive to initialization. Associated compute is minimal during inference time, but is significant during training time, as it requires subtraction of mean, and division by standard deviation to achieve the zero-centered distribution with unit variance.

Furthermore, recurrent layers are characterized by the fact that they contain state over a sequence of input data. There are many different options for the implementation of the recurrence within the layer, starting from simple recurrent layers, to GRUs or LSTM layers, which can be uni-or bidirectional, feature different numbers of feedback gates, and may include numerous specializations such as peepholes.

Beyond, these basic layer types, there are many layer combinations emerging, such as **inception layers** in GoogleNet[51, 52], **residual layers** in ResNet models [53], **fire modules** [54] and more recently **shift** and **shuffle layers** [32]. Even layer execution becomes dynamic with SkipNet [55], non-Euclidian inputs such as point clouds, netlists and protein interaction networks are leveraged, and neural connectivity becomes irregular [56]. As such, neural networks are evolving more and more in direction of becoming general (differentiable) compute and the speed with which the algorithms are changing, is accelerating. While this fact is hard to measure, growing peer-reviewed conference papers (which have increased by 300% in the last 10 years) and AI related conference attendance are a clear indication that this is a highly active field of research and fluctuations are to be expected [57]. For this thesis, we will focus on the more traditional topologies. Again, the interested reader can find a more detailed description in [45] and [44].

2.2.4 Associated Compute and Memory Requirements

In this section, we describe the computational requirements of the various types of neural networks, which form the design requirements for our benchmark and more specifically the basis to performance estimation later in combination with what will become level-0 of the benchmark. We analyze networks with regards to their arithmetic compute, intermediate storage requirement and memory footprint given a specific model leveraging the formulas provided in the previous section. While actual hardware requirements depend on numerous attributes, at this point we are characterizing the theoretical requirements in an architecturally independent way. For example, actual on-chip memory requirements and external memory requirements depend on implementation choices, but can be derived directly, so this analysis is useful to categorize the different requirements. The scope of the analysis is currently constrained to a set of models that are detailed in Chapter 5, which reviews the scope of the evaluation in great detail.

Inference Each NN layer (L0, L1, etc.) requires a specific number of arithmetic operations O_{L0}, O_{L1}, O_{L2} in the form of multiplications, additions etc. We measure these in giga operations (GOPs) and tera operations (TOPs) respectively. The overall compute of a network with n layers, O_{total}, is the sum of the compute in each individual layer (see eq. 2.5). We define the total modelsize W_{total} as the sum of the weight requirements per layer measured in millions of elementss (MEs); this is independent of any choice in numerical representation. The real memory footprint can be derived by multiplying with the size of the given datatype (for example 32b for single precision floating point). We quantify the intermediate buffer requirement T_{total} in an implementation neutral fashion. For this we calculate the sum of the required amount of tensors T_i that precede each layer. In the context of image classification, these are derived as the product of feature map dimensions (w_i, h_i) and number of channels (ch_i). For exact compute requirements of the

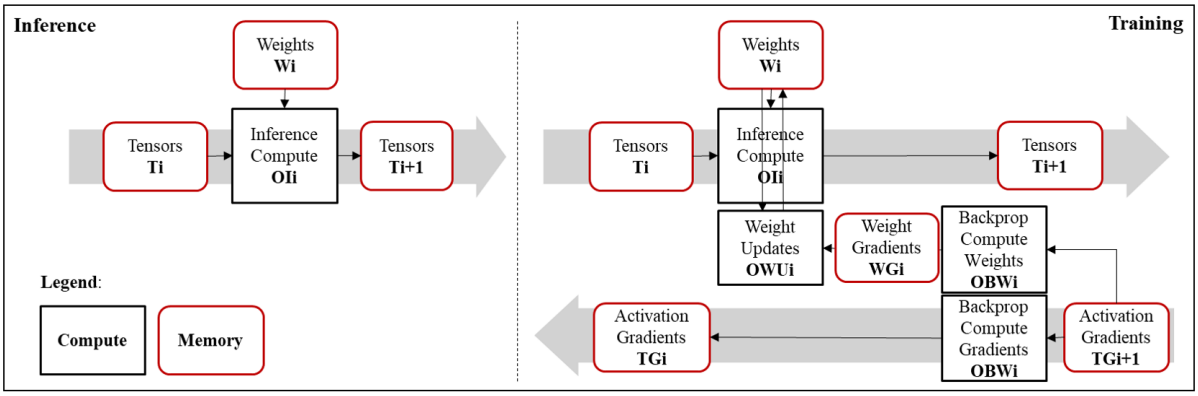


Figure 2.7: Compute, buffer and storage elements in both inference and training. Inference only consists of a forward pass through all layers while training consists of the same forward pass, plus backpropagation through all layers in reverse order with complex data dependencies.

individual fully connected and convolutional layers, please refer back to Equations 2.2 and 2.4. Note that all of this applies to non-linear topologies such as DenseNet [58]; however, our models currently do not reflect graph connectivity. We plan to address this in the future. Figure 2.7 provides an overview of all compute and memory elements.

$$O_{total} = \sum_{i=0}^{n-1} O_i, \quad W_{total} = \sum_{i=0}^{n-1} W_i, \quad T_{total} = \sum_{i=0}^{n-1} T_i, T_i = w_i \times h_i \times ch_i \tag{2.5}$$

Training This thesis evaluates the benchmarking methodology on the basis of inference. However training workloads are critical in the data center context and we expect that it will become increasingly essential in embedded as well as on-line learning takes off. As such we plan to augment our efforts to encompass training in the future. In regards to requirements, we need to consider backpropagation in addition to inference. As depicted in Figure 2.7, training requires numerous more data structures. First of all, symmetrically to the tensors T_i, we need to buffer G_i. Furthermore, so-called weight gradients need to be stored WG_i which are the derivative (in relation to the input weights) of the gradient G_{i+1}. Depending on given optimization strategies, weight updates need to be buffered as well. This results in roughly 3 times the buffer requirements for weights, and double the amount for tensors. Regarding compute, backpropagation requires roughly 3 times the inference compute for a single image of the training data set (plus 1 update operation per weight parameter). Overall compute needs to be multiplied with number of iterations and number of inputs in the training data set. Full details are given in Equation 2.6. Note that data dependencies are significantly more intricate and challenging for training. This is currently not reflected within the theoretical analysis.

$$WG_i \approx WU_i \approx W_i; \tag{2.6}$$

Table 2.2: Ranges and mean requirements for compute (OI and OT) and memory in inference and training, whereby we differentiate weight memory (W), activation tensors in the forward path (T) and backward path (TG) as well as as weight updates (WU).

	Inference			Training		
	$\mathbf{OI_{total}}$ [GOPs]	$\mathbf{W_{total}}$ [MBytes]	$\mathbf{T_{total}}$ [MBytes]	$\mathbf{OT_{total}}$ [GOPs]	$\mathbf{WU_{total}}$ [MBytes]	$\mathbf{TG_{total}}$ [MBytes]
Min	0.00	0.00	0.13	0.00	0.27	0.00
Max	412.17	71.14	138.34	1236.64	276.69	71.14
Mean	62.59	11.9	38.02	187.79	76.05	11.9
SD	107.34	13.51	39.21	322.03	78.41	13.51

We are assuming 8b datatypes for inference and 32b for training.
SD: Standard deviation

Summary of Requirements. The compute and memory requirements of the CNNs form the design requirements for our benchmark. Figure 2.8 and 2.9 visualize initial results, where for Seq2Seq models, we assume a sequence length of 3000 (based on the LSTM test case in DeepBench [59]). All other assumptions are annotated within the figures. The key observations are as follows: First, the compute and memory requirements are on average very high. Mean model size is too big to fit into most on-chip low latency memory (with 71.14MBytes). This is the case even for inference and training is much higher. Compute is in the GOPs range for every single input datum. Secondly, it is obvious that training is much more memory intensive compared to inference (ratios of black bars compared to red bars). Thirdly, there is a significant variation in all requirements for both training and inference as summarized in Table 2.2. No simple generalizations can be made, even within subcategories such as image recognition, as models vary greatly depending on size and complexity of images, number of objects to be recognized, etc. The defined parameters: OI_{total}, W_{total}, T_{total}, OT_{total}, WU_{total}, and TG_{total} help describe the compute requirement for inference and training of each individual network and can be used for baseline computations. When taking architectural constraints into consideration and cross-correlating with roofline models (explained in Chapter 4), we can use them to derive performance guidance.

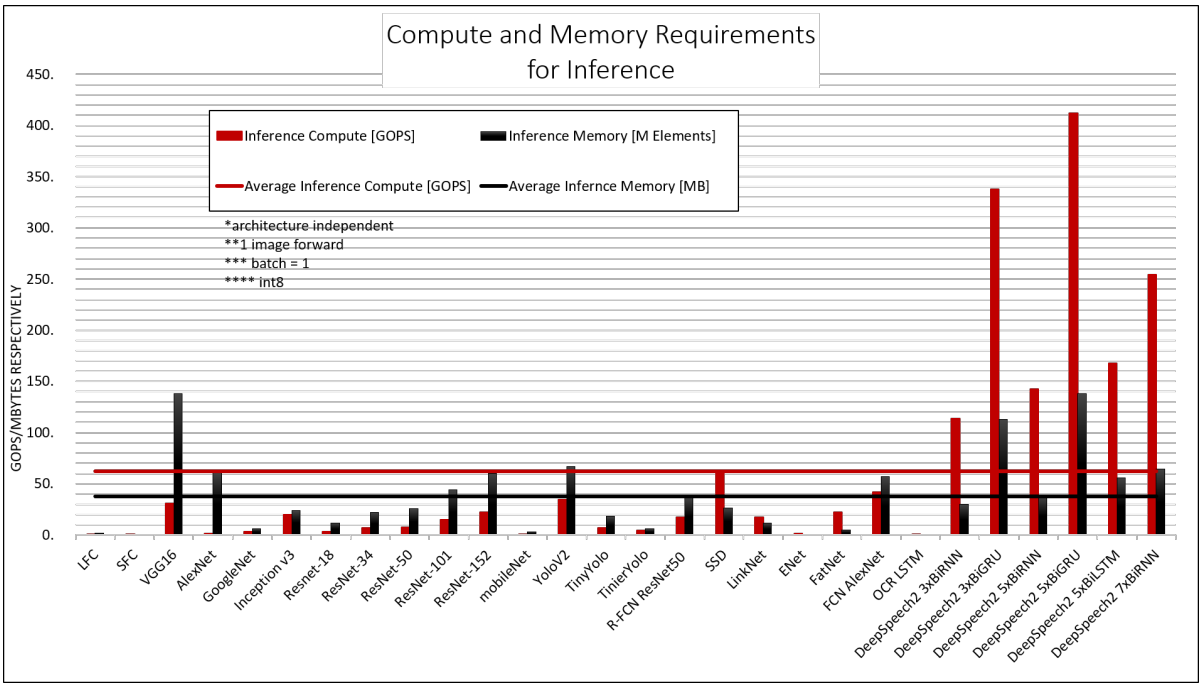


Figure 2.8: Compute and memory requirements for inference for a spectrum of NNs (Visualization of QuTiBench Level-0 - CNN Statistics). We observe on average high requirements plus a high variation which makes it difficult to cater for in hardware architectures.

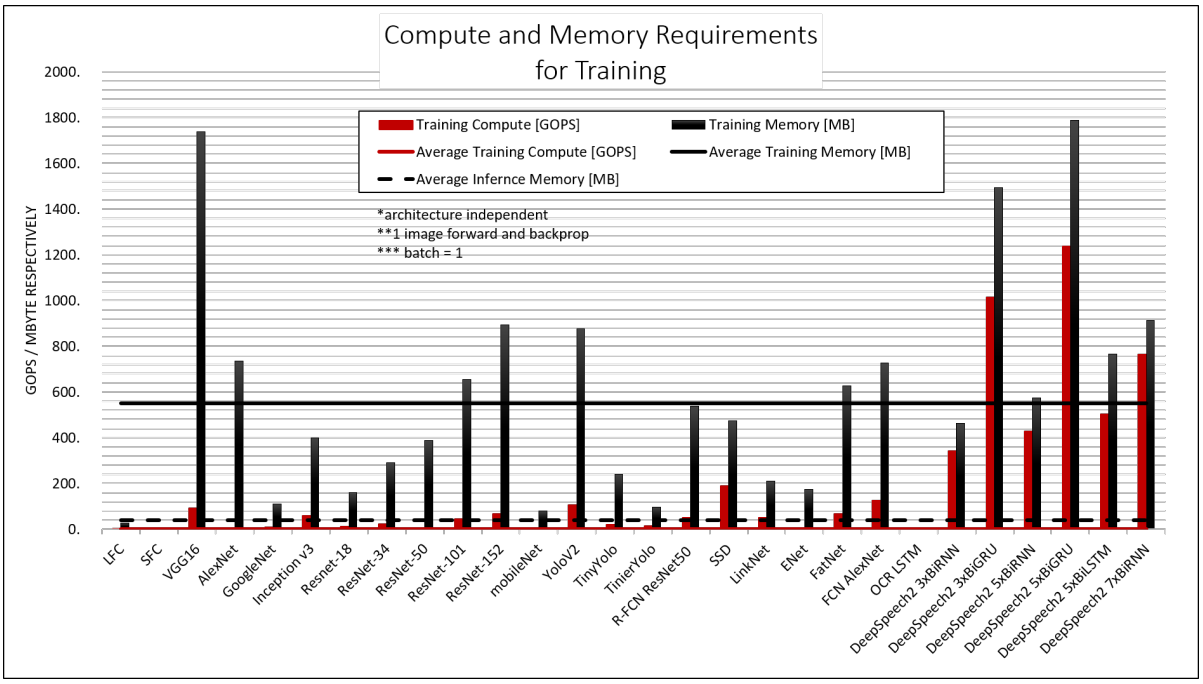


Figure 2.9: Compute and memory requirements for training for a spectrum of NNs (Visualization of QuTiBench Level-0 - CNN Statistics). We observe on average high requirements plus a high variation which makes it difficult to cater for in hardware architectures.

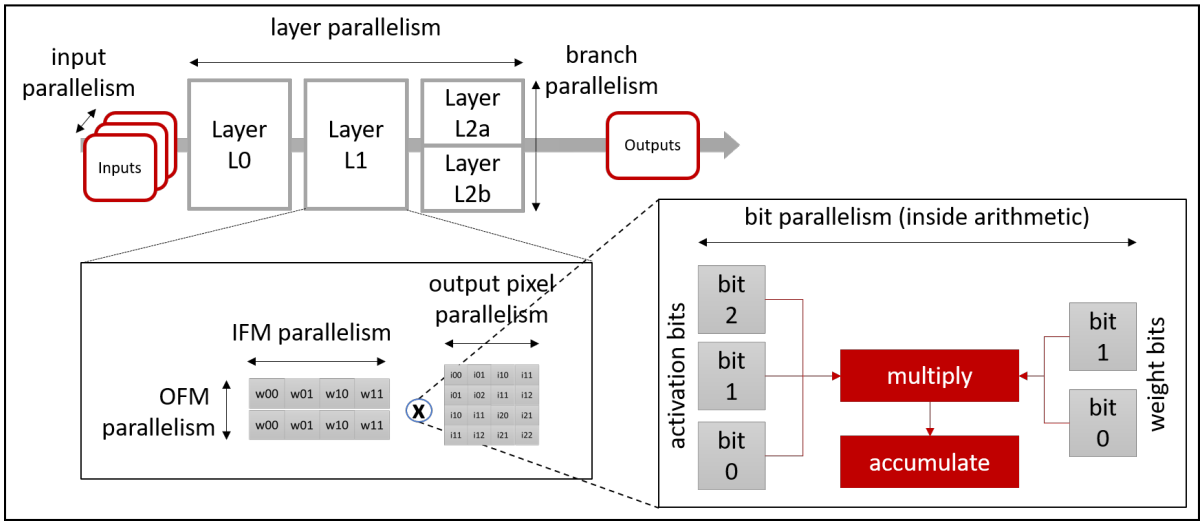


Figure 2.10: Degrees of parallelism in CNNs

2.2.5 Degrees of Parallelism in CNNs

Hardware architectures exploit parallelism in the application to provide compute acceleration. As such it is important to understand what degrees of parallelism are present. Within CNNs, we observe a high degree of parallelism in multiple dimensions. This is illustrated in Figure 2.10 and summarized as follows:

- Coarse-grain topology parallelism between consecutive layers, and parallel branches such as those found in GoogLeNet or in DNN ensembles. We refer to this also as **layer parallelism**.
- Neuron and synapse parallelism inside a layer, such as multiple input/output feature map (IFM/ OFM) channels and pixels in convolutional layers. This is also referred to as **IFM, OFM** or **pixel-parallelism**.
- **Bit-level parallelism** inside arithmetic, when individual bits of weights and activations are viewed separately.

Neuron and synapse parallelism can easily be exploited by all hardware architectures. However, care must be take when parallelizing over layers, as data dependencies exist. This is shown in Figure 2.11. Convolutions between layers create pyramid-shaped data dependencies. this is important when parallelizing this in hardware architectures. For example, pipelined or feed-forward dataflow architectures can adhere to this dependency while parallelizing the

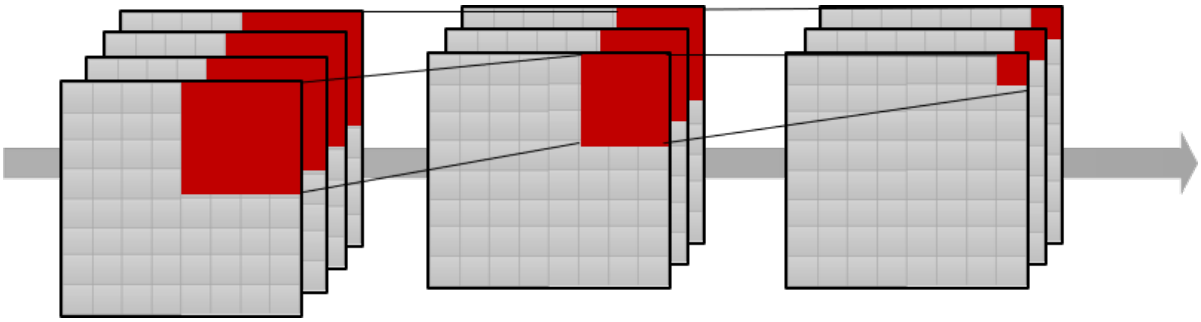


Figure 2.11: Data dependencies between convolutional layers create a pyramid shape.

hardware. However, layer-by-layer compute architectures cannot. Finally bit-level parallelism can only be exploited by architectures that support bit-level processing, such as bit-serial.

2.2.6 Popular Optimization Techniques

As explained in the first parts of this chapter, the challenge lies foremost within the compute and memory requirements For example, classifying an ImageNet image with ResNet50, takes 7.7 billion operations, and requires 101MB of weight storage[2]. Training with backpropagation, assuming 1.2million training images (as in ImageNet), and 100 epochs, requires 10^{18} operations and has a runtime of roughly 2 weeks on a Nvidia P40 GPU. To alleviate the computational burden and maximize performance, many optimizations techniques have been introduced.

1. Loop transformations to minimize memory access [60]
2. Pruning [4]
3. Compression [4]
4. Low-rank transformations [4].
5. Winograd, Strassen and Fast Fourier Transforms [61]
6. Novel layer types (squeeze, shuffle, shift) [62]
7. Quantization & Reduced Precision Numerical Representations [11]

All of these techniques fall under the category of algorithmic optimizations. A representative benchmark must support and measure these, as they are essential for viable deployment solutions.

[2]assuming 32b floating point per parameter

Table 2.3: Accuracy of QNNs: This table shows the accuracy of some QNNs compared to their floating point variants

Network	float top1(top5)	QNN top1(top5)
GoogLeNet	71.4% (90.5%)	63.0% (84.9%)
VGG-like	69.8% (89.3%)	64.1% (85.6%)
ResNet50 [70, 71]	79.26 (94.75%)	64.6% (85.9%)
ResNet50 [72]		64.6% (87.8%)

Particularly successful techniques include pruning and quantization. We discuss both (with specific focus on quantization) in more detail below.

Quantization & Numerical Representations

Transprecision computing is making strides in many application domains [2, 3], and is highly effective for neural network inference, in particular, quantization to reduced precision datatypes, including 8 bit fixed point integer and below, as well as custom floating point formats. On smaller image classification benchmarks such as MNIST, SVHN and CIFAR-10, QNNs achieve state of the art accuracy despite reduction in precision [63, 64], even for partial or full binarization of fully connected and convolutional layers. XNOR-Net [65] applies convolutional BNNs on the ImageNet dataset with topologies inspired by AlexNet, ResNet and GoogLeNet, report top1 accuracies of up to 51.2% for full binarization and 65.5% for partial binarization, while for the more challenging ImageNet benchmark, there is a small but noticable accuracy drop. The resulting solution can run significantly faster in hardware and might still pose an attractive design trade-off. Furthermore, there is significant evidence that increasing network layer size can compensate for this drop in accuracy [40, 66–69].

New quantization schemes show promising results using for example Half-wave Gaussian Quantization (HWGQ) [73] to take advantage of the Gaussian-like distribution of batch normalized activations. Furthermore, new training and optimization techniques [72, 74] can be highly effective to improve accuracy despite the heavy quantization in data types. At the state of this writing, the current lowest error rates for ImageNet classification have been achieved using ternarization [70, 71] as shown in Table 2.3. Quantization has been successfully applied to other tasks including 3D object recognition, facial expression recognition [75, 76], optical character recognition as well as speech [33, 77, 78]. Even in training, research shows that 32bits are not really needed given the typical value ranges for weight and activation gradients and

weight updates involved. Fixed point integers, half precision floating point (FP16), bfloat16, flexpoint or block floating point representations show state-of-the-art performance [79–82]. All of these need to be accurately reflected within the tests.

Pruning

This is another popular optimization technique which has been shown to dramatically reduce both memory and compute requirements, through either synaptic pruning or filter pruning. When synaptic pruning is leveraged, irregular compute patterns result which impact memory access efficiency, thus hardware architectures require support for sparse matrix representations to benefit from this [78]. Filter pruning yields regular compute patterns and benefits thereby a broader selection of platforms [4].

This is another popular optimization which has been shown to dramatically reduce memory and compute requirements, through either synaptic pruning or filter pruning. In the context of synaptic pruning, individual synaptic connections between neurons are removed according to a pruning rule, e.g. when the synapse weight is below a certain threshold. As visualized in Figure 2.12, the resulting compute patterns become irregular and impact both memory and compute efficiency, unless hardware architectures offer support for sparse matrix representations to benefit from this [78]. In the context of convolutional layers, filter pruning (see Figure 2.13) is more popular, as it yields regular compute patterns, which can be easily parallelized and stored efficiently, thereby benefiting a broader selection of platforms [4]. The basic technique computes a so-called **sensitivity** of filters as the sum of magnitude of all included weights and removes all filters below a threshold. In regards to experiments included within this paper, we investigate three of the previously mentioned topologies pruned to different levels, specifically: ResNet50, CNV and MLP. Each pruning variant is given in a percentage such as 100% for the baseline. Unfortunately the definition of the percentage is not consistent. This is an artifact of the associated typically black-box tooling, each with their own definition. In the context of CNV and MLP it relates to the number of inner channels associated with inputs and outputs of hidden layers.

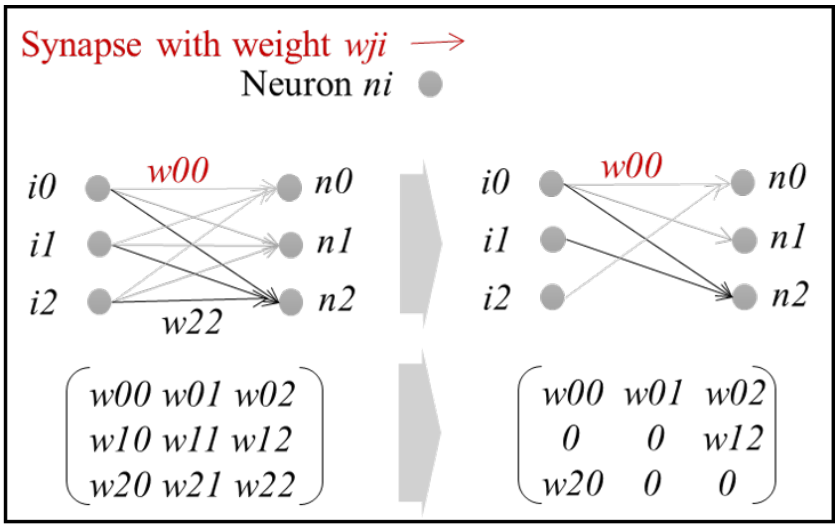


Figure 2.12: Synaptic pruning in the example of fully connected layers: Removal of individual synaptical connections

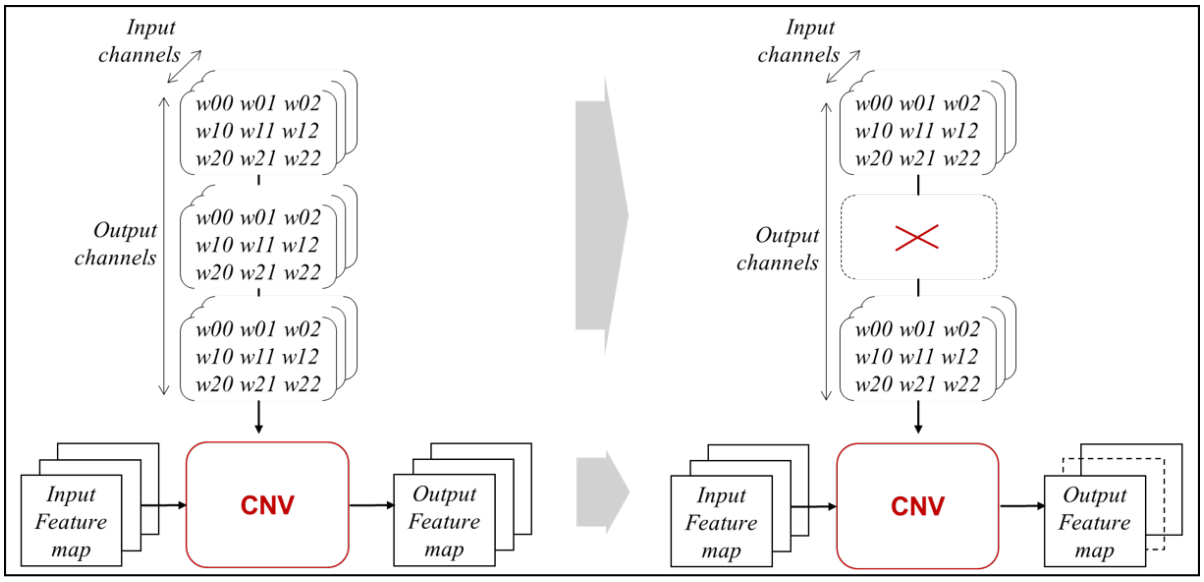


Figure 2.13: Filter pruning in the example of convolutional layers: Removal of complete filers and output channels

2.3 Hardware Architectures for Deep Learning

In the previous section, we considered the application requirements for deployment of CNNs and quantified their associated compute and memory demands, which are huge and growing beyond the limits to where standard silicon-based semiconductors can scale. The reasons behind the scalability challenges in the semiconductor industry are as follows: Firstly, as we approach the End of Moore's Law, transistor cost has been exponentially rising due to the rising chip design cost with shrinking technology nodes (as published by Xilinx and Gartner in 2011 already [83]).

Furthermore, with the end of Dennard scaling we encounter considerable thermal challenges, which saw the era of dark silicon emerge, where not all transistors within a single device can be switched on simultaneously due to overheating [84].

To overcome these challenges and provide sufficient compute capabilities, many disruptive approaches have been proposed. For example Cerebras [85] has introduced the concept of what they call **wafer scale computing**, where chips are built from complete wafers rather than individual dies, bringing with it substantial challenges in regards to manufacturing. Exploring the other dimension, foundries are investigating true 3D stacking as was presented at HotChips'2019 by TSMC [86]. Even analog computing [87, 88], quantum computing [89] and in-memory computing [90, 91] are investigated as well. All of these are on the speculative end of the spectrum.

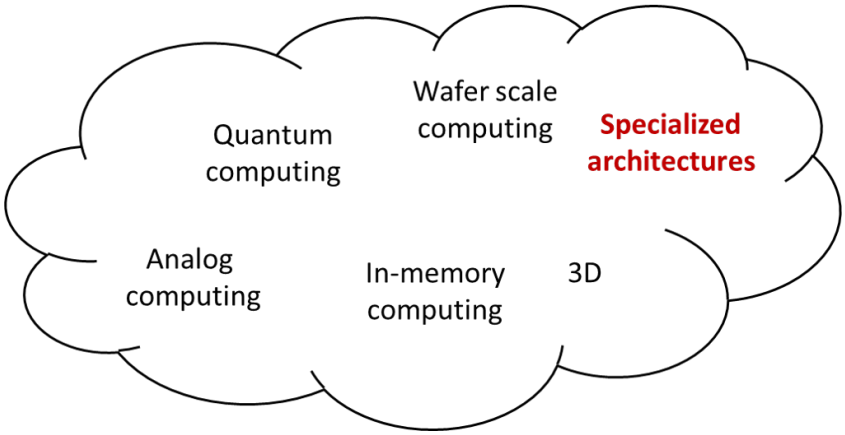


Figure 2.14: Innovative approaches for acceleration of CNN workloads

Less risky approaches focus on moving away from traditional von Neumann architectures, using specialization of compute architectures to provide the necessary performance scaling and energy efficiency. Due to the specialization, the devices become increasingly heterogeneous. A huge range of devices has emerged that all try to address this problem in different ways, whereby the key challenge is: How do we loop transform and unfold the algorithms best to maximize data reuse and compute efficiency, minimize memory bottlenecks, limit power consumption while meeting real-time requirements? We restrict our benchmark to the devices that can be realistically leveraged today, and thereby focus on **specialized architectures**. These build the focus of this chapter. It is important to remember though that these speculative approaches, as discussed above, will materialize in the future and a future benchmark will have to deal with an even greater diversity of hardware platforms.

We begin with a taxonomy of these hardware architectures, and discuss their relevant

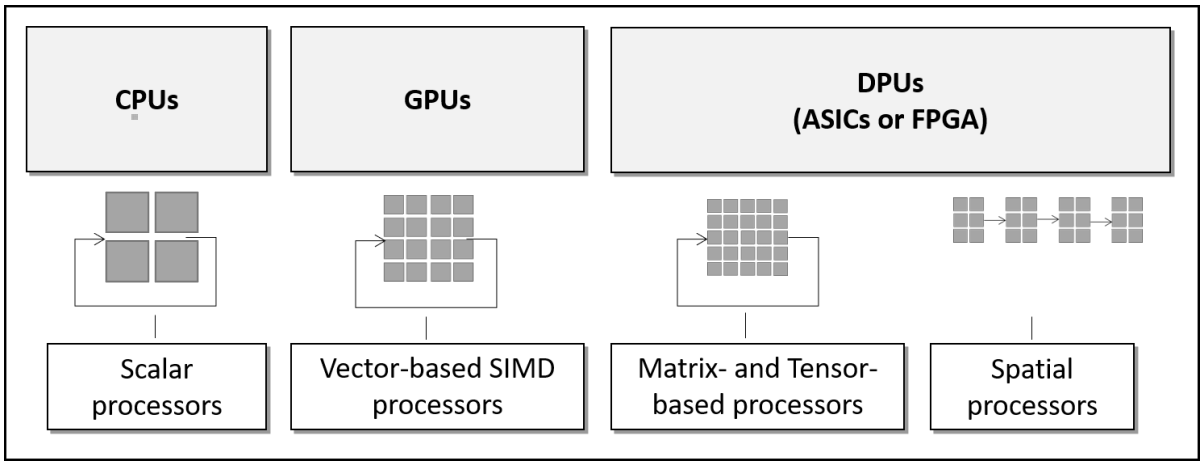


Figure 2.15: Taxonomy of compute architectures, differentiating CPUs, GPUs and DPUs

characteristics when it comes to acceleration of machine learning workloads. This is essential to understand how they will differ in their execution behaviour, what it takes to leverage their unique features and how they can potentially benefit from previously introduced optimization techniques. We will also discuss deployment options that are unique to specific architectures, and may or may not bring additional benefits to CNN inference. At last, we briefly touch on other considerations such as form factors. All of these stipulate another form of design requirements for a benchmarking methodology.

2.3.1 Taxonomy of Compute Architectures for Deep Learning

A broad range of hardware architectures to deploy machine learning algorithms exists today. We can broadly classify them by the following criteria:

1. Basic type of compute operation
2. Inherent support for specific numerical representations
3. External memory capacity (which is mostly relevant for training workloads) [3]
4. External memory access bandwidth
5. Power consumption in form of thermal design power (TDP)
6. Level of parallelism in the architecture and the degree of specialization

[3]For comparison, we treat HBM and HBM2 as external memory as it is used in the same way as DDR4 or GDDR memory.

As is shown in Figure 2.15, we classify the compute architectures into scalar processors, **CPUs**, vector-based processors, **GPUs**, and so-called deep learning processing units (**DPUs**), although realistically these categories blend to some degree. DPUs are specialized for this application domain whereby we distinguish the more generic matrix- or tensor based processor and a spatial processing approach. DPUs can be implemented with either ASICs or Field Programmable Gate Arrays (FPGAs). All of these architectures will be discussed individually below.

CPUs

CPUs are widely used for ML applications, and are viewed as largely serial or scalar compute engines (even though high-end variants for cloud deployment may have up to 10s of cores). They are optimized for single thread performance, with implicitly managed memory hierarchies (including three levels of caches), and support floating point operations (FP64 and FP32) as well as 8bit and 16bit integer formats with dedicated vector units in most recent variants. Theoretical peak performance tops at 6.8TOPs for FP64 assuming boost clock speed (Cascade lake, 56 cores, 3.8GHz). External memory is currently primarily leveraging DDR4 memory banks with large capacities: Intel's Cascade Lake offers up to 4.5 TebiByte (2^{40} Bytes) which is beyond what any of the other device categories can offer. Access is at maximum speed through high end hardened memory controllers, offering 282GBps bandwidth (for example Cascade Lake with 12 DDR4 channels). In regards to memory bandwidth, this is overall at the lower end of the spectrum, however in many application contexts, this can be compensated through the sophisticated multi-level memory hierarchies. Regarding power consumption, CPUs are at the upper end of the spectrum with high end devices range up to 400Watts (Ws) [92]. In the embedded space, ARM processors provide generally popular solutions, in particular when performance requirements are very low and when much alternative functionality is required that is not supported by the specialized device variants. In particular the Ethos family of processing cores is specialized for CNN workloads and as such listed under the DPU category below. Advantages of CPUs are in particular the generality of the hardware, as well as the ease of programming where design environments have matured over centuries. As expected this comes at the cost of lower peak performance and less efficiency compared to the more specialized device families. In regards to quantization, CPUs can only leverage this optimization technique for INT8 and INT16 if supported.

GPUs

GPUs are SIMD-based vector processors that support smaller floating point formats (FP16) natively, most recently fixed point 8bit and 4bit integer formats, and have a mix of implicitly and explicitly managed memory. NVIDIA GPUs are some of the most popular hardware targets for machine learning, and newer families of chips have been introduced to specifically accelerate this workload, with AMD not far behind. Latest devices in NVIDIA's Volta and Turing architecture families, introduced in 2018 and 2019 respectively, offer up 130TOPs in FP16 which is beyond the capabilities of latest CPU generations. As such they are amongst the highest performant devices in the market for the acceleration of DNNs as they can exploit a high degree of parallelism inherent to this application with increasingly specialized features. For example, NVIDIA's Volta is the first generation to incorporate tensor cores as a new feature, as well as improved FP32 and FP64 support for training in a data center setting [93] and also introduced a deep learning accelerator (DLA) in their embedded devices to further reduce power consumption. This specialization brings additional challenge for their usage and with that for benchmarking as there are up to 3 distinct execution units now, namely CUDA cores, tensor cores and the DLA which don't operate on the workload simultaneously, at least not easily or by default. We therefore don't sum up the peak performance of different execution units but use only the maximum. AMD announced the Vega GPU [94] with new deep learning instruction set operations, with the goal of obtaining parity with NVIDIA's high-end Tesla V100 datacenter GPUs, as well as the most recent EPYC family supports customized instructions for deep learning [95]. Both companies offer also low power GPUs for the embedded space, namely the AMD Vega mobile GPU [96] and NVIDIA Jetson TX2 [97] and AGX family [98].

In regards to memory, GPUs leverage specialized and highly pipelined GDDR memory, which reduces capacity, but offers much higher bandwidth (up to 732GBps). With the NVIDIA's Turing family, latest devices include HBM2 DDR memory stacks [99], which scales the memory access bandwidth to 1TBps and beyond. Again this is particularly important to address the needs of training workloads. For the same reason, some of the DPUs introduce HBM2 too, as discussed below. In regards to power consumption, GPUs are high as well, and will consume up to 345Ws.

One general challenge for GPUs is that they need to leverage input parallelism to achieve high utilization of their large compute arrays. Therefore inputs need to be grouped before execution, typically referred to as batches, which has adverse affects on end latency. This is

discussed in more detail below in Section 2.3.3. Further, GPUs are relatively high in power consumption. Regarding quantization, again support is limited to the inherent datatypes, which are INT4 at best in the context of NVIDIA's Turing family, and INT8 for many of the others. Finally, the corresponding software environments for GPUs, while not on the same level as CPUs, have matured significantly and provide increasingly ease of use.

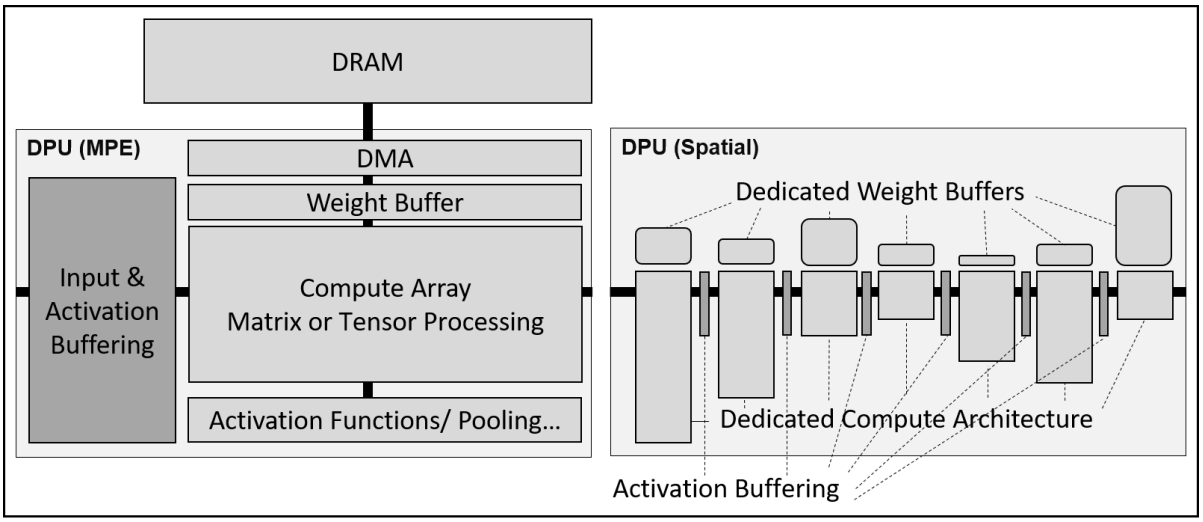


Figure 2.16: DPU architectures: Matrix of Processing Engines (MPE) on the left, and spatial architecture on the right

FPGAs and ASICs

FPGAs and Application Specific Integrated Circuitss (ASICs) customize hardware architectures to the specifics of a given application. They can be adapted in all aspects to suit a use case's specific requirements. This includes their IO capability, their functionality, or even to suit specific performance or efficiency targets. FPGAs can be reprogrammed whereas ASICs are fully hardened. This flexibility allows to amortize design costs of the circuit across many applications, but comes at the expense of hardware resource cost and performance.

FPGAs are a popular choice for acceleration of CNNs. Traditionally, an FPGA compute fabric consists of a sea of lookup tables (LUTs) which are interconnected through a programmable interconnect. Latest generations host millions of LUTs. Furthermore, the fabric is interspersed with specialized hardened compute blocks (DSPs) which accelerate n-bit multiply accumulate operations (MACs), as well as SRAM blocks. The latter are refered to as block RAMs (BRAMs), which hold 36kbits, and Ultra RAMs (URAMs) which store 288kbits. More recent FPGAs generations, combine multiple FPGA dies, referred to as super logic regions (SLRs), and leverage

a silicon interposer to provide connectivity between SLRs. This technology is referred to as stacked silicon interconnect (SSIT) and helps scale device capacity.

DPUs

As mentioned at the beginning, the term DPU (short for deep learning processing unit) refers to a new type of compute architecture, specialized for the acceleration of CNNs. DPUs are customized for these type of applications in a number of ways: types of operations supported, direct support of tensors or matrices, inherent data types and supported numerical representations, macro-architecture, explicitly managed and specialized memory hierarchies, and which levels of parallelism they exploit (input, output pixel, IFM, OFM, bit, and layer and branch parallelism as was introduced in the first part of this chapter. We differentiate two types of DPUs, which can be implemented with both ASIC technology and FPGAs.

DPU (MPE). The first type, as shown in the left side of Figure 2.16, consists of a Matrix of Processing Engines (MPE) that operates on matrices or higher dimensional tensors. The processing engines can be trivial Multiply Accumulate (MAC), vector processors or more complex Very Large Instruction Word (VLIW) cores which can support concurrent execution of different instructions. Popular examples in this category include Google's Tensor Processing Unit (TPU). Introduced in 2016 [100], it was originally designed to accelerate Google's TensorFlow framework. The first generation supported integer arithmetic with a massively parallel INT8 matrix multiply engine. The second generation TPU was announced in May 2017 [15], and the third generation in May 2018 [101]. These newer chips boast improved memory performance as well as support for floating point specifically aimed at training. There are a number of startups introducing custom hardware which fall into this category: Within the cloud space, there are Graphcore, Groq, and Wave Computing. Within the embedded space, where the design constraints are even more stringent, we find even more solutions, as are listed in Table 2.5. Most are secretive about the details of their designs. Intel is investigating several custom accelerators and has for that purpose acquired a number of startups, namely Nervana, Habana and Movidius. Fathom [102] is Movidius' ultra low power Neural Compute Stick (NCS) which operates at about 1W. Also, ARM offers specialized CNN processors in form of their Ethos family, boosting performance up to 4TOPs with support for INT8 and INT16 datatypes.

As mentioned above, DPUs provide specialized datatypes to execute heavily quantized, re-

duced precision CNN implementations. At the extreme, binarized neural networks which are very high throughput at extremely low power are exploited in the following ASICs: BinarEye [103], BNN Custom Fabric [104], and IBM AI Accelerator [105]. Finally, Lattice has announced binarized neural network libraries targetting low power FPGAs and achieving 1TOPs/W [106]. Custom floating point representations are also considered. For example, Microsoft's Brainwave project [107] uses this approach with the aim at applying FPGAs to CNNs at datacenter scale. However, typically the hardened versions in form ASIC only support INT8, as lower precisions could potentially limit their application scope. FPGA-based DPU-MPEs implementations such as Xilinx's xDNN are less constrained and in principle can be customized as needed.

Similarly to the GPU, but perhaps to a lesser degree, DPUs leverage input, IFM and OFM parallelism, which requires buffering of inputs and may have adverse affects on latency too. A particular challenge arises in the context of software environments, which differ for all vendors and are less mature than what we have observed for CPUs and GPUs. Typically, they are limited to support execution of very specific layer types (sometimes even restricted in regards to parameter ranges) and neural networks. A given and increasingly expanding modelzoo is supported, however many limitations exist in the form of layer types and are provided as black box solutions which lack transparency and flexibility. This will become apparent in the experimental chapter of the thesis.

In summary, through their specialization, these implementations minimize hardware cost, maximize performance and optimize efficiency by exploiting specific precision arithmetic with a specialized instruction set and customized memory systems. However in order to gain the performance advantage the algorithms need to be adapted to leverage these features.

Spatial DPUs. The second type of DPU leverages spatial acceleration and exploits layer and branch parallelism, as was introduced in the first part of this chapter. To that extent, the hardware architecture is even further specialized to the specifics of a given CNN topology. This is visualized in the right side of Figure 2.16. The hardware architecture actually mimics the given CNN topology and the inputs are streaming through the architecture. Every layer is instantiated with a dedicated compute data path. Each layer has a dedicated weight buffer, and activation buffers in between layers are FIFOs of minimal size. They buffer just enough data to feed the next set of convolutions in the next layer. This is substantially more efficient compared to the first type of DPUs or GPUs and yields reduced latency. DPUs and GPUs generally perform a layer by layer compute, where a sequence of images has to be buffered in

order to extract maximum compute out of the platform (input, IFM and OFM parallelism). For this the device buffers a batch of images before computing the first layer of all images. Then all intermediate results are buffered, and then the next layer is computed and so on. Hence the latency is heavily dependent on the size of the input batch. As a result, spatial DPUs have an advantage in regards to latency. This level of customization is only possible with programmable hardware architectures with FPGAs, as they can adapt the hardware architecture for different use cases. This wouldn't make sense in the context of an ASIC accelerator, as that would yield the ASIC only capable of accelerating one specific topology, which would be far too restrictive in scope. The limitation in spatial architectures is the scalability with numbers of layers. Each layer comes at a resource cost overhead and there is a maximum number of layers that can be created within a single device. As a result some extremely deep CNNs might not be able to fit into a single device. Microsoft's Brainwave project leverages spatial computing and overcomes this limitation with a distributed approach [107]. Once a spatial acceleration has been leveraged and the architecture is specialized for a very specific CNN, then the architecture can be further customized in regards to minimum precision, supporting only the bits as needed per layer of the CNN thereby achieving even higher performance and efficiency, while with the first type of DPU, the hardware will support the maximum precision that is required over the whole network. In regards to customized precisions and spatial architectures, we have pioneered the first binarized neural network accelerators [34, 40] and provided many proof points for customized reduced precision implementations [11, 14]. This flexibility comes at a cost in form of programming complexity and they are extremely difficult to characterize in general, as the performance characteristics depend on the specifics of the hardware architecture that has been implemented.

Further Variants

Others exploit sparse computing engines, such as EIE and its successor ESE [78], SCNN [108], Cnvlutin [109], Cambricon-S and Cambricon-X [110]. These are the only architectures which can benefit from irregular sparsity.

Finally, another dimension to customization of precision is to optimize over the execution- or run-time of a CNN. In other words, beyond using statically fixed reduced precision, where the hardware operates with a fixed precision for all variables, some approaches explore run-time configurable bit precision which allows for the exploitation of bit-parallelism in the arithmetic.

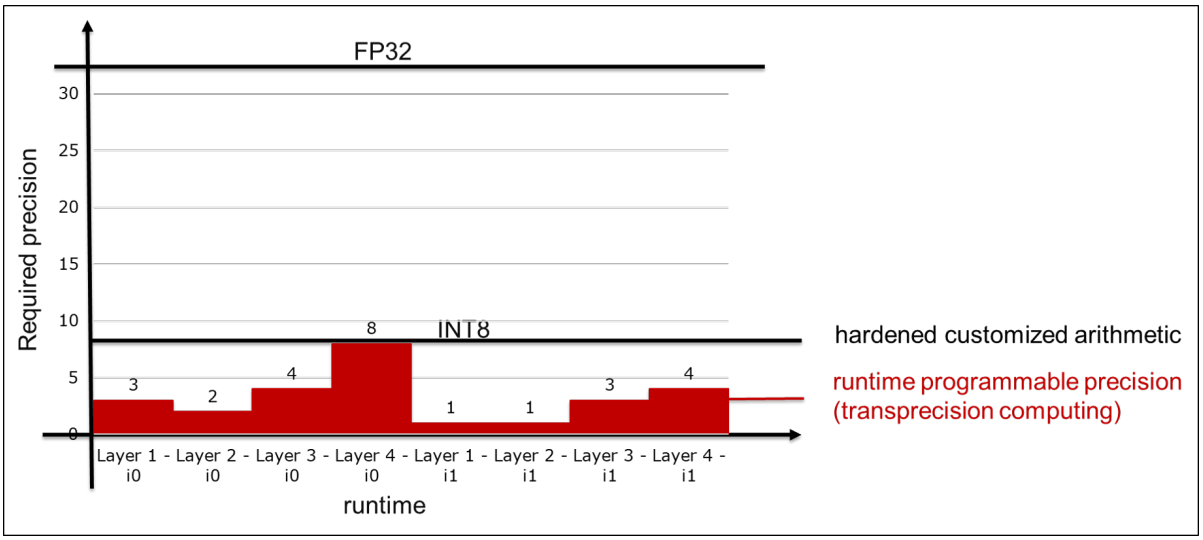


Figure 2.17: Run-time programmable precision.

This is also referred to as transprecision computing and illustrated in Figure 2.17. On the x-axis, we show the execution time of a CNN, where we first execute all layers (layer 1, layer 2 and so on) for input i0. Then we progress to the second input i1 and so on. The y-axis annotates the minimum precision required. In the shown example, layer 1 for i0 requires 3bits, while layer 2 for i0 requires 2bits. This can vary for every input. In case of a FP32 implementation, marked by the top line, the area between the red boxes and line would be representative of the wasted computation. When leveraging hardened customized arithmetic, for example INT8, then the wasted computation is significantly reduced, but there is further scope for optimization. This is what can be exploited with run-time programmable precision and is effective with **bit-serial** implementations. For example Umuroglu et al. [111] demonstrate with BISMO that bit-serial can provide highly attractive performance with minimal overhead on FPGAs, while Judd et al. show the same is true for ASICs with their prototype ASIC called Stripes [112]. While this concept can be applied to both MPE and spatial architectures, it makes most sense for MPEs.

2.3.2 Example Platforms for Cloud and Embedded Systems

We have carried out extensive research on available hardware platforms in this space and collected and summarized them in Tables 2.4 and 2.5 with published performance and power. These tables cover both high-end platforms for cloud deployment as well as devices targeted for IoT.

Table 2.4: Hardware architectures for cloud systems with theoretical performance

Platform	Num. Choice	Throughput [TOPs]	Mem BW [GBps]	Power [W]	Performance/Power [TOPs/W]
CPUs					
Intel CascadeLake 92xx (56cores) [92]	FP64	6.8	282	400	0.017
Intel CascadeLake 82xx (28cores) [92]	FP64	3.5	141	205	0.017
AMD Epyc 7742 (28cores) [95]	FP64	3.5	205	225	0.016
GPUs					
Quadro RTX 6000 [99]	FP16	130.5⋆	624	260	0.5
NVIDIA V100 [93]	FP32	14	250	300	0.06
NVIDIA V100	FP16	112⋆	250	300	0.45
NVIDIA P100 [113]	FP32	8	732		
NVIDIA P100	FP16	16	732		
NVIDIA P40 [114]	INT8	47	200	346	0.24
NVIDIA P4	INT8	22	60	192	0.37
AMD Vega10 [115]	FP32	13.7	484	345	0.04
TPUs					
Google TPUv1 [100]	INT8	92	75	34	1.23
Google TPUv2 [15]	FP16	45		600	
Google TPUv3 [101]	FP16	90			
ASIC DPU (MPE)					
Graphcore	Custom	224		300	0.75
Groq	unknown	400			8
Nervana	custom16	55			
Wavecomputing 1DPU	INT8	181		271	0.7
FPGA-based Spatial DPU					
Xilinx VU9P	2b/8b	93.00	88	100	1.06
Xilinx VU9P	2b/4b	139.88	88	100	1.59
Xilinx VU9P	2b/2b	192.52	88	100	2.19
Microsoft Brainwave Stratix X [107]	FP8	90		125	0.72

Performance of Tensor Cores ⋆

Table 2.5: Low power hardware architectures and theoretical performance

Platform	Num. Choice	Throughput [TOPs]	Mem BW [GBps]	Power [W]	Performance/Power [TOPs/W]
CPUs					
ARM Cortex-A53 using gemmlowp; Ultra96	INT8	0.192	4.26		
Bitserial Cortex-A57; Jetson TX1 [116]	BIN	0.09			0.019
GPUs					
NVIDIA AGX (30W) [98]	FP32	3.59		30W	0.12
NVIDIA AGX (30W) [98]	FP16	7.19		30W	0.24
NVIDIA TX2 (MaxP) [97]	FP32	.575	59.7	15.0	0.038
NVIDIA TX2 (MaxP) [97]	FP16	1.15	59.7	15.0	0.077
ASIC DPU					
Movidius Myriad 2 [102]	INT8	.15		1.2	0.125
Movidius Myriad X [117]	INT8	1		1	1
Kalray MPPA Turbocard3 [118]	FP32	1.6		110	0.014
BinarEye [103]	BIN	0.09 - 2.8⋆			230†
BNN Custom Fabric [104]	BIN	1.4		0.6	2.3
Stripes Bitserial ASIC [112]	BIN	128.5			4.3
IBM AI Accelerator [105][4]	BIN	12			
Eyeriss [60]	INT16	0.084		1.17	†
ARM ML Processor [119]	unknown	4.6			3
DianNao [120]	INT16	0.452	120	0.485	0.93
EIE(28nm) [121]	INT4	3 (0.102 sparse)	2.36	1.27	2.4 (0.08 sparse)
Cambricon-X [110]	INT16	0.544			
FPGA DPU					
Lattice SenseAI [106]	BIN	1.4		0.6	2.3
BISMO biserial on PYNQ [111]	BIN	6.5		4.64	1.4
FINN on ZC706 [34]	BIN	11.6			0.408
ZCU104 (Deephi-666MHz)	INT8	4.60	19.2		
ZCU104 (Theoretical-775MHz)	INT8	5.36	19.2		
GX1150 on HARPv2 [122]	BIN	0.041			0.85

Measured ⋆

Chip level power consumption only †

Comparison

Roughly comparing these architectures, we can distinguish between them in regards to their theoretical peak throughput, latency, power consumption, external memory capacity and memory access bandwidth, their degree of specialization of the hardware towards the workload and the associated ease of use. CPUs count as general purpose devices, even though they also show first signs of specialization in regards to CNN acceleration with specialized instructions in their vector processing units. For example Intel's AVX-512 now offers so-called Vector Neural Network Instructions (VNNI) [123]. However, overall they show the least degree of specialization. CPUs provide high throughput, are the highest in regards to power consumption and memory capacity and the easiest to use. GPUs increasingly specialize for AI workloads and support increasingly reduced precision types (INT4 with the latest Turing family of GPUs) and specialized tensor processing cores. They offer together with DPUs highest performance and excel at external memory bandwidth but also at a high power footprint and very high latency cost. DPUs, due to their increasing degree of specialization, offer top in class in regards to throughput, energy efficiency and with spatial DPUs offering lowest latency. However, increasing hardware complexity is accompanied with increasing complexity in the design entry, which reduces ease of use. We have illustrated these characteristics in Figure 2.18. On the left side, the heatmap visualizes through colour the qualitative characteristics and highlights the strong aspects of each architecture. The number reflects a ranking, whereby higher is better. Similarly, the radar chart depicts the compromises in each approach and illustrates that there is not a single architecture that excels in all aspects.

	CPUs	GPUs	DPU-MPE	DPU-Spatial
Throughput	2	4	4	3
Latency	2	2	3	4
Power	1	3	2	4
Memory Capacity	4	2	3	3
Memory Bandwidth	3	4	4	2
degree of specialization	1	2	3	4
ease of use	4	3	2	1

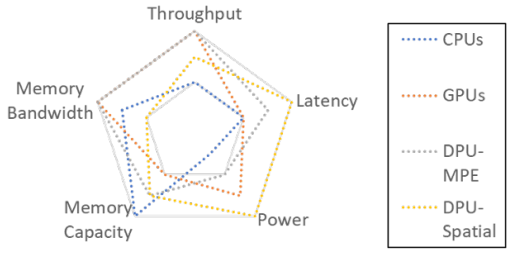


Figure 2.18: Qualitative comparison of hardware architectures

Trends. As already indicated, with newer generations, the boundaries between different hardware architectures are blurring. CPUs are increasingly incorporating vector processing

units and support for reduced precision integer formats. GPUs are adding tensor processing units, and the TPU now supports floating point operations. FPGAs can support any of the above configurations with explicitly managed memory. They are the most flexible of all target hardware, and can be configured to support any numeric representation.

2.3.3 Deployment Options

Many of the described hardware platforms offer different deployment options in order to support different compromises between throughput, latency and power as indicated in Figure 2.19. These are discussed in more detail within this section. In our benchmark, we ensure to cover a systematic exploration of all of the provided deployment options regarding all figures of merit to ensure a fair comparison and thorough understanding of the design compromises.

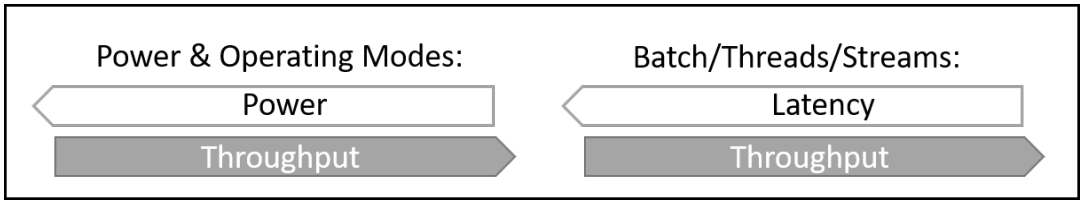


Figure 2.19: Compromises between power, throughput and latency with deployment options

Operating and Power Modes. Many of the chosen platforms offer different power or operating modes which provide a compromise between power consumption and achievable throughput, for example by regulating clock speed or disabling parts of the circuit. A specific example, is the TPU Coral stick from Google which can operate with a fast or a slow clock. The NVIDIA GPU Jetson TX2 platform can run in either maxn, maxp or maxq modes. Maxn is the high performance mode with highest power consumption. Maxp is the most efficient mode, with lowest power but also lowest performance, and maxq is a compromise between maxn and maxp. Similarly, the NVIDIA AGX offers a maxn, 10W, 15W or 30W mode. Examples of this behaviour are shown in Figure 2.20 indicating substantial differences in regards to power and performance for different modes, where naturally higher performance comes with higher power consumption.

Batch Sizes, Thread Counts and Stream Sizes. Many hardware platforms require increased batch sizes, thread counts or stream sizes in order to extract maximum performance out of a

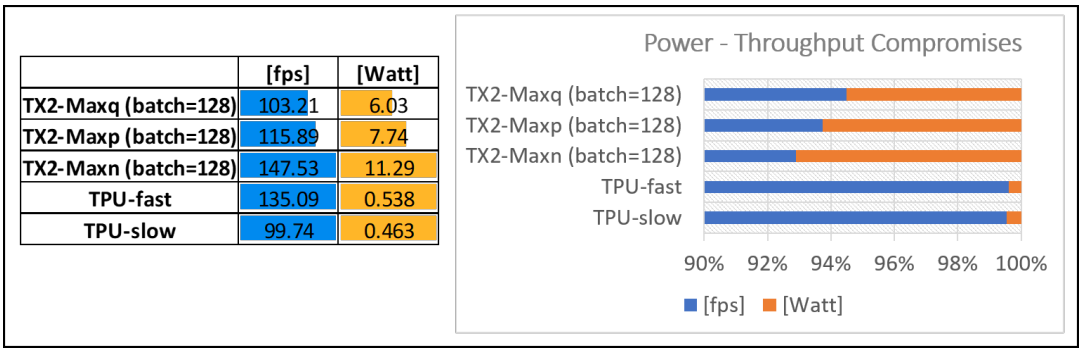


Figure 2.20: Power versus performance for GoogLeNetv1: Different trade-offs are achieved in different operating modes

hardware platform and achieve high compute efficiency. However, this can result in substantially different latency as is shown in Figure 2.21. Layer-by-layer compute architectures including Google's TPU, GPUs such as Nvivida's TX2, and Intel's Neural Compute Stick NCS) have a larger than linear increase in latency with regards to batch size. Typically, a sequence of images has to be buffered in order to extract maximum compute out of the platform. The architecture buffers a batch of images before computing the first layer of all images. Then all intermediate results are buffered, and then the next layer is computed and so on. Hence the latency is heavily dependent on the batch size (or thread count). As the device utilization saturates with increasing batch size, peak performance is reached, and no further benefit can be achieved. This is illustrated by the measured results for the NCS and TX2.

Spatial architectures such as the dataflow implementations with FINN [34] have a fixed latency independent of input stream size which is determined by the length of the given pipeline and is fixed. This is visualized with the gray line in Figure 2.21 which represents the measurements on a spatial architecture. As such when only 1 input is processed (stream size=1) the pipeline is underutilized and throughput is low. Full throughput is achieved when the stream size saturates the pipeline, whereby the latency remains constant independent of the stream size. Finally, the Xilinx DPU example (implemented on a ZCU102 development platform), which deploys a vector of processing engine, utilizes thread counts rather than batches. Latency increases with rising thread count, however at a much gentler slope than batch sizes for GPUs. Example behaviour of latency and performance for a spectrum of batch sizes, thread counts and stream sizes is shown in Figure 2.21 for the above mentioned devices for ResNet50v1 implementations (RN50) and a VGG16 derivative CNN, currently abbreviated as CNV100%. A more systematic exploration can be found in Chapter 7.

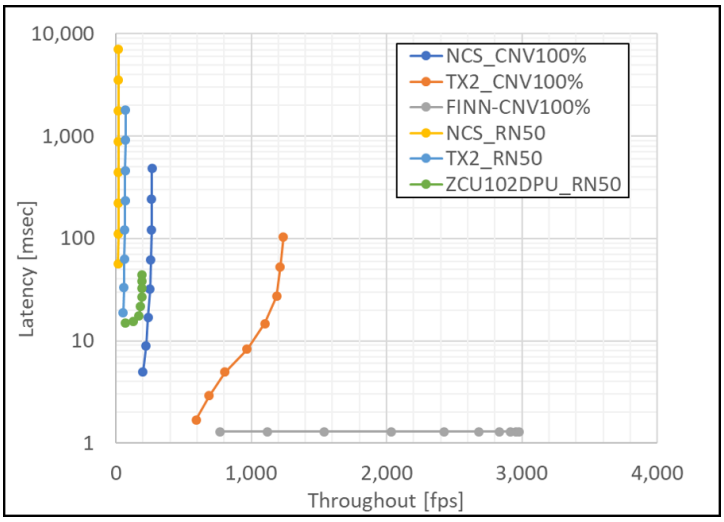


Figure 2.21: Latency versus throughput for different batch sizes, thread counts and stream sizes. For most architectures increasing the latter improves throughput at the cost of latency, with exception of spatial architectures, where latency remains constant.

2.3.4 Other Considerations

Finally, we would also like mention different options in regards to form factors. There is a spectrum of form factors available for all of these accelerators and it is important as it poses additional challenges in regards to ensuring fair measurements. Cloud CPUs are only deployed in motherboard sockets, while GPUs and DPUs (including FPGA-based ones) are typically available in PCIe accelerator form factors. The motherboards hold significant amounts of additional circuitry, which prevents us from measuring power of CPUs in isolation. Further there are specialized AI machines such as Nvidia's DGX-1 [124], however their focus is mostly on training which is beyond the scope of this thesis. Similarly, the TPU is available in customized form factors, so called pods, for Google's data centers, and no power metrics are available. In the embedded space, we see many customized platforms. We believe it's useful to distinguish between USB accelerator options (which are further constrained to USB power budget of 2.5W), and socket-powered boards. Most of these boards are not actually intended as the final product, but more as development or evaluation boards. The actually deployed form factor is heavily customized to the specific use case of the end customer. Similarly to the server space, depending on the complexity of the evaluation boards, power consumption might be inflated given collateral devices located on the same platform.

2.4 Concluding Remarks

This chapter contained two separate parts, the first summarizing the background of CNNs including a discussion of compute and memory requirements, popular optimization strategies and potential scope for parallelization, and the second part describing, analyzing and comparing popular hardware choices. In the first part, we provided key insights about neural networks, their unique requirements and challenges that need to be addressed in the benchmarking approach. We discussed the ever broadening scope of applications for CNNs from image classification to natural language processing with their corresponding figures of merit. We considered the relevance of training datasets and exposed the breadth of CNN topologies together with a discussion around the rapid speed of change in their topologies. Additionally, we provided more of an in-depth look at popular computational patterns, including a quantification of the enormous compute and memory requirements and discussed popular optimization techniques, with specific emphasis on quantization and pruning as the most popular schemes. Together, this illustrates the complexity of the design space and formulates the design objectives for our benchmark in regards to the application.

In the second part of this chapter, we analysed three categories of hardware architectures that are leveraged for CNN inference, namely common CPUs, SIMD based vector processors such as GPUs, and DPUs which are specialized architectures for acceleration of deep learning workloads. For DPUs, we distinguish between tensor processors which leverage a matrix of processing engines and spatial architectures which can be further specialized for specific topologies using FPGAs. CPUs are the most general solution but high in power. GPUs and DPUs offer highest performance, whereby GPU are more expensive in regards to energy cost. Spatial DPU architectures excel at latency and provide highest compute efficiency through maximized customization. CPUs, GPUs and DPUs (MPE) use a sequential layer by layer compute model whereas spatial DPUs execute all layers of the network concurrently. Hardened topologies in form of ASICs, CPU and GPU offer a fixed set of native dataypes, whereas FPGAs can adopt any precision and numerical representation, which provides utmost flexibility and leverages optimization with quantization to the maximum, whereas hardened approaches need to default to the next higher supported precision where the reduced precision variable can be embedded. However the programmability in the FPGA fabric also comes at a speed and energy cost. All architectures can benefit from coarse-grained pruning optimization techniques. Only

sparse execution engines can benefit from irregular pruning, such as synaptic pruning. We also discussed the various deployment options. Many devices offer different power and operating modes as different compromises between throughput and power consumption to adapt to the potentially very different optimization targets of different application settings. Similarly, batch sizes, thread counts and stream sizes offer another compromise in regards to throughput versus latency. Again this is to facilitate a spectrum of different use cases. This creates a unique set of requirements and the whole range of deployment options need to be considered as part of the benchmark in order to create a complete picture of the design space. In the next chapter, we'll take a look at benchmarking requirements in more general terms and study related work.

3 Considerations in Benchmarking & Related Work

3.1 Introduction

In the previous chapter, we considered the unique requirements of the application space in regards to algorithms and hardware architectures. The final missing piece to designing a benchmarking methodology, is looking at key components that constitute a benchmarking suite, understanding the general characteristics we should be striving for, and understanding the specific challenges that need to be addressed within the machine learning context. This will be addressed in the following sections. In addition, we include here a thorough review of related work. Importantly, during the time of this thesis, many other benchmarking attempts have emerged. This validates our problem statement and confirms the need for a specific benchmarking methodology in this space. While this created some overlap, substantial differences remain. We include a detailed comparison with particular focus on MLPerf, which emerged after QuTiBench, and is the largest and most significant industry wide effort in this space.

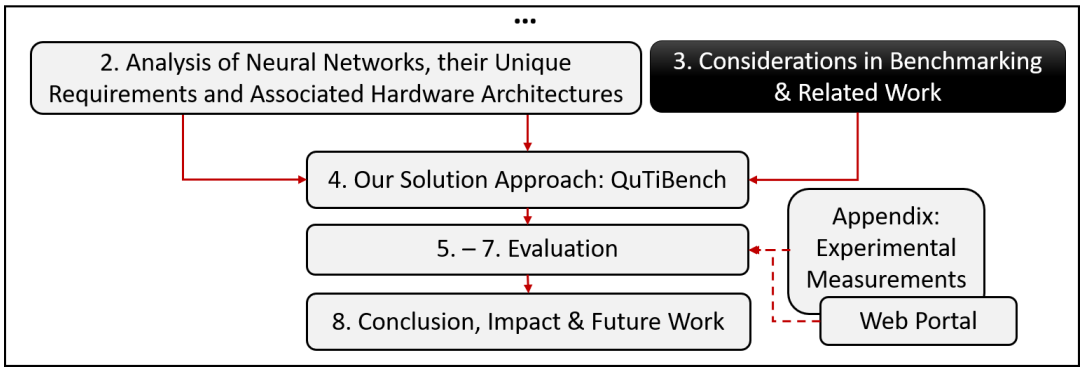


Figure 3.1: Location within the thesis

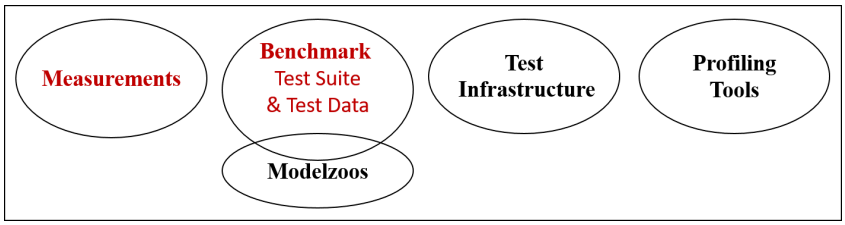


Figure 3.2: Minimum components of a benchmark plus potential collateral

3.2 Key Components, Characteristics & Challenges of a Benchmark

To help with the design of a benchmarking methodology, we start by taking a look at what constitutes a benchmarking suite in general. What are general characteristics we should be striving for and what are the specific challenges that need to be addressed specifically within the machine learning context.

A benchmark can be defined as a set of standards used for evaluating performance or level of quality. A more practical definition implies that the "set of standards" is supplied in the form of a very specific set of executable tests and measured regarding a specific set of figures of merit. The tests combined with the defined figures of merit are the essence of the benchmark.

Sometimes additional items are included such as performance analysis or profiling tools which can help shed light on system bottlenecks. Test infrastructure or a testbed can be provided to ensure reproducibility. This makes particular sense when specialized and not easily available hardware systems are involved. Data management can be handled together with the benchmark suite and stored in an accessible location as for example with DAWNbench [125], MIT's Eyeriss project [126] and the Request tournaments online score card [127]. We differentiate profiling tools, test infrastructure, and measurements from the actual benchmark test suite (see Figure 3.2) which encompasses executable code, associated test input data and definitions of figures of merit. Also, somewhat related to benchmarking are modelzoos, such as OpenAI Gym [128] and rllab [129], which are selections of sample codes. They are not necessarily aiming to be representative, and typically include simplified implementations to teach concepts. In this thesis, we focus on the benchmark suite and measurements only.

To bring maximum benefit, the following characteristics are essential which are discussed in greater detail below:

- Representative of common workloads
- Supportive of algorithmic modifications
- Objective and reproducible
- Portable to heterogeneous hardware systems
- Complexity vs accuracy trade-off
- Adaptive "living" benchmark supported by industry and academia

3.2.1 Representative

Benchmarks need to be **representative** of real world workloads. In machine learning, this requires breadth across a spectrum of applications, algorithms and computational patterns. For example Fathom [130] focuses on computational patterns, while TBD [131] investigates algorithmic breadth. Both of these aspects are important. Computational patterns are important to maximize insights into different hardware architectures. Application coverage is essential as it provides more holistic insights into system level performance which can be hard to predict given the emerging complexity of increasingly heterogeneous hardware systems. In particular, this is essential to provide scope for algorithm optimizations. Other benchmarks, such as Fathom, try to minimize the tests and focus only on a representative set of compute patterns. However this would eliminate the scope for algorithmic freedom.

3.2.2 Support for Algorithmic Modification

Algorithmic modifications are inevitable to extract best possible performance out of diverse hardware systems, for example to take advantage of caching and parallel hardware resources. Licht et al [132] quote up to 400x speed-up, but this is just one of an endless list of publications manifesting this fact. This is even more evident within the machine learning space, where software and hardware co-design is compulsory [133] for energy constrained compute environments. To support this algorithmic freedom within the benchmark suite, application coverage is essential as mentioned above. We correlate hardware performance independent of the algorithm back to application performance, which is equivalent to accuracy in this context. However, optimized performance alone is not sufficient, as not every system designer may be able to achieve it.

Ideally we would also reflect the out-of-the-box, naive performance. Both optimized and naive are representative of a specific hardware platform, and the difference gives a good indication of the development effort involved. We believe both should be part of the benchmarks and be captured together with development time or lines of code. We intend to incorporate this as part of future work. Specifically for neural networks, algorithmic modifications to be considered include quantization, compression, topological changes and pruning techniques, as was detailed in Chapter 2. In our evaluation we focus on pruning and quantization as the most popular optimization schemes.

3.2.3 Objective & Reproducible

To provide clear differentiation between marketing and scientific efforts, reproducible and objective results that do not favour any particular system configuration or hardware architecture are needed. **Reproducible** results are a key ingredient in the move towards **Open Science**, however, what does reproducibility actually entail? In the context of the plethora of esoteric AI accelerators, is it sufficient that an objective third party has validated the results? This is for example done in the context of the Collective Knowledge Framework [134]. Or does it imply that everyone on the planet should be in a position to reproduce the results if they had access to the system at a reasonable cost? Some hardware systems are too expensive; for example, a NVIDIA V100 may be beyond someone's budget. Other hardware choices are only available for rent, such as Google's TPU versions as part of Google cloud. As such it may be appropriate to define reproducible at a certain cost.

3.2.4 Portability

Portability is a challenging subject as specialized hardware architectures come with their own design entry languages and compiler tool stacks. The community is fragmented by a huge choice of frameworks including Caffe, Tensorflow, Mxnet, Theano, pytorch and Darknet. What is more, the prediction accuracy of a network depends on the choice of framework, since training data are passed through different preprocessing stages and numerical inaccuracies accumulate and manifest themselves as discrepancies. These inaccuracies are exacerbated by the characteristics of floating point arithmetic [135]. For example, the non-associativity of floating point arithmetic and differences in rounding operations, can accumulate to differences

in validation error [135]. As a result, models and frameworks are inherently tied together. Although new efforts on interchange formats such as ONNX [136], NNEF [137] and compatibility stacks such as TVM [134, 138] make strides in this domain, currently supporting hardware backends within each of the frameworks that is tied to the selected machine learning model, seems inevitable. There are three basic choices: The first is to constrain ourselves to exactly one framework as was done with Fathom [130]. Second, we could support all frameworks. However, given that we are dealing with different hardware backends, this causes an explosion in test infrastructure, as the number of tests multiplies with the number of frameworks. Furthermore, many specialized AI accelerators would not support integration with all frameworks. The final choice and probably the cleanest, is to support one of the intermediate neural network representations such as ONNX [136], NNEF [137] or TVM [138], which provide translation between all popular frameworks. However, this requires hardware vendor support, which is currently limited. This can be addressed in the future once standards and vendor tools mature.

3.2.5 Complexity vs Speed vs Accuracy

Speed of result is essential, as the key purpose of a benchmark is to provide faster insights than developing the full end-system. There is a trade-off between speed, benchmark complexity and the accuracy of the results. Benchmarks which provide application and algorithmic breadth may require a large number of tests thus making the benchmark suite inherently complex and limit its usefulness. If it takes longer to run the benchmark than the targeted end applications, the whole exercise may become useless. Sometimes it is important to have less accurate predictions at a faster rate, and, for different users, different trade-offs are acceptable.

3.2.6 Adaptive

As machine learning is a highly active research field where algorithms change fast, the benchmark suite should be adaptive and able to incorporate emerging popular algorithms, compute patterns and end applications.

3.2.7 Other Considerations

Workloads in the benchmark can be synthetic or real. Synthetic workloads allow the tailoring of tests to be optimized to expose certain characteristics and thus may not be representative. Real workloads may take longer to benchmark and may provide fewer insights and be less precise at exposing bottlenecks. With QuTiBench we propose a combination of the two.

Another important consideration is whether to adhere to FAIR guiding principles [19] for data management which ensures findability, accessibility, interoperability, and reusability for all experimental and theoretical datapoints through creating of a persistent identifier, a repository, a web portal [20], and creation of project- and data-level metadata. These are important to provide automated access and therefore helps maximise the potential impact on research in general as data becomes much more reusable.

3.3 Types of Benchmarks

We differentiate 3 types of benchmarks to clarify the differences in the various efforts in this space. Specifically, we differentiate between **ML benchmarks**, **performance benchmarks** and **NN system benchmarks**. ML benchmarks exclusively focus on accuracy. No consideration of computational cost is given. Performance benchmarks record performance regarding throughput (measured in processed inputs per second (ips), giga operations per second (GOP/sec) or tera operations per second (TOP/sec)), latency or response time in milliseconds (ms), and power consumption in Watts, however, completely ignore the performance in regard to the machine learning task. Performance benchmarks only look at hardware performance and are agnostic of the application. NN system benchmarks, as shown in Figure 3.3, lie at the intersection and combine both performance and accuracy. They are at the heart of what we are striving for.

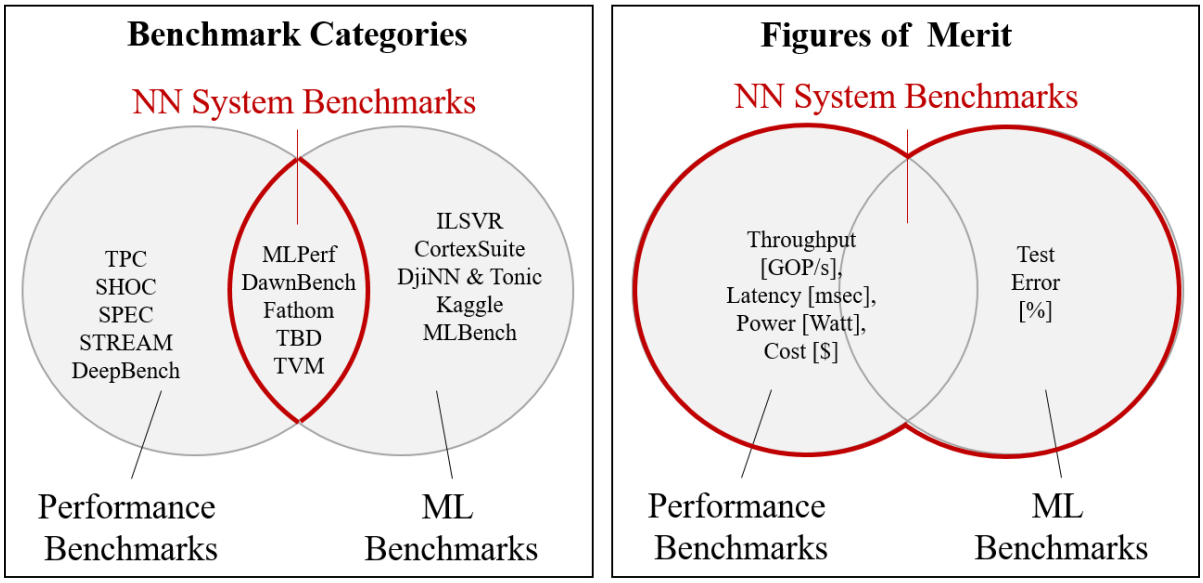


Figure 3.3: Categories of benchmarks (performance, ML and NN system benchmarks) and corresponding figures or merit

3.4 Related Work

We group the related work into the introduced benchmarking types.

3.4.1 ML Benchmarks

The Machine Learning community has defined its own benchmarks which have an exclusive focus on achieved accuracy independent of the required compute, employing ensemble techniques and multi-crop which in essence, linearly scale up the compute load per input data. The most popular of these is the ImageNet Large Scale Visual Recognition (ILSVR) Challenge [139]. The associated compute requirements are unrealistic, particularly when deployed in energy-constrained environments. CortexSuite [140] and BenchNN [141] are limited to measuring accuracy, where CortexSuite is constraint to perception and cognition while BenchNN shows the value of machine learning for approximate computing, based on 5 out of the 12 recognition, mining and synthesis applications from the PARSEC benchmark suite. DjiNN and Tonic [142] focus on deep learning tasks for warehouse scale computers including image, speech processing and natural language processing. While kaggle [143] isn't designed as a benchmark specifically, it hosts a portfolio of data science challenges where the machine learning community competes with the latest topologies and algorithms for highest accuracy. MLBench [18] compares human

derived learning algorithms against machine learning services from Amazon and Microsoft Azur.

3.4.2 Performance Benchmarks

In regards to benchmark for neural network performance, DeepBench [59] is probably the most successful at the state of this writing. DeepBench is a suite of microbenchmarks that measures and compares basic compute operations. Specifically, it benchmarks individually direct convolutions, matrix multiply, and a specific LSTM layer for single precision, half precision floating point and for some operations 8b fixed point integer datatypes on hardware architectures. It currently features cloud deployment and some embedded data points on raspberry pi and iphone. It captures the most popular compute patterns, however lacks support for lower precision datatypes, and exclusively investigates performance. As such it does not provide the mechanisms to tie algorithmic modifications back to the application level, nor provide insights into compute performance for reduced precision representations. As shown in Fathom [130], while these are dominant operations, they only represent a subset of possible operations that are needed to represent machine learning workloads. Furthermore, they do not capture data movement bottlenecks between layers, as well as potential bottlenecks around buffering state, as required for LSTMs for example, where capacity and access latency crucially impact overall speed. To address this issue, researchers from industry and academia contribute to another benchmark suite, MLPerf, that captures complete neural networks and with that system level bottlenecks, further discussed below.

There are more general, machine learning agnostic, hardware benchmarks such as TPC [17] for the data processing community, SHOC [144], SPEC [145] and STREAM [146]. SHOC looks specifically at how to benchmark heterogeneous hardware systems using OpenCL as design entry. Similar to our own efforts, SHOC deploys microbenchmarks combined with application benchmarks and is multi-tiered. SPEC includes a broad range of applications including graphics, MPI, mail servers, virtualization, and storage, and STREAM exclusively focuses on memory bandwidth. None are specifically designed for machine learning, and address the challenges of this application domain. gemmlowp [6], while it is not a benchmark, is specifically designed for matrix multiply operations; it includes low precision operations which may be suitable as a basis for implementation of part of our benchmark suite.

3.4.3 NN System Benchmarks and MLPerf

NN System Benchmarks combine representative machine learning workloads with figure of merit from the hardware performance benchmarks. **BenchIP** [147] is a benchmarking suite which has a broad set of machine learning tasks. Similar to our own efforts, BenchIP adopts a multi-tiered approach with micro- and macro-benchmarks. However, BenchIP does not support the theoretical layer, which we use to cover compute efficiency and track benchmarking results. BenchIP also doesn't cover level-2, namely stacks of layers, which we believe bring great merit in isolating bottlenecks in data movement and highlighting problematic dimensionality in tensors. Finally BenchIP does not offer the concept of comparison via pareto graphs which is essential to a) visualize the full scope of potential solutions within the design spectrum, and b) provide the necessary scope for algorithm optimizations matching the specifics of various accelerators. **Fathom** is probably the first attempt to provide a representative workload for benchmarking that has algorithmic breadth beyond convolution neural networks inference and includes example training and unsupervised learning such as reinforcement learning and recurrent models. However, Fathom does not address the spectrum of numerical representations. It also does not support heterogeneous hardware platforms. In regards to framework strategy, Fathom advocates a unified software package, relying on compatibility software stacks to emerge, and therefore only supports one framework, TensorFlow. Table 3.1 lists all machine learning tasks as covered by Fathom.

Table 3.1: Fathom coverage: applications, datasets and models

	Application		Fathom	
Learning Technique	**Domain**	**Task**	**Dataset**	**Model**
Supervised	**Vision**	**Image Classification**	ImageNet	ResNet
			ImageNet	VGG, AlexNet
	Vision	**Object Detection**	-	-
	Vision	**Semantic Segmentation**	-	-
	NLP	**Machine Translation**	WMT-15	Seq2Seq
	NLP	**Machine Translation**	-	-
	NLP	**Speech Recognition**	TIMIT	DeepSpeech
	NLP	**Sentiment Analysis**	-	-
	NLP	**Language Modeling**	babI	Memory Networks
Unsupervised	**Vision**	**Feature Extraction**	MNIST	Autoencoder
	Vision	**Adversarial Learning**	-	-
	Recommendation	-	-	-
Deep Reinforcement Learning	**Game**	**Go**	-	-
		Atari ALE	Atari ALE	Deep Q

With a primary focus on benchmarking for training and achieving application coverage

rather than algorithmic breadth, **TBD** [131] adopts some of the concepts introduced in Fathom. It supports more frameworks and datasets and covers a range of applications, including image classification, machine translation, object detection, speech recognition, adversarial and deep reinforcement learning. Also it offers more in-depth insights with profiling infrastructure. Table 3.2 lists all machine learning tasks as covered by TBD.

Table 3.2: TBD coverage: applications, datasets and models

	Application		TBD	
Learning Technique	**Domain**	**Task**	**Dataset**	**Model**
Supervised	**Vision**	**Image Classification**	ImageNet1k	ResNet50
			ImageNet1k	InceptionV3
	Vision	**Object Detection**	Pascal VOC 2007	Faster R-CNN
	Vision	**Semantic Segmentation**	-	-
	NLP	**Machine Translation**	IWSLT15	Seq2Seq
	NLP	**Machine Translation**	IWSLT15	Transformer
	NLP	**Speech Recognition**	Librispeech	DeepSpeech2
	NLP	**Sentiment Analysis**	-	-
	NLP	**Language Modeling**	-	-
Unsupervised	**Vision**	**Feature Extraction**	-	-
	Vision	**Adversarial Learning**	Downsampled ImageNet	WGAN
	Recommendation	-	-	-
Deep Reinforcement Learning	**Game**	**Go**	-	-
		Atari ALE	Atari2000	A3C

MLMark [148] is an effort by EEMBC (Embedded Microprocessor Benchmark Consortium) with an emphasis to benchmark IoT CNN inference implementations. Its strengths relate to clear definition of figures of merit and standardized software support. In contrast to our efforts, it has no specific quantization support, nor a multilayered approach including a theoretical analysis.

DAWNBench [125] exclusively looks at ImageNet classification for training and inference. The benchmark sets very clear figures of merit such as "Time taken to train an image classification model to a top5 test accuracy of 93% or greater" and "Latency required to classify one ImageNet image using a model with a top5 test accuracy of 93% or greater" and as such supports the concept of algorithmic optimizations by tying hardware performance to accuracy achieved at the application level, in this case image classification. DAWNBench does not provide further insights beyond the specified figures of merit, is limited in application scope, and doesn't support multiple tiers.

We would also like to include the **Collective Knowledge Framework** [134] in conjunction with the ASPLOS Request Tournament [127] as an additional benchmark. While it is narrow in scope, specifically limited to ImageNet Classification inference, it makes strides in the way it

opens up the design space for esoteric hardware accelerators, specifically facilitating architecture specific algorithm transformations and correlation between accuracy with performance and power within a larger design space. All of this is essential to support heterogeneous hardware architectures. ASPLOS also excels in regards to reproducibility, leveraging ACMs artifact evaluation technology, and provides insight into possible hardware performance and error rate trade-offs, through an online scorecard. Also, ckframework,org have recently embraced MLPerf, discussed in the next paragraph, and help with the visualization of the results [149].

MLPerf [150, 151] is probably the most promising approach at providing system level benchmarks, which emerged just shortly after we started our efforts with QuTiBench. Similarly to Fathom and TBD, MLPerf is dedicated to very specific, yet representative, application scenarios, having selected two image classification and object detection CNNs (one light-weight and one heavy-weight each) plus a machine translation network. Table 3.3 summarizes MLPerf's application coverage as of 2018.

More application scenarios such as sentiment analysis and recommendation are in planning. All 5 applications are tested in 4 different load scenarios (single stream, multi-stream, server and offline). For each of the load scenarios, only 1 specific figure of merit is reported for a specific minimum application accuracy, whereas QuTiBench, as will be shown in the later chapters, rigorously reports everything and is not constrained to specific application scenarios. As such the insights gained with MLperf are much more limited and the fully available design space cannot be represented. MLPerf considered initially only training, but inference was added since end of 2019. Finally, MLPerf's strength is that it's been created by a consortium of industry partners and universities, which addresses objectivity criteria plus its detailed considerations of statistical aspects, as well as reproducibility. Key to this is a test harness which was developed by consortium members and executes the applications in each load scenario. All contributors measure in exactly the same way.

Unlike QuTiBench, as will be shown in later chapters, the performance is always measured at system level and does not represent the performance of the actual accelerator, for example when the data is already present and can be directly streamed from other system components. With this, system-level insights are much more limited.

Also, different to MLPerf, QuTiBench embraces a multi-tiered concept, which is missing from MLPerf. The multi-tiered approach offers a range of compromises for benchmarking in regards to quality of prediction and effort. The microbenchmarks expose system bottlenecks in

Table 3.3: MLPerf coverage: applications, datasets and models

Learning Technique	Application Domain	Task	MLPerf Dataset	MLPerf Model
Supervised	**Vision**	**Image Classification**	ImageNet	ResNet
	Vision	**Object Detection**	COCO	-
	Vision	**Semantic Segmentation**	-	Mask R-CNN
	NLP	**Machine Translation**	WMT Eng-German	Transformer
	NLP	**Machine Translation**	-	-
	NLP	**Speech Recognition**	Librispeech	DeepSpeech2
	NLP	**Sentiment Analysis**	IMDB	Seq-CNN
	NLP	**Language Modeling**	-	-
Unsupervised	**Vision**	**Feature Extraction**	-	-
	Vision	**Adversarial Learning**	-	-
	Recommendation	-	-	-
Deep Reinforcement Learning	**Game**	**Go**	Go	Mini-Go
		Atari ALE		

the various hardware architectures and the theoretical baselines can act as a measuring stick to track compute efficiency and provide performance predictions.

MLPerf and QuTiBench are the only benchmarks that provision for algorithmic optimization, however MLPerf offers a much more limited scope compared to QuTiBench. MLPerf does this by introducing the concept of closed and open models as shown in Figure 3.4. In the open model submission, topologies can be changed, quantization and pruning can be applied whereas in the closed division, models can not be changed and submissions must have an accuracy of 99% compared to provided floating point baseline.

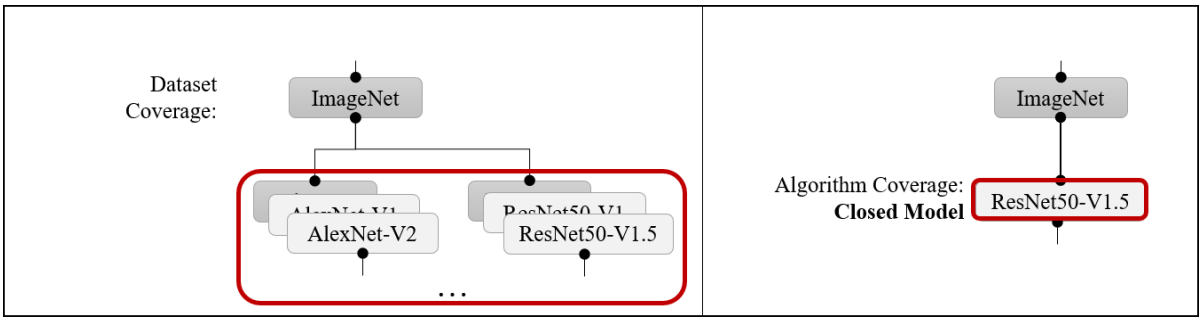


Figure 3.4: Closed vs open models in MLPerf

Furthermore, MLPerf offers 3 levels of maturity for submissions: available, preview, and R&D prototype whereby all submissions that are currently not available are omitted to be submitted with a given time frame. Also, MLPerf has provided a lot of thinking into statistical confidence, auditing, reproducibility and checks for cheats such as caching and taking advantage of the a priori known random distribution. Close to 600 submissions were made for inference in V0.5, considering both cloud and edge scenarios. The only missing submission was GNMT

multistream. Most submissions were on ResNet50v1.5. Only 1 FPGA implementation was contributed by Furiosa.ai, and only 1 submission in the open category leveraged quantization.

In summary, the key difference between QuTiBench and MLPerf can be summarized as follows:

- MLPerf has very specific figures of merit in very specific application scenario; whereas QuTiBench always captures latency and performance, and power (board and idle).
- MLPerf is system-level only, and doesn't capture compute.
- MLPerf only captures application-level performance, whereby QuTiBench is hierarchical with theoretical, level-1 and level-2.
- QuTiBench is single node only; MLPerf is both single- and multi-node.
- MLPerf does not consider visualization of results, for example with pareto graphs.
- MLPerf has much more industry traction, well defined test harness (loadgen) and much more emphasis on statistical confidence, auditing, and reproducibility.

A more detailed comparison is given in the evaluation Chapter 7 which highlights the merits in both efforts. In Chapter 8 we discuss how the best in both efforts could be combined to create a greater version of a NN systems benchmark.

Current benchmarks focus on subsets of algorithms used, mostly on training, and none of them offer flexibility in precision for datatypes specifically (although MLPerf has the open model concept introduced). Also, no other benchmark supports a tiered approach or flexible backends to support heterogeneous hardware platforms and provides understanding of bottlenecks within the system. Overall, support for algorithmic optimization is limited across the whole spectrum of benchmarks, in particular in regards to quantization and none of the other benchmarks None of the other benchmarks ensure FAIR data management to the extent that QuTiBench does. In Table 3.4, we summarize the application scope of existing and our proposed benchmark, as well as the key differentiators between existing benchmarks and our proposal. We will discuss in Chapter 4 how we address these characteristics.

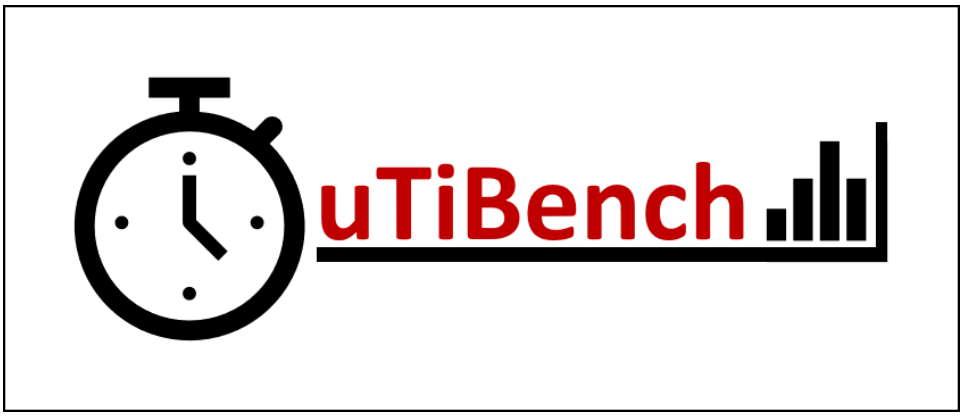
Table 3.4: Feature comparison of existing benchmarks and QuTiBench

Criteria	MLMark	MLPerf	DeepBench	DawnBench	Fathom	TBD	BenchIP	QuTiBench
Machine Learning Task								
Training	no	yes	micro	yes	micro	yes	yes	planned
Inference	yes	planned	micro	yes	micro		yes	yes
Coverage								
Applications	broad	broad		narrow		broad	broad	broad
Compute Patters	broad	broad	medium	narrow	broad	broad	broad	broad
Data Movements								broad
Support for Optimizations	limited	limited		limited				yes
Full Design Space Representation		yes		yes				yes
Deployment Scope								
Cloud	no	yes	yes	yes	yes	yes	yes	planned
Embedded	yes	yes	yes				yes	yes
Benchmark Abstraction								
Theoretical								yes
Microbenchmarks Compute			yes		yes		yes	yes
Microbenchmarks Data Movement								yes
Full Applications	yes	yes		yes		yes	yes	yes
Speed vs Accuracy Trade-off							limited	yes
FAIR/Open Data	open	open	open	open	open	open	open	FAIR
Reproducibility	yes	yes	yes	yes	yes	yes	planned	planned

3.5 Concluding Remarks

In this chapter, we considered the key characteristics in a benchmarking suite which includes being representative of common workloads, supporting of algorithmic modifications, objectivity and reproduciblity, being portable to heterogeneous hardware systems, addressing the complexity vs accuracy trade-off and being agile and adaptive, given the speed of change that we observe within this space. We categorized benchmarks in this space into ML, performance and system benchmarks, whereby we're focusing on systems benchmarks which combine performance with the specific application figures of merit to enable scope for popular optimization schemes. We conduct a thorough review of related work, and expose key gaps in current efforts. This includes in particular lack of system-level insights for computer architects, accuracy-speed trade-offs through the multi-tiered approach, broader representation of the design space, and scope for algorithmic optimizations. Particular emphasis in the comparison to state of the art is put on MLPerf, whereby a full evaluation of the approach in this thesis compared to other system benchmarks is provided in Chapter 7. In the next chapter, we will introduce the key aspects of our proposed approach.

4 Our Solution Approach: QuTiBench



4.1 Introduction

In the previous chapters, we considered the unique requirements of the application space in regards to algorithms and hardware architectures, and considered key characteristics and challenges for benchmarks in this space, including a review of alternative approaches in related work.

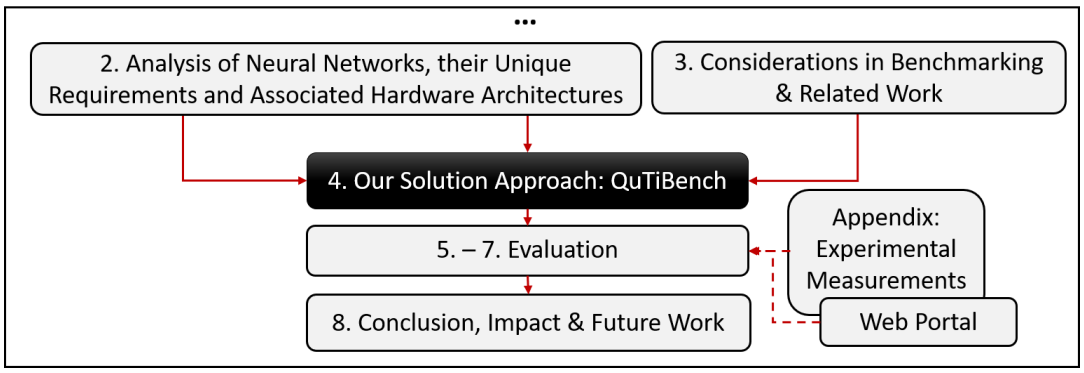


Figure 4.1: Location within the thesis

In this chapter, we propose our benchmarking methodology with the ultimate goal of enabling fair benchmarking of heterogeneous hardware architectures for neural networks which supports a spectrum of algorithmic and architectural co-designed end solutions. The aim is to expose the spectrum of possibilities and accurately reflect the capabilities of the different hardware platforms while offering attractive compromises between accuracy and speed of prediction. In short, our **design goals** are as follows:

1. Representative of the algorithms in this design space (embedded inference to begin with)
2. Representative of the broad spectrum of increasingly complex hardware architectures
3. Support co-designed end-solutions, in particular solutions that have been quantized and pruned
4. Provide a general understanding of capabilities and limitations of these hardware platforms in conjunction with given algorithms
5. Illustrate the full design space with all possibilities of design compromises
6. Offer compromises between accuracy and speed of performance predictions
7. Create true research impact by enabling reuse of our results and facilitate community contributions

With these goals in mind, we created QuTiBench. QuTiBench is a novel multi-tiered (**Ti**) benchmarking methodology that supports algorithmic optimizations such as quantization (**Qu**) and helps system developers understand the benefits and limitations of these novel compute architectures in regard to specific neural networks and will help drive future innovation. Key characteristics of our benchmark are described briefly here and in much greater detail in the following sections:

Firstly, QuTiBench provides a method that allows for algorithmic freedom and can objectively compare customized end-solutions, which is essential when comparing these domain specific and highly versatile hardware platforms. The key idea is to tie performance back to accuracy at the application level, and comparing the various figures of merit (latency, throughput and throughput/power) via pareto graphs whereby one axis is always accuracy in the application space. Optimal solutions can be found along the pareto frontier. This way quantized and pruned models and any other alternative highly customized algorithm can be fairly compared

amongst many other possible optimization strategies. A second key characteristic of QuTiBench evolves around a multi-tiered approach which is shown in Figure 4.3. We tier the benchmark suite with respect to abstraction levels as well as numerical representations for both training and inference tasks. This provides not only attractive compromises in regards to speed versus minimal discrepancy with target workloads, but also brings advantages such as additional system level insights, for example into where difficult data movement patterns lie. A particular important and unique aspect of QuTiBench is that it includes a theoretical level for both hardware platforms and algorithms, that we use to create performance predictions via so-called **roofline models** [21]. These predictions are useful to track compute efficiency over the various levels of the benchmark and they can provide also initial guidance without running any hardware experiments whatsoever. More details on the tiers will be provided in this chapter.

Further, we have specified clear figures of merit and a measurement methodology to provide systematic insights into the design space while considering all deployment settings. Due to the different nature of relevant hardware platforms, for each platform we are faced with many different deployment options such as power modes and batch sizes. It is essential to establish a clear and transparent methodology to help clarify the complex and obfuscated design space. In addition, we have researched different data visualization routines that help create a better understanding of the multi-dimensional design space and present data with clarity. This includes pareto-charts, heatmaps, and so-on.

Finally, we understand the critical need for a community to support this effort as well as open and FAIR [19] data to generate meaningful research impact. As such we have put significant effort into a web portal located here: `https://rcl-lab.github.io/QutibenchWeb`. In addition to providing downloadable access to thousands of measured and theoretical data points, we have included all data analysis and visualizations that were derived within the thesis. This web portal also supports third party contributions. This is essential given the scope of the benchmarking effort which is required within this space. We hope that this web portal can help pull together the research community such that we collectively can have scientific impact.

The key differentiators of QuTiBench are summarized here:

Characteristics:

- *Support for algorithmic optimizations, by correlating everything at the application level's figures of merit via pareto graphs.*
- *A multi-tiered approach towards benchmarking that supports a range of compromises for benchmarking in regards to quality of prediction and effort.*
- *The inclusion of microbenchmarks and combinations of microbenchmarks, which reflect different computational and data movement patterns for different neural networks to understand system bottlenecks. This includes specific support for quantization of compute at all levels*
- *Support of theoretical baselines as a measuring stick to track compute efficiency throughout as well as performance predictions without having to run a single experiment*
- *Clear definition of measurement methodology, figures of merit and consideration of deployment parameters*
- *Data visualization*
- *Open and FAIR data*

In the following subsections, we discuss the proposed benchmark suite in more detail, including the test suites at various abstraction levels, algorithmic optimizations and quantization in particular, considerations in regards to datasets, hyperparameters and framework challenges, reproducibility, adaptability and others.

4.2 Support for Benchmarking Co-designed Solutions

There is no single criteria that decides whether one solution is optimal, as for different use cases, different figures of merit apply. For example using a single criteria such as best inference response time for a single image from ImageNet dataset, with a specific minimum accuracy,

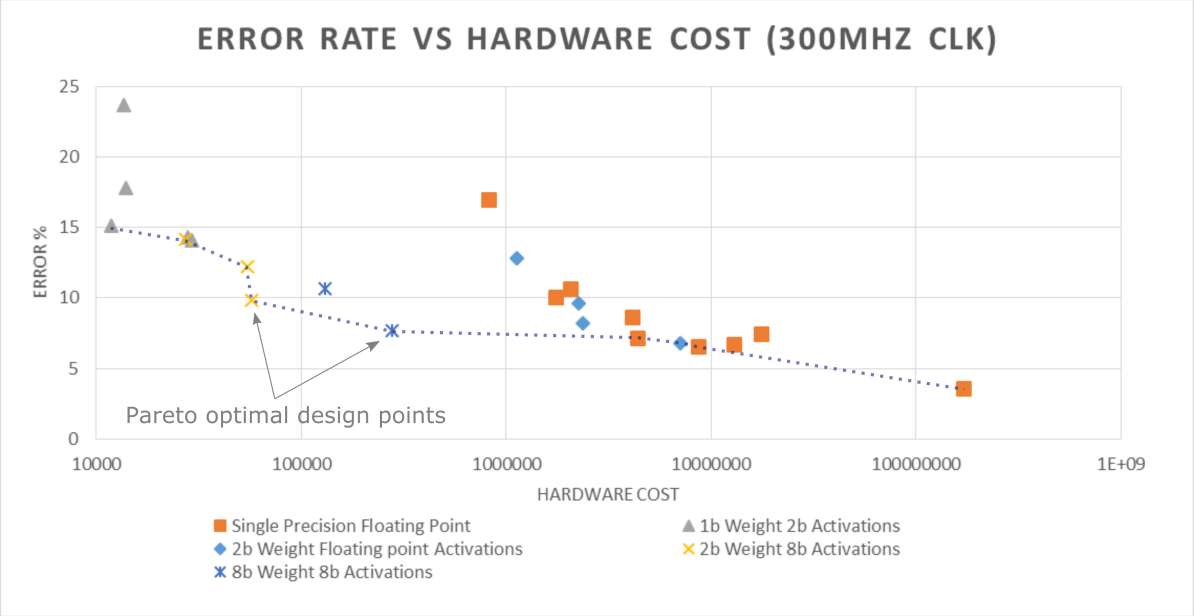


Figure 4.2: System performance evaluation: Pareto graphs are visualizing the optimal solutions within the design space, with best combinations of lowest error and lowest hardware cost

provides limited insights. It could be that another application might need less accuracy and tolerates a lower response time. Also for system designers, it can be extremely useful to have an understanding of the full scope of solutions. All combinations yield different trade-offs within the multidimensional design space. As such, we present all solutions and measurements within multi-dimensional figures, whereby the pareto frontier represents the best possible compromises for different use cases. The charts always leverage accuracy on the y-axis (to provide the algorithmic freedom). The x-axis however can represent many different figures of merit, for example hardware resource cost (as shown in Figure 4.2), performance in inputs per second (ips), frames per second (fps), giga operations per second (GOP/sec) or tera operations per second (TOP/sec), latency in milliseconds (msec) and energy related figures of merit including power consumption or throughput over power. This offers a broad overview of all the advantages and disadvantages of the various solutions. Fig, 4.2 shows an example of such a pareto graph, with hardware resource cost on the x-axis. In this case, we derived the hardware cost as a weighted sum of the different hardware resources used in the Field Programmable Gate Array (FPGA) fabric. For readers familiar with FPGAs, the sum is calculated from LUT and DSP usage, whereby DSPs are weighted a factor of a 100. The general point, however is that any formula can be used that is deemed as representative of the hardware cost. Depending on a specific applications requirement, we can derive the best suited solution from these charts.

This way of benchmarking opens up the opportunity for algorithmic innovations. We

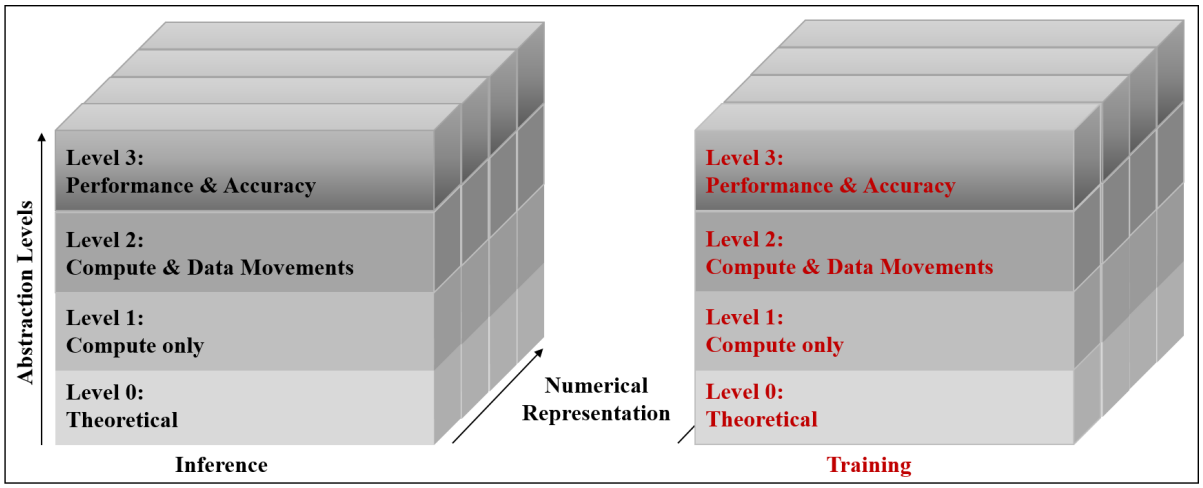


Figure 4.3: QuTiBench offers a multi-layered approach, which constitutes of a theoretical level, two microbenchmark levels (with and without data movements), and an application level, whereby precision support is implemented on all levels

include in this pruning and topological changes, while initially focusing on quantization and numerical representations as one of the most promising techniques. For this, we include, on every level of the benchmark several numerical representations, including FP32, FP16, INT8, INT4, INT2 and allow for arbitrary choices to be included, for example Microsoft's custom floating point [152]. Training each neural network with different quantization approaches and different and potentially esoteric numerical representations is highly time-intensive and is not guaranteed to always deliver converging training results. As such, careful logging of trained quantized models is a high priority for level-3. This way of performance comparison at system-level will form the highest tier (level-3) of our benchmark), as described in the next section.

4.3 The Concept of Multiple Tiers

The multi-tiered approach consists of four tiers which are shown in Figure 4.3. This includes the system or application level, two microbenchmark levels and a theoretical basis. The microbenchmarks run subsets of compute and data movement patterns on the various hardware platforms to provide insights into system bottlenecks. This offers different levels of quality in performance predictions. We leverage a theoretical baseline analysis to provide fast performance estimations without having to run any experimentation. Furthermore this theoretical baseline is highly useful to track compute efficiency on all levels of the benchmark and highlight difficult compute and data movement patterns as well as measure achieved compute efficiency. These

tiers can be applied to both inference and training, whereby we focus initially only on inference. We added an additional dimension to the tiers for quantization, which is one of the most popular and effective method of optimization. We support quantization at all four abstraction levels. We'll discuss the different levels in detail in the following subsections.

4.3.1 Level-0: Predicting and Tracking Performance with Theoretical Baselines

Level-0 is a theoretical base layer with instantly available results that offers performance predictions and provides guidance for optimization efforts and allows to compute metrics such as achievable compute efficiency. At level-0, we already introduce the notion of performance per datatype operation which is essential to support quantization as an algorithmic optimization. For all target hardware backends, level-0 records theoretically possible peak performance in TOP/sec or GOP/sec, external memory bandwidth (giga byte per second (GB/sec)) [5], and thermal design power (Watts). For all models, we record their compute and memory requirements; This includes four values: total number of compute operations for a single input, the model size, the size of the state and the total amount of tensors in between layers that require buffering. These values can be used as a basis to derive memory requirements and compute requirements for both inference and training. Many of these datapoints, for both hardware and neural networks, are presented in Section 2.2.4 and Section 2.3.1.

Combining application requirements with hardware platform characteristics allow us to derive initial performance predictions using **roofline models** [21]. In more detail, a roofline is a 2D line graph using a log-log scale which models the theoretical performance *hw_pp* of a hardware platform, shown on the y-axis, as a a function of the **operational intensity** of applications, whereby the operational intensity represents the ratio between instructions to be executed *cmp_req*, and total memory to be read and written *mem_req* from off-chip memory. We model the theoretical performance *hw_pp* of a hardware platform taking available off-chip memory bandwidth *mem_bw* and peak compute performance *p_hw_pp* into account, whereby we differentiate peak performance at different numerical representations. For example a Graphics Processing Unit (GPU) has a different peak performance for 32-bit Floating Point Representation (FP32), 16-bit Floating Point Representation (FP16), and various fixed point integer formats. The resulting curve resembles a roofline, whereby the slope represents the

[5] HBM is counted as external memory

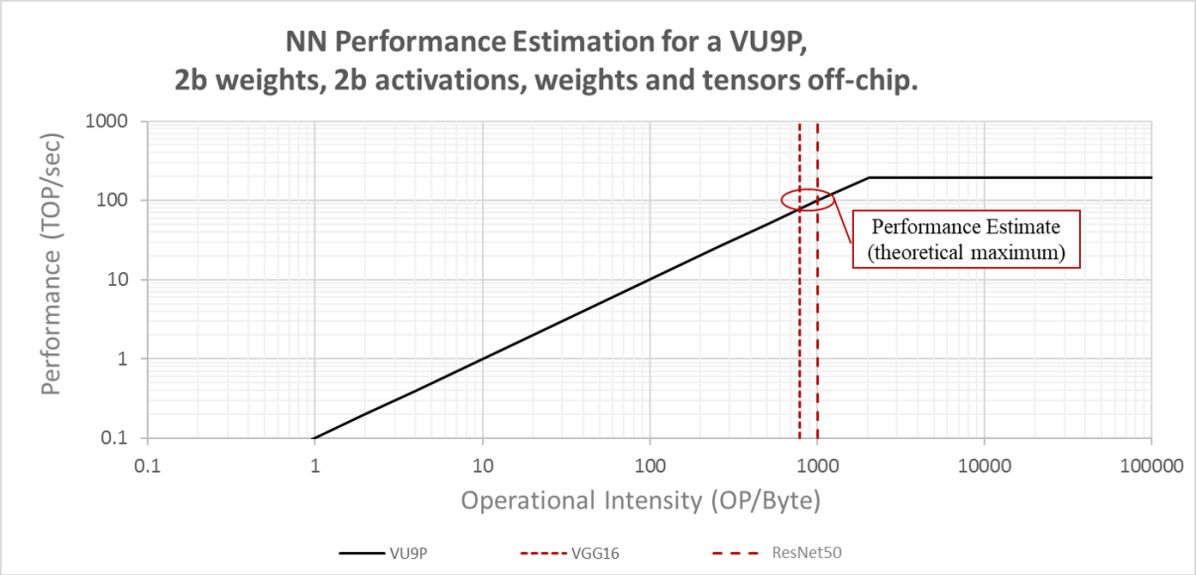


Figure 4.4: Modeling compute and memory bound hardware performance with roofline models as a function of the operational intensity of CNNs.

memory bound area, and the flat top the compute bound area, where applications hit the absolute compute limit of a platform. Combining theoretical peak performance and memory bandwidth in roofline models with the operational intensity of a Convolutional Neural Network (CNN) provides insight as to whether a neural network will be memory or compute bound and what is theoretically possible in regards to performance. The operational intensity of a CNN is derived by making assumptions for where weights, tensors, weight and activation gradients, weight updates and state of a neural network are stored, combined with the size of the datatypes used. The predicted performance is shown at the intersection between the operational intensity and the roofline itself, This is shown in Figure 4.4 at the intersection between red and black lines, annotated by a red circle, whereby the black represents the hardware performance *hw_pp*, and the red line the operational intensity of the CNN.

In more detail, this is how we calaculate the predicted performance for a specific CNN on a given hardware platform. As mentioned above, *hw_pp* is given as a function of the operational intensity (OI) of an application, which in essence represents the ratio between instructions to be executed *cmp_req*, and total memory to be read and written *mem_req* from off-chip, in bytes: $Operational\ Intensity = \frac{cmp\_req}{mem\_req}$. To be precise, *hw_pp* is calculated as the minimum of available compute performance *p_hw_pp* and memory bandwidth *mem_bw* multiplied by an application's operational intensity. This is as defined in the original paper [21] and expressed in

the following formula:

$$hw\_pp = Min(p\_hw\_pp\ , mem\_bw * \frac{cmp\_req}{mem\_req}) \tag{4.1}$$

We can use this equation 4.1 for the performance estimation by calculating the specific operational intensity of a given application. For computing operational intensity, we assume a storage location for weights, tensors, gradients, weight updates and state, either on- or off-chip, and combine these with the size of the datatypes. To compare between different implementation topologies of the same machine learning task (i.e. CIFAR-10 classification) we use performance prediction *pp* measured in ips as a metric for theoretical performance.

pp is calculated as the peak performance of the hardware *hw_pp* (defined in equation 4.1), in operations per second) divided by the compute requirements *cmp_req* to process a single input. This forms the first part in formula 4.2. Replacing *hw_pp* with 4.1, gives the second part of equation 4.2 whereby *p_hw_pp* refers to the hardware's absolute peak performance ignoring memory, and *mem_req* and *mem_bw* refer to the memory requirements in millions of elements (ME) and available hardware bandwidth respectively.)

$$pp = \frac{hw\_pp}{cmp\_req} = Min(\frac{p\_hw\_pp}{cmp\_req}, \frac{mem\_bw}{mem\_req}) \tag{4.2}$$

Results are computed leveraging this equation and visualized in different ways, as described in section 7.6.

Overall, this easy-to-understand model offers insights on performance scaling by performing analysis of bounds and bottlenecks without running any experiments. Beyond performance estimation, we can leverage the performance estimate in the roofline to track the benchmarking and estimate compute efficiency. This is shown in Figure 4.5. The compute efficiency, computed as $\frac{mp}{pp}$, whereby *mp* represents the measured performance. In this particular example, the method estimates that the achieved efficiency is 1% of what is theoretically possible.

4.3.2 Level-1 and Level-2: Identifying Inefficient Compute and Data Movement Patterns

The middle tiers in our benchmark suite are specifically designed to identify inefficient compute and data movement patterns. Level-1 represents the typical compute patterns such as

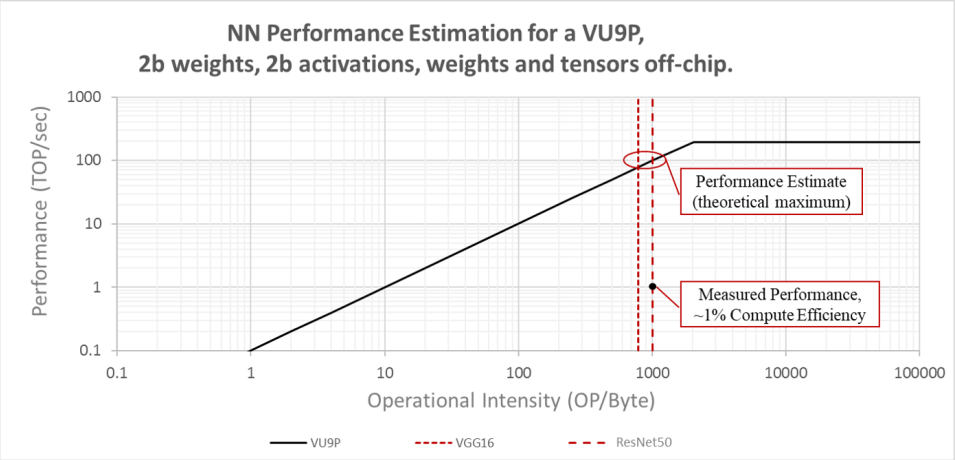


Figure 4.5: Performance estimation and tracking using a hardware platform's roofline model and the operational intensity of CNNs.

convolutional layers, fully connected layers, pooling layers and so on. Level-2 includes combinations of these microbenchmarks with the aim of identifying bottlenecks in data movement, for example, how the dimensionality of the tensors or skip connections affect storage and buffer requirements and potentially hamper system-level performance. The aim is by choosing these microbenchmarks well, we can provide further insights into what causes potential performance drops. These are valuable insights that can be used by machine learning engineers to redesign existing and future machine learning algorithms.

In more detail, level-1 exposes achievable compute performance for typical compute patterns encountered within neural networks, which equates to popular layers including convolutions, fully connected layers, recurrent layers, residual layers, and squeeze layers, over a range of dimensions and with different numerical representations (Chapter 2). These tests are comparable to DeepBench [59], with the significant difference that we provide much broader support for specialized numerical representations. For each of these compute patterns, we record the following figures of merit: measured performance (TOP/sec or GOP/sec), latency (msec), power consumption (Watt) of the full platform in the embedded space, and of the board excluding the host system in the cloud.[6] While level-1 does not capture application level accuracy, the tests will include verification of functional correctness. In the following section, we'll introduce a further differentiation between **system-level** and **compute-level** performance. For the microbenchmarks, the results reflect achievable compute performance only, excluding potential

[6]Power measurements might not always be available and might require specialized test infrastructures and testbeds.

overheads for moving data which are addressed in level-2. While requiring execution, the tests at level-1 are relatively rapid. We include a sweep over batch and thread sizes.

Level-2 is a microbenchmark for compute and data movements. It is comprised of simple combinations of level-1 tests, and can thereby effectively capture potential bottlenecks such as tensor movement between layers, as well as storage requirements. It considers only stacks of 2 and 4 layers and only a subset of all possible combinations to keep test time to a minimum. Exhaustive representation of all layer types would result in far too many test cases. We include mixed precision between layers in these small template stacks. Figures of merit are identical to level-1 and results reflect achievable compute performance only, excluding potential overheads for moving data into the accelerator and returning results. In particular, the latency variation between level-1, level-2 with stacks of 2 layers, and level-2 with stacks of 4 layers will bring insight into data movement and buffering bottlenecks. While more detailed definition of layer combinations is required, overall the benchmarking effort on level-2 is moderate.

We face particular challenges when it comes to measuring individual sub-topologies. Ideally we would represent all level-1 and level-2 tests in form of a prototxt or similar file that represents the model. This would be the fairest form of comparison. However many of the vendor-specific tools operate in a black box mode and don't support execution of parts of the topologies. Some of the tools provided a profiler, which helped and we retrieved datapoints in this variant. However for some of the platforms it was simply impossible to carry out the measurement. More details are described in Chapter 7. Finally, another challenge is that many backend tools perform automated layer fusion such as merging batch normalization with convolutions, which makes testing in isolation inaccurate.

4.3.3 Level-3: Comparing at the Application Level

Application coverage is essential to offer space for algorithmic innovation which can achieve superior system-level performance and can only be validated when combined with application results. As such, coverage in the application space is essential, achieved accuracy becomes the bar for normalizing results, and the neural network (or algorithm) itself can become a variable. We will initially consider some of the datasets and models in (Table 3.3), taken from existing benchmarks, and complement them with models that have been explored to work well with pruning and quantization optimizations. Furthermore, contributors are welcome to provide different models for given machine learning tasks.

For inference, we include performance measurements for a single image. However, the error rate is the reported test error over the whole test dataset. For training, we plan to report throughput, training time (latency), and power for a single image as well (including correctness tests). We also aim to provide measurements over longer training sequences with specific accuracy targets, for example, measure complete training time 90% top-5 error for ImageNet classification with a ResNet50. Finally, we consider the option to optimize the training algorithm and network and record all possible data points in a multi-dimensional graph; for those it is essential to include development time. However, all experimentation is currently for inference only.

4.4 Unified Figures of Merit & Clear Measurement Methodology

In order to provide a fair comparison between these diverse sets of platforms and algorithms, it is essential to provide a clear measurement methodology with well-defined figures of merit. In addition to this, due to the different nature of relevant hardware platforms, we are faced with many different deployment options such as power modes and batch sizes. It is essential to establish clear and transparent guidelines and measure in a systematic way to help clarify the complex and obfuscated design space. We discuss each figure of merit in isolation below.

4.4.1 Latency and Throughput

Latency and throughput are key performance characteristics and there are many ways to measure them. We report and differentiate **compute** and **system** throughput and latency as this allows to crystallize system overheads such as data copy which can have a significant impact on overall performance. The difference between **system** and **compute** throughput or latency is illustrated in 4.6. Unfortunately, it is not always possible to make measurements at exactly the same granularity across all the given platforms as shown in Figure 4.6. In this example, for compute performance on the FPGA, the primitive to execute the compute includes the movement of results at the end. Given that this only contains the resulting probabilities, we believe this is negligible and the results are still representative and fair. In regards to throughput, we consider fps or ips when comparing different algorithms with each other, as we

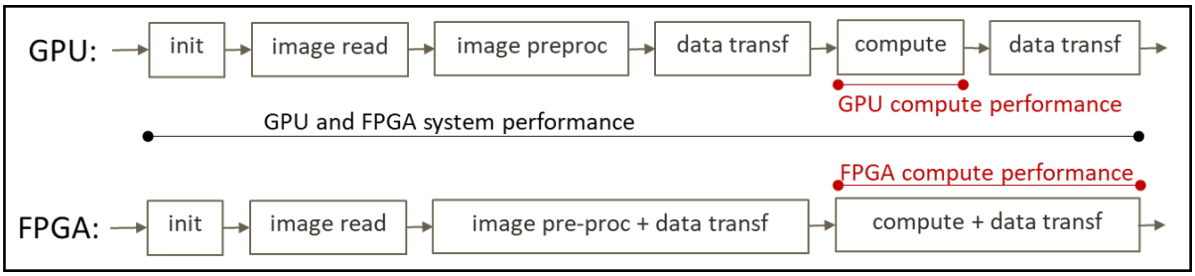


Figure 4.6: Difference in measuring system versus compute performance on different hardware platforms. System performance includes overheads such as image reading, preprocessing of data and data movement.

are not really interested in the actual TOP/sec or GOP/sec, but only the resulting performance in fps or ips when comparing the accuracy. We chose TOP/sec or GOP/sec only when considering compute efficiency aspects. Finally, system or compute latency is reported in msec.

4.4.2 Power and Energy

To represent power and energy cost, we only report platform power measured at the socket. While this is not necessarily accurate, there are strong reasons behind this choice: First, the measurement needs to be fair. Therefore we believe it is essential that subsystems, including memory, need to be taken into account. Second, more detailed current sampling on the platforms may be available on some platforms, but each platform comes with different interfaces, and may or may not provide access to all power rails. While the accuracy of typical socket power meters is around 10%, we found that these results remain representative of the systems. We average the results over 100 measurements to ensure statistically relevant information. The biggest drawback is the lack of temporal resolution and the lack of automation as the meters are being read visually. We will in the future investigate whether there are cost effective means to provide automated power measurements with sufficient sampling rates.

Another consideration is whether to consider power or energy per frame. We settled on using absolute power consumption since when multithreading or batching is applied, it is hard to derive a representative number for energy and would differ depending on whether the end application is latency or throughput driven. Finally, idle power with these platforms, can represent a significant percentage of the overall power budget and would therefore cloud the observation. In particular, two of the FPGA platforms (ZCU104 and ZCU102) are evaluation boards with many superfluous peripherals, which is reflected in high idle power (19.9 Watts) compared to the GPU (between 3.4 to 5.0 Watt depending on power mode). The additional

Figure 4.7: Breakdown of power consumption for both a GPU (TX2) and FPGA platform (Ultra96): While the GPU has a much lower idle consumption (board power and bitstream loaded), the FPGA requires much less power for the actual processing of the CNN.

Table 4.1: Idle power consumption for a GPU in different power modes and one FPGA platform, whereby the FPGA platform has a much higher idle power consumption

Tx2-MAXN [Watt]	Tx2-MAXQ [Watt]	Tx2-MAXP [Watt]	ZCU104 [Watt]
5.0	3.4	4.0	19.9

dynamic power consumption is minimal and yields the FPGA overall as the more efficient platform despite the initial load. This is shown in Figure 4.7.

4.4.3 Figures of Merit at Different Levels

The figures of merit vary in regards to the various benchmark levels with some exception which relate to compute performance only. This is summarized in the following table 4.2. In level-3, we measure both system and compute latency, system and compute performance, both in TOP/sec or GOP/sec as well as ips or fps, and power. Also level-3 is the only level where accuracy measurements make sense as it operates on the application level. For level-2 and level-1, we capture compute only variants of latency and throughput and we omit accuracy (as previously mentioned). Power which was typically impossible to measure due to the short execution time of the benchmark and the coarse temporal resolution of the power meters, therefore only latency and performance have been captured. Level-0 has different figures of merit which include Thermal Design Power (TDP), theoretical peak performance and memory bandwidth for the hardware platform, and modelsize, measured in millions of elements (ME), size of state (number of activations) and total compute requirement.

Table 4.2: Figures of merit for all levels

Figure of Merit	Level-0	Level-1	Level-2	Level-3
System Latency [ms]	no	no	no	yes
Compute Latency [ms]	no	yes	yes	yes
Theo. Peak Performance [GOPsec, TOPsec]	yes	no	no	no
System Performance [GOPsec, TOPsec]	no	no	no	yes
Compute Performance [GOPsec, TOPsec]	no	yes	yes	yes
System Throughput [fps]	no	no	no	yes
Compute Throughput [fps]	no	yes	yes	yes
Accuracy [Top-1 %, Top-5 %]	no	no	no	yes
Base Power [Watt]	no	no	no	yes
Idle Power [Watt]	no	no	no	yes
Full Power [Watt]	no	no	no	yes
TDP [Watt]	yes	no	no	no
Mem. Bandwidth [GBsec]	yes	no	no	no
Modelsize [ME]	yes	no	no	no
Compute [GOP]	yes	no	no	no
Size of State [ME]	yes	no	no	no

4.4.4 Systematic Evaluation

A systematic evaluation is highly beneficial when it comes to creating system-level insights, such as how does batch size impact latency and throughput, or what performance-power compromises can be achieved through different power modes. For this purpose, QuTiBench includes a highly systematic evaluation approach that measures all figures of merit, for all machine learning tasks, for all topologies with all available optimizations, for all deployment settings on all hardware platforms. This is very different to the MLPerf effort, where only one specific figure of merit in each of the scenarios is reported.

4.5 Understanding through Data Visualization

Data Visualization through charts and graphs is critically important to gain insights and to communicate these quickly and efficiently as visual information is known to be processed faster than text. Most of what our brains absorbs on a daily basis is visual information. Graphs provide context and correlate data which helps to drive clarity to an analysis and enhances understanding. As part of our effort, we have researched different data visualization routines that help create a better understanding of the multi-dimensional design space. This includes pareto charts, heatmaps, and so-on. The key challenge in visualizing our benchmarking data

are as follows:

1. High dimensionality of the design space
2. Highly divergent range of data
3. Volume of data points

To address the multi dimensionality aspects and the large volume of data, we leverage two techniques. We only visualize isolated dimensions at a time through projections: For example, we show accuracy versus latency and accuracy versus throughput separately. Secondly we leverage higher dimensional graphs such as box and whiskers. This allows us to compact data significantly and represent the statistical distribution of values rather than absolute values. To address the big differences in data ranges, we normalize the values to the maximum or minimum of the various platforms, which demonstrates the trends of values for different variants of neural networks (for example quantized and pruned networks) and combine these with tables, which provide the absolute numbers.

We chose the following set of visualizations which will be explained in more detail in the subsequent sections. This includes the previously mentioned rooflines (see Section 4.3.1) and pareto charts, bar and stacked bar charts, line and XY scatter graphs, heatmaps and box & whiskers. All of these visualization routines are implemented, available as open source and included as part of our web portal.

4.5.1 Bar Charts, Stacked Bar Charts, Line Charts and Scatter Plots

For many simple comparisons, we leverage bar charts and stacked bar charts. They are easy to understand and visualize nicely one figure of merit (for example power, latency or throughput) for a selection of experiments. With stacked bars, we can show a further breakdown of the figure of merit too, for example system versus compute, or full power versus idle power, and even visualize a range of values. An example is shown in Figure 4.8. We also leverage bar charts for comparisons between theoretical (estimated) performance and measured performance, whereby we annotate the percentage of the theoretical baseline achieved within the chart. For visualization of the statistical distribution, we leverage box & whiskers, which are further explained below, see Section 4.5.2. We also use standard line charts and scatter plots which display information as a series of data points called 'markers' connected by straight line segments

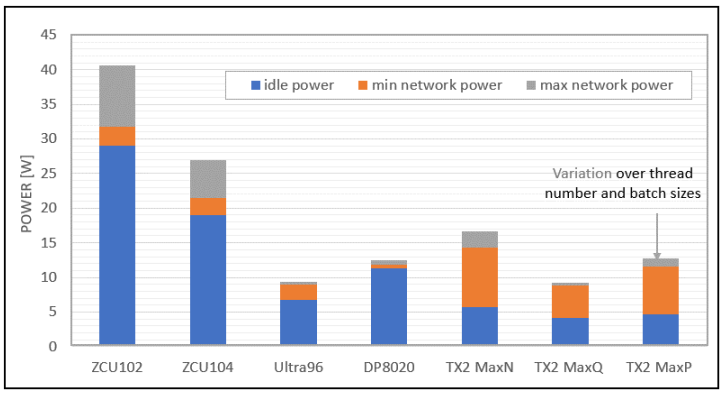


Figure 4.8: Example bar chart for visualizing power consumption of different hardware platforms in different operating modes. Each bar indicates idle power, minimum and maximum power during computation and summarizes all results over all thread numbers, stream sizes and batch sizes. This visualizes range and variation of values.

for ordered or no ordered data points respectively, typically in conjunction with pareto frontiers, as described in Section 4.5.4 below.

4.5.2 Box and Whiskers Charts

As mentioned above, we chose box and whisker graphs as they allow to compress many experimental values and visualize overall behaviours. Figure 4.9 illustrates an annotated example of such a graph. Each figure shows all experimental data for one specific CNN topology. In more detail: We group all datapoints over all batch, stream sizes, thread counts and all deployment parameters per hardware platform and network into one bar. The boxes represent the 1st and second quartile and median, x denotes the average, and the whiskers show the outliers. There are always a group of bars per hardware platform and datatype/precision, and one bar per pruning factor. The size of the box visualizes the large variations of values (for example for GPUs). The benefits of quantization are then visualized in comparison between two groups of box and whiskers, in particular in regards to averages but also minimum and maximum values. The benefits of pruning are shown by comparing the bars within one group. Finally, the first group for latency and throughput shows theoretical compute per input or the inverse to visualize the correlated relationship to the respective figure of merit.

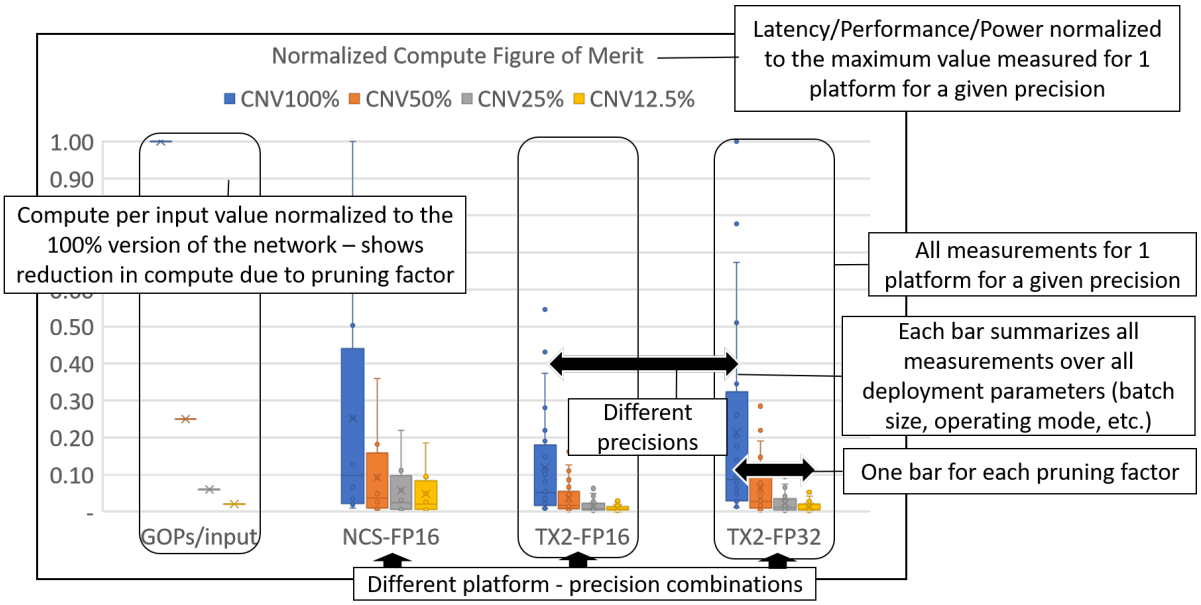


Figure 4.9: Box and whiskers for different pruned and quantized versions of a given CNN topology. This visualizes overall range and variation for pruned and quantized solution variants.

4.5.3 Heatmaps

The heatmap (Figure 4.10) visualizes predicted performance for CIFAR-10 classification across all topologies and platforms. We create one heatmap per machine learning task, indicated in the top left corner. Each column represents a different CNN model, in different pruning scales and with different precisions. Each row represents a hardware platform, whereby we treat hardware platforms in different operating modes and at diffferent datatypes as separate platform, so we can make differentiated performance predictions. Each cell contains the actual numerical value for the performance prediction, whereby the colour scale presents the actual value. We choose a 2-colour scale to emphasize the highest (red) and the lowest (gray) values.

In this example, the results show immediately that we expect a TPU in fast mode at INT8 to provide the best possible performance for ImageNet classification, using MobileNet-V1. From these types of visualizations, impact of operating modes, datatypes and pruning scales are easy to recognize.

4.5.4 Pareto Graphs: Accuracy versus Other Figures of Merit

Per wikipedia, a pareto frontier is defined as a state of allocation of resources from which it is impossible to reallocate so as to make any one individual or preference criterion better

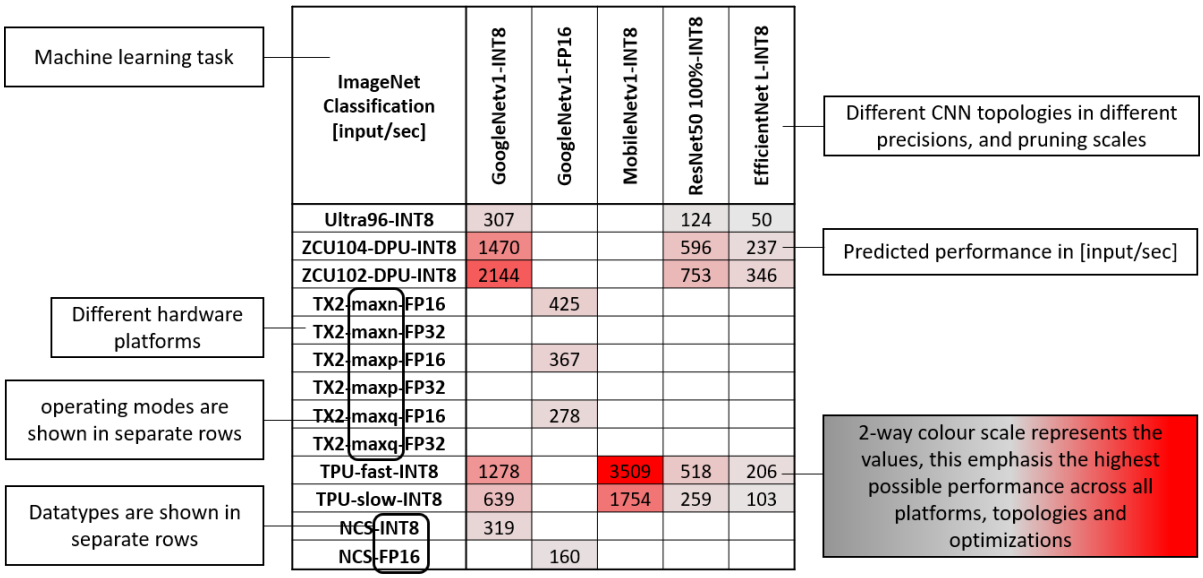


Figure 4.10: Performance predictions for ImageNet classification with heatmaps

off without making at least one individual or preference criterion worse off. As such pareto frontiers are ideal for level-3, where we aim to explore and visualize optimal solutions within the design space regarding application performance independent of model topology and algorithmic optimizations. These are typically compromises between accuracy and either throughput or latency. As base graphs we leverage XY scatter and line graphs. The latter are being used when we are looking to group some of the measurements, for example all experiments over a spectrum of batch sizes, but for the same hardware platform, same CNN topology and under the same operating modes. Below, in Figure 4.11, we show an example which includes results for FPGA implementations with FINN using INT2 and INT4 precision and 3 different pruned variants of the aforementioned CNV CNN. The results are across the spectrum of stream sizes, which are combined by a line. The pareto frontier visualizes the following: Firstly, both quantized and pruned versions provide pareto optimal design points, which increase performance at the cost of a slight accuracy degradation. Optimal design points are achieved for highest stream sizes when the streaming architecture is fully utilized and saturated. In particular, the pruning benefits from 50% to 25% are close to 4x. Also, the performance continues to increases with more pruning or quantization, as the CNN can be further unfolded (parallelized) inside the hardware architecture. Similar graphs are leveraged to visualize the accuracy-latency trade-offs. Finally, we utilize these visualization not only for measurements but also for theoretical performance predictions with accuracy as given by trained CNN variants. Using these charts allows us to predict the pareto-optimal solutions and highlight most promising combinations of optimization

techniques.

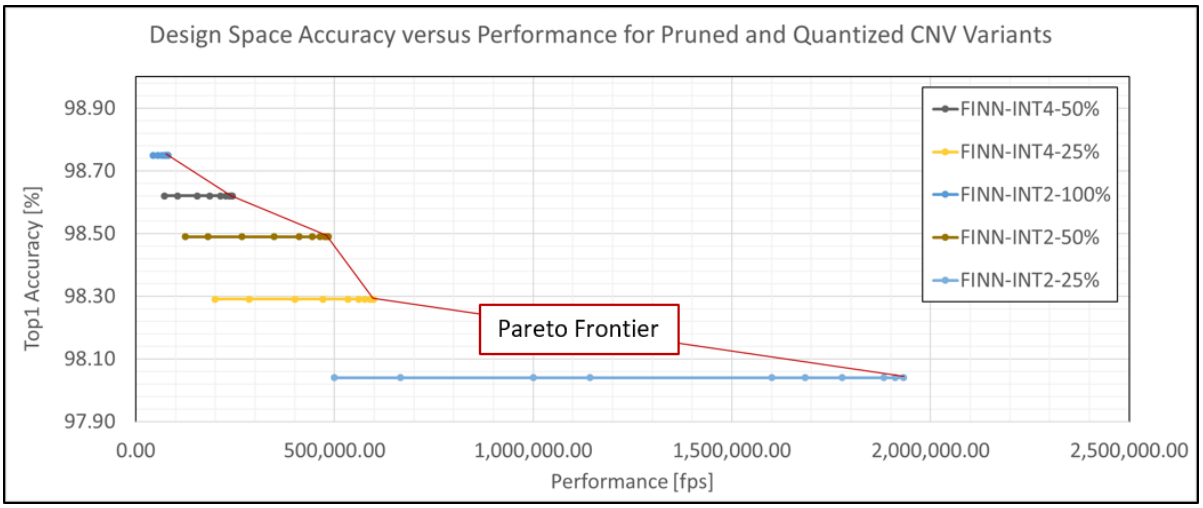


Figure 4.11: Level-3: Pareto frontiers help visualize the different accuracy and performance trade-offs.

4.5.5 Theoretical versus Experimental Comparison Pareto-Frontier

We have created a new chart as part of this effort, with the aim to enable a visual comparison between the predicted pareto-optimal solutions and the measured pareto-optimal solutions. The chart is based on XY scatter and line charts and includes all theoretical and measured data points. We visualize both the pareto frontier for performance predictions and the frontier for the measured datapoints. An example of such a chart is shown in Figure 4.12, whereby the yellow line represents the theoretical pareto frontier and the blue shows the measured pareto frontier. Many of the predicted optimal design points, materialized also in the measurements as optimal. We faced a number of challenges creating these charts, most importantly relating to the huge amount of data involved. To help with clarity we experimented with size, shape and colour of markers. We found best using colour to identify markers belonging to the same hardware choice and using shapes to differentiate experimental versus theoretical. Also, as part of the web portal, we made hardware platforms and CNN topologies selectable and the chart itself interactive, so the user can pan and zoom into particular areas of interest.

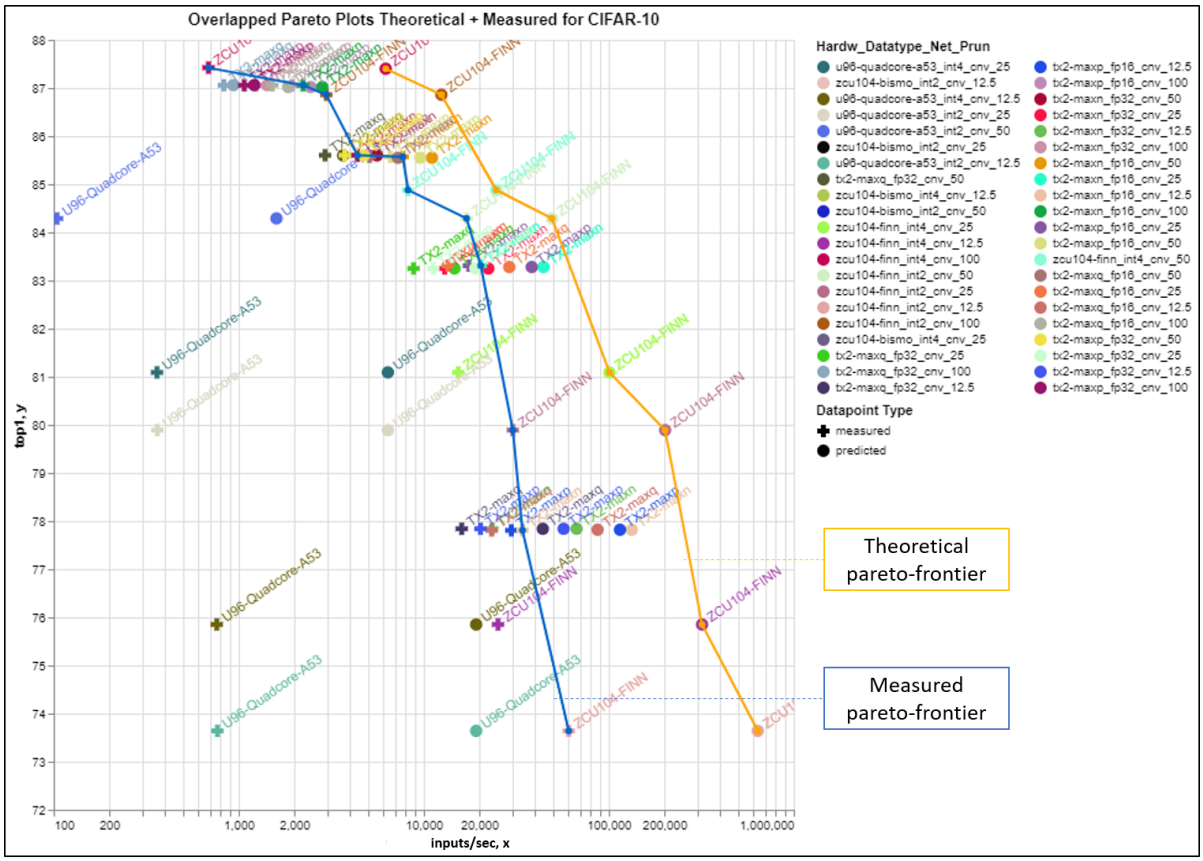


Figure 4.12: Level-3: Pareto comparison charts help with the evaluation of predicted versus measured pareto frontiers.

4.6 Impact on the Wider Research Community through Open and FAIR Data

The nature of the problem that we are trying to address within this thesis is far beyond what an individual can achieve. The sheer scope of experimentation required for the cross product of hardware platforms, CNNs and potential deployment parameters and optimizations is vast. Furthermore the constant addition of new platforms, the rapidly evolving topologies for CNNs requires a high degree of adaptability. Machine Learning is currently a highly dynamic field in research and specific algorithms may become very quickly outdated as new models may emerge and take over rapidly. It will be of essence that the benchmark rapidly adapts to latest models and expands to novel application domains as they emerge. In summary, this effort is well beyond the scope of a thesis. The only way to really have an impact is through a community effort. In order to facilitate this, it was essential to not only share our scientific data, but to also enable third party contributions. Finally, training CNNs is highly time intensive and costly

process. Therefore it is very important, that we carefully record all datapoints.

To address all of these challenges, we have put significant effort into a web portal located here: `https://rcl-lab.github.io/QutibenchWeb`. In addition to providing downloadable access to thousands of measured and theoretical data points and including all data analysis and visualizations that were derived within the thesis, this portal enables third party contributions. For the contribution of additional measured or theoretical datapoints, we've created a separate section on our web portal to facilitate addition of theoretical data and measurements. Furthermore, to allow expansion of the scope of the project, all code and datapoints are organized through an open source repository. This way, maximum flexibility can be provided and new types of tests and new machine learning tasks could be added, as well as new figures of merit and new data visualizations. For example, an interested party is looking into the addition of energy delay products as an additional figure of merit. It is essential that all datapoints are easily accessible and with the right license agreements. Also all data needs to be categorized and annotated with metadata such that it is not only for human consumption, but also possible for third party software to automatically analyse the data and contribute back. The concepts of open and FAIR [19] data have been specifically designed for this. Implemented with these concepts, this provides a gateway to generate meaningful research impact. In the following, we provide some background on open and FAIR data and how we applied them to QuTiBench as well as details on the web portal.

4.6.1 Data Revolution, Open and FAIR Data

The idea behind the **data revolution** [153], and the concept of **Open Data** [154] is that data, in particular when relating to science and governments, should be freely available to everyone to use and republish as they wish, without restrictions from copyright, patents or other mechanisms of control, with the aim of facilitating knowledge discovery and improving research transparency. **FAIR** refers to a set of guiding principles for scientific data management and stewardship. It is an acronym and stands for findability, accessibility, interoperability, and reusability [19]. As such, the intention is to make scientific data easier to find, more accessible and improving interoperability and reuse. In particular, this is aimed to enable direct access from computational systems with minimal human intervention. It is essential in this context of benchmarks for machine learning, where a large, multi-dimensional and complex amount of data is produced and collected. Open and FAIR data are key to turning scientific data into

valuable resources, that will eventually bring new insights and create true benefit to the entire academic community.

At the basis of implementing these principles is making data open and freely available, providing associated meta-data with each data point, and a trusted and persistent repository with a unique and persistent identifier. In order to deliver on this, the following was implemented:

- All measurements are available through an open source repository and accessible through a web portal [20].
- Data can be downloaded as CSV files and from there easily integrated, postprocessed and analysed by third party software.
- All measurements are tagged with metadata which makes it unique.
- We created data-level and project-level documentation through `http://nsteffel.github.io/dublin_core_generator/generator_nq.html` which can be found on the web portal.
- We have created a unique persistent identifier (DOI) with ResearchGate for the repository: DOI: 10.13140/RG.2.2.35785.57448
- License terms are clear and defined through a license file located within the repository.

4.6.2 Web Portal

As mentioned above, we have created a web portal in collaboration with Northeastern University to enable third party contributions in many different forms. In addition, it has a significant amount of analytics and interactive data visualizations (as presented within this thesis) which help understand the best trade-offs in this design space. It is designed and set-up to become a collective place for the research community to collaborate and gather insights.

As shown in Figure 4.13, the web portal has the following content: The main page contains a general description of the project as well as a small dashboard, which holds the current statistics of logged data points. We included a small subsection on open and FAIR data, which provides access to all data including meta data, as well as DOI and so forth. From the main page, one can also access the **Theoretical Analysis**, which contains the following:

- Compute and memory requirements, accuracy of all CNN topologies, in table format, including optimized variants - grouped by machine learning task

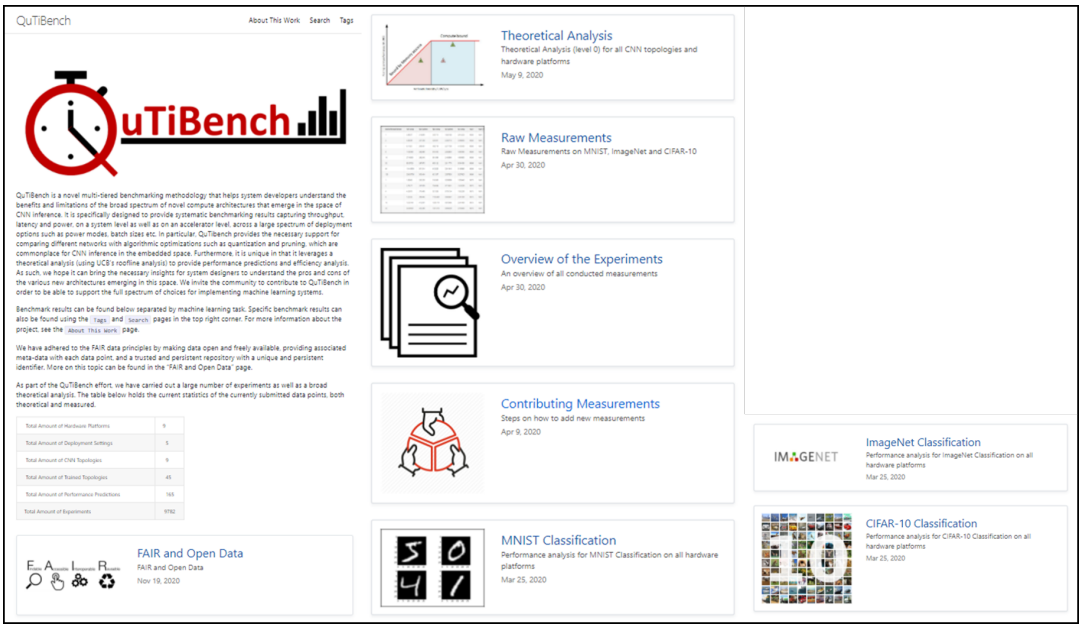


Figure 4.13: Contents of the QuTiBench web portal

- Compute and memory capability of all hardware platforms in table format
- Rooflines for all hardware platforms with operational intensity of given topologies visualized
- Performance predictions for the different machine learning tasks in heatmaps
- Theoretical pareto charts to forecast the most promising implementation options for a given machine learning task.

In addition, there is for each machine learning task, currently **MNIST**, **CIFAR-10** and **ImageNet Classification**, a separate page. Here, the theoretical and experimental data points relating to this specific task are grouped together. In more detail, the reader can find the following:

- Rooflines and performance predictions pertaining to the specific machine learning task
- A table of all experimental datapoints for the task
- Line charts showing the relationship of latency to throughput for all experimental data
- Boxplots visualing throughput, latency and power for all optimizations and platforms. These can highlight the benefits of the various optimization schemes.

- Pareto plots which depict the overall design space compromises offered with the pareto-optimal solutions marked along the frontier.
- The web portal also includes the overlaps between theoretical and measured pareto plots which can be very helpful to visualize how well the performance predictions have worked.
- This section of the portal also shows achieved compute efficiency for all measured data points in box plots. These can be instrumental to decide whether more performance can be achieved.
- Finally, the reader can find much more detailed power information such as for example power consumption over time.

In addition to the above, there are pages for **Raw Measurements**, which allows all measurements to be downloaded in form of CSV files, **Overview of Experiments**, which is designed to provide the reader with a brief overview of all experimental content, and **Contributing Measurements**, for documenting the process for community contributions.

4.7 Handling Vendor-specific Frameworks & Datasets

Datasets are a key input to the benchmark and impact accuracy results. We rely on open source datasets exclusively. Framework support is one of the biggest challenges since each framework is directly connected with a neural network and datasets within an application context and models are not necessarily portable between frameworks. Also, hardware platforms typically only support a subset of frameworks. Furthermore, quantization is not necessarily mainstream in frameworks. It is not yet clear to what extent cross compilation tools such as TVM [138] can help, while exchange formats such as ONNX [136] are still immature, lack adoption and very importantly full quantization support. Finally, it is also important to provide full training scripts exposing all hyperparameters, training initializations and so on, as they can have significant impact on accuracy.

4.8 Concluding Remarks

In this chapter, we introduced the key aspects of our benchmarking methodology, named QuTiBench, in order to address the unique demands of the machine learning application space.

Most notably, QuTiBench supports algorithmic optimizations by correlating everything at the application level's figures of merit. QuTiBench offers a multi-tiered approach towards benchmarking that supports a range of compromises for benchmarking in regards to quality of prediction and effort. This includes microbenchmarks and combinations of microbenchmarks, which reflect different computational and data movement patterns for different neural networks to understand system bottlenecks including specific support for quantization of compute at all levels. The multitiered approach also offers theoretical baselines as a measuring stick and performance predictions and to track compute efficiency. We also contributed a clear definition of measurement methodology and consideration of deployment parameters. Further, we propose data visualization aspects which are essential given the design space complexity. Examples of this are heatmaps, box and whiskers charts, and pareto comparison graphs. Finally, we emphasis the need for a web portal implementing open and FAIR data concepts to advance the true impact on research. In the following chapters, we will evaluate the approach, whereby Chapter 5 first sets the scope. The theoretical analysis can be found in chapter 6 and experimental results and evaluation of the results are in Chapter 7.

5 Scope of Evaluation

5.1 Introduction

In the previous chapter, we introduced the key concepts in our benchmarking methodology. In this chapter, we set the scope of theoretical and experimental analysis to evaluate our approach in Chapter 6 and 7 respectively. This chapter includes a full overview of all experimental platforms, chosen workloads and the conducted experiments. Furthermore, we apply our measurement methodology to the chosen set of experiments including details on experimental setup. In short, this chapter contains everything that the reader needs to know in order to understand the results.

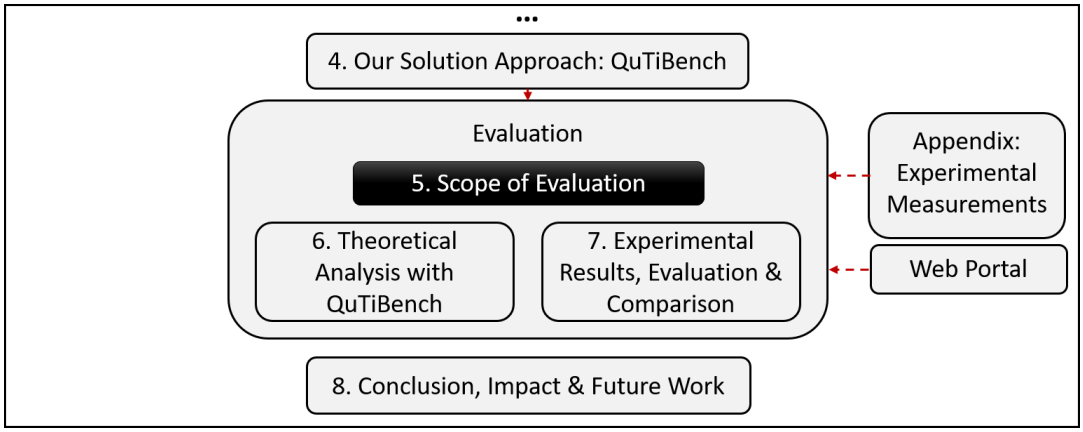


Figure 5.1: Location within the thesis

5.2 Scope of Theoretical and Experimental Evaluation

Overall, we aim to select a minimum of experiments on the various levels of the benchmark to validate our initial claims regarding the benchmarking approach, which were as follows: First of

all, the multi-tiered approach provides interesting trade-offs in regards to representative results versus performance and energy prediction and insights into system bottlenecks. This addresses the value in microbenchmarks and theoretical baseline. Secondly, design space visualization via pareto curves is an effective means to evaluate the design space. Or said differently, for specific machine learning tasks, the benchmark allows to identify optimal solutions within the design space using pareto curves and that we are able to find better solutions than other benchmarks. Thirdly, the proposed concepts are applicable to heterogeneous hardware and specific different types of popular hardware architecture such as GPUs, CPUs, VLIWs, TPU and Field Programmable Gate Arrays (FPGAs), as well as different types of algorithmic choices.

To that extent, we set the following scope for the evaluation: We will benchmark at least three different machine learning tasks, to ensure a certain breadth in regards to applications and datasets, specifically Image Classification for ImageNet, CIFAR-10 and MNIST, using different data sets and different topologies, including convolutional neural networks and multilayer perceptrons. We aim to use at least four different neural network topologies for one of these tasks to explore the usability of the pareto curves for design space exploration (namely ResNet50, GoogleNetV1, MobileNetv1, EfficientNet for ImageNet Classification). We target multiple different hardware architecture groups, GPUs, CPUs, VLIWs, TPU and FPGAs, as these are some of the most popular approaches within the embedded design space. Finally, we are using both quantization and pruning as forms of algorithmic optimization strategy for all of these machine learning tasks. We have pruned and quantized variants of CNN topologies in all three machine learning tasks to ensure that the methodology can adequatly expose the benefits of the various design space trade-offs and identify the best optimization strategies.

In total, we ran close to a thousand experiments on 9 different hardware platforms including 5 different FPGA implementations, GPU, TPU, VLIW and CPU implementations, targeting specifically the embedded space. We leveraged 9 different topologies, and trained 45 CNNs for different pruning factors and for different precisions. For all of these combinations, we swept over all deployment parameters, specifically batch sizes, thread counts, stream sizes, in all possible operating modes, measuring throughput, power, accuracy and latency. In the following, we provide more details on this.

5.2.1 Selected Applications, Topologies & Optimizations

Applications and Topologies. For evaluation, we leverage the following three machine learning tasks: ImageNet, CIFAR-10 and MNIST classification. As CNN topologies, we utilize ResNet50 (RN50) [53], MobileNetv1(MNv1) [155], EfficientNet [156] in three different sizes (large, medium and small), and GoogLeNetv1 (GN) for ImageNet, a VGG16 derivative named CNV [11] with five convolutional layers and three fully connected layers for CIFAR-10, and finally a multilayer perceptron (MLP) for MNIST classification with four fully connected layers [11]. The mapping of topologies to machine learning task is summarized in Table 5.1. We believe that this scope provides sufficient application breadth to carry out a meaningful evaluation of the proposed benchmarking methodology. Please note, as training experiments are highly time-intensive with no necessarily guaranteed outcome, we only trained topologies at the given precision for the cases where the hardware platform could actually support it. (For example, we didn't train ResNet50 in INT2 or INT4 variants.)

Table 5.1: Machine learning tasks and corresponding CNN topologies

	ImageNet Classification	**CIFAR-10 Classification**	**MNIST Classification**
GoogLeNetv1	yes	-	-
MobileNetv1	yes	-	-
EfficientNet Edge small	yes	-	-
EfficientNet Edge medium	yes	-	-
EfficientNet Edge large	yes	-	-
ResNet50	yes	-	-
CNV	-	yes	-
MLP	-	-	yes -

Optimizations. In regards to optimizations, we focus on pruning and quantization specifically, as two of the most popular forms of optimizations, which were both explained in detail in Chapter 2. In regards to quantization experiments, we include the following versions of CNN topologies with different precisions: GoogLeNetv1, and ResNet50 in half and full floating point precision and fixed point 8-bit integer (FP16, FP32, INT8), MobileNetv1 in INT8, and CNV and MLP in 4-bit and 2-bit fixed point integer (INT4, INT2) and FP16. In regards to pruning, we restricted ourselves to channel pruning exclusively. We express the degree of pruning as a percentage, for example 100% for the baseline. Unfortunately the definition of the percentage is not perfectly consistent which is an artifact of the associated typically black-box tooling. Each tool has its own definition of the pruning percentage. In the context of CNV and MLP it relates to the number of inner channels associated with inputs and outputs of hidden layers. In

regards to ResNet50, the pruning percentage is as reported by the tool [157]. All model variants and their corresponding application-level accuracies, for example top1 and top5 classification accuracy for ImageNet, are summarized in Table 5.2.

Table 5.2: Experimental CNNs and their accuracy over all pruning and quantization variants

	Accuracy				
	INT2	**INT4**	**INT8**	**FP16**	**FP32**
	top1 (top5) [%]	**top1 (top5) [%]**	**top1 (top5) [%]**	**top1 (top5) [%]**	**top1 (top5) [%]**
GoogLeNetv1	nm	nm	**69.24 (88.45)**	**66.93 (87.83)**	**66.96 (87.84)**
MobileNetv1	nm	nm	**69.57 (87.71)**	nm	nm
EfficientNet small	nm	nm	**77.0**	nm	nm
EfficientNet medium	nm	nm	**78.6**	nm	nm
EfficientNet large	nm	nm	**80.2**	nm	nm
ResNet50 100%	nm	nm	**73.29 (91.26)**	**75.14 (92.12)**	**75.15 (92.11)**
ResNet50 80%	nm	nm	**73.30 (91.40)**	nm	nm
ResNet50 50%	nm	nm	**69.49 (91.00)**	nm	nm
ResNet50 30%	nm	nm	**68.83 (90.16)**	nm	nm
CNV 100%	**86.86**	**87.40**	nm	**87.02**	**87.06**
CNV 50%	**84.29**	**84.88**	nm	**85.55**	**85.60**
CNV 25%	**79.89**	**81.09**	nm	**83.28**	**83.25**
CNV 12.5%	**73.64**	**75.85**	nm	**77.82**	**77.84**
MLP 100%	**98.75**	**98.77**	nm	**97.30**	**97.31**
MLP 50%	**98.49**	**98.62**	nm	**97.45**	**97.46**
MLP 25%	**98.04**	**98.29**	nm	**97.49**	**97.44**
MLP 12.5%	**96.85**	**97.54**	nm	**97.95**	**97.15**

CNN Sources and Training Techniques. Wherever possible, the FP32 models (specifically ResNet50 and GoogLeNetV1) were chosen from publicly available sources. The FP32-based CNV and MLP required training from scratch; these were trained using Caffe. In all cases, quantization and pruning of models was performed using the available tools for the target platform. Specifically, the INT8 models attained for the DPU were quantized (with retraining) and pruned using the DECENT tool in DNNDK [157]. The FP16 models which were attained for the TX2 and NCS were directly quantized from FP32 using TensorRT [158] and OpenVINO [159] respectively. The 8-bit models attained for the TPU were sources from TensorFlow's own quantized modelzoo[7] and EfficientNet Edge variant from here: [160]. The 4-bit and 2-bit models attained for FINN and BISMO were trained using the techniques described by Su et al [36] using the code provided in the BNN-PYNQ Github repository with a few modifications in order to achieve a desired number of channels[8].

[7]https://coral.withgoogle.com/models/
[8]https://github.com/Xilinx/BNN-PYNQ

5.2.2 Selected Hardware Platforms

We chose the following selection of platforms, which are summarized in Table 5.4. In total, this includes 9 different hardware platforms including 5 different FPGA implementations, GPU, TPU, VLIW and CPU implementations, targeting specifically the embedded space. As was possible, we ran all 45 trained CNNs on all hardware platforms, whereby for each combination of CNN and hardware, we swept over batch sizes, thread counts, stream sizes, in all possible operating modes, measuring throughput, power, accuracy and latency.

In more detail, for FPGA implementation, we use Xilinx's commercially available DPU platforms, called Vitis AI, implemented on the ZCU102, ZCU104 and Ultra96. We leverage FINN as an example of a fully synchronous dataflow architecture with arbitrary precision support, and BISMO [111] which offers run-time programmable precision (although it targets matrix operations in a more general sense). All FPGA resource data is given in Table 5.3 with exception of the DPU data which is not public. The amount of utilized resources inside the FPGA reflects the compute potential of the implementation. Therefore, we expect that BISMO will be less performant than the other FPGA implementation by orders of magnitudes.

Table 5.3: Resource requirements for FPGA based implementations for all MLP and CNV variants

	FPGA Resources in LUTs, FFs, BRAMs, URAMs			
	MLP100%	**MLP50%**	**MLP25%**	**MLP12.5%**
ZCU104-BISMO	**35.7k, 57k, 275.5, 4**			
ZCU104-FINN-INT4	nm	125k,80k,312,0	166k,122k,312,0	163k, 120k, 312,0
ZCU104-FINN-INT2	nm	158k,230k,282,64	156k,224k,154,64	153k,221k,154,64
	CNV100%	**CNV50%**	**CNV25%**	**CNV12.5%**
ZCU104-BISMO	**35.7k, 57k, 275.5, 4**			
ZCU104-FINN-INT4	nm	153k, 189k, 242.5, 0	114k,148k,137.5,0	69k,90k,63,0
ZCU104-FINN-INT2	148k,158k,445,123	99k,102k,278.5,68	50k,54k,77.5,0	27k,37k,28.5,0
	MLP100%	**MLP50%**	**MLP25%**	**MLP12.5%**

Table 5.4 summarizes all hardware platforms used in the experimentation including all performance characteristics. For FPGAs in regards to theoretical peak performance, we assume 500MHz for LUTs and 666MHz for DSP, 100% LUT and DSP utilization and hardware cost as detailed in [11], independent of the implementation. In regards to CPU implementation, we include an ARM Cortex A53 processor using gemmlowp [6]. Furthermore, we run experiments on Google's TPU Coral USB stick (TPU) and Intel's Neural Compute Stick (NCS) which is an example of a VLIW processor, and finally Nvidia's Jetson TX2 platform as a popular example of a GPU. While the USB sticks are almost a separate category in regards to compute

performance and power consumption ($< 2Watt$), we opted to include them as no comparable TPU or VLIW processor is available in the larger embedded category. With that, all platforms are roughly 1-10TOPs in performance and 1-20Watt in regards to power consumption. Finally, note that we provide separate values for different datatype peak performance and the hardware performance scales with the reduced hardware cost of lower precision operations. Certainly smaller datatypes can be emulated in larger carrier datatypes such as INT2 in INT8, whereby the peak performance equates to peak performance of the carrier datatype.

Table 5.4: Summary of all experimental hardware platforms

Hardware Platforms	INT2	INT4	INT8	FP16	FP32	Power
	[TOPs]	[TOPs]	[TOPs]	[TOPs]	[TOPs]	[Watt]
Ultra96-DPU	na	na	0.96	na	na	na
ZCU104-DPU	na	na	4.60	na	na	na
ZCU102-DPU	na	na	6.71	na	na	na
ZCU104-FINN	30.7	8.8	na	na	na	na
ZCU104-BISMO	30.7	8.8	na	na	na	na
TX2 - maxn	na	na	na	1.33	0.67	15
TX2 - maxp	na	na	na	1.15	0.57	15
TX2 - maxq	na	na	na	0.87	0.44	15
TPU-fast	na	na	4	na	na	2
TPU-slow	na	na	2	na	na	2
NCS (MyriadX)	na	na	1	0.5	na	1
U96-Quadcore A53-INT8	na	na	0.192	na	na	na

5.2.3 Overview of all Experiments

Tables 5.5 and 5.6 summarize all included experiments. While we strive for a complete systematic evaluation of all combinations, we are constrained by what the various hardware platforms offer and many have limitations in regards to supported layer types and with that supported topologies. This applies to the precisions that are supported in both hardware and run-time software and also type of layers. This is the number one reason why there is not a single network that can be run in all precisions on all hardware platforms. Instead many solutions are black box solutions that come with a supported modelzoo and CNNs, but different topologies or precisions cannot be executed on these platforms. For example, the TPU failed to execute both the CNV and MLP, while the FPGA platforms did not support MobileNetv1 at the time of this study. Also, TPU, VLIW, and CPU cannot support precisions below INT8, for example INT2 or INT4, and the GPU does not support any datatype other than FP16 and FP32. Despite these limitations, we believe that the presented datapoints provide sufficient proof to demonstrate the benefits of pruning and quantization on the different platforms. In Tables 5.5 and 5.6 we use the following notation: [selection of datatypes] x [pruning percentage] denotes the tested

cross product of datatypes at different pruning scales for this specific topology and hardware platform combination.

Table 5.5: Overview of all ImageNet experiments

		ImageNet					
		ResNet50	**GoogLeNetV1**	**MobileNet**	**EfficientNet L**	**EfficientNet M**	**EfficientNet S**
FPGA	**ZCU102-DPU**	INT8 x [100%,80%,50%,30%]	INT8	nm	nm	nm	nm
	ZCU104-DPU	INT8	INT8	nm	nm	nm	nm
	Ultra96-DPU	INT8 x [100%,80%,50%,30%]	INT8	nm	nm	nm	nm
	ZCU104-FINN	nm	nm	nm	nm	nm	nm
	ZCU104-BISMO	nm	nm	nm	nm	nm	nm
GPU	**TX2-maxn**	[FP16,FP32]	[FP16,FP32]	nm	nm	nm	nm
	TX2-maxp	[FP16,FP32]	[FP16,FP32]	nm	nm	nm	nm
	TX2-maxq	[FP16,FP32]	[FP16,FP32]	nm	nm	nm	nm
TPU	**TPU-fast clk**	nm	INT8	INT8	INT8	INT8	INT8
	TPU-slow clk	nm	INT8	INT8	INT8	INT8	INT8
VLIW	**NCS**	FP16	nm	nm	nm	nm	nm
CPU	**U96-Quadcore A53**	nm	nm	nm	nm	nm	nm

Table 5.6: Overview of all MNIST and CIFAR-10 experiments

		MNIST	**CIFAR-10**
		MLP	**CNV**
FPGA	**ZCU102-DPU**	nm	nm
	ZCU104-DPU	nm	nm
	Ultra96-DPU	nm	nm
	ZCU104-FINN	[INT2,INT4] x [100%, 75%, 50%, 25%]	[INT2,INT4] x [100%, 75%, 50%, 25%]
	ZCU104-BISMO	[INT2,INT4] x [100%, 75%, 50%, 25%]	[INT2,INT4] x [100%, 75%, 50%, 25%]
GPU	**TX2-maxn**	[FP16,FP32] x [100%, 75%, 50%, 25%]	[FP16,FP32] x [100%, 75%, 50%, 25%]
	TX2-maxp	[FP16,FP32] x [100%, 75%, 50%, 25%]	[FP16,FP32] x [100%, 75%, 50%, 25%]
	TX2-maxq	[FP16,FP32] x [100%, 75%, 50%, 25%]	[FP16,FP32] x [100%, 75%, 50%, 25%]
TPU	**TPU-fast clk**	nm	nm
	TPU-slow clk	nm	nm
VLIW	**NCS**	FP16 x [100%, 75%, 50%, 25%]	FP16 x [100%, 75%, 50%, 25%]
CPU	**U96-Quadcore A53**	[INT2,INT4] x [100%, 75%, 50%, 25%]	[INT2,INT4] x [100%, 75%, 50%, 25%]

5.2.4 Envisioned Future Scope

While the current plan already involves a significant amount of experimentation, it will only address a small part of the overall design space. Ideally the benchmark would address a much broader scope of ML applications and CNN topologies. A provisional list for the future is included in Table 5.7.

Table 5.7: Planned applications, datasets and models

Learning Technique	**Application**		**QuTiBench**	
			Dataset	**Model**
Supervised	**Vision**	**Image Classification**	ImageNet, MNIST	ResNet50, MobileNet(V1), GoogleNet, MLP
	Vision	**Object Detection**	Pascal VOC	SSD-ResNet34, YoloV2
	Vision	**Semantic Segmentation**	Pascal VOC	Mask R-CNN, SSD-MobileNet
	NLP	**Machine Translation**	WMT'14 English-to-French&German	GNMT [161]
	NLP	**Speech Recognition**	Librispeech	DeepSpeech2
	NLP	**Sentiment Analysis**	SST, IMDB, SemEval2018	Multiplicative LSTM
	NLP	**Language Modeling**	babI	Memory Network
	Recommendation	**Movies**	Movielens 20M	NCF
Unsupervised	**Vision**	**Feature Extraction**	MNIST	autoencoder
	Vision	**Adversarial Learning**	ImangeNet	WGAN
Deep Reinforcement Learning	**Game**	**Go**	Go	MiniGo
		Atari ALE	Atari ALE	DeepQ

5.3 Applying our Measurement Methodology

We introduced the measurement methodology in Chapter 4. When applying the methodology to the chosen hardware platforms, there are a few important details to be noted, which we discuss below: In regards to power, we only report platform power measured at the socket to ensure memory subsystems are taken into account. For the USB devices, we used a USB power meter (Innovateking, A3A3-B). While not as accurate as current sampling on the board, we typically found the numbers to be within 10%. Also it allowed us to adopt a consistent way of measuring across all platforms. We conduct all our experiments across the spectrum of all operating and power modes unless specifically annotated. Specifically: The TPU Coral stick from Google can operate with a fast or a slow clock. The Nvidia GPU Jetson TX2 platform can run in either maxn, maxp or maxq modes. Maxn is the high performance mode with highest power consumption. Maxp is the most efficient mode, with lowest power but also lowest performance, and maxq. In regards to other deployment settings, we run a large sweep over batch sizes, thread counts and stream sizes, which we believe to expose representative behaviour of the test system. Table 5.8 summarizes the collected datapoints per experiment with regard to the spectrum of these deployment settings.

Table 5.8: Deployment settings for all measurements

Hardware Platforms	**Deployment Settings**	
	Batch, Thread, Stream Size	**Power or Operating Mode**
Ultra96-DPU	1,2,3,4,5,6,7,8	na
ZCU104-DPU	1,2,3,4,5,6,7,8	na
ZCU102-DPU	1,2,3,4,5,6,7,8	na
ZCU104-FINN	1,2,4,8,16,32,64,128, 256, 512	na
ZCU104-BISMO	2,4,8,16,32,64,128	na
TX2	1,2,4,8,16,32,64,128	maxn, maxp, maq
TPU	1	fast, slow
NCS (MyriadX)	1,2,4,8,16,32,64,128	na
U96-Quadcore A53	2,4,8,16,32,64,128	na

5.4 Concluding Remarks

In this chapter, we set the scope of theoretical and experimental analysis to provide an overview of the experimental effort which included 3 machine learning tasks (ImageNet, CIFAR-10, MNIST classification), with different types of topologies (ResNet50, MobileNetv1, GoogleNetev1, EfficientNet, CNV and MLP) on 9 different hardware platforms. This included 5 different

FPGA based implementations as they can provide the broadest sets of architectures, including a matrix of processing engine based approaches, synchronous feed forward dataflow architectures as well as bitserial implementations. We also included an ARM processor (A53), a Nvidia GPU (Tx2), Google's coral TPU and the Intel Neural compute stick. All experimentation was conducted over a sweep of batch sizes, stream sizes and thread counts and over all possible operating modes, both in respects to system and compute only behaviour, including accuracy, throughput in frames per second (fps) and giga operations per second (GOP/sec), latency in msec, and power consumption breakdown in Watts. Also we explore comparisons with systematically quantized and pruned CNNs. Overall close to a thousand experiments were conducted. In the next chapter, we will evaluate these scenarios theoretically, followed by an experimental analysis in Chapter 7. A key challenge is finding fair comparisons and carrying out systematic experimentation as we are constrained by what the various hardware platforms offer and many have limitations in regards to supported models. Finally, we discussed how the measurement methodology was applied to the hardware platforms. In the next two chapters, we will evaluate our approach using the in this chapter outlined scope of test cases and methods.

6 Theoretical Analysis with QuTiBench

6.1 Introduction

This chapter provides the theoretical analysis of applying level-0 of the benchmarking methodology to all the experiments. It covers compute and memory analysis of the given set of algorithms and platforms, performance predictions with roofline models, as well as an analysis of the expected behaviour of quantization and pruning. This provides instant estimates for expected performance for given use cases, whereby the real experiments are provided in the next section. The scope of theoretical and experimental analysis is as defined in the previous chapter.

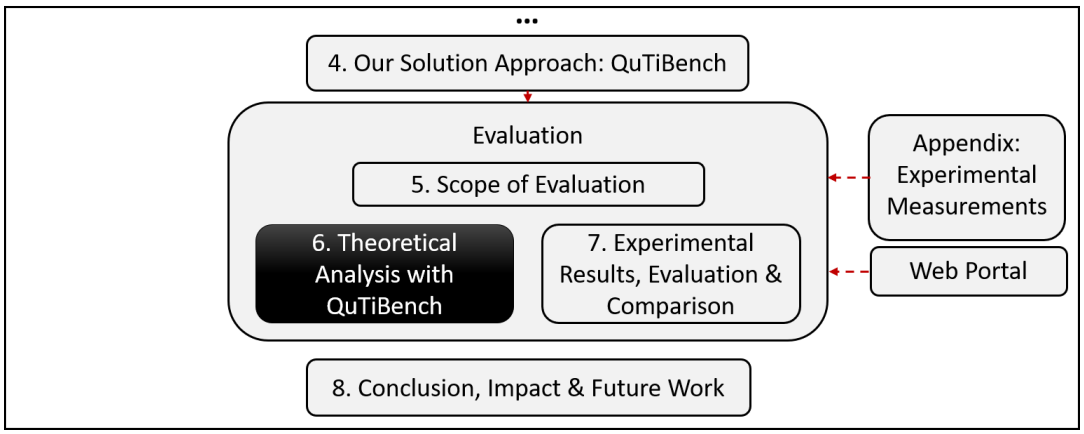


Figure 6.1: Location within the thesis

6.2 Compute and Memory Requirements in the Chosen Applications

As introduced in Section 5, we leveraged 6 different topologies with numerous variants across 3 different categories of machine learning tasks. Table 6.1 summarizes the level-0 results for these topologies. This includes their compute requirements in number of operations per a single input in giga operations (GOPs), their associated model size in millions of elements (ME) independent of datatype used, plus the specific Operational or Arithmetic Intensity (OI) for each individual datatype. Corresponding accuracy figures, which are not part of the theoretical analysis, can be found in Chapter 5. Trained topologies are highlighted in green and untrained topologies are shown in black. For those, we have no corresponding accuracy figures. ResNet50 is by far the highest in regards to compute and memory requirements, followed by GoogleNetv1, which is in complexity somewhere between a 30% and 50% pruned ResNet50. In regards to OI, we show values for all leveraged datatypes on the assumption that the weights reside off-chip and all intermediate results are kept on-chip and batch size equals 1. GoogLeNet is the highest and therefore the least likely to be bound by memory subsystems, while the MLP variants are the lowest. This is expected as MLP consists of exclusively fully connected layers which are memory intensive. We will leverage these numbers for performance predictions in Section 6.4.

Table 6.1: Experimental CNNs and their requirements over all pruning and quantization variants (batch size = 1). This includes compute, model size and operational intensity for different datatypes.

	Total OPs	Total Model Size	OI (INT2)	OI (INT4)	OI (INT8)	OI (FP16)	OI (FP32)
	GOPs	[ME]	[OPs/Byte]	[OPs/Byte]	[OPs/Byte]	[OPs/Byte]	[OPs/Byte]
GoogLeNetv1	3.1	6.0	2,093.97	1,046.99	523.49	261.75	130.87
MobileNetv1	1.1	4.2	1,075.47	537.74	268.87	134.43	67.22
ResNet50 100%	7.7	25.5	1,210.84	605.42	302.71	151.36	75.68
ResNet50 80%	6.5	23.7	1,086.59	543.30	271.65	135.82	67.91
ResNet50 50%	3.8	15.8	949.85	474.93	237.46	118.73	59.37
ResNet50 30%	2.5	10.1	970.16	485.08	242.54	121.27	60.64
EfficientNet Edge S	4.7	5.4	3481.48	1740.74	870.37	435.18	217.59
EfficientNet Edge M	7.4	6.9	4289.86	2144.93	1072.46	536.23	268.12
EfficientNet Edge L	19.4	10.6	7313.21	3656.60	1828.30	914.15	457.08
CNV 100%	0.47	6.16	304.95	152.48	76.24	38.12	19.06
CNV 50%	0.12	1.54	308.32	154.16	77.08	38.54	19.27
CNV 25%	0.03	0.39	315.01	157.51	78.75	39.38	19.69
CNV 12.5%	0.01	0.10	332.61	166.30	83.15	41.58	20.79
MLP 100%	0.02	10.01	8.00	4.00	2.00	1.00	0.50
MLP 50%	0.01	2.91	8.00	4.00	2.00	1.00	0.50
MLP 25%	0.00	0.93	8.00	4.00	2.00	1.00	0.50
MLP 12.5%	0.00	0.33	8.00	4.00	2.00	1.00	0.50

6.3 Theoretical Peak Performance of Selected Hardware Platforms

We have introduced the selection of hardware platforms in the previous chapter. In Table 6.2, we summarize the level-0 characteristics of these platforms, which include theoretical peak performance for different datatypes, memory bandwidth, memory capacity and thermal design power. Graphics Processing Units (GPUs) are highest in regards to memory subsystem for both access bandwidth and capacity. In regards to power consumption, the USB accelerators (Neural Compute Stick (NCS) and Tensor Processing Unit (TPU)) stand out with ranges from 1-2Watt, but are also by far the lowest in regards to performance (maximum of 4 tera operations per second (TOP/sec))[9].

Table 6.2: Experimental hardware platforms and their peak performance, power and memory bandwidth and capacity

Hardware Platforms	INT2	INT4	INT8	FP16	FP32	Memory Bandwidth	Memory Capacity	Power
	[TOP/sec]	[TOP/sec]	[TOP/sec]	[TOP/sec]	[TOP/sec]	[GB/sec]	[GByte]	[Watt]
Ultra96-DPU	na	na	0.96	na	na	4.26	2	na
ZCU104-DPU	na	na	4.60	na	na	19.20	4	na
ZCU102-DPU	na	na	6.71	na	na	19.20	4	na
ZCU104-FINN	30.7	8.8	na	na	na	19.20	4	na
ZCU104-BISMO	30.7	8.8	na	na	na	19.20	4	na
TX2 - maxn	na	na	na	1.33	0.67	59.70	8	15
TX2 - maxp	na	na	na	1.15	0.57	59.70	8	15
TX2 - maxq	na	na	na	0.87	0.44	59.70	8	15
TPU-fast	na	na	4	na	na	25.6	1	2
TPU-slow	na	na	2	na	na	25.6	1	2
NCS (MyriadX)	na	na	1	0.5	na	12.8	2	1
U96-Quadcore A53	0.192	0.192	0.192	na	na	4.26	2	na

For performance, the highest performance is achievable with the lowest precisions in Field Programmable Gate Arrays (FPGAs), which is the only device where hardware can leverage the reduced cost of the corresponding smaller operators. The compute performance scales with the smaller datatypes as far as they are natively supported in all platforms. For example for the TX2, the 16-bit Floating Point Representation (FP16) performance is double compared to 32-bit Floating Point Representation (FP32). Peak performance and memory bandwidth feed into the performance predictions in the next section. Whereby when the lower precision datatype is not supported in a given hardware platform, it can be emulated in the carrier datatype. The performance in Table 6.2 is the shown as equal to the carrier datatype. For example, INT2 and INT4 are emulated in an INT8 carrier datatype for the Quadcore ARM processor. For FPGA platforms, we assume 100% resource utilization. For INT8, as assume 2 operations (vector

[9]We only show the peak performance for the datatypes that were leveraged in the experimentation.

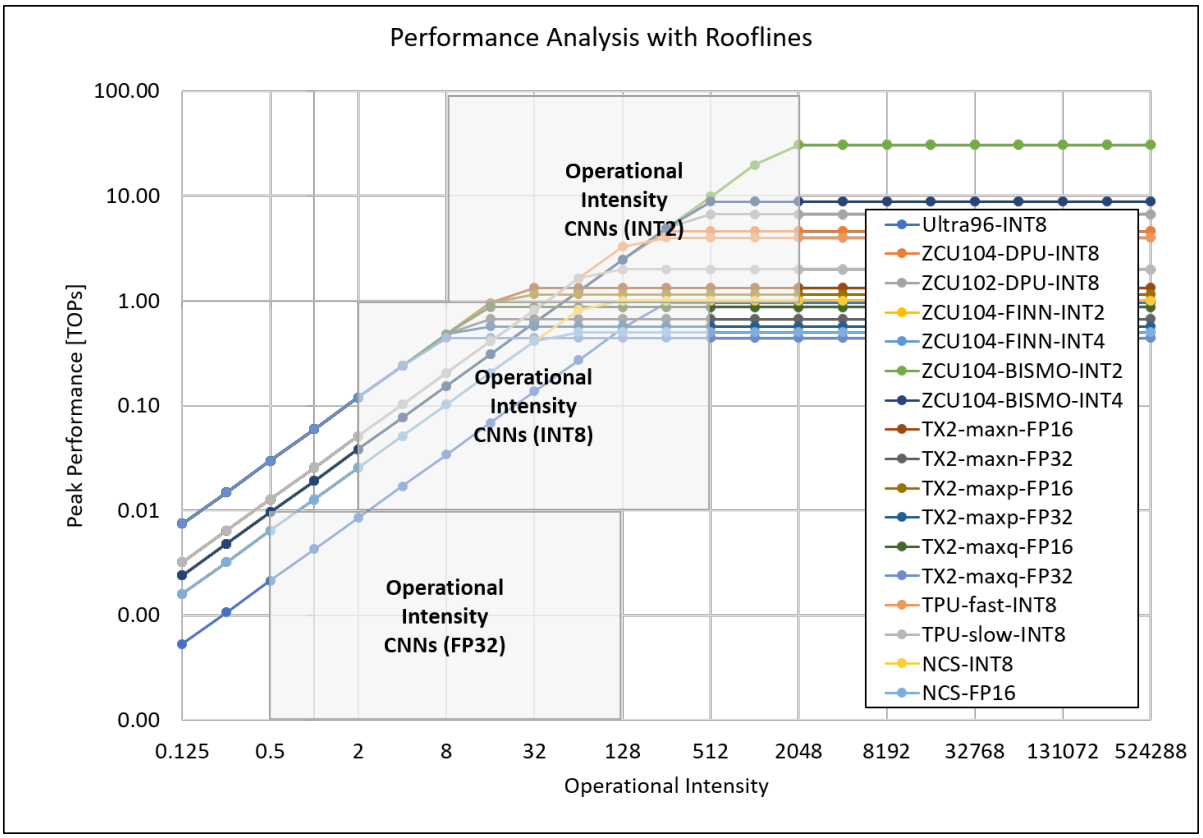


Figure 6.2: Effect of quantization on operational intensity and potential performance impact for different hardware platforms - visualized with rooflines

width of 2) for each DSP slice. Finally clock frequency is estimated at 666MHz for DSPs and 500MHz for LUTs.

6.4 Performance Predictions with Rooflines

As discussed in Chapter 4, rooflines can be leveraged for performance prediction whereby the maximum peak performance achievable for a given CNN lies at the intersection point between the operational intensity and the roofline of the hardware platform. In Figure 6.2, we visualize this for all Convolutional Neural Network (CNN) topologies that are part of the evaluation for a specific datatype. For example the CNN's operational intensity for INT8 ranges from 2 - 523 OPs/Byte. We show the range for INT2, and INT8 and FP32. From this it is visible that for FP32, the execution for the majority of all CNNs will be memory bound on the majority of platforms. With reduced precision, the intersection points migrate to the compute bound part of the rooflines, however even for INT2, the topologies are still memory bound on half of the

platforms. Again, this is under the assumption that weights have to reside off-chip and batch = 1. With larger batch sizes, the operational intensity moves to the right.

6.5 Expected Performance

With the following heatmaps, we visualize the theoretical peak performance for all experiments split over the various machine learning tasks, ImageNet, CIFAR-10 and MNIST classification to be specific. These are computed as described in Chapter 4.

Here are some of the computed results and visualized as heatmaps in Figures 6.3, 6.4 and 6.5. The donation in the heatmaps was also explained in Chapter 4. For ImageNet, we expect MobileNetv1 to deliver the highest performance, in particular on the TPU, which is as expected as it is significantly lower in compute than alternative networks. We can observe how the pruned and quantized variants should bring significant performance scaling. For example comparing ResNet50 pruned variants and GoogleNetv1 in INT8, FP16 and FP32. For CIFAR-10 and MNIST, combined quantization and pruning should deliver the highest performance. FPGAs which are the only platform offering native support for INT2 and INT4 are expected to shine here. We will discuss the reasons in more detail in the following subsection.

ImageNet Classification [input/sec]	GoogleNetv1-INT8	GoogleNetv1-FP16	GoogleNetv1-FP32	MobileNetv1-INT8	ResNet50 100%-INT8	ResNet50 100%-FP16	ResNet50 100%-FP32	ResNet50 80%-INT8	ResNet50 50%-INT8	ResNet50 30%-INT8	EfficientNet S-INT8	EfficientNet M-INT8	EfficientNet L-INT8
Ultra96-INT8	307				124			147	256	392	204	130	50
ZCU104-DPU-INT8	1470				596			703	1216	1878	979	622	237
ZCU102-DPU-INT8	2144				753			798	1216	1901	1428	907	346
TX2-maxn-FP16		425				172							
TX2-maxn-FP32			214				87						
TX2-maxp-FP16		367				149							
TX2-maxp-FP32			182				74						
TX2-maxq-FP16		278				113							
TX2-maxq-FP32			141				57						
TPU-fast-INT8	1278			3509	518						851	541	206
TPU-slow-INT8	639			1754	259						426	270	103
NCS-INT8	319												
NCS-FP16		160				65							

Figure 6.3: Theoretical performance predictions for ImageNet via heatmaps

CIFAR-10 Classification [input/sec]	CNV 100%-INT2	CNV 100%-INT4	CNV 100%-FP16	CNV 100%-FP32	CNV 50%-INT2	CNV 50%-INT4	CNV 50%-FP16	CNV 50%-FP32	CNV 25%-INT2	CNV 25%-INT4	CNV 25%-FP16	CNV 25%-FP32	CNV 12.5%-INT2	CNV 12.5%-INT4	CNV 12.5%-FP16	CNV 12.5%-FP32
ZCU104-FINN-INT2	12458				49331				201600				638592			
ZCU104-FINN-INT4		6229				24666				100800				319296		
ZCU104-BISMO-INT2	12458				49331				201600				638592			
ZCU104-BISMO-INT4		6229				24666				100800				319296		
TX2-maxn-FP16			2830				11083				44333				133000	
TX2-maxn-FP32				1426				5583				22333				67000
TX2-maxp-FP16			2447				9583				38333				115000	
TX2-maxp-FP32				1213				4750				19000				57000
TX2-maxq-FP16			1851				7250				29000				87000	
TX2-maxq-FP32				936				3667				14667				44000
NCS-FP16			1038				4111				16667				50000	

Figure 6.4: Theoretical performance predictions for CIFAR-10 via heatmaps

MNIST Classification [input/sec]	MLP 100%-INT2	MLP 100%-INT4	MLP 100%-FP16	MLP 100%-FP32	MLP 50%-INT2	MLP 50%-INT4	MLP 50%-FP16	MLP 50%-FP32	MLP 25%-INT2	MLP 25%-INT4	MLP 25%-FP16	MLP 25%-FP32	MLP 12.5%-INT2	MLP 12.5%-INT4	MLP 12.5%-FP16	MLP 12.5%-FP32
ZCU104-FINN-INT2	7680				25600				38400				229709			
ZCU104-FINN-INT4		3840				12800				38400				114855		
ZCU104-BISMO-INT2	7680				25600				76800				229709			
ZCU104-BISMO-INT4		3840				12800				38400				114855		
TX2-maxn-FP16			2985				9950				29850				89281	
TX2-maxn-FP32				1493				4975				14925				44641
TX2-maxp-FP16			2985				9950				29850				89281	
TX2-maxp-FP32				1493				4975				14925				44641
TX2-maxq-FP16			2985				9950				29850				89281	
TX2-maxq-FP32				1493				4975				14925				44641
NCS-FP16			640				2133				6400				19142	
U96-Quadcore A53	1704	852			5680	2840			17040	8520			50967	25483		

Figure 6.5: Theoretical performance predictions for MNIST via heatmaps

6.5.1 Expected Behavior of Optimizations

In this subsection we analyze mathematically what the expected impact of pruning and quantization should be, beginning with quantization. When reducing the precision, the amount of external memory required for weight storage naturally decreases, and therefore the operational intensity increases and the resulting implementation is more likely to be compute bound. At the same time, the peak performance of the platform increases when native hardware support for the datatype is available and expected performance jumps to the reduced precision peak performance. Therefore, the benefit is compounded. For a ZCU104, from INT16 to INT4, this implies an improvement by a factor of 16x. These trade-offs are depicted in Figure 6.6.

In regards to pruning, the characteristics of the hardware platform do not change, however

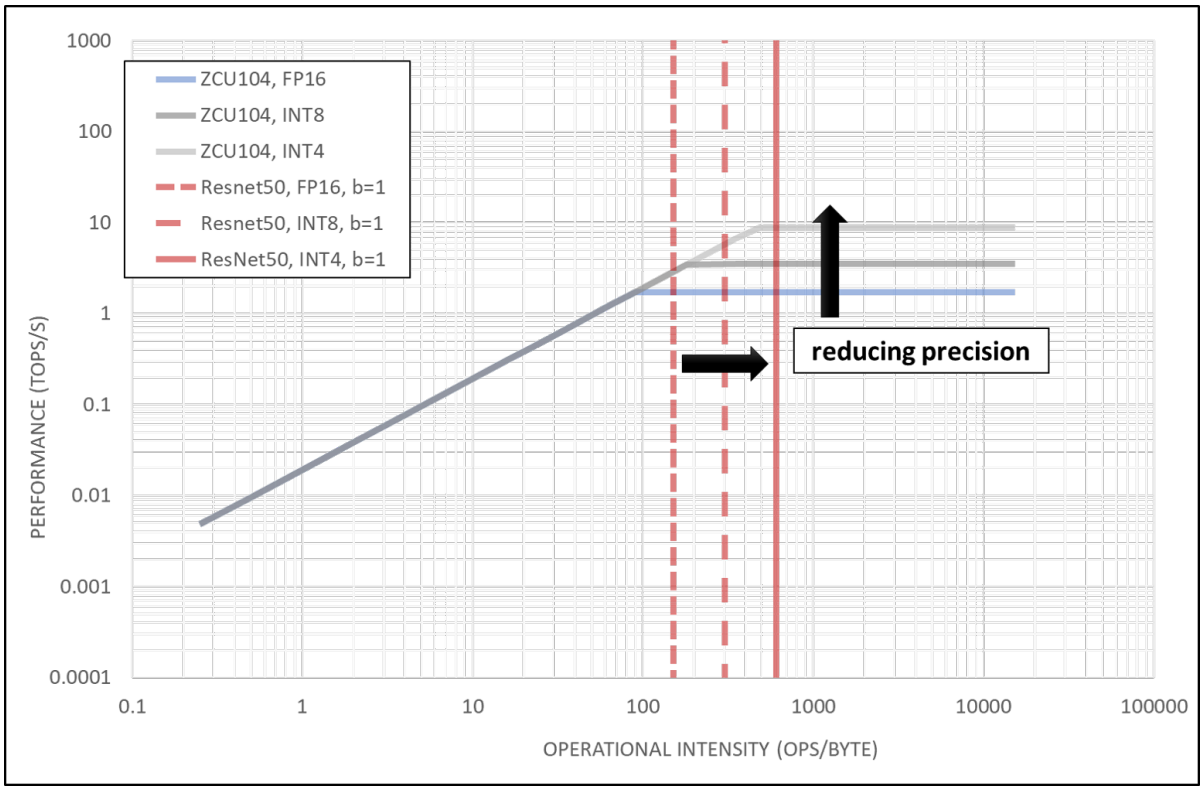


Figure 6.6: Reducing precision scales performance and increases OI

both the required modelsize as well as the compute load are reduced, and in fact the resulting operational intensity of the algorithms is roughly the same. This is detailed in Table 6.3. In regards to achievable performance, this means the following: The overall compute performance in regards to operations per second does not change as per roofline analysis, but the required amount of compute per input has reduced. Therefore the resulting throughput measured in fps should scale linearly with the reduction achieved through pruning in compute operations. For example, performance for a 50% pruned variant shows 4x less compute. As a result, we expect the performance to quadruple. Table 6.4 visualizes the reduction in compute, memory and behaviour of the operational intensity as a function of the pruning percentage.

The graph in Figure 6.7 shows that there is a significant reduction in both compute and memory, even quadratic in regard to pruning percentage for CNV and MLP, while the arithmetic density, which is defined as compute operation per memory byte read or written[10], remains mostly level.

[10]assuming only the model remains externally in memory

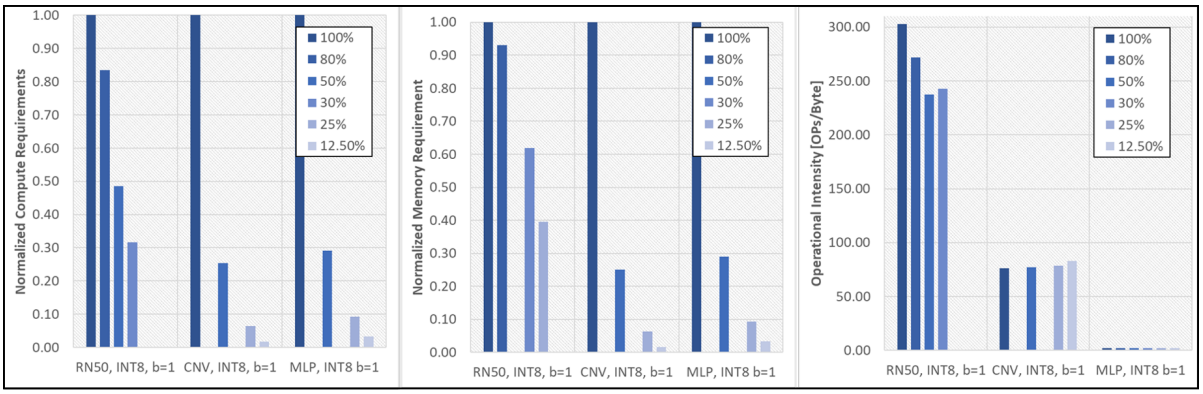


Figure 6.7: Impact of filter pruning on compute, memory and operational intensity

Table 6.3: Impact of pruning on OI

	Operational Intensity [OPs/Byte]					
Scale	**100%**	**80%**	**50%**	**30%**	**25%**	**12.50%**
ResNet50, INT8, b=1	302.71	271.65	237.46	242.54	na	na
CNV, INT8, b=1	76.24	na	77.08	na	78.75	83.15
MLP, INT8, b=1	2.00	na	2.00	na	2.00	2.00

Table 6.4: Impact of pruning on compute requirements

	Absolute and Relative Compute [MOPS]					
Scale	**100%**	**80%**	**50%**	**30%**	**25%**	**12.5%**
RN50	7,720 (100%)	6,450 (84%)	3,750 (48%)	2450 (32%)	na	na
CNV	469 (100%)	na	119 (25%)	na	31 (6%)	8 (2%)
MLP	20 (100%)	na	6 (29%)	na	2 (9%)	1 (3%)

Table 6.5: Impact of pruning on memory requirements

	Absolute and Relative Memory Requirements [ME]					
Scale	**100%**	**80%**	**50%**	**30%**	**25%**	**12.5%**
RN50	25.5 (100%)	23.73 (93%)	15.8 (62%)	10.09 (40%)	na	na
CNV	6.16 (100%)	na	1.54 (25%)	na	0.39 (6%)	0.10 (2%)
MLP	10.01 (100%)	na	2.91 (29%)	na	0.93 (9%)	0.33 (3%)

6.5.2 Theoretical Estimate of Pareto Optimal Solutions

Leveraging the performance predictions from Section 6.4 and combining them with the accuracy from the trained topologies, including the optimized variants, allows to predict potentially pareto optimal solutions. These are visualized in the Figures 6.8, 6.9 and 6.10. According to these, we would expect pruned and quantized INT4 and INT2 variants on the FPGA-based platforms to dominate MNIST and CIFAR-10 classification. For ImageNet classification, we expect the highest performance from the MobileNetv1 implementation on the TPU with fast clock. Furthermore, we anticipate that the pruned ResNet50 variants on the FPGA platform ZCU102 with INT8 are pareto dominant. We will evaluate these compared to the real measured

results in Chapter 7, where we superimpose the theoretical with the measured pareto frontiers.

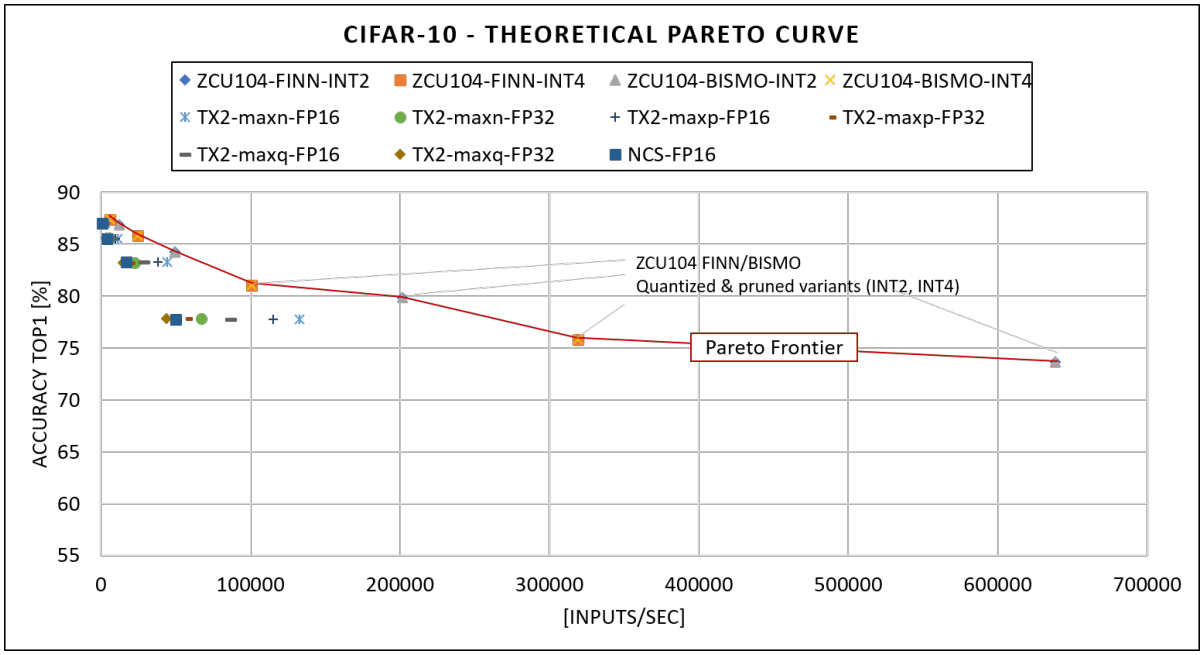


Figure 6.8: Theoretical estimate of pareto-optimal design points for CIFAR-10

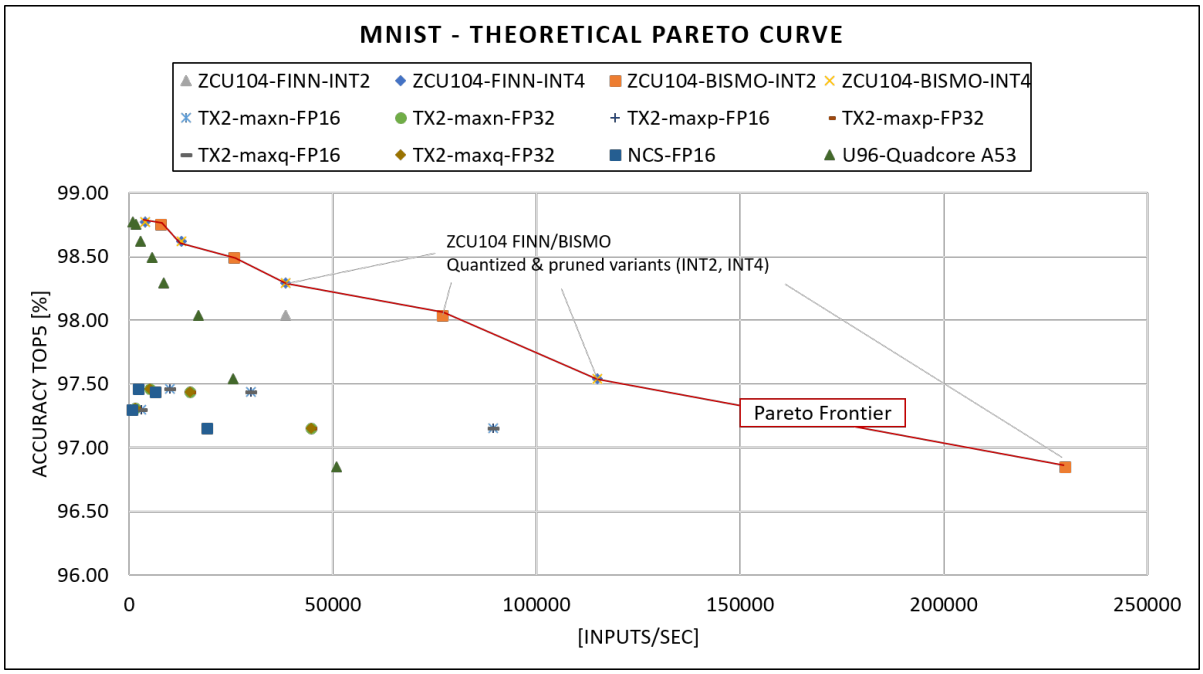


Figure 6.9: Theoretical estimate of pareto-optimal design points for MNIST

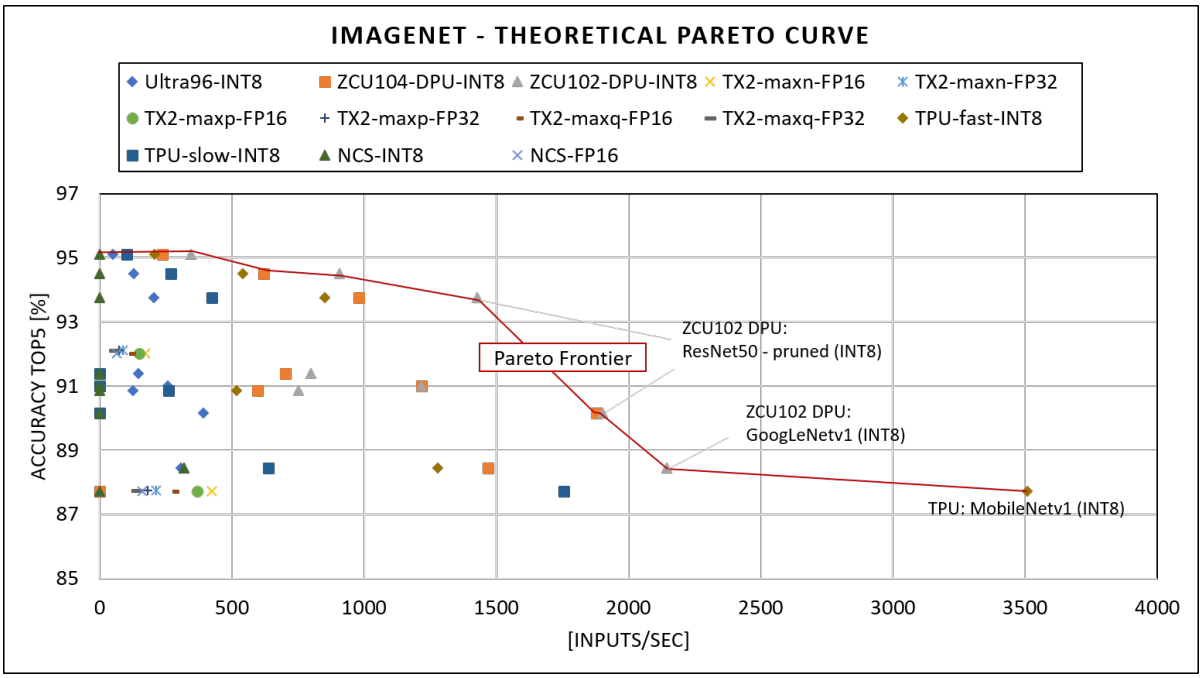


Figure 6.10: Theoretical estimate of pareto-optimal design points for ImageNet

6.6 Concluding Remarks

In this chapter, we submitted all use cases to level-0 analysis as defined in QuTibench. We first analyzed all CNNs for their compute and memory requriements. The MLP and CNV networks were lightest on compute requirements, with comptue requirements being below 1 GOPs per input, while ImageNet classification models required up to 20 GOPs. Model size varied up to 25.5 MEs in the case of ResNet50. The roofline analysis showed that the majority of the workload is memory bound, whereby MLPs have typically the lowest OI. In regards to the hardware platforms, we can roughly divide them into 2 performance-power categories. While all platforms perform in the range of 1-10TOPs (8-bit Fixed Point INTEGER (INT8)) with power consumption being between 1-20Watts, the selected USB sticks are below 2Watts and in the bottom range of the performance together with the ARM processor and BISMO.

In regards to optimizations, we can make the following predictions: Pruning has no effect on operational intensity but as the compute per frame is reduced, the overall frame rate of the algorithm should increase in proportion to the pruning scale. Quantization doubles the benefits as it lowers the amount of data read and written from memory, thereby increasing the operational intensity. Furthermore, quantization also increases the theoretical performance of the platform if the new datatypes are natively supported. Resulting from this, we can carry out

performance predictions: We expect MobileNetv1 to outperform on ImageNet Classification, and reduced precision and pruned variants to outperform for CIFAR-10 and MNIST using our theoretical pareto charts.

In the next chapter, we will present the actual measured data. Correlating this with the performance predictions from this chapter, should give us a good indication whether the level-0 data is representative of the actually achievable performance and as such whether level-0 analysis is useful as a first estimate. Further, the performance predictions may also be useful to guide future optimizations as they provide a reference point of what the hardware platform could be in principal capable of.

7 Experimental Results, Evaluation & Comparison

7.1 Introduction

In this section we evaluate the key characteristics of QuTiBench with experimental data, whereby the scope of the evaluation was defined in previous chapters. We restrict ourselves to a subset of the datapoints and present only what is relevant to the evaluation. The complete measurements together with all data visualizations can be found in the appendix as well as on our web portal located here: `https://rcl-lab.github.io/QutibenchWeb`. In the following, we will revisit each contribution of the benchmark separately and evaluate QuTiBench towards the specific aspects. In particular, these are:

- Multi-tiered approach with theoretical analysis for performance predictions, to track compute efficiency throughout and predict pareto-optimal design points
- Microbenchmarks to help understand system bottlenecks
- Clear definition of measurement methodology and consideration of deployment parameters
- Enabling comparisons within a multi-dimensional design space to support algorithmic optimizations using data visualization

We will show that the theoretical predictions can give reasonable insights without having to run any experimentation including predicting performance which optimizations in 7.2 and optimal design points in 7.3. Regarding the microbenchmarking (level-1 and level-2 results), we ran into many practical constraints that limited the scope of the evaluation, however some interesting results could be obtained: 7.4. The measurement methodology with the distinction

between system and compute level performance has been shown to be very useful to illustrate data movement constraints on the various hardware platforms 7.5. The systematic exploration of other deployment settings provides more clarity in the design space. Additionally, the chosen comparisons and data visualizations in Section 7.6 aid providing fair comparisons between fundamentally different hardware platforms, different topologies and include the scope of potential optimizations. All of this is discussed in greater detail within this chapter. We end with a detailed comparison to the most prominent existing benchmark 7.7.

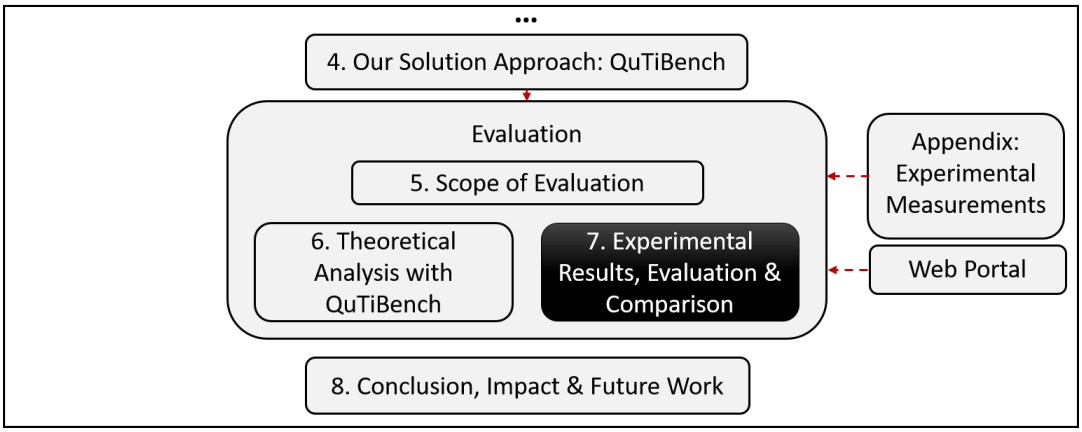


Figure 7.1: Location within the thesis

7.2 Relating Theoretical to Measured Performance

The aim of the multi-tiered approach is to provide a range of compromises for benchmarking in regard to quality of prediction and effort, whereby the theoretical level serves as a rough first guess for performance on different systems as well as a measuring stick for experimental data. The key question in this regard is: How representative are the performance predictions and can we achieve performance close to this in real deployments? In the next section we will analyze whether the theoretical analysis can predict the pareto-optimal solutions with level-0 and get meaningful estimates for optimization schemes, but first we will focus on achieved performance compared to predicted performance and compute efficiency.

For this, we will use efficiency charts in forms of bar charts, which show on the y-axis predicted performance in inputs per second (ips) versus measured performance in ips for the same topology and hardware combination. Further, we are annotating the measured performance with achieved percentage out of predicted performance. When the measured performance exceeds

the predicted performance, then I'm adding another bar for theoretical peak performance as for example for the quantized MLP models on FINN. As a quick reminder, the performance predictions are based on the assumption that all weights reside completely off-chip and that all activations remain on-chip. As such there is no variation over batch size, thread counts and stream sizes. As experimental data points, we will use the best measured performance measured over all batch sizes, thread counts, and stream size. Furthermore, we compare compute performance only, in inputs/sec, as the theoretical model does not include system level assumptions, which would involve potential preprocessing of data, moving data to the accelerator and retrieving results. We'll discuss the different types of hardware platforms separately. Finally, we leverage correlation coefficient to analyze the prediction quality.

7.2.1 FPGA Platforms

DPU

Clearly the performance trends are well estimated for the different topologies for both shown platforms (Ultra96 and ZCU102 using the DPU architecture). The biggest concern is that there is a large difference between predicted and measured performance. As shown in Figure 7.2, the efficiency is fairly low (between 12% and 25%). However, it is consistent across all topologies with correlation coefficient = 0.78. The discrepancy is attributable to the enormous flexibility in FPGAs. The performance predictions assume 100% utilization, whereby the DPU implementation would use only a fraction of that. Also, clock frequency might not be at theoretical peak. Nevertheless, the performance trend is clearly captured by the predictions. The quality of prediction drops with increasingly pruned variants of ResNet50. This is most likely caused by the reduced overall compute in the pruned networks which results in lower achievable utilization of compute resources. Hence, the discrepancy between predicted and measured performance increases with higher pruning scale. This can be addressed in future by introducing an expected compute efficiency factor to peak compute performance and scaling this factor with the pruning degree.

FINN

Similar observations can be made when analyzing performance predictions for FINN for pruned and quantized variants of the neural networks. Overall, efficiency averages at 19%, which is

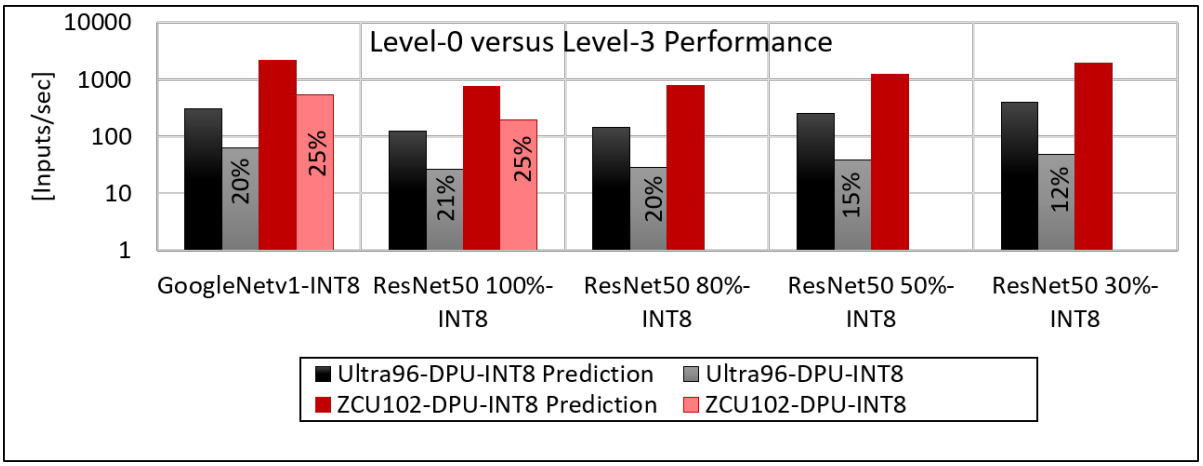


Figure 7.2: Performance: Level-0 versus level-3 for FPGAs with DPU on ImageNet. Actual measurements are within 12-25% of predicted performance.

similar to the above, however with stronger variations. Similarly to the DPU, the efficiency drops with increased pruning scale. However, it seems that for quantization, the smaller precision has slightly better predictions. Finally, for CIFAR-10 the performance predictions seem to represent the trends for both pruning and quantization well, apart from the overall relatively low efficiency. The correlation efficient overall is very high with 0.96. Overall, we expect that predictions can be improved by adjusting resource utilization and clock frequency for peak performance within the assumptions.

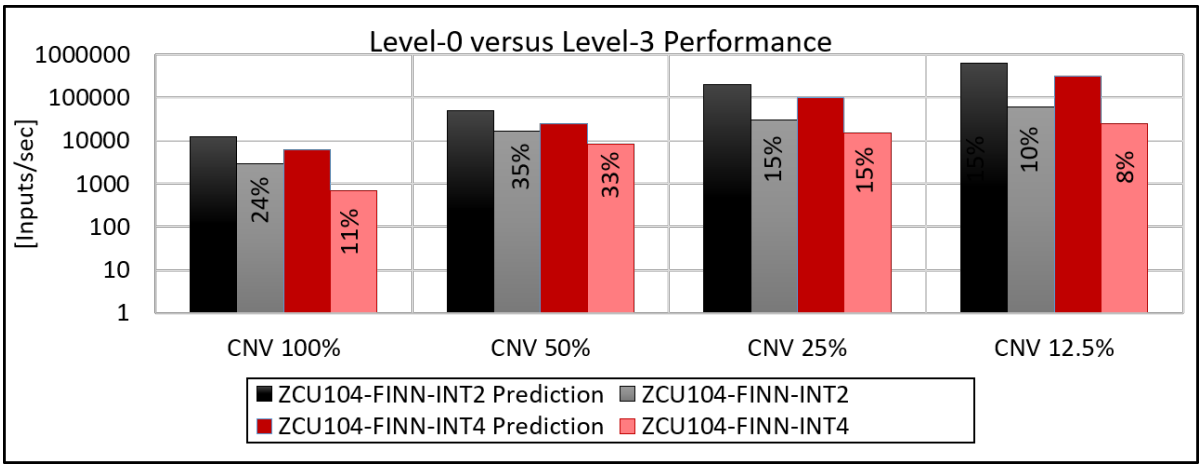


Figure 7.3: Performance: Level-0 versus level-3 for FPGAs with FINN on CIFAR-10. Actual measurements are within 8%-35% of predicted performance.

Regarding MLP topologies, again the predictions relative to each other, across quantized and pruned variants are representative, with a correlation coefficient of 0.93 for pruning and 0.99 for quantization. However, there are substantial differences in the behaviour for MLP networks as shown in Figure 7.4. First, the measured performance exceeds the predictions by 20x on average

as the theoretical analysis assumes the weight memory to be off-chip and the computation being memory bottlenecked. This is due to the performance prediction using a very low operational intensity for these types of networks, which are memory heavy. Therefore the prediction foresees the performance being memory bottlenecked. This is not the case, as the heavily quantized models end up being small enough to fit in on-chip memory (which is not consistent with the original assumption for the theoretical model). As this eradicates the memory bottleneck completely, the measured performance was substantially above the predicted performance. It therefore makes more sense to leverage the theoretical peak compute performance *p_hw_pp*, shown as the striped bars in Figure 7.4, as a reference point. With disregard of memory interfaces, the measured performance in regards ips averages around 11%. To improve the prediction quality for extremely quantized models, we could either ignore memory constraints, or alternatively, refine the model and introduce an on-chip memory capacity assumption.

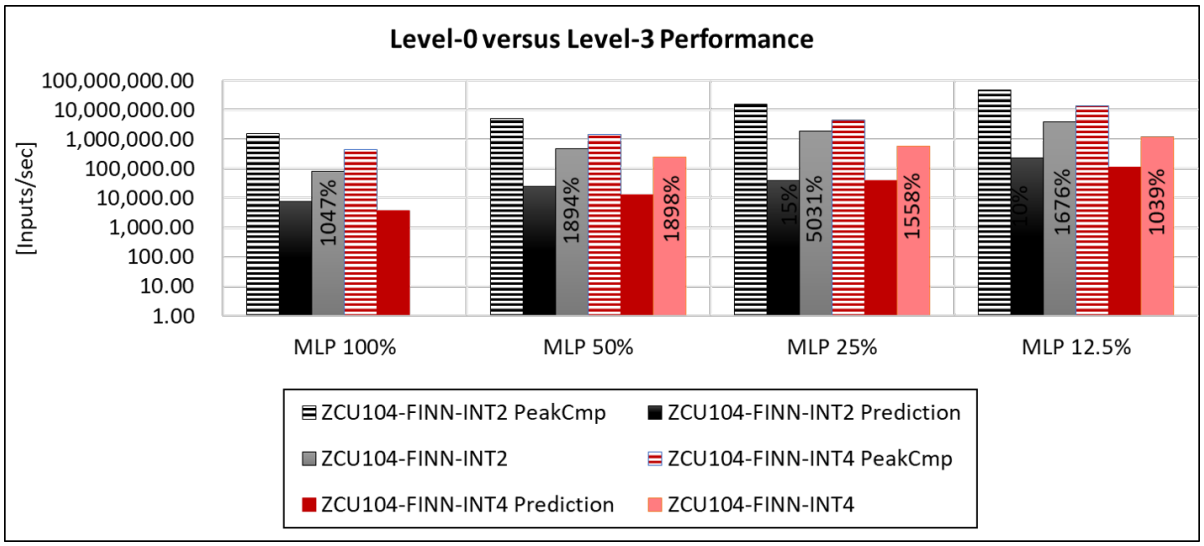


Figure 7.4: Performance: Level-0 versus level-3 for FPGAs with FINN on MNIST, with peak compute performance. Actual measurements are greater than predicted.

Summary for FPGAs

Overall it seems that the level-0 predictions reflect achievable performance well whereby the trends for pruning and quantization are reflected, and achieved measured performance is typically around 10-20% of what is predicted across all types of models. However with increasing pruning scale, the predictions becomes less accurate for all networks and the same effect can be observed in regard to quantization. This is most likely because overheads are starting to dominate and compute efficiency reduces with smaller model sizes and compute complexity. An interesting

artefact was observed for MLP type of topologies, where the quantized models became so small that they were completely retained in on-chip memory. This was not reflected in the level-0 model. As this eradicated the memory bottleneck completely, the measured performance was substantially above the predicted performance in the end, however on average 11% of the theoretical peak compute performance. In future, the performance predictions could be improved by taking actual resource utilization and achieved clock frequency of the FPGA circuit into account. In addition integrating an on-chip memory model for level-0 analysis could provide more accurate estimates for heavily quantized CNNs. This would simply require a model size versus on-chip memory capacity comparison. If the model fits on-chip, we would then ignore the memory access completely.

7.2.2 GPU Platforms

Regarding ImageNet Classification, as illustrated in Figure 7.5, we observe that efficiency for GoogleNet is up to 37% for FP16 and 52% for FP32 across operating modes. For ResNet50, it seems to be much higher with 60% for FP16 and 71% for FP32. Operating modes are nicely reflected in the performance predictions. However, there seems to be larger variation in efficiency between the topologies. This could be caused by a poor estimation of the operational intensity which when memory bottlenecked can have a significant impact on the performance prediction. For example if intermediate activations exceed on-chip memory capacity, they need to spill to external memory, which would lower operational intensity and correlated efficiency. Alternatively this could relate to topological features. Furthermore, the higher precision variant (FP32) achieves higher efficiency, as do larger topologies (ResNet50 is about twice the size to GoogLeNetv1 - see Chapter 5). In summary, it seems that the performance predictions from level-0 vary greatly, whereby the cause for this is most likely the assumptions for operational intensity. To improve this, similar to the FPGA conclusions, we could refine the memory model in the theoretical analysis, or perhaps exclude it and work exclusively of the theoretical peak performance.

For CIFAR-10, where we can leverage the systematically quantized and pruned topologies, as shown in Figure 7.6, we observe similar trends with an overall high correlation coefficient of 0.96. Predicted performance accurately matches that of experimental results, although we observe that the prediction accuracy diminishes with growing pruning scales and shrinking sizes of the CNNs. The smaller the network, the lower is the efficiency that we achieve (3%-7%).

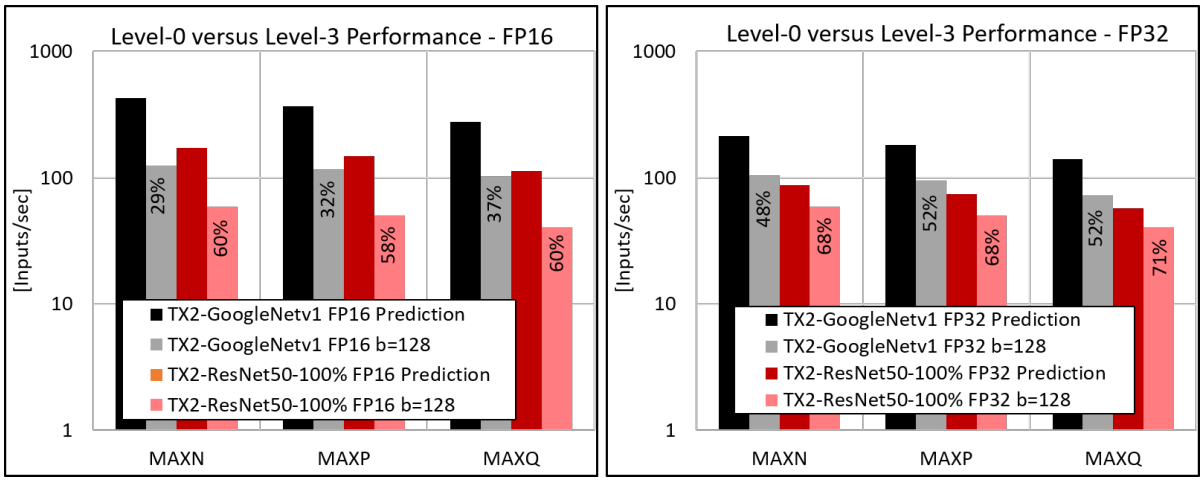


Figure 7.5: Performance: Level-0 versus level-3 for GPUs for ImageNet. Actual measurements are within 23-71% of predicted performance.

However, the correlation coefficient for pruned variants is very high with 0.98. This is consistent with our results for ImageNet and most likely relating to the fact, that with lower compute, we will get increasing effects of overhead. Similarly, for quantization, the efficiency is also lower with smaller datatypes. For example the CNV-12.5% shows FP16 efficiency of 55% - 57%, while the FP32 efficiency is between 69% and 72%, across the various operating modes. Overall, in regards to quantization, the correlation coefficient remains high with 0.98 for FP32 and 0.97 for FP16. Operating modes are nicely reflected in the performance predictions.

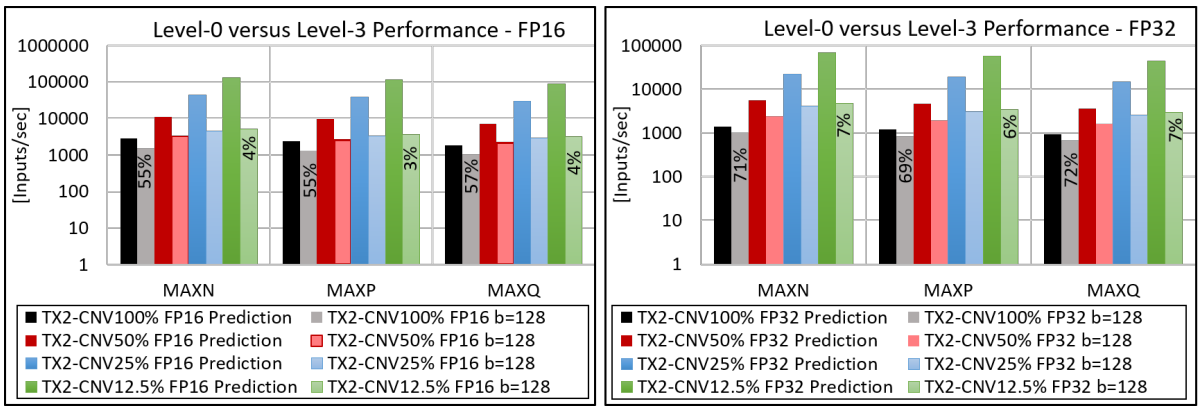


Figure 7.6: Performance: Level-0 versus level-3 for GPUs for CIFAR-10, visualizing a much higher discrepancy in prediction of heavily pruned variants.

Finally, for MNIST classification, large mismatches between predictions and actually measured performance can be observed, similar to our observations with regards to FPGAs. These discrepancies relate to the assumptions in regards to where data is stored, which results in large variations in operational intensity which can easily result in performance predictions being incorrect, and particularly in measured performance being beyond predicted peak. To prove this

point, we added the peak compute performance of the platforms into the graphs for the 100% variants in Figure 7.7, which is well beyond predicted and measured performance. Secondly, similar to all previous results we can observe again that the measured performance is closer to the predicted performance for FP32, compared to FP16. Finally, performance for smaller networks is predicted too optimistically, and efficiency results in being lower.

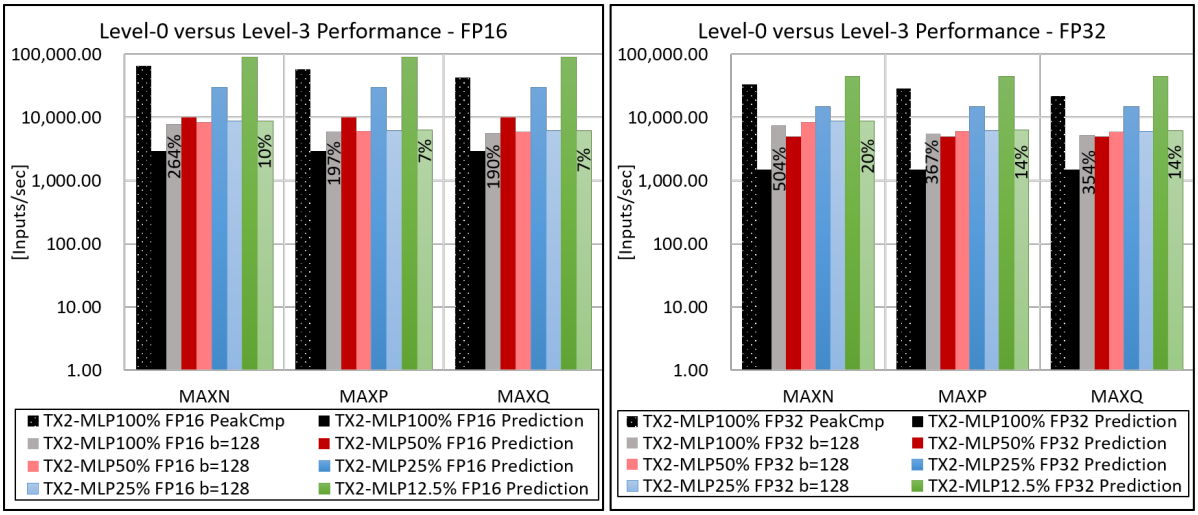


Figure 7.7: Performance: Level-0 versus level-3 for GPUs for MNIST, visualizing a much higher discrepancy in prediction of heavily pruned variants.

In summary, theoretical predictions work for GPUs as well and reflect benefits gained through optimizations (even though a bit too optimistic) and operating modes. Also, we expect that the built-in memory assumptions distort the image and is a key contributor to discrepancies between predictions and measured performance.

7.2.3 TPU & NCS

For ImageNet classification on the TPU, as shown in Figure 7.8, we observe similar behaviour. The general trends are predicted by the theoretical analysis, however the actual efficiency is low and the discrepancies are largest for the smallest CNNs (which is MobileNetv1 in this case). For the NCS with ResNet50, which is not shown in the graph but can be found on our web portal, we achieve 27% of the predicted performance.

In regards to CIFAR-10 classification, again it appears that performance predictions for smaller (more optimized CNNs) are overly optimistic, and efficiency numbers for the most pruned topology, CNV12.5%, is the lowest, whereby overall efficiency is between 1% and 18%. Discrepancies are largest for the smallest CNNs (e.g. MobileNetv1). Introducing an overall

efficiency factor for peak performance in equation 4.2, that decreases with the size of the network could alleviate these discrepancies. Overall, the efficiency numbers for both the TPU and the NCS are on average even lower than for FPGAs and GPUs, but the general trends are predicted by the theoretical analysis with a correlation coefficient=0.78 for TPU and 0.64 for NCS2.

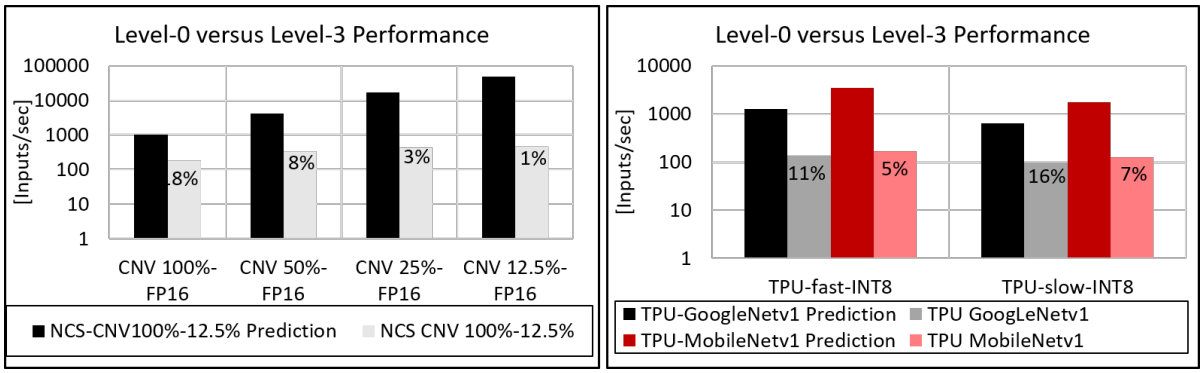


Figure 7.8: Performance: Level-0 versus level-3 for NCS & TPU, showing similar trends in the predictions with large discrepancies.

7.2.4 Summary

In summary, theoretical predictions work. They are representative of achievable performance and reflect improvements achievable through optimization techniques and operating modes. while overall discrepancies exist, the results are well correlated. In the following Table 7.1, we have summarized the correlation coefficients over these experiments, which are consistently high.

Table 7.1: Correlation Coefficient for ImageNet (IN), CIFAR_10 (C) and MNIST (MN) over selected hardware platforms

	DPU (IN)	**FINN (C)**	**FINN (MN)**	**TX2 (C)**	**NCS2 (C)**	**TPU (IN)**
cc	0.78	0.96	0.90	0.96	0.64	0.88

There are clearly limitations in regards to the memory model. As an outcome of this research, we would suggest either refinement of the memory model and assumptions in regard to what data is stored on and off-chip, or alternatively removing of memory bottleneck as given through the roofline model and using only peak compute performance for performance predictions. Having said this, the memory aspect of the roofline is still useful to highlight expected memory challenges in the platforms, but not for quantitative performance predictions unless further

refined. Another limitation which has been consistently observed, is that for smaller CNNs, clearly overheads which are not modelled as part of the theoretical analysis, start dominating performance and as such achieved efficiency is lower than for larger topologies. We expect that we can improve this in future versions by introducing a scalar efficiency factor for performance, which decreases with smaller model size and compute complexity to counterbalance this effect. Finally, for FPGAs specifically the estimates could be much more refined when factoring in actual resource usage and achieved clock frequency.

7.3 Predictions of Pareto-optimal Design Points with Theoretical Analysis

The other key question that we would like to address is whether the theoretical analysis is able to correctly predict the pareto-optimal design points as were presented in Chapter 6. For this we consider the three different classification tasks separately.

For CIFAR-10, the theoretical pareto charts predicted the pruned CNV variants on the ZCU104 with FINN and BISMO using INT2 and INT4 precision as building the pareto frontier. This was shown in Figure 6.8. The actual measurements confirm that ZCU104-FINN with INT2 and INT4 result in pareto-optimal implementations. However, BISMO performance is significantly lower than estimated with the theoretical model. This was anticipated as BISMO leverages only a fraction of the device resources. Clearly, a refinement of the peak performance for the FPGA platforms can avoid this false prediction. Secondly, in the actual measurements, We also observe 2 TX2 with FP16 and pruned variants on the pareto frontier. This is consistent with the observation in the previous Section 7.2.1 that indicated that the actually achieved performance for GPUs was closer to the prediction than for FPGAs. If we were to refine the FINN performance estimates using actually used resources and clock speed, the TX2 data points would have started to emerge in pareto curves of the theoretical analysis. We show an overlapped pareto curve that contains both theoretical predictions as well as the corresponding measured data points in one chart in Figure 7.9. Experimental data points can be correlated with their theoretical counterparts through colour. The theoretical pareto-optimal datapoints and the measured pareto-optimal datapoints are connected through two separate lines in yellow and blue respectively.

For MNIST classification, shown in Figure 7.10, the exact same predictions were made and

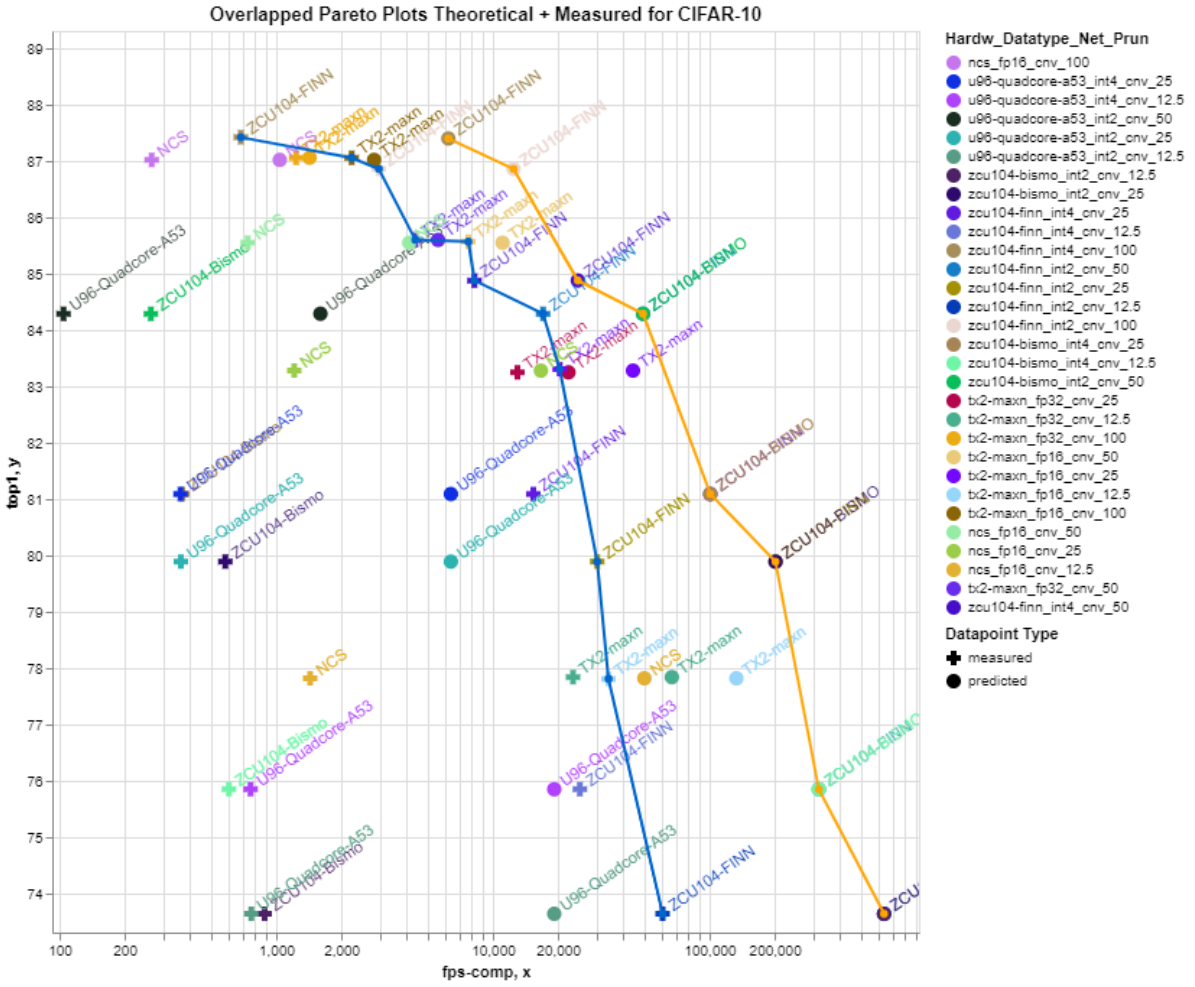


Figure 7.9: CIFAR-10 classification design space with predicted versus measured pareto-optimal design points

the actual measured results are consistent with the predictions with the exception of BISMO, for the reasons as explained above. Finally for ImageNet Classification, the theoretical analysis predicted the following optimal solutions to be ZCU104 and ZCU102-DPU variants of ResNet50, ZCU102-DPU with GoogleNetv1 as well as the TPU for MobileNetv1. This reflects fairly well the actual measurements. Small discrepancies observed are due to missing measurements for the pruned ZCU102 ResNet50 variants and the overly optimistic predictions for FPGAs. Rsults are visualized in Figure 7.11.

7.3.1 Summary

In summary, while the performance predictions where quantitatively perhaps not accurate, the predictions for pareto-optimal design points can save substantial implementations and provide

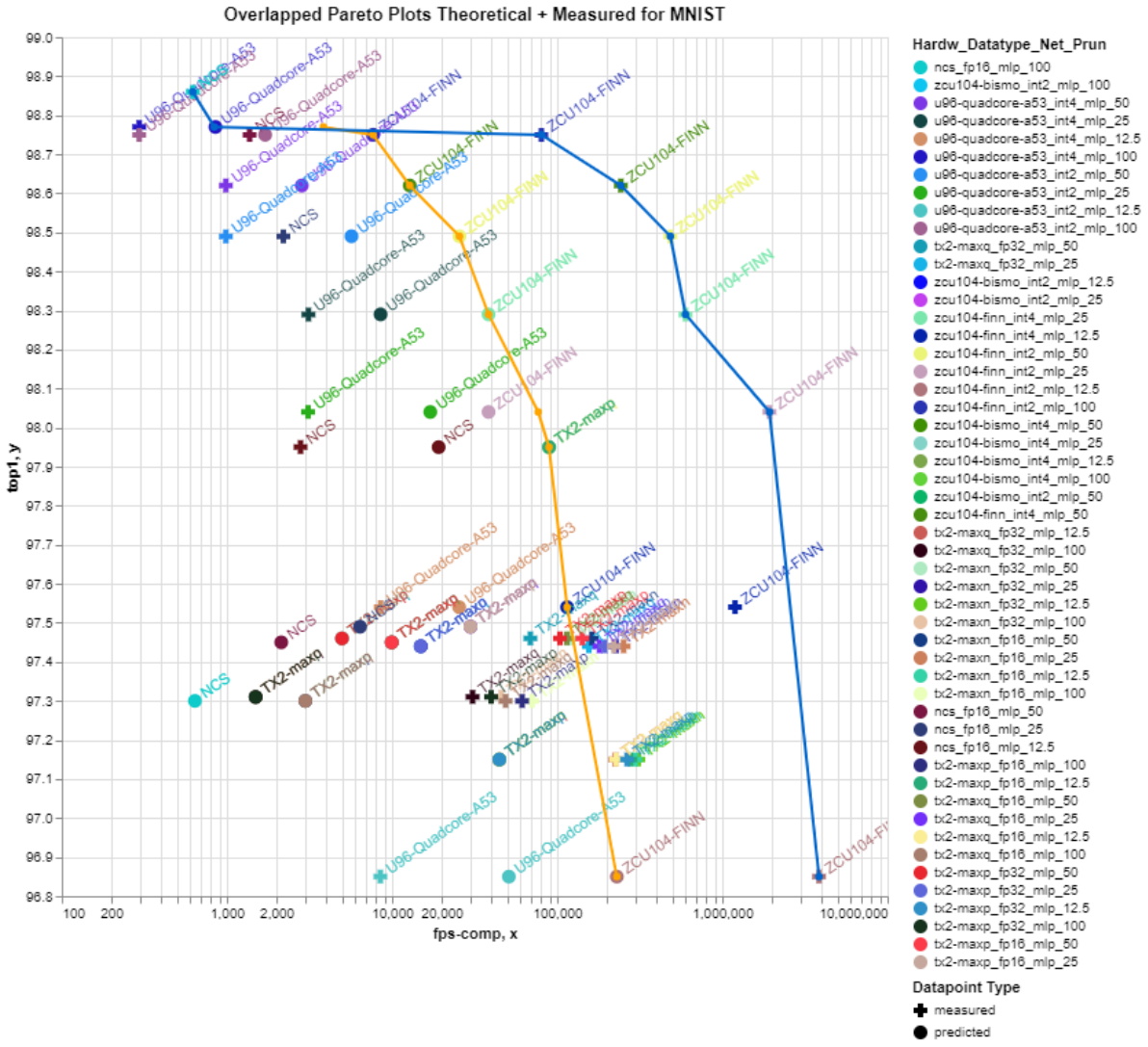


Figure 7.10: MNIST classification design space with predicted versus measured pareto-optimal design points

good guidance on what combinations of topologies, optimization strategies and complementary hardware platforms can provide best implementation alternatives, especially when applying the proposed refinements of the theoretical models.

7.4 Benefits and Challenges with Microbenchmarks

As discussed before, microbenchmarks at level-1 expose achievable compute performance for typical compute patterns for specific layer types, encountered within neural networks. Microbenchmarks at level-2 are comprised of simple combinations of level-1 tests to capture

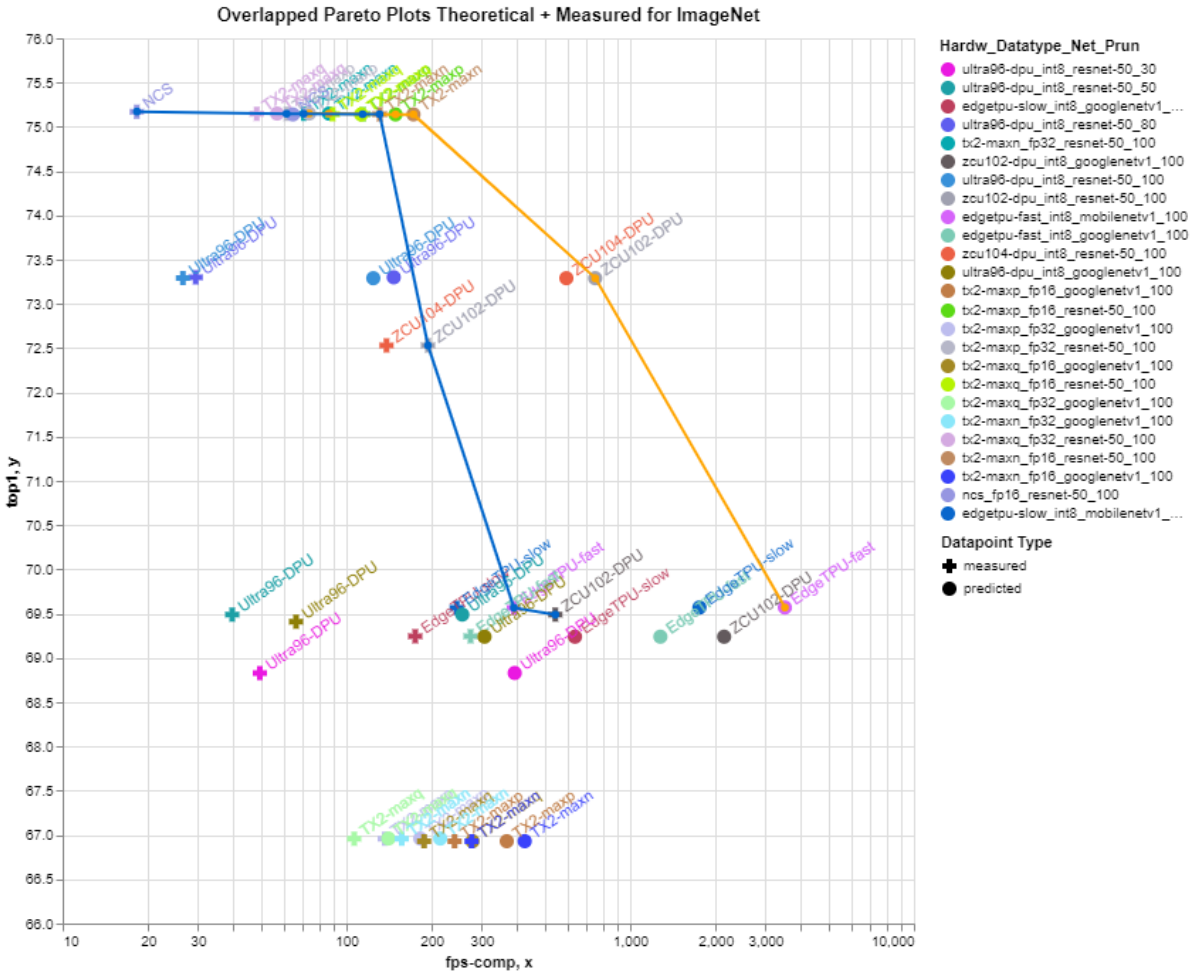


Figure 7.11: ImageNet classification design space with predicted versus measured pareto-optimal design points

potential bottlenecks caused by tensor movement between layers, as well as storage requirements.

We restrict the evaluation of level-1 and level-2 to ResNet50, as this is sufficient to make the key observations. The ResNet50 topology is relatively regular in structure, consisting of a top convolutional layer with pooling combination, 16 residual blocks, and a fully connected layer. Each residual block is comprised of thresholding layers, convolutions, and elementwise additions. Since the platform-specific frameworks perform layer fusion as network optimization, level-1 represents the smallest possible fused layer structure. We restrict level-1 to convolutions of different sizes and select the residual layers res2a, res3a, res4a and res5a to get an overview over the whole network. Results are shown in Table 7.2. The table contains level-1 and level-2 latency results for one TX2 hardware configuration (MaxN, FP16) with different batch sizes as well as level-1 results for ZCU104 with different thread numbers. Complete results can be found in the appendix and the web portal.

One of the key challenges encountered relates to limited support by the hardware-specific tool chains. The cleanest way to describe fair microbenchmarks would be to create topologies of subsets of layers. However many of the vendor-specific toolchains did not support execution of this. These tools are restricted to execution of full topologies which process inputs to classification results rather than converting tensors. We ran into such limitations with most of the hardware platforms. We tried to work around this issue in whatever way possible. To give one example, for the chosen FPGA DPU implementation, the only way to produce level-1 results was through profiling a full network which shows latency per layer. While one could argue that this is okay for level-1 results, this is pointless for level-2 as these would be a simple sum of level-1 equivalent numbers and as such render the results as superfluous with no added information. This limitation through vendor specific tool chains was the largest hindering block to many of the level-1 and level-2 experiments.

In regards to achieved measurements, we observe a large discrepancy in execution time for different residual stacks, even though the compute requirements within each are similar. This is in direct conflict with performance prediction assumptions in level-0, which is based on compute requirements. It is likely that data movement varies significantly depending on the incoming and outgoing tensor dimensions. Therefore, it is important to include as many layer types inside level-1 and 2 testing. We would expect this to be even more pronounced for other topologies, as they may be less balanced than ResNet50. We also observe a large discrepancy between the performance of different convolutional layers (Table 7.2, level-1). Unlike the residual blocks, this is anticipated, as they come with very different compute requirements. Furthermore, the differences are more pronounced with larger batch size. As such it would be essential to include the full spectrum of convolutional layers within level-1, which will make the mircobenchmarking a substantial task. It can further be observed that level-2 layers show latency in strong excess over the sum of the latencies of the contributing level-1 components, clearly highlighting the contribution of data movements to latency. For example, when summing up all level-1 results for **res2a**, with batch size 1 on level-1, this would result in a predicted latency for level-2 of 0.64msec, whereas level-2 reports 1.37msec, which is a factor of 2x overhead. As such level-2 can really help characterize intra-layer bottlenecks.

How well do the performance predictions of the microbenchmarks approximate system and compute only performance at level-3? Figure 7.12 depicts the performance measurements of the various levels for MaxN, FP16 configuration on TX2, and a subset of microbenchmarks on level-1 and level-2, for a spectrum of batch sizes. Note that the theoretical peak performance is

Table 7.2: Level-1 and 2 - discrepancy between latency of different convolutions and residual layers

Level-2				Level-1					
Residual Layer	**[MOP]**	**TX2, MaxN, FP16 b=1 [ms]**	**b=128 [ms]**	**Conv. Layer**	**[MOP]**	**TX2, MaxN, FP16 b=1 [ms]**	**b=128 [ms]**	**ZCU104,INT8 t=1 [ms]**	**t=8 [ms]**
res2a	462.44	1.37	119.12	res2a_branch2a, 1x1	25.70	0.06	5.05	0.06	0.08
res2b	436.74	1.12	108.24	res2a_branch2b, 3x3	231.20	0.19	22.78	0.19	0.19
res2c	436.74	1.12	108.07	res2a_branch2c, 1x1	102.80	0.18	20.15	0.22	0.26
res3a	590.88	1.39	133.09	res2a_branch1, 1x1	102.80	0.21	23.66	0.43	0.46
res3b	436.74	1.03	87.72	res3a_branch2a, 1x1	51.40	0.09	7.19	0.09	0.13
res3c	436.74	1.05	87.87	res3a_branch2b, 3x3	231.20	0.21	24.63	0.21	0.21
res3d	436.74	1.04	88.66	res3a_branch2c, 1x1	102.80	0.15	15.18	0.21	0.25
res4a	590.88	1.20	104.66	res3a_branch1, 1x1	205.50	0.29	30.35	0.33	0.39
res4b	436.74	1.03	71.33	res4a_branch2a, 1x1	51.40	0.08	7.10	0.12	0.13
res4c	436.74	1.03	72.52	res4a_branch2b, 3x3	231.20	0.20	23.12	0.21	0.23
res4d	436.74	1.02	72.01	res4a_branch2c, 1x1	102.80	0.15	13.01	0.29	0.38
res4e	436.74	1.02	72.39	res4a_branch1, 1x1	205.50	0.28	29.23	0.43	0.50
res4f	436.74	1.02	71.76	res5a_branch2a, 1x1	51.40	0.14	7.61	0.12	0.19
res5a	590.88	1.73	95.61	res5a_branch2b, 3x3	231.20	0.31	24.90	0.33	0.49
res5b	436.74	1.24	61.55	res5a_branch2c, 1x1	102.80	0.27	12.53	0.47	0.60
res5c	436.74	1.23	61.58	res5a_branch1, 1x1	205.50	0.51	30.92	0.52	0.69
Min		1.02	61.55	**Min**		0.06	5.05	0.06	0.08
Max		1.73	133.09	**Max**		0.51	30.92	0.52	0.69
Var		0.04	454.94	**Var**		0.01	79.42	0.02	0.03

significantly higher than measured performance, as already discussed in Section 7.2 and only within reach of individual layers that fit the hardware architecture well. The system (level-3) achieves from 41.1% to 60.7% efficiency, where larger batch sizes achieve higher performance. Level-2 results are on average more negative than achieved performance (level-3) and a fairly good approximation within 16% of the achievable level-3 system performance, but far off level-3 compute performance. Level-1 results have usually higher performance than the level-2 results. This makes intuitively sense, as a limited amount of bottlenecks are exposed during execution of the benchmark. In particular lower weight storage is required, which is most likely contained on-chip, thereby alleviating any potential memory bottlenecks. Also it can be said that the averaged level-1 results provide a good estimation of possible compute performance on level-3. As already mentioned, for level-1 and 2 results, we observe large variations in performance ranges for different dimensions of convolutions. The insight is that to provide a good projection from level-1 or level-2 to level-3, we need to provide full coverage of convolutional layers. Another challenge is that many backend tools perform automated layer fusion such as merging batch normalization with convolutions, which makes testing in isolation inaccurate.

7.4.1 Summary

Clearly the microbenchmarks at level-1 and level-2 can offer early insights into system level bottlenecks caused by data movement. Furthermore, level-1 can highlight specific bottleneck

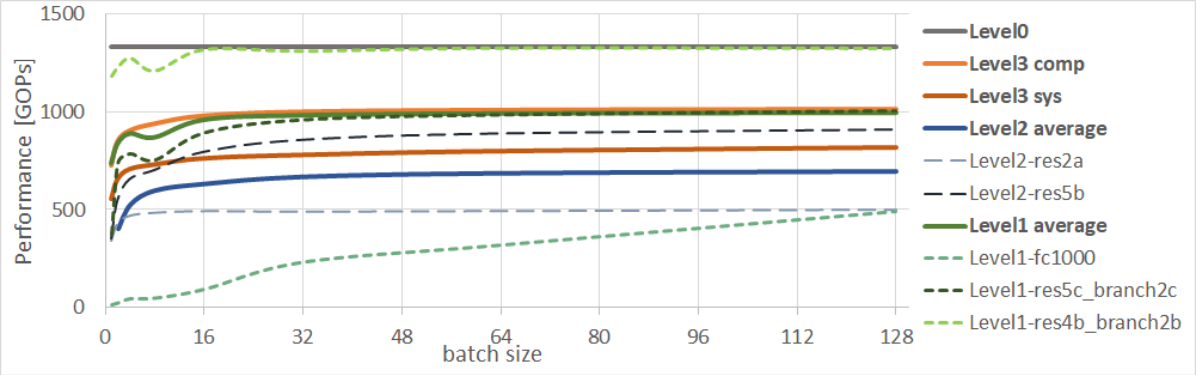


Figure 7.12: Performance comparison level-0, level-1, level-2 and level-3 for TX2 (MaxN, FP16 configuration)

layers which cannot be anticipated from the theoretical performance predictions (the reported level-1 latencies differ substantially even though the compute requirements are very similar). Finally, as expected and shown for ResNet50 on TX2, level-1 and level-2 microbenchmarks can clearly provide much more accurate performance predictions than level-0, whereby it is arguable whether the added benefit from level-2 is worth the additional effort. Perhaps the greatest advantage of level-2 comes from characterizing individual hardware platforms. Although level-1 and 2 provide these useful platform insights and improved performance predictions, the Achilles heal of this approach is the vendor specific and highly limited toolchain support which prevents the execution of topologies which only consist of a subset of the layers. In addition, the scope of testing required is significant, in particular for level-2. Perhaps this could be addressed with larger community support. As such level-1 and level-2 results will be very challenging until toolchains of the various vendors mature and enable consistent support for microbenchmarking.

7.5 Gains from the Measurement Methodology

QutTiBench provides clear guidelines on how to measure power, latency and throughput on fundamentally different hardware platforms, differentiates system and compute only measurements, and proposes how to include deployment parameters such as batch sizes and operating modes in a fair manner. This brings specific benefits which are discussed individually in the following sections.

7.5.1 Systematic Measurement of all Figures of Merit for all Deployment Parameters

Many of the hardware platforms offer different deployment options in order to support different compromises between power and throughput as well as latency and throughput. Firstly, many of the platforms require large batch sizes in order to ensure high compute utilization, however this comes at the expense of latency. Separately, low power modes are offered, which can for example run at reduced clock speed, however this will also reduce overall compute performance. In our benchmark, we ensure to cover a systematic exploration of all of the provided deployment options regarding all figures of merit to ensure a fair comparison and thorough understanding of the design compromises and avoid cherry picking. We evaluate this below for latency vs throughput and power vs throughput separately.

Latency versus Performance

We consider the compromise between latency and performance in the following charts: 7.13, and 7.14[11]. More datapoints are visualized in the web portal, where the charts are also interactive and easier to parse. The lines correspond to all measurements over a spectrum of batch sizes and similar, for one hardware platform and for one topology. FINN delivers lowest latency with no jitter for increasing stream sizes, whereby the performance in regards to fps will saturate when the pipeline is fully utilized. This is reflected in the right part of Figure 7.13, where the FINN results almost overlap with the actual y-axis. For the layer-by-layer compute platforms such as the NCS, BISMO, A53, and TX2, one can observe that the latency always increases with larger batch size and thread counts respectively, because a sequence of images has to be buffered before each of the layers can be executed which adds significant delays. The compute performance flattens out once peak performance has been reached. The extreme variant is the NCS which has already achieved full throughput at batch=1. Therefore with increasing batch size only latency rises but throughput stays constant. The DPU implementations reach performance saturation for very small thread counts. To show that it is important to visualize the full spectrum of batch sizes, please consider in Figure 7.14 the lines for ZCU102-INT8-RN100% and TX2-FP32-GNv1. Depending on latency requirements either the ZCU102 or the TX offers the better performance

[11]We had to split up hardware platforms into different charts due to the large difference in ranges in Figures 7.13 and 7.14.

which is determined by the batch size or respective thread count. If the latency is constraint to 12msec, then the batch size will be limited and with that the corresponding throughput. In this scenario, the ZCU102 offers higher throughput at lower latency while for larger latency constraints, the TX2 is the better option. As such it is really important to benchmark the full spectrum and visualize all options.

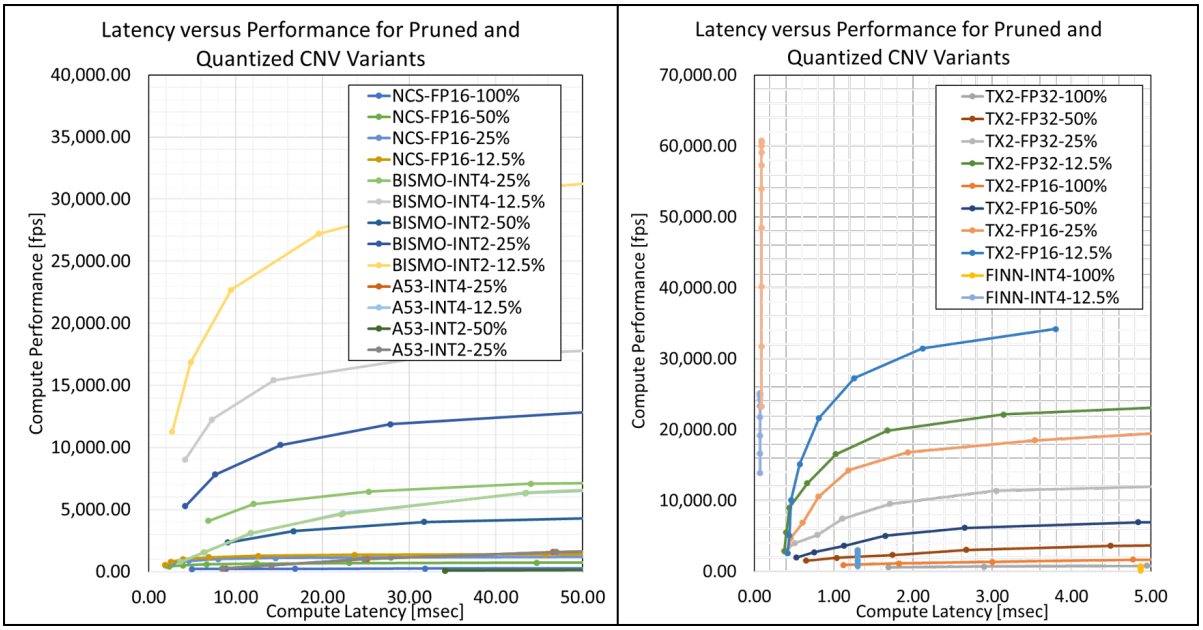


Figure 7.13: CIFAR-10 classification design space: Trade-offs between latency and throughput, visualizing the impact of increased batch sizes, stream sizes and thread counts

Power versus Performance

As discussed in Chapter 2, many of the chosen platforms offer different power or operating modes which provide a compromise between power consumption and achievable throughput, for example by regulating clock speed or disabling parts of the circuit. A specific example, is the TPU Coral stick from Google which can operate with a fast or a slow clock. The NVIDIA GPU Jetson TX2 platform can run in either maxn, maxp or maxq modes. Maxn is the high performance mode with highest power consumption. Maxq is the most efficient mode, with lowest power but also lowest performance, and maxp is the middle compromise. Examples of this behaviour are shown in Figure 7.15 and 7.16 indicating substantial differences in regards to power and performance for different modes and batch sizes, where naturally higher performance comes with higher power consumption. In this regard the benchmark really aids the system designer by highlighting the behaviour of the platform and shows the power cost of the different

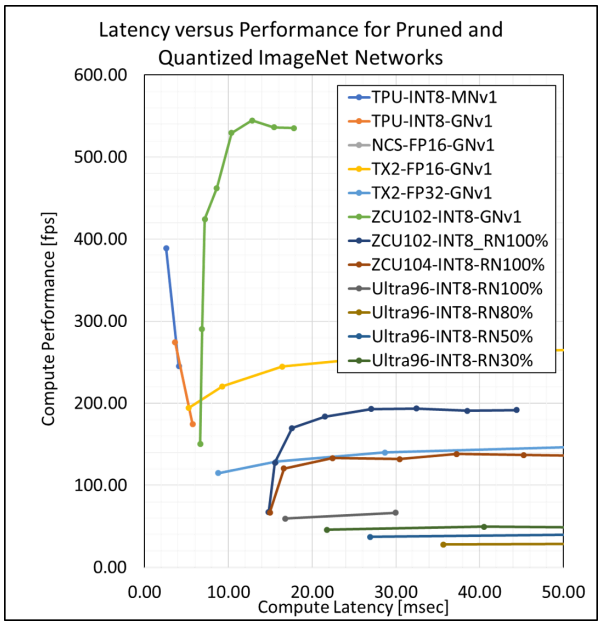


Figure 7.14: ImageNet classification design space: Trade-offs between latency and throughput, visualizing the impact of increased batch sizes, stream sizes and thread counts

operating modes. Furthermore the benchmark results illustrate that for the TX2, the batch size impact (which is essential to scaling performance) comes at a limited power penalty of 1-2W, which shrinks further with the low power modes.

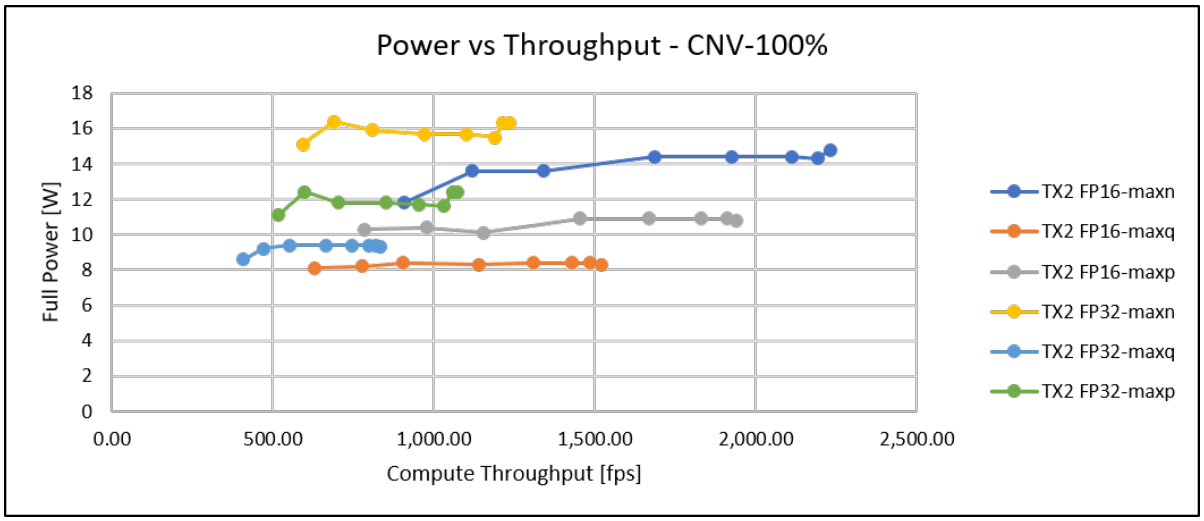


Figure 7.15: Power versus performance for CNV: Higher performance comes at higher power consumption, while higher precision is higher in power consumption and lower in performance

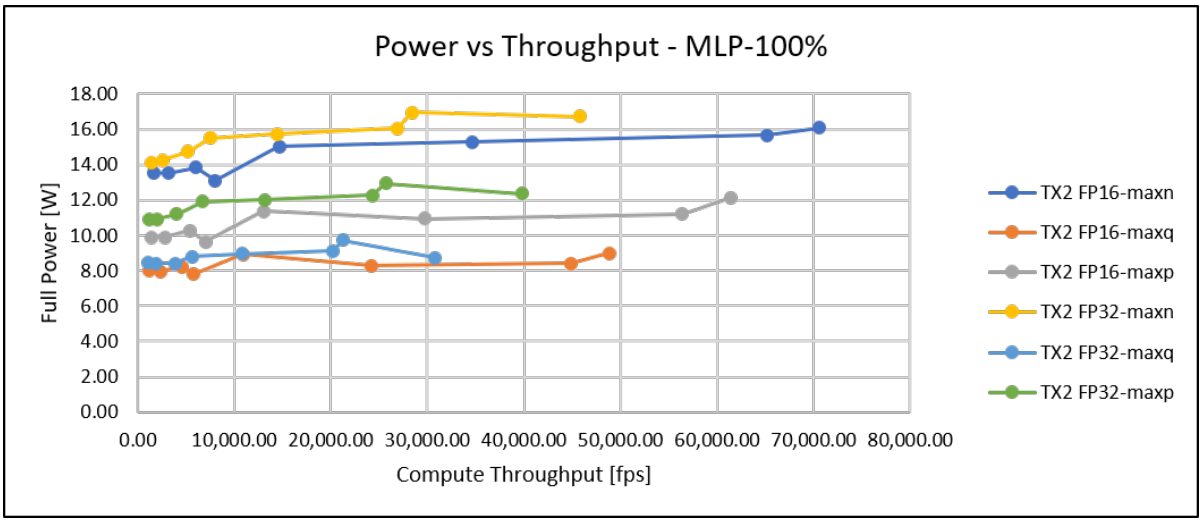


Figure 7.16: Power versus performance for MLP, while higher precision is higher in power consumption and lower in performance. Performance benefits of lower precision are marginal for low batch sizes.

7.5.2 System versus Compute

Reporting system and compute only performance separately brings two distinct benefits. Firstly, it brings clarity regarding how the data was measured and whether it includes data movement to accelerators or not. Secondly, it can show systematically the cost of data movement for different platforms, topologies and optimization strategies over a spectrum of deployment settings. In the following Figures 7.17, 7.18, 7.19, which show the difference between system and compute latency and throughput, we will highlight some of the observations that can be made thanks to this benchmarking methodology. For example, Figure 7.17 shows latency and throughput for CNV on both the TX2 and the NCS platforms. Here it can be observed that data movement overheads for TX2 are much lower compared to NCS, which is as expected, as for NCS data has to be transferred via USB to the accelerator first. Secondly, for both platforms, overheads directly correlate to batch size. In regards to throughput, the overhead seems to be relatively constant, only small increase over batch size until platform reaches peak performance, for both NCS and TX2. In the case of the MLP networks, as shown in Figure 7.18, we can observe that although quantization should half the amount of data to be transferred, the data movement overhead seems to be roughly the same, both for FP16 and FP32, and consistent over all power modes. Throughput almost doubles for FP16 compared to FP32 which is as expected thanks to the native support for FP16 in the TX2. Another curious artefact can be observed in regards to the TX2 for throughput: While the compute throughput starts to level off with batch size 64

and 128, the system-level throughput depends heavily on the data movement overhead and as such the difference continues to increase with batch size. Finally, in Figure 7.19 we can observe the effect of pruning on the data movement overhead. In the case of the FPGA platform, pruning clearly improves the data movement overheads. In regards to throughput, for thread count 1[12], there are larger discrepancies, while for higher thread count, the difference seems to become constant. Also, unlike GPUs, the difference decreases with thread count, which clearly shows, that the FPGA platform increasingly overlaps data movement with compute for higher thread count. However, this is less for increased pruning scale.

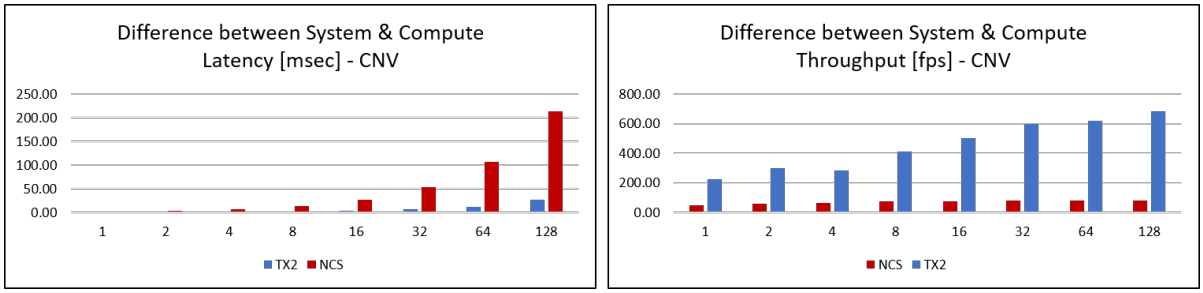


Figure 7.17: Difference between system and compute latency and throughput for CNV

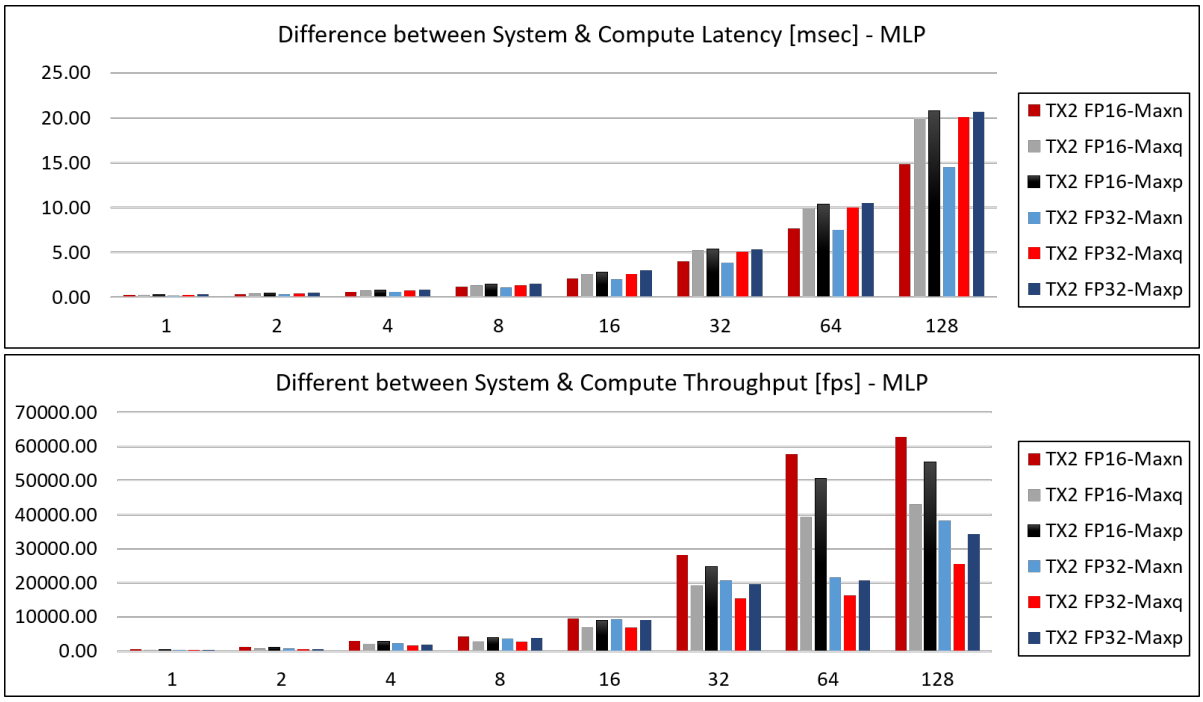


Figure 7.18: Difference between system and compute latency and throughput for MLP with different precisions

[12]There is a missing datapoint for the 30% pruned variant, where the experiment produced a hardware error.

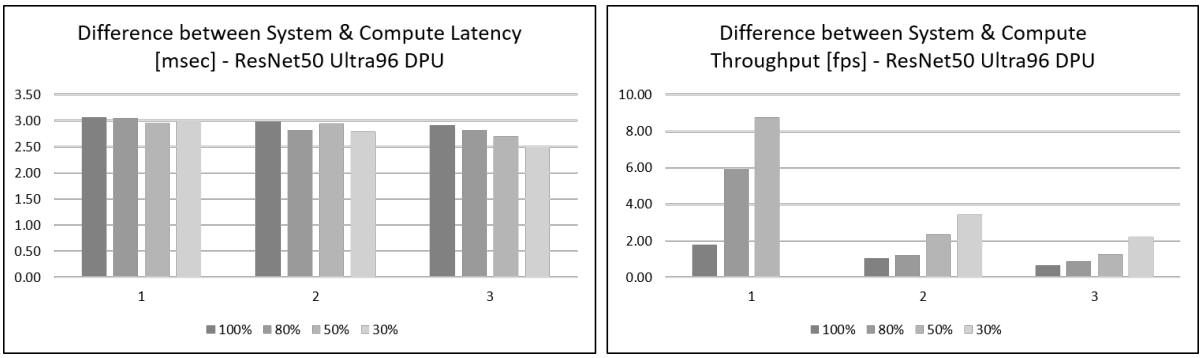


Figure 7.19: Difference between system and compute latency and throughput for pruned ResNet50 variants

These are some examples of observations that can be made with this benchmark and are not possible with other benchmarks such as MLPerf. We believe it is clear from this and the fundamentally different behaviours showcased in the presented figures, that reporting system and compute latency and performance separately can really help to create a better platform understanding, as the benchmark intends. Note, that this is just a selection of datapoints. The remaining data is available on the web portal and in the appendix.

7.5.3 Summary

Overall, we showed that the defined measurement methodology is instrumental for categorizing fundamentally different hardware platforms for different neural networks. The systematic measurements over all deployment parameters provides a clear and quantified understanding of the various system-level trade-offs between the various figures of merit such as power, latency and throughput and shows the impact of low power operating modes, or latency penalties associated with large batch sizes. Finally, differentiating system and compute-only measurements, can yield surprising insights in regard to data movement overheads: for example no savings in regards to quantization for the TX2 GPU, while pruning visibly reduces overhead for all hardware platforms. The FPGA DPU clearly overlaps data movement with compute and thereby reduces overhead. Overall we observe significantly larger overheads for MLPs compared to CNVs, which makes sense as they are more memory intensive. The aim here is not to provide a complete understanding of the insights gathered but rather to illustrate the type of insights that can be gathered with this benchmark.

7.6 Benefits of Full Design Space Visualization for Optimization Strategies

In this section, we want to illustrate that the chosen data visualizations can be an effective method to provide interesting insights and answer important design questions up front. For example, system designers might wonder, for the same level of accuracy, do we derive more benefit from pruning or using reduced precision for ImageNet, CIFAR-10 and MNIST classification? QutiBench aims to identify these optimal solutions along the pareto frontier in the graphs between accuracy and performance or latency. We show example graphs in Figures 7.20, 7.22 and 7.21. These visualizations allow system designers to draw conclusions about the potential of various optimization strategies. In the following discussion, we present a few example conclusions, that can be drawn for our three example machine learning tasks using the proposed methods of comparison and visualization.

When looking at the design space of platforms and optimization techniques, we can observe that overall, combining pruning and quantization delivers the best result. Pruning seems to be more effective than quantization across a broader spectrum of platforms and topologies, however reducing precision is more effective for MLPs as shown in Figure 7.21. For example, FINN-INT2-25% outperforms FINN-INT4-12.5%. In this scenario, FINN also outperforms other architectures by far. This is due to the memory bound nature of the topology for FP16 and FP32 which throttles the compute performance on NCS and TX2. For INT4 and INT2, the model can remain on-chip for FINN while still offering competitive accuracy for this dataset and topology.

We also make several other observations from these graphs: The USB devices are far lower in overall performance as expected from the theoretical analysis as can be seen in Figure 7.20. Secondly, the bit-serial implementations do not feature on the Pareto frontier, which is as expected, as these implementations are not performance optimized. Thirdly, FPGAs outperform others with regards to throughput in particular where quantization is available.

In the following, we will discuss the results in more detail for the various ML tasks: For ImageNet classification, as shown in Figure 7.22, we have a few more topologies to compare with, and we can see that they have a massive impact in regards to performance and accuracy and in fact outweigh benefits from optimization techniques. Quantization from FP32 to FP16

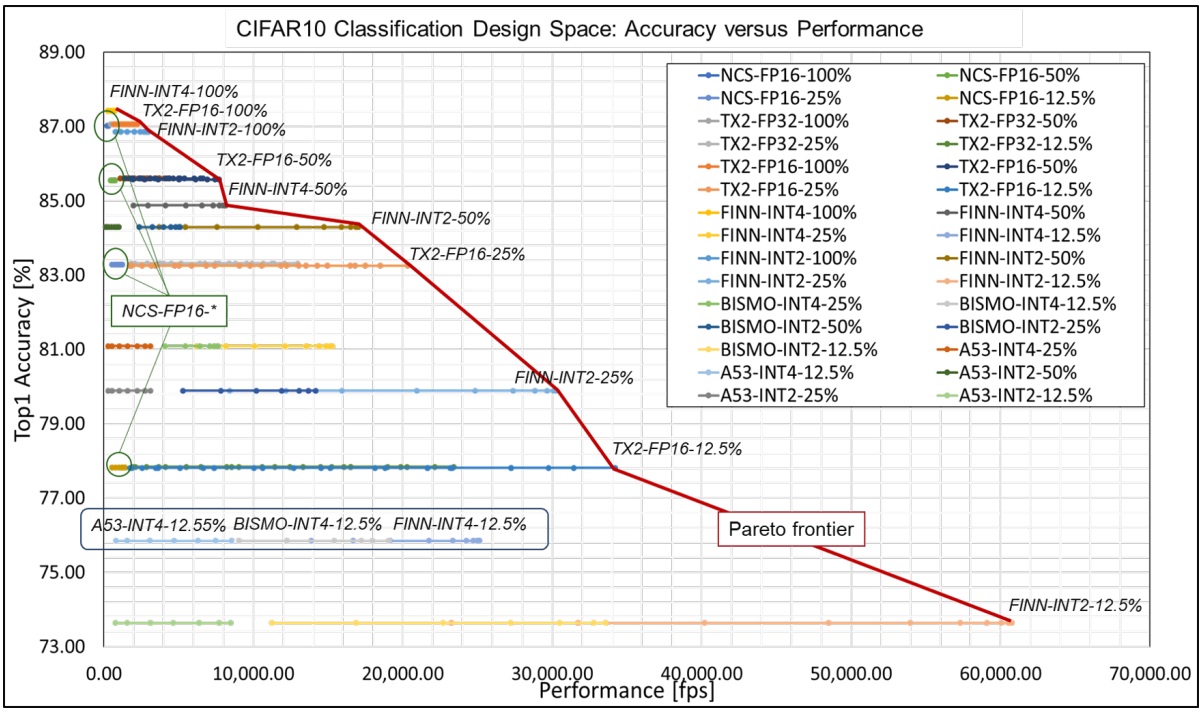


Figure 7.20: Visualization of pareto-optimal solutions in the CIFAR-10 classification design space

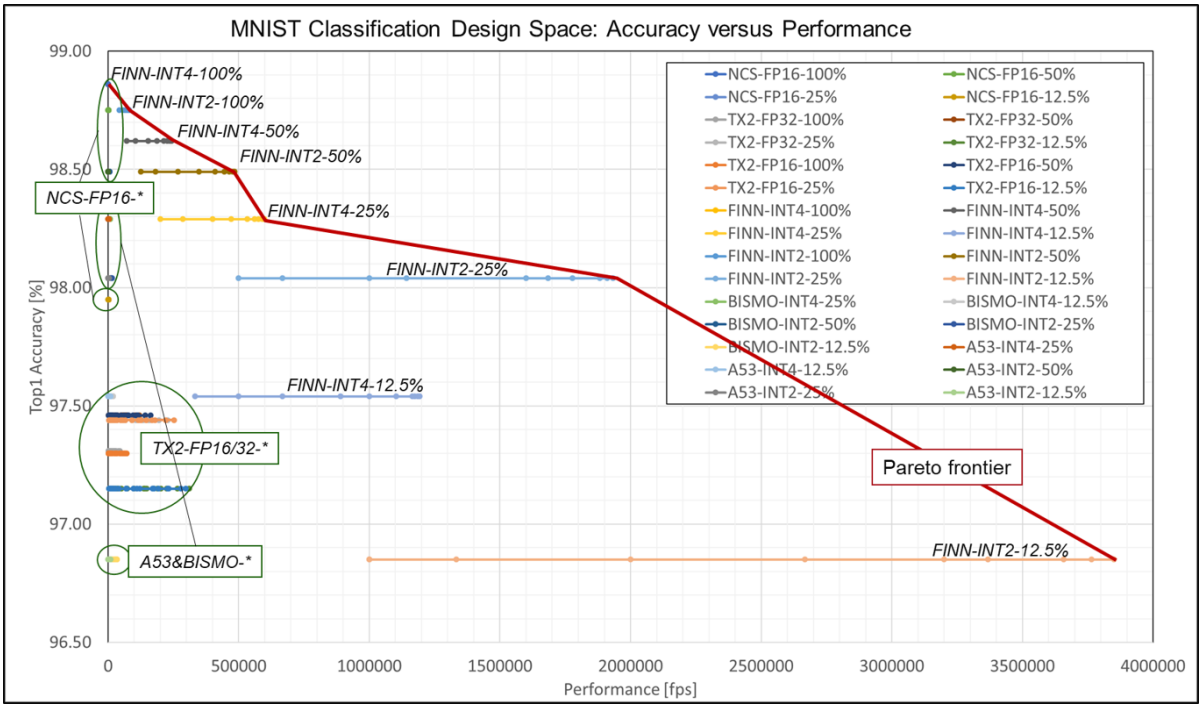


Figure 7.21: Visualization of pareto-optimal solutions in the MNIST classification design space

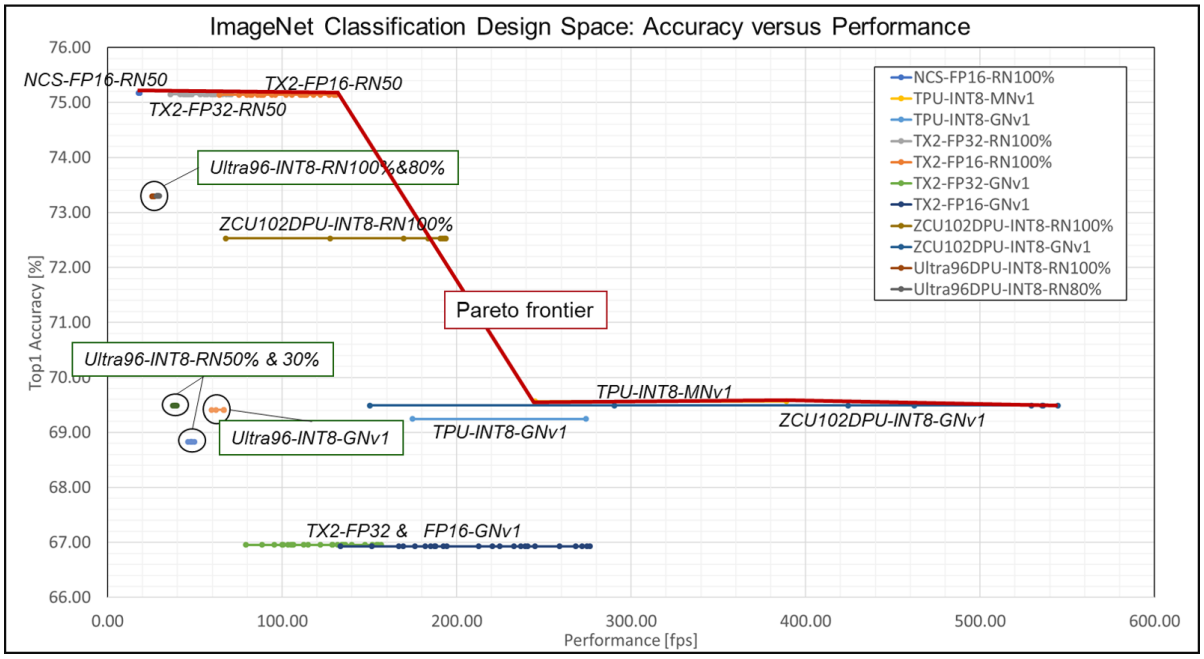


Figure 7.22: Visualization of pareto-optimal solutions in the ImageNet classification design space

has in essence no impact on accuracy and FP16 variants bring massive performance benefits. Pruning impacts accuracy when dropping below 50%, but up to then the performance benefits come basically for free. The NCS is far below competitive devices in regards to performance overall. The TPU offers GoogLeNetv1 performance on par with the TX2 whereby the INT8 accuracy is higher than TX2's FP16 and FP32, most likely due to superior image preprocessing in TensorFlow. The TPU provides pareto optimal design points with MobileNetv1, clearly showing that MobileNetv1 is an exceptional topology with regard to compute cost-accuracy compromise and outperforms GoogLeNetv1 variants on other devices. For the Ultra96, one can observe that pruning brings performance benefits at small accuracy reduction, except for the 80% variant which remains equivalent to the original version. Finally, the ZCU102-DPU provides pareto optimal implementations with both GoogLeNet and ResNet variants. Overall, this figure visualizes clearly the impact that topology variations, which are in essence a form of algorithmic variations, have on design space trade-offs. Further, the charts visualize many design points that would be invisible if accuracy would be a hard target as is the case in other benchmarks. These are significant benefits that QuTiBench can offer to system designers and hardware architects.

7.7 QuTiBench and MLPerf

One of the most promising industry-wide efforts in regards to benchmarking NNs is MLPerf [150] with 600 reproducible inference performance measurements from 14 organizations, representing over 30 hardware platforms. A summary of MLPerf was provided in the Related Work Section in 3. MLPerf emerged shortly after the start of my PhD and has significantly unfolded during the course of the thesis. Yet, despite a similar motivation, namely driving clarity into this highly complicated design space, there are substantial differences with the approaches taken. In this section, we discuss these key differences. With QuTiBench we aim to provide more **system-level insights**, offer more **generality** and clarity regarding **optimizations**. Compared to MLPerf, QuTiBench brings the following additional benefits:

- QuTiBench's multi-tiered approach includes microbenchmarking and theoretical analysis and performance predictions ($\rightarrow$ system-level insights)
- QuTiBench's measurement methodology includes systematic testing across all deployment parameters, system vs compute, and reporting of all figures of merit ($\rightarrow$ generality and system-level insights)
- QuTiBench does not have constraints towards specific application scenarios and considers a more complete design space ($\rightarrow$ generality)
- QuTiBench has specific support for optimization techniques, which are only added to MLPerf as an afterthought in the open section ($\rightarrow$ optimization support)
- Qutibench provides additional consideration of data visualization ($\rightarrow$ system-level insights)

In summary, QuTiBench has more focus on providing a deeper system level understanding for the various hardware platforms, through a much more general and systematic measurement of all figures of merit, and with that allows to provide predictions and guidance for new application scenarios. In contrast, MLPerf is dedicated to very specific, yet representative, application scenarios, having selected two image classification and object detection CNNs (one light-weight and one heavy-weight each) plus a machine translation network. All 5 of them are tested in 4 different load scenarios (single stream, multi-stream, server and offline). For each of the load scenarios, only 1 specific figure of merit is reported for a specific minimum application accuracy, whereas QuTiBench rigorously reports everything and is not constraint to these 5 application

scenarios. As such QuTiBench can illustrate and identify potential system-level trade-offs which provide optimal solutions under application constraints that may not be covered by MLPerf. Furthermore, through its theoretical analysis and performance predictions, QuTiBench can provide predictions and insights for completely new topologies, such as which optimization techniques are the most likely to provide optimal solutions, or how certain techniques will impact performance and power consumption without having to try every single one. Finally, the explicit support for optimizations at all tiers of the benchmark are vital in particular in the embedded space, where most solutions are co-designed with the hardware and heavily optimized. However, to be real, QuTiBench is just a PhD project and as such cannot compete with the general level of industry support and level of adoption that MLPerf has achieved. Also, MLPerf has substantially more detail when it comes to providing statistical relevance and ensuring reproducibility, and system level evaluation, addressing potential cheat techniques such as taking advantage of on-the-fly caching detection. Ideally, as the techniques on QuTiBench and MLPerf complement each other, the benchmarks could get combined. We discuss the details of this and how this could manifest itself in the last Chapter 8.

7.8 Concluding Remarks

In this chapter, we evaluated the benchmarking methodology. We have learned that the theoretical predictions can give reasonable insights for new application scenarios without having to run any experimentation, depending on the type of accelerator platform. We also highlighted a couple of options regarding how to make the predictions more accurate in the future and illustrated that the theoretical pareto plots could predict many of the most beneficial optimization techniques. The measurement methodology with the distinction between system and compute level performance has been shown to be very useful to illustrate the individual data movement characteristics of the various hardware platforms. The systematic exploration of other deployment settings provides more clarity in the design space. And finally, the chosen comparisons and data visualizations aid providing fair comparisons between fundamentally different hardware platforms, different topologies and include the scope of potential optimizations. A key challenge was microbenchmarking level-1 and level-2 results. While we could show the benefits and more accurate performance predictions, we ran into too many practical constraints that rendered this as limited in scope due to limitations of current vendor-specific tooling. Further, we found that the chosen data visualizations were effective to break-down the complexity

of the design space. Finally, we discussed how QuTiBench compares to MLPerf, which is the number one industry-wide effort in this space. Most notably, QuTiBench brings additional system-level insights, offers optimization support and brings more generality compared to MLPerf. In the next chapter, which concludes the thesis, we'll discuss how to create an impact on the research efforts within this space, in particular in comparison and conjunction with MLPerf, offer lessons learned and provide a future outlook.

8 Conclusions, Impact & Future Work

8.1 Introduction

This chapter concludes the thesis. It summarizes the overall effort and offers a critical review of the key findings and lessons learned. This includes a section on interesting insights for the various hardware architectures that we have gathered through the experimentation. We also incorporate a discussion around how we aim to maximize our impact on the wider research community with FAIR data, a web portal and specific suggestions on how to integrate QuTiBench with MLPerf. We finish with an outlook into our future efforts.

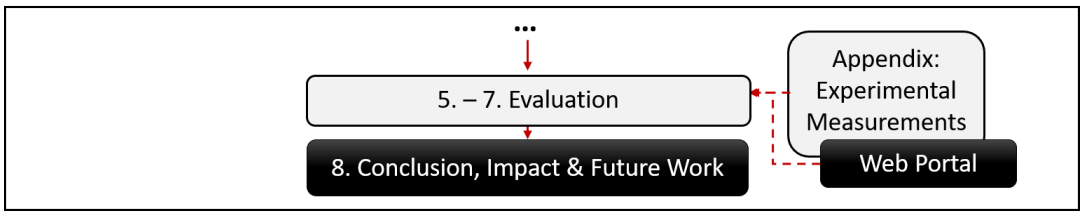


Figure 8.1: Location within the thesis

8.2 QuTiBench

Neural networks are fast gaining popularity across an increasing number of applications. However, they are accompanied by challenging compute and memory requirements, which is seriously challenging the semiconductor industry which is facing performance scalability issues. This is of particular importance for embedded computing environments, where real estate, power and available compute and memory resources are at a premium. The industry is turning to both algorithmic innovation in the form of new topologies, quantization, and pruning strategies, as well as architectural innovation with increasingly complex and heterogeneous devices. The

latter resulted in specialized compute architectures for Convolutional Neural Networks (CNNs) which are typically referred to as Deep Learning Processing Units (DPUs). To facilitate better insights into the increasingly complex space of end solutions which involve hardware-software codesign and evaluate new concepts in computer architecture, novel CNN system benchmarks are needed.

QuTiBench is a proposed novel benchmarking methodology to help represent all possibilities within the solution spectrum fairly and to drive hardware innovation by providing valuable insights for system level designers. QuTiBench creates an understanding of possible performance accuracy trade-offs for newly devised and fine-tuned algorithms combined with highly customized accelerators.

Key contributions are the concepts that allow benchmarking of highly optimized algorithms by tying hardware characteristics back to the end-application, thereby providing the necessary algorithmic freedom. Another unique characteristic of this benchmarking concept is the introduction of the multi-tiered approach including a theoretical level and consideration of a spectrum of numerical representations at all levels. As such the benchmark can provide insights at various abstraction levels. This brings three key advantages: a) It provides a spectrum of insights. Users can choose from instant but perhaps crude results, which can be elaborated upon at a later stage with real implementations and more in-depth evaluation; b) The multi-tiered approach provides insights into system bottlenecks. For example, are the recurrent or the fully connected layers the challenge? Or is the bottleneck the data movement in between?; c) We can track performance optimizations with the theoretical analysis and predict potentially pareto-optimal design points. This can save substantial experimental efforts. Furthermore, QuTiBench is enabling understanding by providing thoughtful data visualizations such as the pareto-comparison charts, box and whiskers charts as well as heatmaps. Other differentiators of our approach include the clear measurement methodology, which investigates both system and compute throughput and latency, and the systematic evaluation across all optimization techniques, topologies, hardware platforms with all deploymentment parameters. Finally, similar to other benchmarking efforts, we understand the critical need for a community to support this effort as well as open and FAIR data to generate meaningful research impact. As such we have put significant effort into the data storing and access (more detail below) and a web portal located here: `https://rcl-lab.github.io/QutibenchWeb`. This portal supports third party contributions and provides downloadable access to all measured and theoretical data points. It also includes all data analysis and visualizations that were derived within the thesis in open

source format such that the community can leverage and further build on top of it.

8.3 Lessons Learned

Through careful evaluations with close to a thousand experiments over a spectrum of hardware platforms, optimizations, and machine learning tasks with different topologies, we have learned a number of lessons which we can be grouped into two categories: The first set of lessons relates to benchmarking ML applications and which aspects of QuTiBench have worked well and which haven't. The second relates to actual insights regarding the various hardware architectures. While this is perhaps slightly tangential to the thesis, it might still be interesting to many readers. Both sets are discussed in greater detail within the following subsections.

8.3.1 Lessons in Benchmarking

One of the most important lessons we have learned is that the implemented theoretical predictions can give representative insights for performance in new application scenarios which are well correlated with the actual measurements. This is really significant as it allows to narrow down the design space without having to run any time-consuming experimentation. In addition, we found that the theoretical aspect of the benchmark with its pareto plots could predict many of the most promising solutions and beneficial optimization techniques. Again this can save significant amount of time as it narrows the scope of experimentation. We have identified a number of options on how to make the predictions more accurate in the future. First of all in regards to Field Programmable Gate Arrays (FPGAs), we can take resource cost into account (how much of the available compute resources was actually used). Secondly, we can improve the estimates by considering memory capacity for calculating operational intensity of the various CNN topologies. If the required memory fits into on-chip resources, external memory considerations should be removed from the model assumptions. Finally, throughout the experimentation we have observed that with smaller model size the achieved compute efficiency decreases. This insight could be leveraged to improve the performance predictions by introducing a simple scalar factor that is inversely proportional to the size of the CNN.

In addition to the theoretical analysis, we've provided experimental proof that the measurement methodology with the distinction between system and compute level performance has been

shown to be very useful to illustrate the individual data movement characteristics of the various hardware platforms. Also, the systematic exploration of other deployment settings provided more clarity in the design space in regards to throughput, latency and power compromises. Finally, we found that the chosen data visualizations can significantly aid with gaining valuable insights from the vast amount of complex and multi-dimensional data points. They aid with fair comparisons between fundamentally different hardware platforms, different topologies and include the scope of potential optimizations.

A substantial challenge was encountered in regard to microbenchmarking level-1 and level-2 results. While they clearly showed benefits, including interesting system-level insights as well as much more accurate performance predictions, we ran into too many practical constraints that rendered this as limited in scope. This is due to two reasons: Firstly, benchmarking all layer types and combinations thereof, is highly time-intensive and it is questionable whether it is worth the effort compared to a refined theoretical estimate. Alternatively, perhaps through community effort, this could become feasible. Secondly, we ran into many constraints regarding the vendor-specific toolchains. In more detail: Ideally, one would describe the microbenchmarks as individual topologies that consist only of one to a few layers. However, many of the available tools did not support execution of these. The alternative was to use profiling tools, but again these are vendor specific and as such inconsistent between the various hardware platforms. The resulting conclusion here is that it would make sense to postpone the microbenchmarks to the future, until the available tools have matured. But even then, in particular the level-2 results are still highly time-intensive, and it is not clear whether they are worth the effort unless this becomes a one-time effort that a whole community is going to leverage going forward.

Finally, we compared in-depth QuTiBench to MLPerf, which is the number one industry-wide effort in this space. Firstly, QuTiBench emphasizes providing a deeper system level understanding for the various hardware platforms, through a much more general and systematic measurement of all figures of merit. This way allows QuTiBench to provide predictions and guidance for new application scenarios. MLPerf on the other hand is dedicated to very distinct application scenarios, having selected two image classification and object detection CNNs (one light-weight and one heavy-weight each) plus a machine translation network at the time of this writing. All 5 of them are tested in 4 different load scenarios (single stream, multi-stream, server and offline). For each of the load scenarios, only 1 specific figure of merit is reported for a specific minimum application accuracy, whereas QuTiBench rigorously reports everything and is not constraint to these 5 application scenarios. As such QuTiBench can illustrate and

identify potential system-level trade-offs which provide optimal solutions under application constraints that may not be covered by MLPerf. Furthermore, through its theoretical analysis and performance predictions, QuTiBench can provide predictions and insights for completely new topologies, such as which optimization techniques are the most likely to provide optimal solutions, or how certain techniques will impact performance and power consumption without having to try every single one. Finally, the explicit support for optimizations at all tiers of the benchmark are vital in particular in the embedded space, where most solutions are co-designed with the hardware and heavily optimized. However, to be real, QuTiBench is just a PhD project and as such cannot compete with the general level of industry support and level of adoption that MLPerf has achieved. Also, MLPerf has substantially more detail when it comes to providing statistical relevance and ensuring reproducibility, and system level evaluation, addressing potential cheat techniques such as taking advantage of on-the-fly caching detection. Ideally, as the techniques on QuTiBench and MLPerf complement each other, the benchmarks could get combined. We analyze this aspect (on how adding some aspects of QuTiBench to MLPerf) further below. We expect that this could significantly improve the scope of MLPerf, in particular for system designers, and for solutions that are more focused on the embedded space.

Overall, the theoretical analysis was clearly worth the effort, as was the systematic exploration across all figures of merit. This provides real system-level insights for future system and hardware architects, for example the observed data movement overheads over batch sizes for the different hardware platforms. The application-level visualization, which offers complete algorithmic freedom, is essential for benchmarking embedded devices where co-design is standard. Implementations in the embedded space are often written for specific platforms and cannot be executed on other platforms. Particularly within this context, MLPerf, with its strict accuracy requirements, is too rigid and potentially doesn't expose the interesting parts of the design space. On a side note, I would also like to add, that from working within the industrial context, I can observe that these type of results that have been generated through QuTiBench are of high interest in order to design next generation devices for this application space, as well as to understand the competitive landscape. These observations are also confirmed by initial interest I'm receiving from colleagues in the field. However, it is clear that while the scope of experimentation was sufficient to prove benchmarking concepts, it is overall too limited, as there are only a few individual participants. In order to create real impact, a community effort is essential. While the web portal opens up this possibility, it might be more advantageous to integrate the complementary aspects of QuTiBench with the existing

MLPerf to create the biggest impact within the field. This includes in particular the theoretical analysis, the systematic and more general capturing of all figures of merit, and eventually the microbenchmarks once the vendor-tool chains mature enough to support sub-topologies. We will discuss the details of this further below.

8.3.2 Insights for the Various Hardware Architectures

While the thesis is dedicated to the benchmarking methodology, we have also gained some interesting insights in regards to the behaviour of a broad spectrum of customized and heterogeneous compute accelerators and what optimizations, specifically pruning and quantization, are of greatest benefits in what situations. In the following, we'll summarize our observations regarding latency, performance, power and efficiency with quantization and pruning:

Latency

In this section, we'll investigate the effects of pruning and quantization on latency for the different hardware architectures. As expected by our theoretical analysis, all platforms benefit from pruning in equal measure and directly related to the remaining compute in the network. This is visualized in Figure 8.2 and 8.3 annotated through the red arrows. (Please refer back to Section 4 regarding the details of the visualization techniques.) This applies to all tested topologies[13]. Quantization benefits, highlighted by the blue arrows in Figure 8.2 and 8.3 are shown for the hardware platforms with the supported datatypes; for example, A53 derives no benefit. TX2 benefits from going from FP32 to FP16 almost by 2x for larger batch sizes, less so for batch=1. FINN derives the greatest benefit (3.7x) due to the greater than linear reduction in hardware cost when going from INT4 to INT2, however for smaller networks (as we increase the pruning factor), the speedup converges to 1.3x, as we increasingly encounter hardware overheads. BISMO improves by up to 2x for larger batch sizes in reduction from INT4 to INT2. It is worth noting that quantization is beneficial independent of whether we have a layer-by-layer compute architecture or a full dataflow for all tested topologies. Furthermore, there is a significant difference in regards to latency variation to be observed between dataflow architectures (FINN), which shows no jitter, and all others. This is as predicted by the theoretical analysis.

[13]however for ResNet50, we could only use the Ultra96-INT8 platform, as this was the only one which would execute the pruned CNNs.

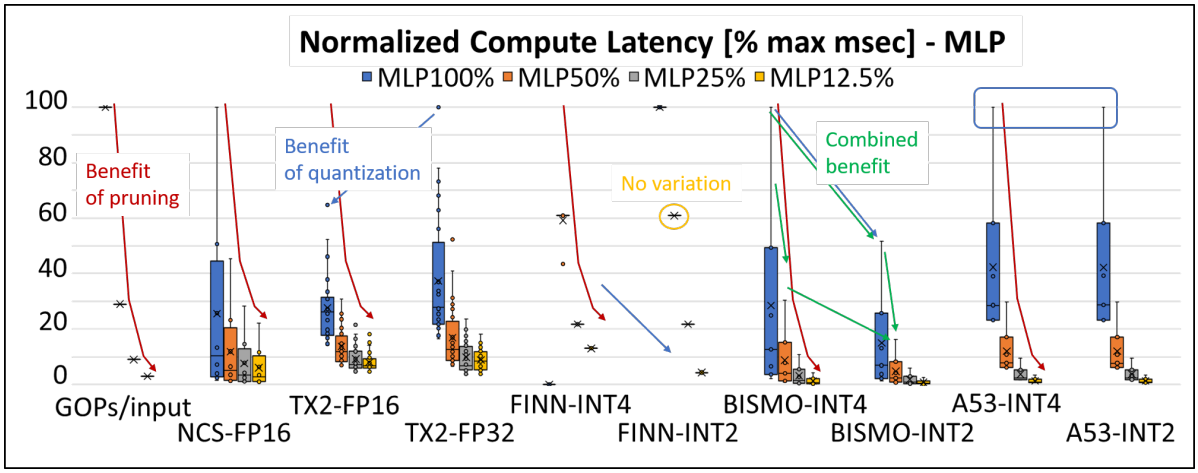


Figure 8.2: Effect on latency with pruning and quantization for MLP topologies

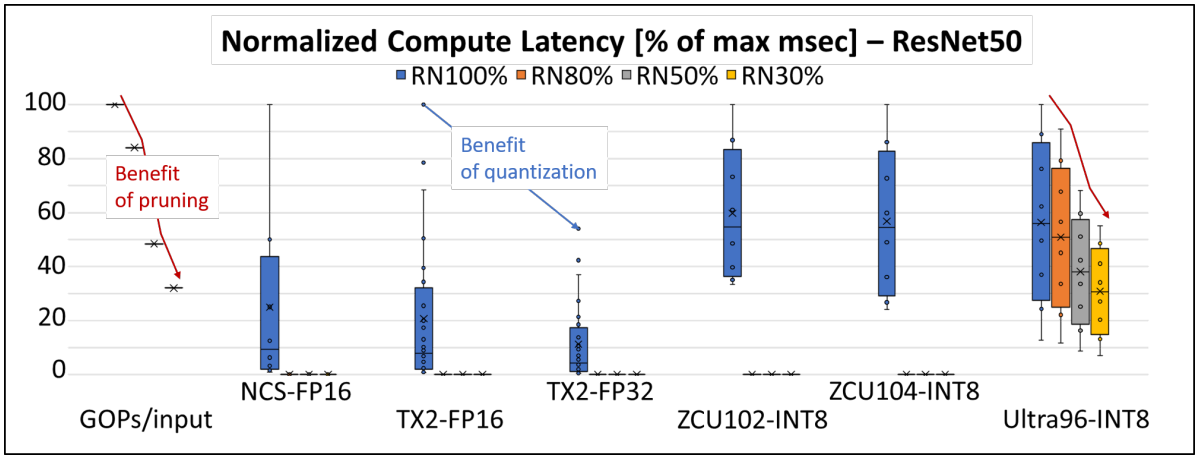


Figure 8.3: Effect on latency with pruning and quantization for ResNet50

Finally, it can be observed that quantization and pruning can be effectively combined, and the overall speedup and latency reduction is an almost direct combination of the individual gains. We have visualized this with the green arrows in Figure 8.2, and discuss this further below under the heading "Orthogonality of Pruning and Quantization".

Performance

Similar to latency, direct speedup can be derived from both quantization and pruning. This is visualized in Figure 8.4 and Figure 8.5 for CNV, MLP and ResNet50. The speed-up is mostly correlated to the amount of compute per input required (which has been included in the figures as the first column and is annotated in red. Quantization brings the greatest benefit for FINN;

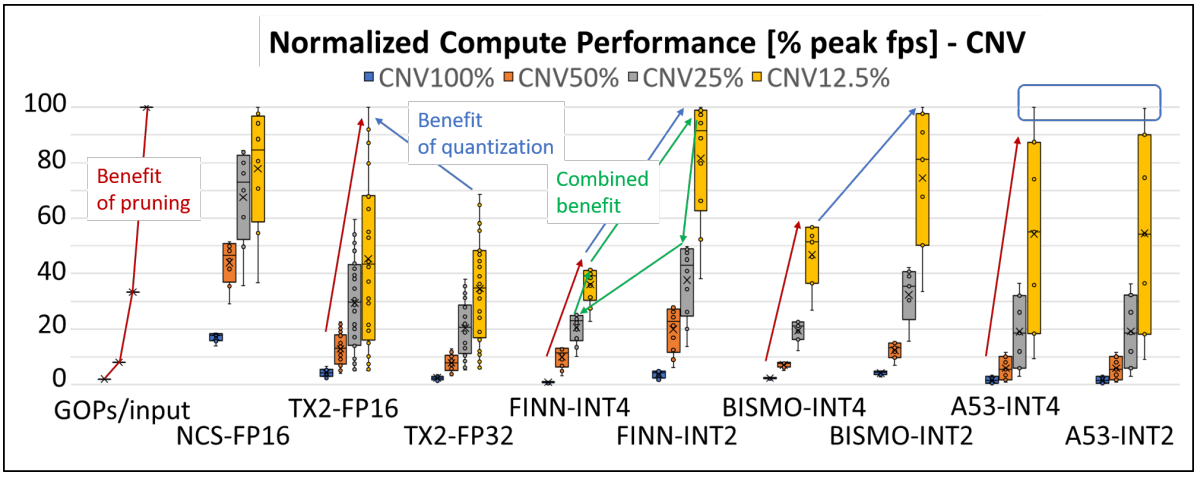


Figure 8.4: Effect on throughput with pruning and quantization for CNV

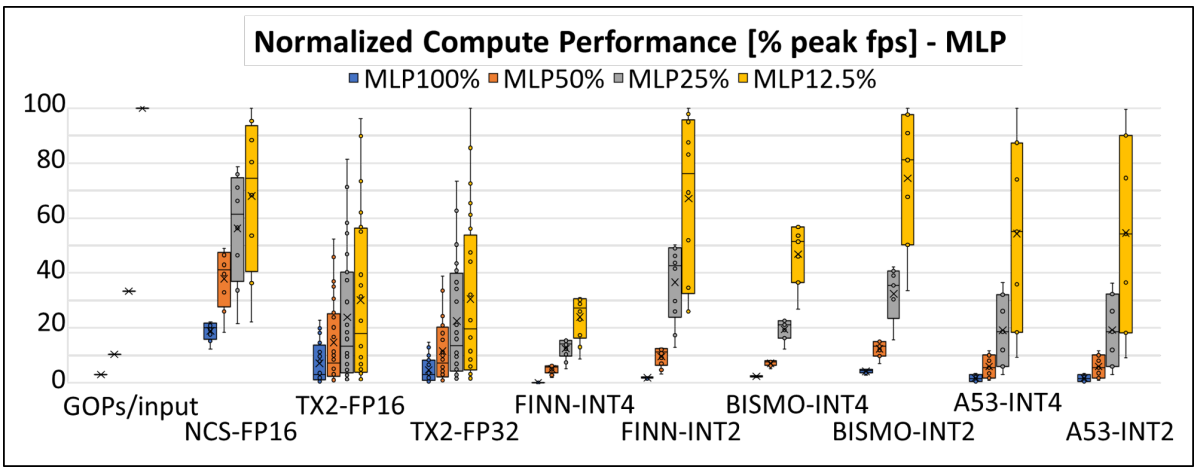


Figure 8.5: Effect on throughput with pruning and quantization for MLP

2x on average for all pruned versions and all networks (annotated in blue). TX2 performance also improves between FP32 and FP16 but only by a factor of 1.42x for CNV and much less for MLP. We assume this is because MLP is, according to the theoretical analysis, memory bound given the operational intensity shown in Figure 2.13. The DPU on Ultra96 gets a significant but not quite linear improvement. Specifically, by taking the pruning scale factor for ResNet50 from 100% to 30% the speed-up is 1.81x. Furthermore, it can again be observed that pruning and quantization can be effectively combined (green annotation). We analyze this further below under the heading "Orthogonality of Pruning and Quantization".

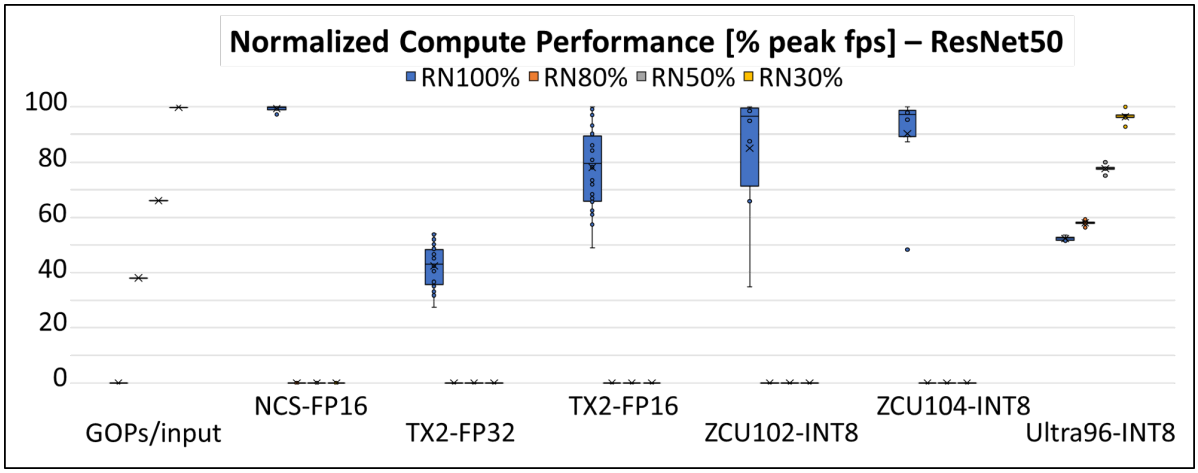


Figure 8.6: Effect on throughput with pruning and quantization for ResNet50

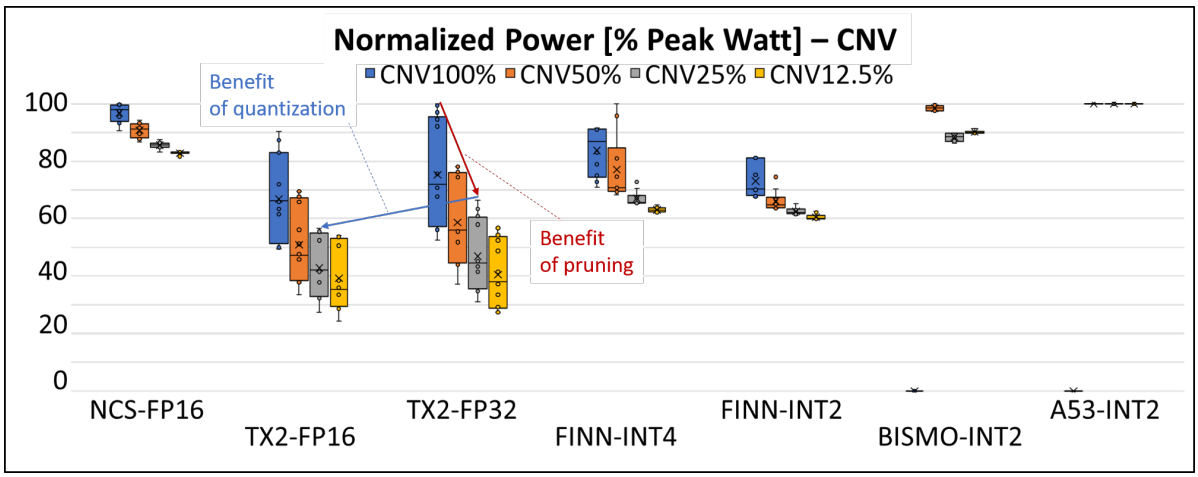


Figure 8.7: Effect on power with pruning and quantization for CNV

Power

The effects of pruning and quantization on overall board power are naturally much more modest, as there is significant overhead in regards to board peripherals with idle power being a substantial component, as shown in Figure 4.7. This applies in particular for FPGA devices (FINN and BISMO). But also the USB device shows only modest improvements with more pruning and quantization. The GPU platform derives the greatest benefits from pruning, with quantization bringing only limited benefits in regards to power (see Figure 8.7, blue and red annotations). Also note that power measurements are done at the socket and come with a 10% error margin, which may explain some of the erratic measurements.

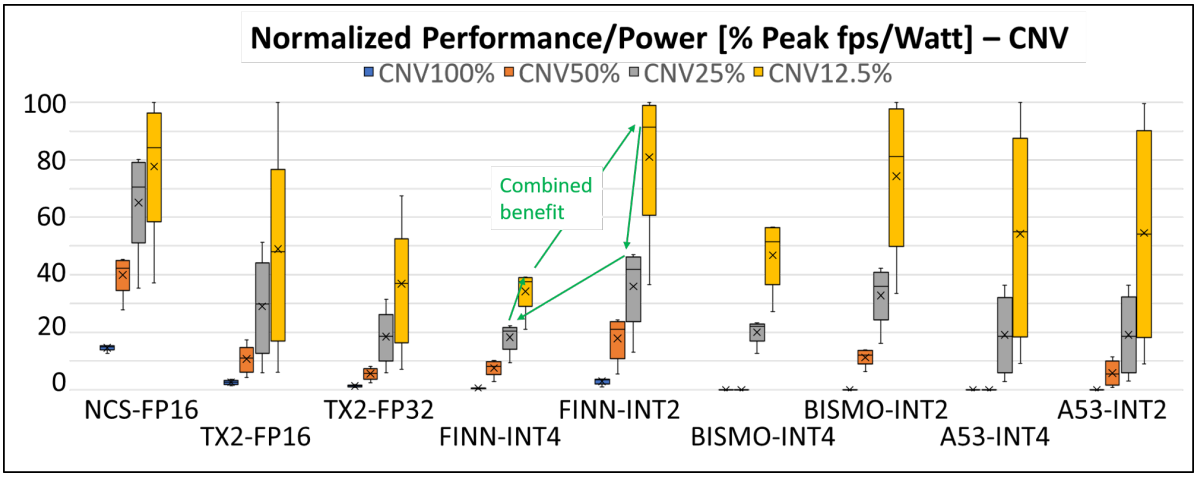


Figure 8.8: Effect on efficiency (throughput/power) with pruning and quantization for CNV

Efficiency

For NCS, the power efficiency seems to be linear, while TX2, FINN, BISMO and A53 improve more in line with the reduction in compute relating to pruning which is the square of the pruning scale. In addition, quantization for Tx2, provides almost a 2x improvement going from FP32 to FP16, while for FPGAs this is even more pronounced, where the improvement is greater than 2x, and for BISMO it is almost exactly 2x. Furthermore, for A53 below INT8 compute, there is naturally no difference as the compute is carried out in INT8 carrier datatypes using the gemmlowp library. For MLPs, thus the greatest benefit can be seen for the FPGA designs. We depict only the MLP and CNV example in Figure 8.9 and Figure 8.8, however the same applies to other topologies which can be observed on the web portal. Finally, as with the other figures of merit, it is shown that pruning and quantization can be effectively combined (annotated in green), as discussed below under the heading "Orthogonality of Pruning and Quantization".

Orthogonality of Pruning and Quantization

As previously mentioned, quantization and pruning can be effectively combined. In this subsection we quantify this in more detail. In order to do so, we must first introduce the nomenclature shown in Figure 8.10. For each topology and each platform, we provide datapoints (latency and throughput) for the original variant (A), a quantized variant (B), a pruned variant (C) and a quantized and pruned variant (D). $p1, q1, p2, q2$ symbolize the improvements achieved with first degree and second degree optimization, whereby $p1, p1$ are the first degree

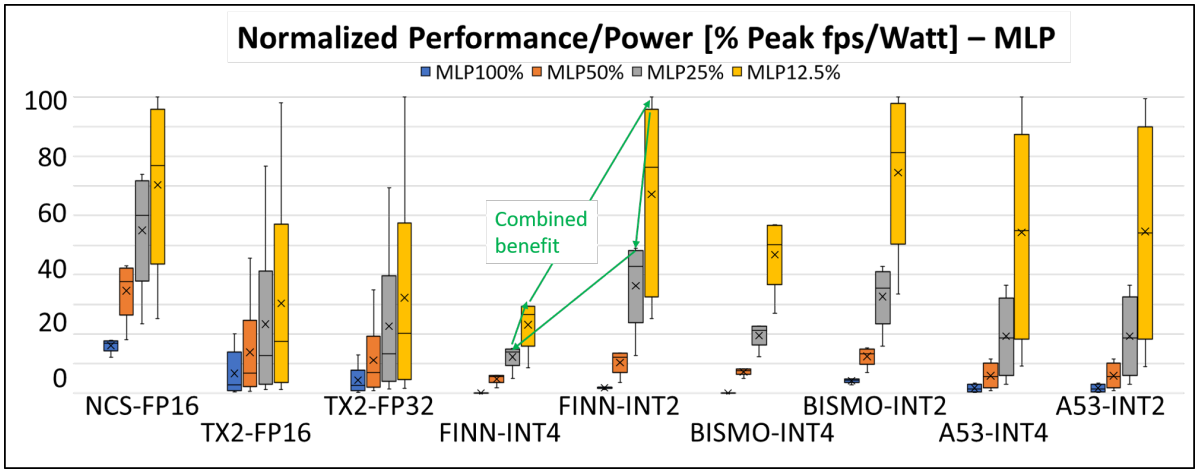


Figure 8.9: Effect on efficiency (throughput/power) with pruning and quantization for MLP

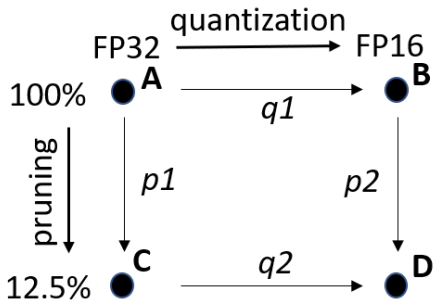


Figure 8.10: Orthogonality of pruning and quantization

optimizations, and $p2, q2$ the second degree optimizations. p stands for pruning and q stands for quantization. If D is better than C and B (lower for latency, and higher for throughput), then the optimizations are complimentary. If $p2/p1 = 1$ and $q2/q1 = 1$, then the optimizations are perfectly orthogonal.

In Table 8.1 we show some examples for the CNV and MLP topologies. We only included examples where optimizations were effective and both pruning and quantization on the same CNN can be applied [14]. As can be seen, the measured results show that D is always greater than B and C and the ratios $p2/p1$ and $q2/q1$ are, apart from a couple exceptions, close to 100%, showing that they are orthogonal and can be effectively combined.

[14]Note: We report for latency batch size = 1 and for throughput highest batch size.

Table 8.1: Orthogonal optimizations

Metric	CNN	Hardware Scenario	A	B	C	D	p1	q1	p2	q2	p2/p1	q2/q1
Latency [msec]	CNV	TX2, 100%-50%, FP32-FP16	1.69	1.12	0.65	0.53	0.38	0.66	0.47	0.82	**123%**	**123%**
		FINN, 100%-50%, INT4-INT2	4.87	0.51	1.30	0.27	0.27	0.10	0.53	0.21	**198%**	**198%**
		BISMO, 50%-25%,INT4-INT2	6.86	4.21	4.18	2.66	0.61	0.61	0.63	0.64	**104%**	**104%**
	MLP	TX2, 100%-50%, FP32-FP16	0.72	0.63	0.29	0.29	0.40	0.88	0.46	1.00	**114%**	**114%**
		BISMO, 100%-50%,INT4-INT2	3.26	2.15	1.19	0.85	0.37	0.66	0.40	0.71	**108%**	**108%**
		A53, 100%-50%,INT4-INT2	101.38	100.79	26.78	26.68	0.26	0.99	0.26	1.00	**100%**	**100%**
Throughput [fps]	CNV	TX2, 100%-50%, FP32-FP16	1,236	2,231	4,385	7,707	3.55	1.81	3.45	1.76	**97%**	**97%**
		FINN, 100%-50%, INT4-INT2	683	2,979	8,149	17,030	11.93	4.36	5.72	2.09	**48%**	**48%**
		BISMO, 100%-50%,INT4-INT2	7,643	19,044	14,172	33,542	1.85	2.49	1.76	2.37	**95%**	**95%**
	MLP	TX2, 100%-50%, FP32-FP16	45,810	70,611	120,414	162,308	2.63	1.54	2.30	1.35	**87%**	**87%**
		FINN, 50%-25%, INT4-INT2	242,884	484,849	598,131	1,932,075	2.46	2.00	3.98	3.23	**162%**	**162%**
		BISMO, 100%-50%,INT4-INT2	828	1,605	2,728	5,104	3.29	1.94	3.18	1.87	**97%**	**97%**

Overall Design Space

The remaining questions are, first, for the same level of accuracy, do we derive more benefit from pruning or using reduced precision for ImageNet, CIFAR-10 and MNIST classification? The optimal solutions can be identified along the pareto frontier in the graph between accuracy and performance (or latency). Second, in embedded applications, latency is typically critical, and it is essential to understand what performance can be achieved within a given latency constraint. This can potentially limit the exploitation of batch- or thread-level parallelism. As such it is important to understand to what extend pruning and quantization can help and affect overall behaviour. In order to answer this questions, we look at at accuracy versus performance trade-offs and latency versus throughput separately below.

Accuracy versus Performance. We first examine the trade-off between accuracy and performance in regards to fps and ips. When looking at the design space of platforms and optimization techniques, we can observe that overall, combining pruning and quantization delivers the best result. Pruning seems to be more effective than quantization across a broader spectrum of platforms and topologies, however reducing precision is more effective for MLPs. For example, FINN-INT2-25% outperforms the FINN-INT4-12.5%. In this scenario, FINN also outperforms other architectures by far. This is due to the memory bound nature of the topology for FP16 and FP32 which throttles the compute performance on NCS and TX2. For INT4 and INT2, the model can remain on-chip for FINN while still offering competitive accuracy for this dataset and topology. We also make several other observations from these graphs. First, the USB devices are far lower in overall performance as expected from the theoretical analysis. Second, the bit-serial implementations do not feature on the pareto frontier, which is expected, as these implementations are not performance optimized. Third, FPGAs outperform others with regards to throughput in particular where quantization is available.

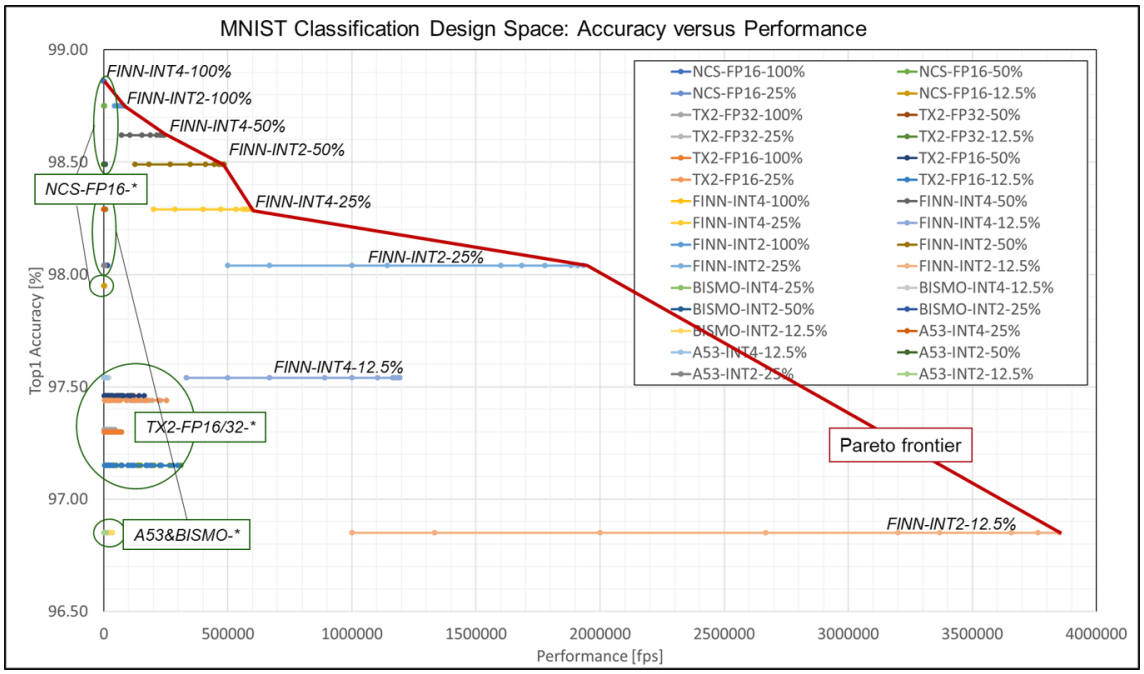


Figure 8.11: MNIST classification design space

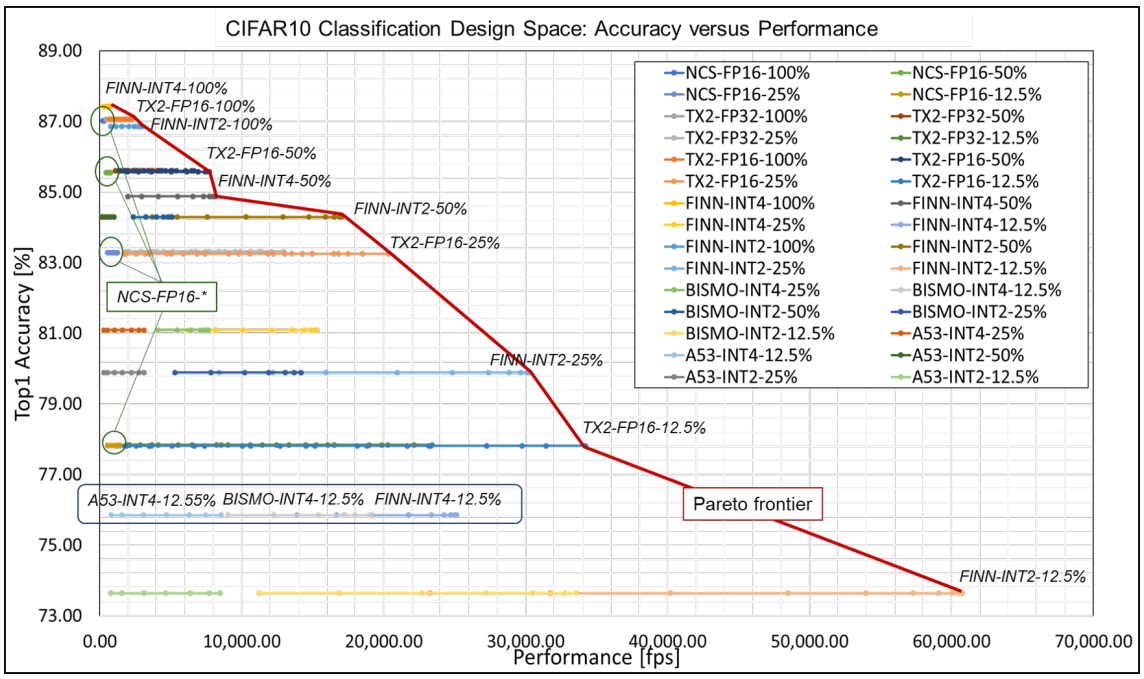


Figure 8.12: CIFAR-10 classification design space

For ImageNet classification, as shown in Figure 8.13, we have a few more topologies to compare with, and we can see that they have a massive impact in regards to performance and accuracy and in fact outweigh benefits from optimization techniques. Quantization from FP32 to

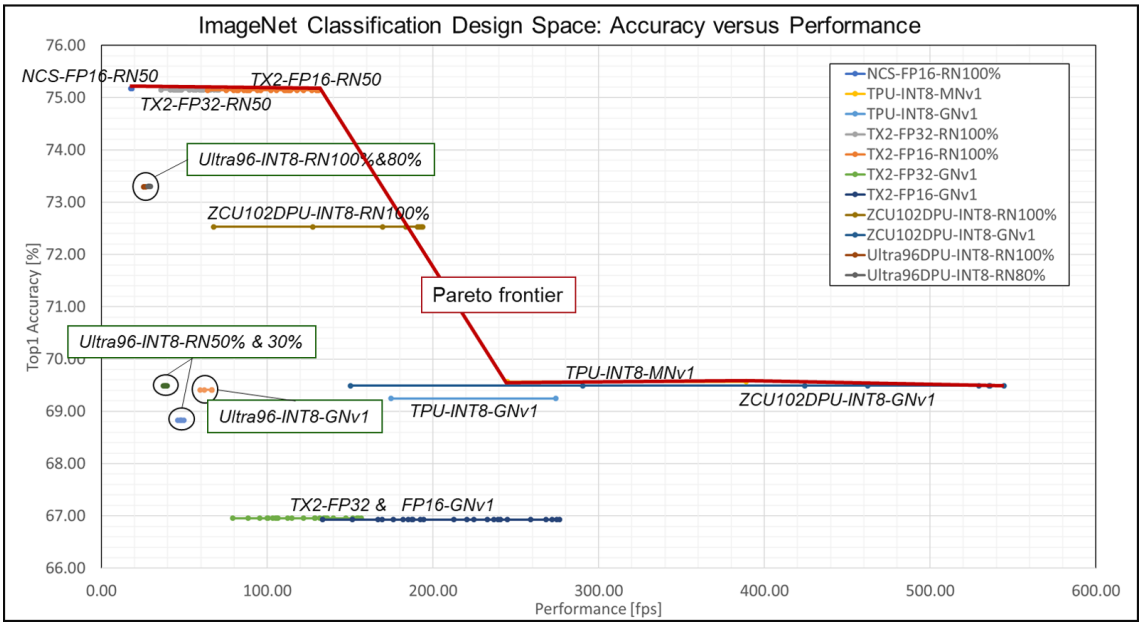


Figure 8.13: ImageNet classification design space

FP16 has in essence no impact on accuracy and FP16 variants bring massive performance benefits. Pruning impacts accuracy when dropping below 50%, but up to then the performance benefits come basically for free. The NCS is far below competitive devices in regards to performance overall. The TPU offers GoogLeNetv1 performance on par with the TX2 whereby the INT8 accuracy is higher than TX2's FP16 and FP32, most likely due to superior image preprocessing in TensorFlow. The TPU provides a pareto optimal design point with MobileNetv1, clearly showing that MobileNetv1 is an exceptional topology with regard to compute cost-accuracy compromise and outperforms GoogLeNetv1 variants on other devices. For the Ultra96, one can observe that pruning brings performance benefits at small accuracy reduction, above 80%, however overall the Ultra96 is too low in performance to offer optimal solutions. Finally, the ZCU102-DPU provides pareto optimal implementations with both GoogLeNet and ResNet variants. Unfortunately, in regards to ImageNet classification, we were not able to carry out a more systematic evaluation as many models were not supported by the various hardware platforms. Many more comparisons could be conducted by correlating power and latency with accuracy, or latency with throughput. This will be addressed in future work.

Latency versus Performance. Finally, we consider the compromise between latency and performance in Figures 7.13 and 7.14. More datapoints are visualized in the web portal, where the charts are also interactive and easier to parse. As detailed in Section 3, many hardware platforms require increased batch sizes, thread counts or stream sizes in order to

extract maximum performance out of a hardware platform and achieve high compute efficiency. However, this can result in substantially different latency as is visualized in the following graphs, whereby we had to split up hardware platforms into different charts due to the large difference in ranges. As shown in previous results and explained in Section 3, FINN delivers lowest latency with no jitter for increasing stream sizes, whereby the performance in regards to fps will saturate when the pipeline is fully utilized. This is reflected in the right part of Figure 7.13, where the FINN results almost overlap with the actual y-axis. This is true for all platforms. Shown here is only a subset, namely CIFAR-10 and ImageNet classification. For the layer-by-layer compute platforms such as NCS, BISMO, A53, and TX2, one can observe that the latency always increases with larger batch size and thread counts respectively, because a sequence of images has to be buffered before each of the layers can be executed which adds significant delays. The compute performance flattens out once peak performance has been reached. The extreme variant is the NCS which has already achieved full throughput at batch=1. Therefore with increasing batch size only latency rises but throughput stays constant. The DPU implementations reach performance saturation for very small thread counts. Finally, for the TPU we have only 2 datapoints: one for slow clock and one for fast clock operation, hence the unusual shape in the diagram.

Summary

In summary, the following observations were made:

- Pruning brings performance, latency and power benefits across all platforms for all tested topologies in direct correlation to the reduction in giga operation (GOP) per input.
- Quantization brings benefits on all topologies but only to the extent that the hardware platform provide native support for the reduced precision datatype. As a result, FPGAs can benefit the most from this technique, as arbitrary datatype operators can be implemented (as long as fixed overlays are not used), even more so than from pruning. Compute performance scales directly with the precision reduction, in particular for MLP where the reduction in model size helps lower the inherent memory bottleneck. The chosen GPU, TPU, NCS, and ARM processor can only support a subset of INT8, FP16, and FP32, and when moving from FP32 to FP16 on these platforms where possible, linear speedup is achieved.

- Power improvements are limited with FPGAs as idle power dominates overall power consumption, whereas for all other devices we see a small improvement with pruning or quantization.

- Furthermore, we provide the experimental proof that quantization and pruning are orthogonal to each other and that the overall speedup and latency reduction is a direct combination of the individual gains. Combined pruning and quantization delivers the best result in the overall design space. This might not necessarily be intuitive as we expect there is a limited amount of redundancy within the CNN.

- Among the chosen test platforms, FPGAs outperform, in particular for MLPs, in regards to latency and throughput in particular when quantization yields acceptable accuracy.

- Layer-by-layer compute approaches result in much higher latency and latency variation, as a lot of buffering is required in order to achieve high compute efficiency.

- Finally, CNN topologies have a massive impact on compute performance and accuracy and can outweigh benefits from optimization techniques.

8.4 Impact on the Wider Research Community

In order to maximize the impact of our efforts, we applied the following ideas:

1. Firstly, we adhere to open and FAIR data principles. This way we provide free open access to all data in a specific format, catalogued with meta data and under clear and suitable licenses.

2. All code is open source. This allows the community to adopt existing visualization routines with ease and expand upon the various implementations.

3. We enable community contributions through a web portal through a separate section where third party contributions both theoretical and measured can be contributed. The web portal also serves as a platform to bring a community together.

4. We describe a concrete proposal to integrate QuTiBench with MLPerf. We will discuss the details of this further below.

8.4.1 Combining QuTiBench and MLPerf

As mentioned before, one of the most important industry-wide efforts in regards to benchmarking NNs is MLPerf, which emerged after the start of my PhD and has significantly unfolded during the course of the thesis. Yet, despite a similar motivation, namely driving clarity into this highly complicated design space, there are substantial differences with the approaches taken. The details of this have been discussed in the previous Chapter 7. In summary, QuTiBench has more focus on providing a deeper system level understanding for the various hardware platforms, through a much more general and systematic measurement of all figures of merit. With that, QuTiBench provides predictions and guidance for new application scenarios. In contrast, MLPerf is dedicated to very specific, yet representative, application scenarios, having selected two image classification and object detection CNNs (one light-weight and one heavy-weight each) plus a machine translation network. All 5 of them are tested in 4 different load scenarios (single stream, multi-stream, server and offline). For each of the load scenarios, only 1 specific figure of merit is reported for a specific minimum application accuracy, whereas QuTiBench rigorously reports everything and is not constraint to these 5 application scenarios. As such QuTiBench can illustrate and identify potential system-level trade-offs which provide optimal solutions under application constraints that may not be covered by MLPerf. Furthermore, through its theoretical analysis and performance predictions, a missing concept from MLPerf, QuTiBench can provide predictions and insights for completely new topologies, such as which optimization techniques are the most likely to provide optimal solutions, or how certain techniques will impact performance and power consumption without having to try every single one. Finally, the explicit support for optimizations at all tiers of the benchmark are vital in particular in the embedded space, where most solutions are co-designed with the hardware and heavily optimized. However, to be real, QuTiBench is just a PhD project and as such cannot compete with the general level of industry support and level of adoption that MLPerf has achieved. Also, MLPerf has substantially more detail when it comes to providing statistical relevance and ensuring reproducibility, and system level evaluation, addressing potential cheat techniques such as taking advantage of on-the-fly caching detection.

Ideally, as the techniques on QuTiBench and MLPerf mostly complement each other, the benchmarks could get combined. There is real scope of adding some of the QuTiBench aspects to MLPerf. We discuss below the specifics on how this could work and have visualized this in Figure 8.14. MLPerf is shown with its 5 different machine learning tasks, in 4 different

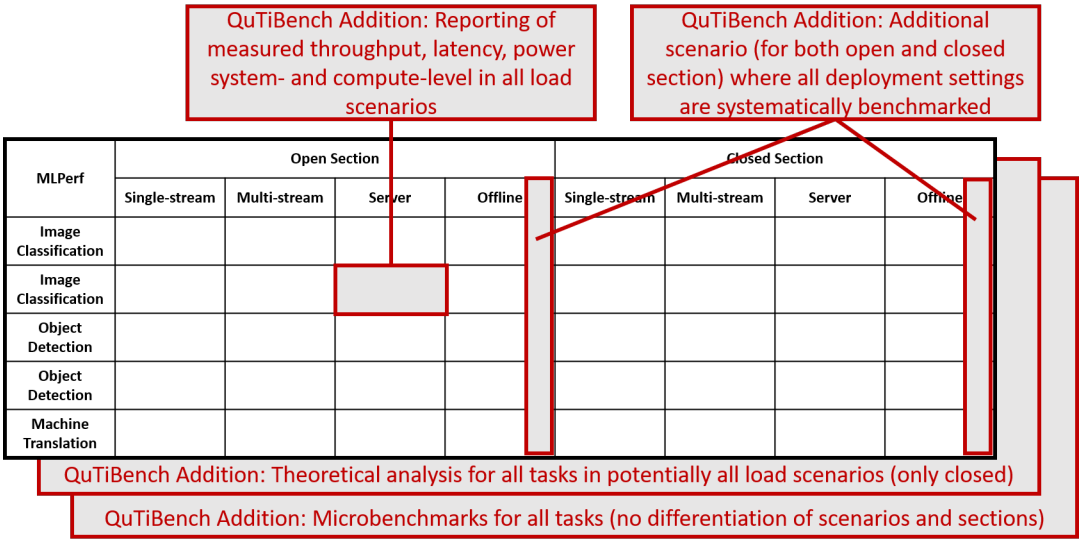


Figure 8.14: Proposal on how MLPerf and QuTiBench could be potentially combined

load scenarios (offline, server, single- and multi-stream) for both closed and open sections. Firstly, adding theoretical performance and corresponding analysis in MLPerf can help with tracking of achieved compute efficiency within the given application contexts and also enable performance predictions for system designers. A key consideration in this regard is whether to provide theoretical analysis for all load scenarios or just for a subset. Either way, this could be considered as another tier to the existing MLPerf benchmark. Similarly, systematic capturing of all figures of merit over all deployment settings would offer much more generality to MLPerf. This could be considered as separate category, an additional mode for both closed and open division, where systematically all figures of merit, both system and compute only aspects, and latency, throughput and power are captured, across all possible deployment sections. Furthermore, beyond the single figure of merit, we would propose to report all figures of merit, in all other load scenarios. In addition, microbenchmarking, as shown in the previous chapter (as well as the reporting of compute only performance), can bring highly useful system-level insights and characterize difficult compute patterns, data movement bottlenecks, and buffer limitations. Again this would form a complete additional tier, for all machine learning tasks, although in this context, it would not make sense to differentiate between the load scenarios and open and closed section. The microbenchmarking aspect of QuTiBench could augment MLPerf in a way that would make it much more interesting for system designers, whereby there wouldn't be any differentiation between scenarios and sections. Finally, the support for co-designed implementations is very restricted due to accuracy limitations within the closed section, and only exposes a small fraction of the design space. The MLPerf open section helps,

however reports only a subset of the figures of merit. As shown in the previous chapter, the systematic reporting of all potential solutions and the visualizations over the full design space, and separate reporting of system and compute performance and latency, can provide highly interesting system-level insights and expose additional and essential solutions within the design space. As such, a combination of both approaches could bring significant benefits.

8.5 Future Work

This effort is just at the beginning and there are so many aspects that we couldn't complete as part of this thesis. Future work will focus on a number of those which we summarize here and discussed in greater detail below:

1. Broadening application scope, including new topologies, data center platforms, and CNN training
2. Refining theoretical models
3. Out-of-the-box versus optimized performance
4. Testbeds, reproducibility, & recorded measurements
5. Outreach

Broadening Scope. Firstly in regard to broadening scope: The design space in this field is extremely complex and increasing further every day, as other algorithms are getting replaced with CNNs. To provide some examples: According to OpenAI, compute has been doubling every 3.4 months[15], in particular through the rise of language processing with Alexa, Siri, and GPT-3 to name a few examples. Typical models in this space are 10x to 200x larger than ResNet50. In particular, Turing-NLG is a 17-billion-parameter language model by Microsoft (Feb,2020)[16], and OpenAI's GPT-3 has 175 billion-parameters. The continuous growth of model size is further fueled by **Reinforcement Learning** and **Network Architecture Search**. This needs to be carefully addressed and continuously evolve. Not only the topologies are changing and getting bigger and more complex, but also novel hardware architectures are

[15]https://blog.openai.com/ai-and-compute
[16]https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft

continually emerging. Also, we have so far only focused on embedded deployments. There are as many server-class accelerator platforms available, that have not been considered in our experimentation. Benchmarking with these brings new challenges, in particular in regards to isolating power measurements for accelerators, and comparing CPUs on motherboards with accelerator cards. Finally, as part of this thesis, we have only addressed inference. CNN training will be a significant aspect to our future work. Training is even more complex compared to inference due to the numerous hyperparameters that influence the behaviours. Examples are optimization strategy, learning rates, and initialization. Overall, we plan to proactively broaden the scope of machine learning tasks and topologies and include novel accelerator approaches and optimization techniques as they become available and expand to data center platforms and include training.

Refinements. In regard to refinement, we plan to expand on level-0 first, by refining theoretical predictions as was suggested in previous chapters. Most importantly, this would leverage percentage of compute resources used for FPGAs, and perhaps entail for other platforms some form of expected utilization. Also, the refining of the memory model, by taking on-chip memory into account, is easy to carry out and should bring significant benefits. Finally, we intend to introduce a scalar factor that ameliorates the effect of lower compute efficiency with smaller model size.

Out-of-the-box versus Optimized Performance. For all hardware platforms, one could always argue that there is scope for improving the performance. As such the question whether a given measured performance is representative is hard to answer. Furthermore, is it fair to compare a solution which has been fine-tuned over a year, to one that you get from a naive implementation? As a consequence, it is really important to take development effort into account. We plan to extend our efforts by bringing development effort in as an additional figure of merit which can help qualify the results. Furthermore, we intend to include naive or out-of-the-box performance too. Bringing both of these aspects into the benchmarking will bring additional insights to the community and relativise the measurements

Testbeds, Reproducibility, & Recorded Measurements. In order to improve the usefulness of the scientific results, it would be very helpful to validate all experiments and measurements. Specifically, all input data to the test suites should be openly accessible and all test code

open source. Furthermore, an open testbed may be advisable where a few instances of each accelerator type is available through the cloud. This could be considered as an extension to these benchmark which enables the community to run experimentation on a broad selection of diverse hardware accelerators and brings considerable benefits in regard to reproducibility (when all users run the same code on the same platforms). As the higher levels of benchmarks may require a long time to run and hardware may not be available, we advocate recording of results, whereby each entry would be ideally validated by a third party such that results are guaranteed to be a) reproducible and b) correct. Our colleagues in the Request Tournament effort [127] leverage ACM's rigorous artifact evaluation technology and the Collective Knowledge Workflow Framework [134] do an outstanding job addressing this. We aim to adopt the same principles in the future.

Outreach. In addition to the specific extensions to our work, as explained above, we plan to work on outreach. The success of this effort depends on its adoption within the industry. Specifically, we're planning to work on the proliferation of this effort through talks and publications. We intend to take an active roll in the MLPerf working groups to drive adoption of QuTiBench aspects into MLPerf. Finally, we intend to kick off a social media campaign for QuTiBench as the web portal has slowly matured into a state where it can handle community contributions.

8.6 Closing Remarks.

With this, we reach the end of the thesis. In conclusion, we hope that the benchmarking methodology brings useful insights to the community and helps drive fair and objective benchmarking of an increasingly diverse and complex spectrum of hardware architectures in the future. Finally, we hope that the numerous results that we have provided create a wider understanding of the current state of the art in CNN accelerators, in what can and cannot be achieved, and the individual strengths and weaknesses of the various hardware platforms.

List of Acronyms

ADAS	Advanced Driver Assistance System
ASIC	Application Specific Integrated Circuits
BNN	Binary Neural Network
CNN	Convolutional Neural Network
DPU	Deep Learning Processing Unit
FP32	32-bit Floating Point Representation
FP16	16-bit Floating Point Representation
fps	frames per second
FPGA	Field Programmable Gate Array
GB/sec	giga byte per second
GOP	giga operation
GOP/sec	giga operations per second
GPU	Graphics Processing Unit
ips	inputs per second
INT8	8-bit Fixed Point INTEGER
MAC	Multiply Accumulate
ME	millions of elements
ML	Machine Learning

MPE	Matrix of Processing Engines
msec	milliseconds
NCS	Neural Compute Stick
NLP	Natural Language Processing
NN	Neural Network
OI	Operational or Arithmetic Intensity
QNN	Quantized Neural Networks
SGD	Stochastic Gradient Descent
TDP	Thermal Design Power
TOP	tera operation
TOP/sec	tera operations per second
TPU	Tensor Processing Unit
VLIW	Very Large Instruction Word
W	Watt

Appendices

A Measurements

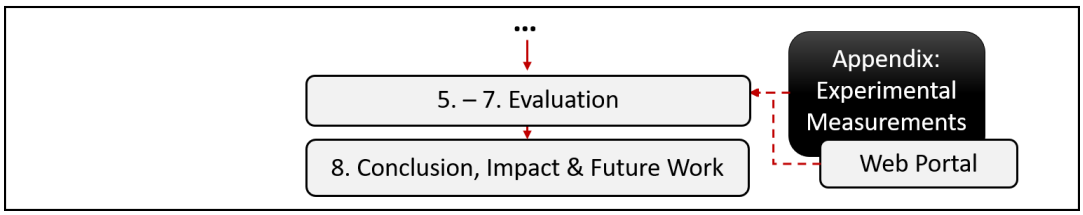


Figure A.1: Measurements

In this chapter, we provide all experimental data points. These are however only a snapshot in time while our web portal at: `https://rcl-lab.github.io/QutibenchWeb` will be continuously updated and where also third party contributions are located. In more detail, the reader can find all level-1, level-2 and level-3 results in this part. The corresponding theoretical analysis is captured in chapter 6.

Table A.1: Level-1 -ZCU104 inference results ResNet50 individual convolutional layers

ZCU104	Network Parameters						Figures of Merit			
							thread=1		thread=8	
Layer	[MOP]	in dim	in ch	filter	stride	out ch	Latency [ms]	Throughput (Eff) [GOPs]	Latency [ms]	Throughput (Eff) [GOPs]
res2a_branch2a	25.7	56	64	1	1	64	0.060	428.05 (0.09)	0.082	630.21 (0.14)
res2a_branch2b	231.2	56	64	3	1	64	0.190	1216.84 (0.26)	0.190	2428.70 (0.53)
res2a_branch2c	102.8	56	64	1	1	256	0.220	467.23 (0.10)	0.258	798.45 (0.17)
res2a_branch1	102.8	56	64	1	1	256	0.430	239.08 (0.05)	0.464	443.34 (0.10)
res2b_branch2a	102.8	56	256	1	1	64	0.142	725.63 (0.16)	0.196	1049.57 (0.23)
res2b_branch2b	231.2	56	64	3	1	64	0.190	1216.84 (0.26)	0.190	2428.19 (0.53)
res2b_branch2c	102.8	56	64	1	1	256	0.429	239.64 (0.05)	0.463	443.98 (0.10)
res2c_branch2a	102.8	56	256	1	1	64	0.140	734.13 (0.16)	0.193	1063.14 (0.23)
res2c_branch2b	231.2	56	64	3	1	64	0.190	1216.84 (0.26)	0.190	2428.32 (0.53)
res2c_branch2c	102.8	56	64	1	1	256	0.435	236.20 (0.05)	0.462	444.69 (0.10)
res3a_branch2a	51.4	28	256	1	2	128	0.090	571.05 (0.12)	0.128	800.12 (0.17)
res3a_branch2b	231.2	28	128	3	1	128	0.210	1100.95 (0.24)	0.214	2159.44 (0.47)
res3a_branch2c	102.8	28	128	1	1	512	0.210	489.52 (0.11)	0.247	832.79 (0.18)
res3a_branch1	205.5	28	256	1	2	512	0.330	622.71 (0.14)	0.390	1052.87 (0.23)
res3b_branch2a	102.8	28	512	1	1	128	0.120	856.45 (0.19)	0.148	1391.07 (0.30)
res3b_branch2b	231.2	28	128	3	1	128	0.210	1100.95 (0.24)	0.214	2165.20 (0.47)
res3b_branch2c	102.8	28	128	1	1	512	0.320	321.24 (0.07)	0.353	582.88 (0.13)
res3c_branch2a	102.8	28	512	1	1	128	0.120	856.60 (0.19)	0.151	1361.41 (0.30)
res3c_branch2b	231.2	28	128	3	1	128	0.210	1100.95 (0.24)	0.215	2154.81 (0.47)
res3c_branch2c	102.8	28	128	1	1	512	0.303	339.02 (0.07)	0.354	580.14 (0.13)
res3d_branch2a	102.8	28	512	1	1	128	0.120	856.52 (0.19)	0.149	1383.86 (0.30)
res3d_branch2b	231.2	28	128	3	1	128	0.210	1100.95 (0.24)	0.214	2165.50 (0.47)
res3d_branch2c	102.8	28	128	1	1	512	0.301	341.20 (0.07)	0.353	582.72 (0.13)
res4a_branch2a	51.4	14	512	1	2	256	0.120	428.80 (0.09)	0.133	774.21 (0.17)
res4a_branch2b	231.2	14	256	3	1	256	0.210	1100.95 (0.24)	0.230	2011.48 (0.44)
res4a_branch2c	102.8	14	256	1	1	1024	0.290	354.46 (0.08)	0.379	541.92 (0.12)
res4a_branch1	205.5	14	512	1	2	1024	0.430	477.87 (0.10)	0.500	821.34 (0.18)
res4b_branch2a	102.8	14	1024	1	1	256	0.130	790.71 (0.17)	0.162	1271.41 (0.28)
res4b_branch2b	231.2	14	256	3	1	256	0.210	1100.95 (0.24)	0.229	2015.61 (0.44)
res4b_branch2c	102.8	14	256	1	1	1024	0.350	293.69 (0.06)	0.436	471.20 (0.10)
res4c_branch2a	102.8	14	1024	1	1	256	0.130	790.71 (0.17)	0.163	1263.60 (0.27)
res4c_branch2b	231.2	14	256	3	1	256	0.210	1100.95 (0.24)	0.231	2002.86 (0.44)
res4c_branch2c	102.8	14	256	1	1	1024	0.360	285.52 (0.06)	0.438	469.22 (0.10)
res4d_branch2a	102.8	14	1024	1	1	256	0.130	790.65 (0.17)	0.164	1251.14 (0.27)
res4d_branch2b	231.2	14	256	3	1	256	0.210	1100.95 (0.24)	0.229	2019.57 (0.44)
res4d_branch2c	102.8	14	256	1	1	1024	0.350	293.76 (0.06)	0.425	484.02 (0.11)
res4e_branch2a	102.8	14	1024	1	1	256	0.130	790.71 (0.17)	0.162	1267.18 (0.28)
res4e_branch2b	231.2	14	256	3	1	256	0.210	1100.90 (0.24)	0.230	2014.73 (0.44)
res4e_branch2c	102.8	14	256	1	1	1024	0.350	293.68 (0.06)	0.438	469.19 (0.10)
res4f_branch2a	102.8	14	1024	1	1	256	0.130	790.53 (0.17)	0.162	1265.78 (0.27)
res4f_branch2b	231.2	14	256	3	1	256	0.210	1100.95 (0.24)	0.230	2007.12 (0.44)
res4f_branch2c	102.8	14	256	1	1	1024	0.360	285.49 (0.06)	0.421	488.23 (0.11)
res5a_branch2a	51.4	7	1024	1	2	512	0.120	427.94 (0.09)	0.188	546.52 (0.12)
res5a_branch2b	231.2	7	512	3	1	512	0.330	699.93 (0.15)	0.493	937.72 (0.20)
res5a_branch2c	102.8	7	512	1	1	2048	0.470	218.66 (0.05)	0.600	342.79 (0.07)
res5a_branch1	205.5	7	1024	1	2	2048	0.517	397.60 (0.09)	0.691	594.55 (0.13)
res5b_branch2a	102.8	7	2048	1	1	512	0.170	604.28 (0.13)	0.272	755.16 (0.16)
res5b_branch2b	231.2	7	512	3	1	512	0.331	698.07 (0.15)	0.499	926.34 (0.20)
res5b_branch2c	102.8	7	512	1	1	2048	0.500	205.56 (0.04)	0.628	327.34 (0.07)
res5c_branch2a	102.8	7	2048	1	1	512	0.170	604.49 (0.13)	0.265	775.00 (0.17)
res5c_branch2b	231.2	7	512	3	1	512	0.340	679.68 (0.15)	0.503	918.94 (0.20)
res5c_branch2c	102.8	7	512	1	1	2048	0.500	205.55 (0.04)	0.632	325.22 (0.07)

Table A.2: Level-1 -TX2 (MaxN, FP16) inference results ResNet50 individual convolutional layers

TX2	Network Parameters						Figures of Merit			
							MaxN, FP16, batch=1		MaxN, FP16, batch=128	
Layer	[MOP]	in dim	in ch	filter	stride	out ch	Latency [ms]	Throughput (Eff) [GOPs]	Latency [ms]	Throughput (Eff) [GOPs]
res2a_branch2a	25.7	56	64	1	1	64	0.06	414.52 (0.31)	5.05	651.15 (0.49)
res2a_branch2b	231.2	56	64	3	1	64	0.19	1197.93 (0.90)	22.78	1299.39 (0.97)
res2a_branch2c	102.8	56	64	1	1	256	0.18	577.53 (0.43)	20.15	653.15 (0.49)
res2a_branch1	102.8	56	64	1	1	256	0.21	487.20 (0.37)	23.66	556.19 (0.42)
res2b_branch2a	102.8	56	256	1	1	64	0.13	778.79 (0.58)	13.74	957.60 (0.72)
res2b_branch2b	231.2	56	64	3	1	64	0.19	1210.47 (0.91)	22.87	1293.82 (0.97)
res2b_branch2c	102.8	56	64	1	1	256	0.21	489.52 (0.37)	23.68	555.77 (0.42)
res2c_branch2a	102.8	56	256	1	1	64	0.13	784.73 (0.59)	13.74	957.88 (0.72)
res2c_branch2b	231.2	56	64	3	1	64	0.19	1210.47 (0.91)	22.85	1295.01 (0.97)
res2c_branch2c	102.8	56	64	1	1	256	0.21	489.52 (0.37)	23.67	555.82 (0.42)
res3a_branch2a	51.4	28	256	1	2	128	0.09	584.09 (0.44)	7.19	915.30 (0.69)
res3a_branch2b	231.2	28	128	3	1	128	0.21	1095.73 (0.82)	24.63	1201.33 (0.90)
res3a_branch2c	102.8	28	128	1	1	512	0.15	694.59 (0.52)	15.18	866.82 (0.65)
res3a_branch1	205.5	28	256	1	2	512	0.29	718.53 (0.54)	30.35	866.60 (0.65)
res3b_branch2a	102.8	28	512	1	1	128	0.13	767.16 (0.58)	12.07	1090.45 (0.82)
res3b_branch2b	231.2	28	128	3	1	128	0.21	1106.22 (0.83)	24.66	1199.92 (0.90)
res3b_branch2c	102.8	28	128	1	1	512	0.16	634.57 (0.48)	16.68	788.87 (0.59)
res3c_branch2a	102.8	28	512	1	1	128	0.13	767.16 (0.58)	12.10	1087.11 (0.82)
res3c_branch2b	231.2	28	128	3	1	128	0.21	1100.95 (0.83)	24.47	1209.19 (0.91)
res3c_branch2c	102.8	28	128	1	1	512	0.16	634.57 (0.48)	16.71	787.65 (0.59)
res3d_branch2a	102.8	28	512	1	1	128	0.13	767.16 (0.58)	12.12	1085.95 (0.81)
res3d_branch2b	231.2	28	128	3	1	128	0.21	1106.22 (0.83)	24.69	1198.56 (0.90)
res3d_branch2c	102.8	28	128	1	1	512	0.16	630.67 (0.47)	16.67	789.35 (0.59)
res4a_branch2a	51.4	14	512	1	2	256	0.08	642.50 (0.48)	7.10	926.26 (0.69)
res4a_branch2b	231.2	14	256	3	1	256	0.20	1185.64 (0.89)	23.12	1279.89 (0.96)
res4a_branch2c	102.8	14	256	1	1	1024	0.15	708.97 (0.53)	13.01	1011.64 (0.76)
res4a_branch1	205.5	14	512	1	2	1024	0.28	728.72 (0.55)	29.23	899.87 (0.68)
res4b_branch2a	102.8	14	1024	1	1	256	0.13	784.73 (0.59)	11.55	1139.45 (0.85)
res4b_branch2b	231.2	14	256	3	1	256	0.20	1179.59 (0.88)	22.33	1325.28 (0.99)
res4b_branch2c	102.8	14	256	1	1	1024	0.15	680.79 (0.51)	13.75	957.18 (0.72)
res4c_branch2a	102.8	14	1024	1	1	256	0.13	778.79 (0.58)	11.62	1132.10 (0.85)
res4c_branch2b	231.2	14	256	3	1	256	0.20	1173.60 (0.88)	22.99	1287.35 (0.97)
res4c_branch2c	102.8	14	256	1	1	1024	0.15	680.79 (0.51)	13.76	956.14 (0.72)
res4d_branch2a	102.8	14	1024	1	1	256	0.13	778.79 (0.58)	11.57	1137.09 (0.85)
res4d_branch2b	231.2	14	256	3	1	256	0.20	1185.64 (0.89)	22.92	1291.17 (0.97)
res4d_branch2c	102.8	14	256	1	1	1024	0.15	680.79 (0.51)	13.76	956.00 (0.72)
res4e_branch2a	102.8	14	1024	1	1	256	0.13	778.79 (0.58)	11.59	1135.32 (0.85)
res4e_branch2b	231.2	14	256	3	1	256	0.20	1185.64 (0.89)	22.85	1295.41 (0.97)
res4e_branch2c	102.8	14	256	1	1	1024	0.15	680.79 (0.51)	13.78	954.89 (0.72)
res4f_branch2a	102.8	14	1024	1	1	256	0.13	784.73 (0.59)	11.65	1129.96 (0.85)
res4f_branch2b	231.2	14	256	3	1	256	0.20	1179.59 (0.88)	22.40	1321.26 (0.99)
res4f_branch2c	102.8	14	256	1	1	1024	0.15	680.79 (0.51)	13.78	955.17 (0.72)
res5a_branch2a	51.4	7	1024	1	2	512	0.14	372.46 (0.28)	7.61	864.77 (0.65)
res5a_branch2b	231.2	7	512	3	1	512	0.31	748.22 (0.56)	24.90	1188.59 (0.89)
res5a_branch2c	102.8	7	512	1	1	2048	0.27	386.47 (0.29)	12.53	1049.90 (0.79)
res5a_branch1	205.5	7	1024	1	2	2048	0.51	406.93 (0.31)	30.92	850.85 (0.64)
res5b_branch2a	102.8	7	2048	1	1	512	0.22	475.93 (0.36)	11.35	1159.74 (0.87)
res5b_branch2b	231.2	7	512	3	1	512	0.30	763.04 (0.57)	24.91	1188.26 (0.89)
res5b_branch2c	102.8	7	512	1	1	2048	0.27	382.16 (0.29)	13.21	995.87 (0.75)
res5c_branch2a	102.8	7	2048	1	1	512	0.22	473.73 (0.36)	11.39	1155.36 (0.87)
res5c_branch2b	231.2	7	512	3	1	512	0.31	753.09 (0.56)	24.91	1187.88 (0.89)
res5c_branch2c	102.8	7	512	1	1	2048	0.28	371.12 (0.28)	13.09	1005.53 (0.75)

Table A.3: Level-2 - Inference results ResNet50 residual layers

HW	Layer	Parameters	MaxN Lat [ms]	MaxN Throughput (Eff) [GOPs] ([%])	MaxQ Lat [ms]	MaxQ Throughput (Eff) [GOPs] ([%])	MaxP Lat [ms]	MaxP Throughput (Eff) [GOPs] ([%])
TX2	res2a	FP16, b=1	1.37	431.27 (0.32)	1.90	292.13 (0.25)	1.58	371.40 (0.42)
TX2	res2a	FP16, b=2	2.17	464.25 (0.35)	3.08	314.23 (0.27)	2.50	401.64 (0.46)
TX2	res2a	FP16, b=4	3.97	481.73 (0.36)	5.83	325.90 (0.28)	4.61	418.69 (0.48)
TX2	res2a	FP16, b=8	7.69	491.73 (0.37)	11.30	330.22 (0.29)	8.91	426.23 (0.49)
TX2	res2a	FP16, b=16	15.11	495.85 (0.37)	22.39	333.24 (0.29)	17.48	428.54 (0.49)
TX2	res2a	FP16, b=32	30.39	436.04 (0.33)	44.20	333.95 (0.29)	34.49	430.49 (0.49)
TX2	res2a	FP16, b=64	60.41	492.24 (0.37)	88.98	334.18 (0.29)	68.93	430.10 (0.49)
TX2	res2a	FP16, b=128	119.12	495.33 (0.37)	177.37	333.95 (0.29)	137.24	430.10 (0.49)
TX2	res2b	FP16, b=1	1.12	443.23 (0.33)	1.57	303.00 (0.26)	1.32	382.90 (0.44)
TX2	res2b	FP16, b=2	1.95	481.92 (0.36)	2.77	333.25 (0.29)	2.29	416.41 (0.48)
TX2	res2b	FP16, b=4	3.63	502.50 (0.38)	5.24	343.22 (0.30)	4.19	437.17 (0.50)
TX2	res2b	FP16, b=8	6.95	509.95 (0.38)	10.06	352.14 (0.31)	7.99	445.00 (0.51)
TX2	res2b	FP16, b=16	13.62	515.82 (0.39)	19.90	354.66 (0.31)	15.86	448.12 (0.51)
TX2	res2b	FP16, b=32	27.19	518.82 (0.39)	39.57	356.64 (0.31)	31.05	451.28 (0.52)
TX2	res2b	FP16, b=64	54.32	515.82 (0.39)	78.80	356.35 (0.31)	62.09	452.20 (0.52)
TX2	res2b	FP16, b=128	108.24	517.02 (0.39)	158.53	355.50 (0.31)	124.29	450.38 (0.52)
TX2	res2c	FP16, b=1	1.12	446.33 (0.33)	1.59	301.77 (0.26)	1.32	380.94 (0.44)
TX2	res2c	FP16, b=2	1.96	483.48 (0.36)	2.75	333.75 (0.29)	2.27	419.53 (0.48)
TX2	res2c	FP16, b=4	3.60	504.19 (0.38)	5.16	346.15 (0.30)	4.17	438.88 (0.50)
TX2	res2c	FP16, b=8	6.89	512.87 (0.38)	10.12	347.49 (0.30)	8.04	446.33 (0.51)
TX2	res2c	FP16, b=16	13.59	516.42 (0.39)	19.86	355.50 (0.31)	15.70	451.28 (0.52)
TX2	res2c	FP16, b=32	27.01	518.22 (0.39)	39.22	356.92 (0.31)	31.14	451.74 (0.52)
TX2	res2c	FP16, b=64	54.36	515.82 (0.39)	78.66	356.64 (0.31)	62.06	448.57 (0.51)
TX2	res2c	FP16, b=128	108.07	516.42 (0.39)	158.11	355.22 (0.31)	123.99	449.92 (0.51)
TX2	res3a	FP16, b=1	1.39	475.68 (0.36)	1.96	323.22 (0.28)	1.66	406.90 (0.47)
TX2	res3a	FP16, b=2	2.38	523.41 (0.39)	3.45	356.13 (0.31)	2.81	449.19 (0.51)
TX2	res3a	FP16, b=4	4.39	555.61 (0.42)	6.43	374.19 (0.33)	5.18	476.43 (0.55)
TX2	res3a	FP16, b=8	8.46	563.90 (0.42)	12.39	385.63 (0.34)	9.80	490.72 (0.56)
TX2	res3a	FP16, b=16	16.53	575.15 (0.43)	24.45	390.61 (0.34)	19.22	495.95 (0.57)
TX2	res3a	FP16, b=32	32.86	576.80 (0.43)	48.62	391.37 (0.34)	38.07	498.40 (0.57)
TX2	res3a	FP16, b=64	65.46	577.35 (0.43)	96.24	393.92 (0.34)	75.66	500.05 (0.57)
TX2	res3a	FP16, b=128	133.09	568.13 (0.43)	194.49	389.61 (0.34)	151.72	496.77 (0.57)
TX2	res3b	FP16, b=1	1.03	511.11 (0.38)	1.41	347.49 (0.30)	1.21	438.45 (0.50)
TX2	res3b	FP16, b=2	1.69	562.54 (0.42)	2.40	385.54 (0.34)	1.96	488.23 (0.56)
TX2	res3b	FP16, b=4	3.05	597.89 (0.45)	4.43	407.31 (0.35)	3.58	517.02 (0.59)
TX2	res3b	FP16, b=8	5.80	616.86 (0.46)	8.45	420.72 (0.37)	6.75	533.04 (0.61)
TX2	res3b	FP16, b=16	11.26	629.89 (0.47)	16.50	426.74 (0.37)	12.95	544.73 (0.62)
TX2	res3b	FP16, b=32	22.24	630.78 (0.47)	33.51	428.78 (0.37)	25.61	547.40 (0.63)
TX2	res3b	FP16, b=64	44.38	628.12 (0.47)	65.79	434.20 (0.38)	51.07	546.72 (0.63)
TX2	res3b	FP16, b=128	87.72	636.16 (0.48)	129.87	434.62 (0.38)	101.77	548.74 (0.63)
TX2	res3c	FP16, b=1	1.05	506.48 (0.38)	1.42	343.22 (0.30)	1.22	436.74 (0.50)
TX2	res3c	FP16, b=2	1.70	561.84 (0.42)	2.43	380.61 (0.33)	1.96	486.64 (0.56)
TX2	res3c	FP16, b=4	3.06	598.69 (0.45)	4.46	406.57 (0.35)	3.56	515.82 (0.59)
TX2	res3c	FP16, b=8	5.81	613.47 (0.46)	8.47	419.93 (0.37)	6.70	536.24 (0.61)
TX2	res3c	FP16, b=16	11.19	630.78 (0.47)	16.40	428.78 (0.37)	12.97	541.43 (0.62)
TX2	res3c	FP16, b=32	22.05	632.56 (0.47)	32.55	430.43 (0.37)	25.68	546.72 (0.63)
TX2	res3c	FP16, b=64	43.94	637.07 (0.48)	64.34	434.20 (0.38)	50.46	552.13 (0.63)
TX2	res3c	FP16, b=128	87.87	636.16 (0.48)	129.02	431.68 (0.38)	101.34	550.09 (0.63)
TX2	res3d	FP16, b=1	1.04	504.19 (0.38)	1.40	347.76 (0.30)	1.22	438.45 (0.50)
TX2	res3d	FP16, b=2	1.70	561.13 (0.42)	2.40	385.20 (0.34)	1.95	488.77 (0.56)
TX2	res3d	FP16, b=4	3.08	592.35 (0.44)	4.43	408.05 (0.36)	3.55	515.23 (0.59)
TX2	res3d	FP16, b=8	5.84	613.47 (0.46)	8.41	423.10 (0.37)	6.71	536.88 (0.61)
TX2	res3d	FP16, b=16	11.25	624.61 (0.47)	16.48	427.15 (0.37)	12.93	546.72 (0.63)
TX2	res3d	FP16, b=32	22.04	631.67 (0.47)	32.74	430.85 (0.37)	25.51	551.45 (0.63)
TX2	res3d	FP16, b=64	44.27	633.46 (0.48)	64.21	435.89 (0.38)	50.51	553.49 (0.63)
TX2	res3d	FP16, b=128	88.66	630.78 (0.47)	129.91	431.27 (0.38)	101.91	549.41 (0.63)

Table A.4: Level-2 - Inference results ResNet50 residual layers

HW	Layer	Parameters	MaxN Lat [ms]	MaxN Throughput (Eff) [GOPs] ([%])	MaxQ Lat [ms]	MaxQ Throughput (Eff) [GOPs] ([%])	MaxP Lat [ms]	MaxP Throughput (Eff) [GOPs] ([%])
TX2	res4a	FP16, b=1	1.20	575.15 (0.43)	1.66	386.37 (0.34)	1.39	496.77 (0.57)
TX2	res4a	FP16, b=2	2.09	604.46 (0.45)	3.01	411.33 (0.36)	2.41	524.32 (0.60)
TX2	res4a	FP16, b=4	3.74	654.12 (0.49)	5.46	446.87 (0.39)	4.34	569.20 (0.65)
TX2	res4a	FP16, b=8	7.00	688.35 (0.52)	10.16	472.33 (0.41)	8.03	600.26 (0.69)
TX2	res4a	FP16, b=16	13.40	706.02 (0.53)	19.71	480.59 (0.42)	15.37	621.85 (0.71)
TX2	res4a	FP16, b=32	26.40	713.52 (0.54)	38.60	493.12 (0.43)	30.30	627.01 (0.72)
TX2	res4a	FP16, b=64	52.32	722.89 (0.54)	76.46	494.33 (0.43)	60.23	628.96 (0.72)
TX2	res4a	FP16, b=128	104.66	723.76 (0.54)	152.47	496.77 (0.43)	119.12	633.57 (0.72)
TX2	res4b	FP16, b=1	1.03	599.49 (0.45)	1.40	407.31 (0.35)	1.16	514.64 (0.59)
TX2	res4b	FP16, b=2	1.53	644.41 (0.48)	2.11	437.60 (0.38)	1.74	562.54 (0.64)
TX2	res4b	FP16, b=4	2.60	706.51 (0.53)	3.81	474.76 (0.41)	3.03	610.96 (0.70)
TX2	res4b	FP16, b=8	4.87	740.43 (0.56)	7.16	497.46 (0.43)	5.58	648.15 (0.74)
TX2	res4b	FP16, b=16	9.33	763.18 (0.57)	13.65	520.03 (0.45)	10.82	660.60 (0.76)
TX2	res4b	FP16, b=32	18.13	769.75 (0.58)	26.64	526.15 (0.46)	20.84	674.54 (0.77)
TX2	res4b	FP16, b=64	36.31	776.43 (0.58)	52.95	530.51 (0.46)	41.50	675.56 (0.77)
TX2	res4b	FP16, b=128	71.33	780.49 (0.59)	105.55	529.88 (0.46)	82.15	680.70 (0.78)
TX2	res4c	FP16, b=1	1.03	598.69 (0.45)	1.40	408.42 (0.36)	1.16	511.11 (0.58)
TX2	res4c	FP16, b=2	1.52	647.21 (0.49)	2.12	437.60 (0.38)	1.74	557.63 (0.64)
TX2	res4c	FP16, b=4	2.63	695.53 (0.52)	3.79	474.76 (0.41)	3.06	602.72 (0.69)
TX2	res4c	FP16, b=8	5.17	697.69 (0.52)	7.11	499.69 (0.43)	5.61	644.41 (0.74)
TX2	res4c	FP16, b=16	9.29	764.48 (0.57)	13.64	518.82 (0.45)	10.74	660.60 (0.76)
TX2	res4c	FP16, b=32	18.15	773.74 (0.58)	26.63	528.63 (0.46)	21.05	666.50 (0.76)
TX2	res4c	FP16, b=64	35.81	780.49 (0.59)	52.86	530.51 (0.46)	41.59	674.54 (0.77)
TX2	res4c	FP16, b=128	72.52	773.74 (0.58)	105.65	527.39 (0.46)	82.21	680.70 (0.78)
TX2	res4d	FP16, b=1	1.02	599.49 (0.45)	1.40	408.05 (0.36)	1.16	514.64 (0.59)
TX2	res4d	FP16, b=2	1.52	650.98 (0.49)	2.12	438.02 (0.38)	1.74	558.33 (0.64)
TX2	res4d	FP16, b=4	2.62	703.18 (0.53)	3.82	474.26 (0.41)	3.02	613.47 (0.70)
TX2	res4d	FP16, b=8	4.84	745.37 (0.56)	7.13	503.06 (0.44)	5.60	644.41 (0.74)
TX2	res4d	FP16, b=16	9.29	763.18 (0.57)	13.58	521.24 (0.45)	10.72	661.57 (0.76)
TX2	res4d	FP16, b=32	18.32	769.75 (0.58)	26.49	529.88 (0.46)	20.92	673.53 (0.77)
TX2	res4d	FP16, b=64	36.17	775.08 (0.58)	53.11	528.63 (0.46)	41.41	678.64 (0.78)
TX2	res4d	FP16, b=128	72.01	779.13 (0.58)	105.92	526.15 (0.46)	82.10	679.67 (0.78)
TX2	res4e	FP16, b=1	1.02	601.10 (0.45)	1.40	407.31 (0.35)	1.16	514.64 (0.59)
TX2	res4e	FP16, b=2	1.53	649.09 (0.49)	2.12	437.17 (0.38)	1.75	560.43 (0.64)
TX2	res4e	FP16, b=4	2.64	696.61 (0.52)	3.81	474.26 (0.41)	3.03	610.13 (0.70)
TX2	res4e	FP16, b=8	4.85	740.43 (0.56)	7.19	500.81 (0.44)	5.63	644.41 (0.74)
TX2	res4e	FP16, b=16	9.33	758.00 (0.57)	13.67	518.22 (0.45)	10.68	663.53 (0.76)
TX2	res4e	FP16, b=32	18.10	771.07 (0.58)	26.59	526.15 (0.46)	20.95	671.51 (0.77)
TX2	res4e	FP16, b=64	35.83	780.49 (0.59)	52.81	531.14 (0.46)	41.24	676.58 (0.77)
TX2	res4e	FP16, b=128	72.39	775.08 (0.58)	105.40	529.25 (0.46)	82.50	679.67 (0.78)
TX2	res4f	FP16, b=1	1.02	599.49 (0.45)	1.39	408.42 (0.36)	1.17	514.05 (0.59)
TX2	res4f	FP16, b=2	1.51	650.03 (0.49)	2.12	437.17 (0.38)	1.75	559.03 (0.64)
TX2	res4f	FP16, b=4	2.64	699.88 (0.53)	3.80	474.76 (0.41)	3.02	610.96 (0.70)
TX2	res4f	FP16, b=8	4.83	741.66 (0.56)	7.10	500.81 (0.44)	5.60	642.56 (0.74)
TX2	res4f	FP16, b=16	9.27	763.18 (0.57)	13.57	521.85 (0.45)	10.76	664.52 (0.76)
TX2	res4f	FP16, b=32	18.24	767.10 (0.58)	26.66	525.52 (0.46)	21.02	669.49 (0.77)
TX2	res4f	FP16, b=64	36.10	771.07 (0.58)	52.77	532.41 (0.46)	41.20	675.56 (0.77)
TX2	res4f	FP16, b=128	71.76	776.43 (0.58)	105.63	533.68 (0.46)	82.10	677.61 (0.78)
TX2	res5a	FP16, b=1	1.73	413.29 (0.31)	2.43	281.55 (0.25)	1.98	357.60 (0.41)
TX2	res5a	FP16, b=2	2.05	634.24 (0.48)	2.88	430.04 (0.37)	2.34	552.57 (0.63)
TX2	res5a	FP16, b=4	3.68	664.17 (0.50)	5.44	447.20 (0.39)	4.29	576.80 (0.66)
TX2	res5a	FP16, b=8	7.30	659.82 (0.49)	10.75	445.88 (0.39)	8.45	572.43 (0.65)
TX2	res5a	FP16, b=16	13.52	707.67 (0.53)	19.64	484.05 (0.42)	15.51	615.53 (0.70)
TX2	res5a	FP16, b=32	25.37	746.07 (0.56)	36.80	514.95 (0.45)	29.04	656.96 (0.75)
TX2	res5a	FP16, b=64	48.45	778.71 (0.58)	71.37	532.16 (0.46)	56.06	675.29 (0.77)
TX2	res5a	FP16, b=128	95.61	791.96 (0.59)	140.12	540.72 (0.47)	110.13	686.01 (0.78)

Table A.5: Level-2 - Inference results ResNet50 residual layers

HW	Layer	Parameters	MaxN Lat [ms]	MaxN Throughput (Eff) [GOPs] ([%])	MaxQ Lat [ms]	MaxQ Throughput (Eff) [GOPs] ([%])	MaxP Lat [ms]	MaxP Throughput (Eff) [GOPs] ([%])
TX2	res5b	FP16, b=1	1.24	456.82 (0.34)	1.71	307.16 (0.27)	1.41	391.96 (0.45)
TX2	res5b	FP16, b=2	1.60	666.50 (0.50)	2.29	447.22 (0.39)	1.84	572.63 (0.66)
TX2	res5b	FP16, b=4	2.66	704.29 (0.53)	3.85	472.75 (0.41)	3.12	607.64 (0.70)
TX2	res5b	FP16, b=8	4.98	723.66 (0.54)	7.40	481.40 (0.42)	5.79	626.36 (0.72)
TX2	res5b	FP16, b=16	8.79	810.18 (0.61)	13.01	540.78 (0.47)	10.17	704.29 (0.81)
TX2	res5b	FP16, b=32	16.33	861.70 (0.65)	24.27	577.06 (0.50)	18.76	752.90 (0.86)
TX2	res5b	FP16, b=64	31.47	892.66 (0.67)	47.00	596.29 (0.52)	36.27	775.08 (0.89)
TX2	res5b	FP16, b=128	61.55	907.14 (0.68)	92.43	606.82 (0.53)	71.35	787.36 (0.90)
TX2	res5c	FP16, b=1	1.23	460.58 (0.35)	1.71	312.74 (0.27)	1.38	398.95 (0.46)
TX2	res5c	FP16, b=2	1.59	665.51 (0.50)	2.28	448.57 (0.39)	1.84	571.89 (0.65)
TX2	res5c	FP16, b=4	2.66	705.40 (0.53)	3.85	469.77 (0.41)	3.10	610.13 (0.70)
TX2	res5c	FP16, b=8	4.99	720.16 (0.54)	7.48	480.37 (0.42)	5.80	626.36 (0.72)
TX2	res5c	FP16, b=16	8.79	811.66 (0.61)	13.08	540.78 (0.47)	10.11	703.18 (0.80)
TX2	res5c	FP16, b=32	16.42	863.36 (0.65)	24.29	576.32 (0.50)	18.80	746.62 (0.85)
TX2	res5c	FP16, b=64	31.34	892.66 (0.67)	47.02	596.29 (0.52)	36.12	777.78 (0.89)
TX2	res5c	FP16, b=128	61.58	907.14 (0.68)	92.32	603.54 (0.53)	71.16	787.36 (0.90)

Table A.6: Level-1 and 2 - Discrepancy between latency of different convolutions and residual layers

level-2				level-1					
Residual Layer	[MOP]	TX2, MaxN, FP16 b=1 [ms]	TX2, MaxN, FP16 b=128 [ms]	Conv. Layer	[MOP]	TX2, MaxN, FP16 b=1 [ms]	TX2, MaxN, FP16 b=128 [ms]	ZCU104,INT8 t=1 [ms]	ZCU104,INT8 t=8 [ms]
res2a	462.44	1.37	119.12	res2a_branch2a, 1x1	25.70	0.06	5.05	0.06	0.08
res2b	436.74	1.12	108.24	res2a_branch2b, 3x3	231.20	0.19	22.78	0.19	0.19
res2c	436.74	1.12	108.07	res2a_branch2c, 1x1	102.80	0.18	20.15	0.22	0.26
res3a	590.88	1.39	133.09	res2a_branch1, 1x1	102.80	0.21	23.66	0.43	0.46
res3b	436.74	1.03	87.72	res3a_branch2a, 1x1	51.40	0.09	7.19	0.09	0.13
res3c	436.74	1.05	87.87	res3a_branch2b, 3x3	231.20	0.21	24.63	0.21	0.21
res3d	436.74	1.04	88.66	res3a_branch2c, 1x1	102.80	0.15	15.18	0.21	0.25
res4a	590.88	1.20	104.66	res3a_branch1, 1x1	205.50	0.29	30.35	0.33	0.39
res4b	436.74	1.03	71.33	res4a_branch2a, 1x1	51.40	0.08	7.10	0.12	0.13
res4c	436.74	1.03	72.52	res4a_branch2b, 3x3	231.20	0.20	23.12	0.21	0.23
res4d	436.74	1.02	72.01	res4a_branch2c, 1x1	102.80	0.15	13.01	0.29	0.38
res4e	436.74	1.02	72.39	res4a_branch1, 1x1	205.50	0.28	29.23	0.43	0.50
res4f	436.74	1.02	71.76	res5a_branch2a, 1x1	51.40	0.14	7.61	0.12	0.19
res5a	590.88	1.73	95.61	res5a_branch2b, 3x3	231.20	0.31	24.90	0.33	0.49
res5b	436.74	1.24	61.55	res5a_branch2c, 1x1	102.80	0.27	12.53	0.47	0.60
res5c	436.74	1.23	61.58	res5a_branch1, 1x1	205.50	0.51	30.92	0.52	0.69
Min		1.02	61.55	**Min**		0.06	5.05	0.06	0.08
Max		1.73	133.09	**Max**		0.51	30.92	0.52	0.69
Var		0.04	454.94	**Var**		0.01	79.42	0.02	0.03

Table A.7: Level-3 - Inference results CNV TX2

NN_Topology	hw_quant_prun	Op mode	batch	lat-sys	lat-comp	fps-system	fps-comp	tp-system	tp-comp	top1	top5 [%]	Base_Pwr_W	Idle_Pwr_W	Full_Pwr_W
CNV	GPU_FP16_100%	maxp	1	1.78	1.304400	567.942	785.639	266.620372	368.818229	87.06	na	1.8	4.7	10.3
CNV	GPU_FP16_100%	maxp	2	2.89	2.086500	695.180	976.935	326.352251	458.622136	87.06	na	1.8	4.7	10.4
CNV	GPU_FP16_100%	maxp	4	4.68	3.468830	867.797	1153.660	407.387302	541.585687	87.06	na	1.8	4.7	10.1
CNV	GPU_FP16_100%	maxp	8	7.78	5.500290	1048.110	1454.760	492.035239	682.937082	87.06	na	1.8	4.7	10.9
CNV	GPU_FP16_100%	maxp	16	14.14	9.843110	1149.270	1666.980	539.524801	782.563761	87.06	na	1.8	4.7	10.9
CNV	GPU_FP16_100%	maxp	32	26.19	17.631200	1232.250	1831.750	578.479762	859.915037	87.06	na	1.8	4.7	10.9
CNV	GPU_FP16_100%	maxp	64	50.40	33.654900	1286.430	1909.280	603.914563	896.311496	87.06	na	1.8	4.7	10.9
CNV	GPU_FP16_100%	maxp	128	101.61	66.364800	1302.800	1939.600	611.599460	910.545220	87.06	na	1.8	4.7	10.8
CNV	GPU_FP32_100%	maxp	1	2.35	1.940500	430.252	515.510	201.981801	242.006169	87.06	na	1.8	4.7	11.1
CNV	GPU_FP32_100%	maxp	2	3.99	3.345520	501.224	598.786	235.299607	281.100088	87.06	na	1.8	4.7	12.4
CNV	GPU_FP32_100%	maxp	4	6.86	5.684520	585.812	703.573	275.009443	330.292345	87.06	na	1.8	4.7	11.8
CNV	GPU_FP32_100%	maxp	8	11.61	9.380380	691.425	851.171	324.589466	399.582226	87.06	na	1.8	4.7	11.8
CNV	GPU_FP32_100%	maxp	16	21.17	16.990300	757.396	954.766	355.559552	448.214899	87.06	na	1.8	4.7	11.7
CNV	GPU_FP32_100%	maxp	32	39.81	31.282800	809.486	1031.790	380.013203	484.373816	87.06	na	1.8	4.7	11.6
CNV	GPU_FP32_100%	maxp	64	77.63	61.085700	829.822	1057.700	389.559938	496.537265	87.06	na	1.8	4.7	12.4
CNV	GPU_FP32_100%	maxp	128	152.63	119.453000	839.344	1072.660	394.030041	503.560237	87.06	na	1.8	4.7	12.4
CNV	GPU_FP16_50%	maxp	1	1.03	0.608011	965.127	1740.870	114.776412	207.030590	85.57	na	1.8	4.7	7.8
CNV	GPU_FP16_50%	maxp	2	1.58	0.862916	1272.050	2359.870	151.276811	280.644321	85.57	na	1.8	4.7	7.7
CNV	GPU_FP16_50%	maxp	4	2.63	1.312840	1568.150	3121.110	186.490100	371.173750	85.57	na	1.8	4.7	8.4
CNV	GPU_FP16_50%	maxp	8	4.22	1.890530	1906.890	4365.680	226.774292	519.182539	85.57	na	1.8	4.7	8.4
CNV	GPU_FP16_50%	maxp	16	7.57	3.090030	2155.790	5392.770	256.374385	641.327817	85.57	na	1.8	4.7	7.6
CNV	GPU_FP16_50%	maxp	32	13.75	5.447190	2316.740	6025.160	275.515144	716.533935	85.57	na	1.8	4.7	7.5
CNV	GPU_FP16_50%	maxp	64	26.79	10.151100	2432.300	6478.430	289.257960	770.438451	85.57	na	1.8	4.7	7.5
CNV	GPU_FP16_50%	maxp	128	51.16	19.256300	2528.400	6618.150	300.686521	787.054462	85.57	na	1.8	4.7	8.4
CNV	GPU_FP32_50%	maxp	1	1.18	0.751665	853.333	1327.070	101.481463	157.819990	85.60	na	1.8	4.7	8.5
CNV	GPU_FP32_50%	maxp	2	1.91	1.201480	1058.950	1675.520	125.934184	199.258931	85.60	na	1.8	4.7	9.2
CNV	GPU_FP32_50%	maxp	4	3.25	2.028070	1245.740	2001.730	148.147930	238.053010	85.60	na	1.8	4.7	9.2
CNV	GPU_FP32_50%	maxp	8	5.42	3.060010	1508.100	2638.560	179.348735	313.787149	85.60	na	1.8	4.7	9.1
CNV	GPU_FP32_50%	maxp	16	9.52	5.215470	1678.690	3124.470	199.635919	371.573333	85.60	na	1.8	4.7	9.3
CNV	GPU_FP32_50%	maxp	32	17.46	9.299470	1818.830	3455.320	216.301877	410.919218	85.60	na	1.8	4.7	9.2
CNV	GPU_FP32_50%	maxp	64	33.53	17.423000	1903.350	3713.930	226.353303	441.674059	85.60	na	1.8	4.7	9.3
CNV	GPU_FP32_50%	maxp	128	65.09	33.716400	1943.070	3744.290	231.076949	445.284581	85.60	na	1.8	4.7	9.3
CNV	GPU_FP16_25%	maxp	1	1.00	0.571742	1025.030	1869.410	31.275441	57.038937	83.31	na	1.8	4.7	6.3
CNV	GPU_FP16_25%	maxp	2	1.32	0.627033	1542.170	3449.150	47.054278	105.239540	83.31	na	1.8	4.7	6.2
CNV	GPU_FP16_25%	maxp	4	1.89	0.709281	2133.330	5862.890	65.091593	178.886928	83.31	na	1.8	4.7	6.8
CNV	GPU_FP16_25%	maxp	8	3.24	0.931445	2534.650	9056.340	77.336562	276.324619	83.31	na	1.8	4.7	6.9
CNV	GPU_FP16_25%	maxp	16	5.67	1.401970	2813.190	12232.400	85.835299	373.231711	83.31	na	1.8	4.7	7.0
CNV	GPU_FP16_25%	maxp	32	10.51	2.365910	3084.340	14952.400	94.108555	456.223622	83.31	na	1.8	4.7	6.9
CNV	GPU_FP16_25%	maxp	64	19.95	4.128690	3230.280	16486.900	98.561438	503.043874	83.31	na	1.8	4.7	6.9
CNV	GPU_FP16_25%	maxp	128	38.90	7.459370	3335.500	17483.400	101.771882	533.448815	83.31	na	1.8	4.7	7.0
CNV	GPU_FP32_25%	maxp	1	0.92	0.513143	1109.430	2075.170	33.850631	63.317031	83.25	na	1.8	4.7	6.8
CNV	GPU_FP32_25%	maxp	2	1.23	0.563732	1635.780	3575.820	49.910481	109.104462	83.25	na	1.8	4.7	7.4
CNV	GPU_FP32_25%	maxp	4	2.13	0.901777	1917.600	4504.010	58.509297	137.425146	83.25	na	1.8	4.7	7.1
CNV	GPU_FP32_25%	maxp	8	3.48	1.271340	2327.270	6430.590	71.009039	196.208439	83.25	na	1.8	4.7	7.3
CNV	GPU_FP32_25%	maxp	16	6.33	1.979410	2572.860	8374.360	78.502415	255.516228	83.25	na	1.8	4.7	7.4
CNV	GPU_FP32_25%	maxp	32	11.56	3.308250	2852.370	9900.220	87.030749	302.072859	83.25	na	1.8	4.7	7.3
CNV	GPU_FP32_25%	maxp	64	22.18	6.260000	2925.710	10667.300	89.268479	325.477799	83.25	na	1.8	4.7	7.3
CNV	GPU_FP32_25%	maxp	128	42.80	11.409000	3038.580	11247.100	92.712339	343.168501	83.25	na	1.8	4.7	7.2
CNV	GPU_FP16_12.50%	maxp	1	0.96	0.552353	1051.330	1888.200	8.430275	15.140864	77.81	na	1.8	4.7	5.7
CNV	GPU_FP16_12.50%	maxp	2	1.29	0.593285	1590.060	3644.020	12.750176	29.220216	77.81	na	1.8	4.7	5.6
CNV	GPU_FP16_12.50%	maxp	4	1.80	0.614004	2295.960	7367.220	18.410559	59.075350	77.81	na	1.8	4.7	5.5
CNV	GPU_FP16_12.50%	maxp	8	2.90	0.654227	2860.340	12703.000	22.936140	101.861241	77.81	na	1.8	4.7	5.5
CNV	GPU_FP16_12.50%	maxp	16	5.27	0.963234	3084.340	18132.600	24.732323	145.399444	77.81	na	1.8	4.7	5.9
CNV	GPU_FP16_12.50%	maxp	32	9.60	1.466280	3390.730	23181.600	27.189165	185.885740	77.81	na	1.8	4.7	6.3
CNV	GPU_FP16_12.50%	maxp	64	18.36	2.444440	3555.560	27244.900	28.510884	218.468026	77.81	na	1.8	4.7	6.4
CNV	GPU_FP16_12.50%	maxp	128	36.44	4.405000	3631.210	29763.100	29.117496	238.660656	77.81	na	1.8	4.7	6.3
CNV	GPU_FP32_12.50%	maxp	1	0.95	0.508646	1116.680	2095.260	8.954295	16.801211	77.84	na	1.8	4.7	5.5
CNV	GPU_FP32_12.50%	maxp	2	1.20	0.526775	1706.670	4135.700	13.685234	33.162838	77.84	na	1.8	4.7	6.3
CNV	GPU_FP32_12.50%	maxp	4	1.70	0.537918	2386.950	8205.260	19.140179	65.795321	77.84	na	1.8	4.7	6.9
CNV	GPU_FP32_12.50%	maxp	8	3.01	0.781062	2737.970	10589.700	21.954894	84.915373	77.84	na	1.8	4.7	6.8
CNV	GPU_FP32_12.50%	maxp	16	5.41	1.148410	3065.870	14488.200	24.584218	116.176182	77.84	na	1.8	4.7	6.2
CNV	GPU_FP32_12.50%	maxp	32	10.26	2.177810	3210.030	16511.900	25.740191	132.403576	77.84	na	1.8	4.7	6.3
CNV	GPU_FP32_12.50%	maxp	64	19.36	3.606190	3368.420	18975.300	27.010269	152.156783	77.84	na	1.8	4.7	6.2
CNV	GPU_FP32_12.50%	maxp	128	38.41	6.471000	3471.190	20283.300	27.834348	162.645211	77.84	na	1.8	4.7	6.1

Table A.8: Level-3 - Inference results CNV NCS

NN_Topology	hw_quant_prun	Op mode	batch	lat-sys	lat-comp	fps-system	fps-comp	tp-system	tp-comp	top1	top5 [%]	Base_Pwr_W	Idle_Pwr_W	Full_Pwr_W
CNV	NCS_FP16_100%	na	1	6.60	4.97914	151.555	200.838	71.147495	94.283399	87.02	na	0.53	1.2	1.728796
CNV	NCS_FP16_100%	na	2	12.29	8.93829	162.738	223.757	76.397354	105.042724	87.02	na	0.53	1.2	1.780300
CNV	NCS_FP16_100%	na	4	23.35	16.84730	171.284	237.426	80.409274	111.459636	87.02	na	0.53	1.2	1.823024
CNV	NCS_FP16_100%	na	8	45.13	31.86580	177.259	251.053	83.214238	117.856831	87.02	na	0.53	1.2	1.860921
CNV	NCS_FP16_100%	na	16	88.10	61.77320	181.621	259.012	85.261978	121.593183	87.02	na	0.53	1.2	1.879459
CNV	NCS_FP16_100%	na	32	174.80	121.82500	183.067	262.673	85.940803	123.311840	87.02	na	0.53	1.2	1.903725
CNV	NCS_FP16_100%	na	64	348.38	242.09800	183.706	264.356	86.240782	124.101924	87.02	na	0.53	1.2	1.903671
CNV	NCS_FP16_100%	na	128	694.75	481.67500	184.240	265.739	86.491468	124.751174	87.02	na	0.53	1.2	1.911020
CNV	NCS_FP16_50%	na	1	3.97	2.38549	251.922	419.201	29.959480	49.852907	85.55	na	0.53	1.2	1.655401
CNV	NCS_FP16_50%	na	2	7.03	3.93902	284.481	507.740	33.831515	60.382287	85.55	na	0.53	1.2	1.676720
CNV	NCS_FP16_50%	na	4	12.86	6.68938	310.979	597.963	36.982753	71.111934	85.55	na	0.53	1.2	1.710000
CNV	NCS_FP16_50%	na	8	24.89	12.41530	321.433	644.367	38.225981	76.630467	85.55	na	0.53	1.2	1.730000
CNV	NCS_FP16_50%	na	16	47.84	23.13690	334.428	691.537	39.771394	82.240094	85.55	na	0.53	1.2	1.750000
CNV	NCS_FP16_50%	na	32	94.62	44.70660	338.178	715.779	40.217357	85.123041	85.55	na	0.53	1.2	1.770000
CNV	NCS_FP16_50%	na	64	187.07	87.24440	342.120	733.571	40.686154	87.238931	85.55	na	0.53	1.2	1.780000
CNV	NCS_FP16_50%	na	128	370.83	173.15800	345.169	739.209	41.048753	87.909422	85.55	na	0.53	1.2	1.800000
CNV	NCS_FP16_25%	na	1	3.50	1.95433	285.517	511.684	8.711618	15.612365	83.28	na	0.53	1.2	1.590000
CNV	NCS_FP16_25%	na	2	5.80	2.80634	344.874	712.672	10.522703	21.744857	83.28	na	0.53	1.2	1.620000
CNV	NCS_FP16_25%	na	4	10.73	4.62598	372.730	864.682	11.372638	26.382945	83.28	na	0.53	1.2	1.620000
CNV	NCS_FP16_25%	na	8	20.31	7.97099	393.913	1003.640	12.018968	30.622795	83.28	na	0.53	1.2	1.630000
CNV	NCS_FP16_25%	na	16	39.23	14.64030	407.873	1092.870	12.444912	33.345357	83.28	na	0.53	1.2	1.640000
CNV	NCS_FP16_25%	na	32	76.52	27.92100	418.169	1146.090	12.759060	34.969191	83.28	na	0.53	1.2	1.650000
CNV	NCS_FP16_25%	na	64	151.57	53.29490	422.250	1200.870	12.883579	36.640624	83.28	na	0.53	1.2	1.650000
CNV	NCS_FP16_25%	na	128	299.80	105.63000	426.946	1211.780	13.026862	36.973507	83.28	na	0.53	1.2	1.670000
CNV	NCS_FP16_12.50%	na	1	3.40	1.90004	293.812	526.305	2.355983	4.220269	77.82	na	0.53	1.2	1.560000
CNV	NCS_FP16_12.50%	na	2	5.61	2.55363	356.569	783.197	2.859211	6.280203	77.82	na	0.53	1.2	1.580000
CNV	NCS_FP16_12.50%	na	4	10.09	3.94931	396.432	1012.840	3.178860	8.121636	77.82	na	0.53	1.2	1.590000
CNV	NCS_FP16_12.50%	na	8	19.18	6.92008	417.172	1156.060	3.345167	9.270071	77.82	na	0.53	1.2	1.590000
CNV	NCS_FP16_12.50%	na	16	37.01	12.60460	432.293	1269.380	3.466418	10.178747	77.82	na	0.53	1.2	1.580000
CNV	NCS_FP16_12.50%	na	32	72.74	23.68960	439.912	1350.800	3.527512	10.831628	77.82	na	0.53	1.2	1.580000
CNV	NCS_FP16_12.50%	na	64	143.09	45.67720	447.278	1401.140	3.586577	11.235288	77.82	na	0.53	1.2	1.590000
CNV	NCS_FP16_12.50%	na	128	284.73	89.22510	449.542	1434.570	3.604732	11.503352	77.82	na	0.53	1.2	1.580000

Table A.9: Level-3 - Inference results CNV U96 A53

NN_Topology	hw_quant_prun	Op mode	batch	lat-sys	lat-comp	fps-system	fps-comp	tp-system	tp-comp	top1	top5 [%]	Base_Pwr_W	Idle_Pwr_W	Full_Pwr_W
CNV	U96-Quadcore A53_INT2_50%	na	2	na	34.132	na	58.596039	na	6.968454	84.29	na	9.2	na	14.5
CNV	U96-Quadcore A53_INT2_50%	na	4	na	52.668	na	75.947444	na	9.031946	84.29	na	9.2	na	14.5
CNV	U96-Quadcore A53_INT2_50%	na	8	na	89.495	na	89.390469	na	10.630640	84.29	na	9.2	na	14.5
CNV	U96-Quadcore A53_INT2_50%	na	16	na	164.939	na	97.005560	na	11.536254	84.29	na	9.2	na	14.5
CNV	U96-Quadcore A53_INT2_50%	na	32	na	320.240	na	99.925056	na	11.883451	84.29	na	9.2	na	14.5
CNV	U96-Quadcore A53_INT2_50%	na	64	na	620.131	na	103.204000	na	12.273395	84.29	na	9.2	na	14.5
CNV	U96-Quadcore A53_INT2_50%	na	128	na	1227.069	na	104.313612	na	12.405354	84.29	na	9.2	na	14.5
CNV	U96-Quadcore A53_INT4_25%	na	2	na	8.550	na	233.918129	na	7.137247	81.09	na	9.2	na	14.5
CNV	U96-Quadcore A53_INT4_25%	na	4	na	14.137	na	282.945462	na	8.633156	81.09	na	9.2	na	14.5
CNV	U96-Quadcore A53_INT4_25%	na	8	na	25.200	na	317.460317	na	9.686264	81.09	na	9.2	na	14.5
CNV	U96-Quadcore A53_INT4_25%	na	16	na	46.580	na	343.495062	na	10.480629	81.09	na	9.2	na	14.5
CNV	U96-Quadcore A53_INT4_25%	na	32	na	90.412	na	353.935318	na	10.799180	81.09	na	9.2	na	14.5
CNV	U96-Quadcore A53_INT4_25%	na	64	na	178.848	na	357.845769	na	10.918494	81.09	na	9.2	na	14.5
CNV	U96-Quadcore A53_INT4_25%	na	128	na	353.671	na	361.918280	na	11.042754	81.09	na	9.2	na	14.5
CNV	U96-Quadcore A53_INT2_25%	na	2	na	8.433	na	237.163524	na	7.236270	79.89	na	9.2	na	14.5
CNV	U96-Quadcore A53_INT2_25%	na	4	na	14.103	na	283.627597	na	8.653969	79.89	na	9.2	na	14.5
CNV	U96-Quadcore A53_INT2_25%	na	8	na	24.808	na	322.476620	na	9.839320	79.89	na	9.2	na	14.5
CNV	U96-Quadcore A53_INT2_25%	na	16	na	46.974	na	340.613957	na	10.392722	79.89	na	9.2	na	14.5
CNV	U96-Quadcore A53_INT2_25%	na	32	na	90.564	na	353.341283	na	10.781055	79.89	na	9.2	na	14.5
CNV	U96-Quadcore A53_INT2_25%	na	64	na	178.144	na	359.259925	na	10.961643	79.89	na	9.2	na	14.5
CNV	U96-Quadcore A53_INT2_25%	na	128	na	351.970	na	363.667358	na	11.096121	79.89	na	9.2	na	14.5
CNV	U96-Quadcore A53_INT4_12.50%	na	2	na	3.625	na	551.724138	na	4.424097	75.85	na	9.2	na	14.5
CNV	U96-Quadcore A53_INT4_12.50%	na	4	na	6.410	na	624.024961	na	5.003854	75.85	na	9.2	na	14.5
CNV	U96-Quadcore A53_INT4_12.50%	na	8	na	11.712	na	683.060109	na	5.477238	75.85	na	9.2	na	14.5
CNV	U96-Quadcore A53_INT4_12.50%	na	16	na	22.370	na	715.243630	na	5.735307	75.85	na	9.2	na	14.5
CNV	U96-Quadcore A53_INT4_12.50%	na	32	na	43.397	na	737.378160	na	5.912797	75.85	na	9.2	na	14.5
CNV	U96-Quadcore A53_INT4_12.50%	na	64	na	85.380	na	749.590068	na	6.010720	75.85	na	9.2	na	14.5
CNV	U96-Quadcore A53_INT4_12.50%	na	128	na	168.071	na	761.582903	na	6.106887	75.85	na	9.2	na	14.5
CNV	U96-Quadcore A53_INT2_12.50%	na	2	na	3.651	na	547.795125	na	4.392592	73.64	na	9.2	na	14.5
CNV	U96-Quadcore A53_INT2_12.50%	na	4	na	6.306	na	634.316524	na	5.086379	73.64	na	9.2	na	14.5
CNV	U96-Quadcore A53_INT2_12.50%	na	8	na	11.731	na	681.953798	na	5.468367	73.64	na	9.2	na	14.5
CNV	U96-Quadcore A53_INT2_12.50%	na	16	na	22.210	na	720.396218	na	5.776624	73.64	na	9.2	na	14.5
CNV	U96-Quadcore A53_INT2_12.50%	na	32	na	43.493	na	735.750581	na	5.899746	73.64	na	9.2	na	14.5
CNV	U96-Quadcore A53_INT2_12.50%	na	64	na	85.127	na	751.817872	na	6.028584	73.64	na	9.2	na	14.5
CNV	U96-Quadcore A53_INT2_12.50%	na	128	na	167.092	na	766.045053	na	6.142667	73.64	na	9.2	na	14.5

Table A.10: Level-3 - Inference results CNV ZCU104 BISMO

NN_Topology	hw_quant_prun	Op mode	batch	lat-sys	lat-comp	fps-system	fps-comp	tp-system	tp-comp	top1	top5 [%]	Base_Pwr_W	Idle_Pwr_W	Full_Pwr_W
CNV	ZCU104-BISMO_INT2_50%	na	2	na	9.114334	na	219.434577	na	26.095958	84.29	na	9.2	na	15.000
CNV	ZCU104-BISMO_INT2_50%	na	4	na	16.680774	na	239.797026	na	28.517534	84.29	na	9.2	na	15.150
CNV	ZCU104-BISMO_INT2_50%	na	8	na	31.695208	na	252.404088	na	30.016812	84.29	na	9.2	na	15.250
CNV	ZCU104-BISMO_INT2_50%	na	16	na	63.422265	na	252.277334	na	30.001738	84.29	na	9.2	na	15.475
CNV	ZCU104-BISMO_INT2_50%	na	32	na	121.255614	na	263.905307	na	31.384579	84.29	na	9.2	na	15.475
CNV	ZCU104-BISMO_INT2_50%	na	64	na	247.773838	na	258.300071	na	30.717984	84.29	na	9.2	na	15.600
CNV	ZCU104-BISMO_INT2_50%	na	128	na	498.197604	na	256.926165	na	30.554594	84.29	na	9.2	na	15.750
CNV	ZCU104-BISMO_INT4_25%	na	2	na	6.862861	na	291.423671	na	8.891841	81.09	na	9.2	na	14.975
CNV	ZCU104-BISMO_INT4_25%	na	4	na	12.062617	na	331.603010	na	10.117782	81.09	na	9.2	na	15.025
CNV	ZCU104-BISMO_INT4_25%	na	8	na	25.337448	na	315.738192	na	9.633719	81.09	na	9.2	na	15.225
CNV	ZCU104-BISMO_INT4_25%	na	16	na	44.035643	na	363.342036	na	11.086195	81.09	na	9.2	na	15.400
CNV	ZCU104-BISMO_INT4_25%	na	32	na	88.454574	na	361.767614	na	11.038156	81.09	na	9.2	na	15.500
CNV	ZCU104-BISMO_INT4_25%	na	64	na	188.719159	na	339.128260	na	10.347391	81.09	na	9.2	na	15.475
CNV	ZCU104-BISMO_INT4_25%	na	128	na	347.028579	na	368.845703	na	11.254121	81.09	na	9.2	na	15.650
CNV	ZCU104-BISMO_INT2_25%	na	2	na	4.206040	na	475.506641	na	14.508531	79.89	na	9.2	na	14.825
CNV	ZCU104-BISMO_INT2_25%	na	4	na	7.654349	na	522.578719	na	15.944782	79.89	na	9.2	na	14.975
CNV	ZCU104-BISMO_INT2_25%	na	8	na	15.170309	na	527.345897	na	16.090237	79.89	na	9.2	na	15.000
CNV	ZCU104-BISMO_INT2_25%	na	16	na	27.868153	na	574.132053	na	17.517763	79.89	na	9.2	na	15.275
CNV	ZCU104-BISMO_INT2_25%	na	32	na	55.855311	na	572.908811	na	17.480440	79.89	na	9.2	na	15.475
CNV	ZCU104-BISMO_INT2_25%	na	64	na	114.312915	na	559.866750	na	17.082504	79.89	na	9.2	na	15.475
CNV	ZCU104-BISMO_INT2_25%	na	128	na	220.039205	na	581.714518	na	17.749117	79.89	na	9.2	na	15.500
CNV	ZCU104-BISMO_INT4_12.50%	na	2	na	4.180616	na	478.398456	na	3.836122	75.85	na	9.2	na	14.850
CNV	ZCU104-BISMO_INT4_12.50%	na	4	na	7.281409	na	549.344265	na	4.405014	75.85	na	9.2	na	14.825
CNV	ZCU104-BISMO_INT4_12.50%	na	8	na	14.385140	na	556.129466	na	4.459422	75.85	na	9.2	na	14.950
CNV	ZCU104-BISMO_INT4_12.50%	na	16	na	31.355124	na	510.283428	na	4.091797	75.85	na	9.2	na	15.275
CNV	ZCU104-BISMO_INT4_12.50%	na	32	na	56.983773	na	561.563375	na	4.502995	75.85	na	9.2	na	15.475
CNV	ZCU104-BISMO_INT4_12.50%	na	64	na	119.823216	na	534.120199	na	4.282937	75.85	na	9.2	na	15.500
CNV	ZCU104-BISMO_INT4_12.50%	na	128	na	211.559943	na	605.029469	na	4.851535	75.85	na	9.2	na	15.525
CNV	ZCU104-BISMO_INT2_12.50%	na	2	na	2.664596	na	750.582799	na	6.018680	73.64	na	9.2	na	14.850
CNV	ZCU104-BISMO_INT2_12.50%	na	4	na	4.831999	na	827.814775	na	6.637978	73.64	na	9.2	na	14.775
CNV	ZCU104-BISMO_INT2_12.50%	na	8	na	9.481912	na	843.711735	na	6.765451	73.64	na	9.2	na	14.875
CNV	ZCU104-BISMO_INT2_12.50%	na	16	na	19.575715	na	817.339260	na	6.553979	73.64	na	9.2	na	15.125
CNV	ZCU104-BISMO_INT2_12.50%	na	32	na	37.490798	na	853.542770	na	6.844283	73.64	na	9.2	na	15.475
CNV	ZCU104-BISMO_INT2_12.50%	na	64	na	76.311300	na	838.670023	na	6.725023	73.64	na	9.2	na	15.475
CNV	ZCU104-BISMO_INT2_12.50%	na	128	na	144.856425	na	883.633570	na	7.085571	73.64	na	9.2	na	15.500

Table A.11: Level-3 - Inference results CNV ZCU104 FINN

NN_Topology	hw_quant_prun	Op mode	batch	lat-sys	lat-comp	fps-system	fps-comp	tp-system	tp-comp	top1	top5 [%]	Base_Pwr_W	Idle_Pwr_W	Full_Pwr_W
CNV	ZCU104-FINN_INT4_100%	na	1	na	4.874	na	205.170	na	96.317057	87.42	na	9.2	11.8	15.0250
CNV	ZCU104-FINN_INT4_100%	na	2	na	6.872	na	291.078	na	136.646567	87.42	na	9.2	11.8	15.4000
CNV	ZCU104-FINN_INT4_100%	na	4	na	9.884	na	404.694	na	189.983598	87.42	na	9.2	11.8	15.8750
CNV	ZCU104-FINN_INT4_100%	na	8	na	15.704	na	509.359	na	239.118583	87.42	na	9.2	11.8	16.7000
CNV	ZCU104-FINN_INT4_100%	na	16	na	27.344	na	584.988	na	274.622617	87.42	na	9.2	11.8	17.5750
CNV	ZCU104-FINN_INT4_100%	na	32	na	50.656	na	631.874	na	296.633249	87.42	na	9.2	11.8	19.2500
CNV	ZCU104-FINN_INT4_100%	na	64	na	97.216	na	658.267	na	309.023443	87.42	na	9.2	11.8	19.3000
CNV	ZCU104-FINN_INT4_100%	na	128	na	190.336	na	672.301	na	315.611704	87.42	na	9.2	11.8	19.2750
CNV	ZCU104-FINN_INT4_100%	na	256	na	376.832	na	679.548	na	319.013809	87.42	na	9.2	11.8	19.3500
CNV	ZCU104-FINN_INT4_100%	na	512	na	749.568	na	683.229	na	320.741854	87.42	na	9.2	11.8	19.3750
CNV	ZCU104-FINN_INT4_100%	na	10000	na	14561.129	na	686.760	na	322.399482	87.42	na	9.2	11.8	19.7625
CNV	ZCU104-FINN_INT2_100%	na	1	na	1.299	na	769.823	na	361.393407	86.86	na	9.2	11.6	14.3000
CNV	ZCU104-FINN_INT2_100%	na	2	na	1.784	na	1121.076	na	526.289128	86.86	na	9.2	11.6	14.4250
CNV	ZCU104-FINN_INT2_100%	na	4	na	2.600	na	1539.053	na	722.508431	86.86	na	9.2	11.6	14.4750
CNV	ZCU104-FINN_INT2_100%	na	8	na	3.928	na	2034.588	na	955.137337	86.86	na	9.2	11.6	14.3500
CNV	ZCU104-FINN_INT2_100%	na	16	na	6.592	na	2425.345	na	1138.578210	86.86	na	9.2	11.6	14.8250
CNV	ZCU104-FINN_INT2_100%	na	32	na	11.936	na	2682.763	na	1259.423090	86.86	na	9.2	11.6	14.9500
CNV	ZCU104-FINN_INT2_100%	na	64	na	22.592	na	2832.987	na	1329.945747	86.86	na	9.2	11.6	15.9250
CNV	ZCU104-FINN_INT2_100%	na	128	na	43.904	na	2914.655	na	1368.284790	86.86	na	9.2	11.6	17.1750
CNV	ZCU104-FINN_INT2_100%	na	256	na	86.528	na	2957.315	na	1388.311527	86.86	na	9.2	11.6	17.1750
CNV	ZCU104-FINN_INT2_100%	na	512	na	172.032	na	2979.082	na	1398.530045	86.86	na	9.2	11.6	17.1750
CNV	ZCU104-FINN_INT4_50%	na	1	na	0.507	na	1972.387	na	234.563434	84.88	na	9.2	12.0	14.6750
CNV	ZCU104-FINN_INT4_50%	na	2	na	0.680	na	2941.176	na	349.775344	84.88	na	9.2	12.0	14.4750
CNV	ZCU104-FINN_INT4_50%	na	4	na	0.972	na	4115.226	na	489.397639	84.88	na	9.2	12.0	14.7500
CNV	ZCU104-FINN_INT4_50%	na	8	na	1.456	na	5486.968	na	652.530185	84.88	na	9.2	12.0	15.0000
CNV	ZCU104-FINN_INT4_50%	na	16	na	2.432	na	6576.243	na	782.070729	84.88	na	9.2	12.0	14.9750
CNV	ZCU104-FINN_INT4_50%	na	32	na	4.384	na	7304.268	na	868.650109	84.88	na	9.2	12.0	14.9750
CNV	ZCU104-FINN_INT4_50%	na	64	na	8.250	na	7732.270	na	919.549663	84.88	na	9.2	12.0	15.8000
CNV	ZCU104-FINN_INT4_50%	na	128	na	16.128	na	7965.152	na	947.244837	84.88	na	9.2	12.0	17.1250
CNV	ZCU104-FINN_INT4_50%	na	256	na	31.744	na	8086.935	na	961.727714	84.88	na	9.2	12.0	20.2750
CNV	ZCU104-FINN_INT4_50%	na	512	na	62.976	na	8149.233	na	969.136419	84.88	na	9.2	12.0	21.1750
CNV	ZCU104-FINN_INT4_50%	na	10000	na	1218.125	na	8209.332	na	976.283611	84.88	na	9.2	12.0	21.3575
CNV	ZCU104-FINN_INT2_50%	na	1	na	0.270	na	3703.704	na	440.457946	84.29	na	9.2	12.0	13.7500
CNV	ZCU104-FINN_INT2_50%	na	2	na	0.366	na	5449.591	na	648.085176	84.29	na	9.2	12.0	13.7250
CNV	ZCU104-FINN_INT2_50%	na	4	na	0.528	na	7575.758	na	900.936687	84.29	na	9.2	12.0	13.4750
CNV	ZCU104-FINN_INT2_50%	na	8	na	0.776	na	10309.278	na	1226.016824	84.29	na	9.2	12.0	13.4750
CNV	ZCU104-FINN_INT2_50%	na	16	na	1.248	na	12892.828	na	1533.261984	84.29	na	9.2	12.0	13.6750
CNV	ZCU104-FINN_INT2_50%	na	32	na	2.176	na	14739.751	na	1752.904783	84.29	na	9.2	12.0	13.5000
CNV	ZCU104-FINN_INT2_50%	na	64	na	4.032	na	15880.893	na	1888.613538	84.29	na	9.2	12.0	13.7750
CNV	ZCU104-FINN_INT2_50%	na	128	na	7.808	na	16516.129	na	1964.158113	84.29	na	9.2	12.0	14.1000
CNV	ZCU104-FINN_INT2_50%	na	256	na	15.104	na	16855.412	na	2004.506881	84.29	na	9.2	12.0	14.8750
CNV	ZCU104-FINN_INT2_50%	na	512	na	30.208	na	17029.769	na	2025.242050	84.29	na	9.2	12.0	15.7750
CNV	ZCU104-FINN_INT4_25%	na	1	na	0.161	na	6211.180	na	189.513860	81.09	na	9.2	11.5	13.8500
CNV	ZCU104-FINN_INT4_25%	na	2	na	0.244	na	8163.265	na	249.075354	81.09	na	9.2	11.5	13.8500
CNV	ZCU104-FINN_INT4_25%	na	4	na	0.396	na	10075.567	na	307.423000	81.09	na	9.2	11.5	14.2000
CNV	ZCU104-FINN_INT4_25%	na	8	na	0.664	na	12121.212	na	369.839172	81.09	na	9.2	11.5	13.8000
CNV	ZCU104-FINN_INT4_25%	na	16	na	1.184	na	13547.841	na	413.368094	81.09	na	9.2	11.5	13.8750
CNV	ZCU104-FINN_INT4_25%	na	32	na	2.240	na	14388.489	na	439.017720	81.09	na	9.2	11.5	13.9250
CNV	ZCU104-FINN_INT4_25%	na	64	na	4.288	na	14852.634	na	453.179588	81.09	na	9.2	11.5	13.9000
CNV	ZCU104-FINN_INT4_25%	na	128	na	8.448	na	15094.340	na	460.554457	81.09	na	9.2	11.5	14.2500
CNV	ZCU104-FINN_INT4_25%	na	256	na	16.896	na	15219.071	na	464.360216	81.09	na	9.2	11.5	14.9500
CNV	ZCU104-FINN_INT4_25%	na	512	na	33.280	na	15282.214	na	466.286818	81.09	na	9.2	11.5	15.4000
CNV	ZCU104-FINN_INT4_25%	na	10000	na	651.779	na	15342.620	na	468.129910	81.09	na	9.2	11.5	16.6250
CNV	ZCU104-FINN_INT2_25%	na	1	na	0.119	na	8403.361	na	256.401099	79.89	na	9.2	11.5	13.4750
CNV	ZCU104-FINN_INT2_25%	na	2	na	0.164	na	12195.122	na	372.094294	79.89	na	9.2	11.5	13.3750
CNV	ZCU104-FINN_INT2_25%	na	4	na	0.252	na	15936.255	na	486.242742	79.89	na	9.2	11.5	13.0750
CNV	ZCU104-FINN_INT2_25%	na	8	na	0.384	na	20942.408	na	638.989140	79.89	na	9.2	11.5	13.0750
CNV	ZCU104-FINN_INT2_25%	na	16	na	0.640	na	24844.720	na	758.055438	79.89	na	9.2	11.5	13.0250
CNV	ZCU104-FINN_INT2_25%	na	32	na	1.184	na	27373.824	na	835.222782	79.89	na	9.2	11.5	12.9500
CNV	ZCU104-FINN_INT2_25%	na	64	na	2.240	na	28854.824	na	880.410657	79.89	na	9.2	11.5	13.0750
CNV	ZCU104-FINN_INT2_25%	na	128	na	4.352	na	29657.090	na	904.889182	79.89	na	9.2	11.5	13.2250
CNV	ZCU104-FINN_INT2_25%	na	256	na	8.448	na	30075.188	na	917.646076	79.89	na	9.2	11.5	13.3250
CNV	ZCU104-FINN_INT2_25%	na	512	na	16.896	na	30286.897	na	924.105684	79.89	na	9.2	11.5	13.8250
CNV	ZCU104-FINN_INT4_12.50%	na	1	0.07	0.072	na	13888.889	na	111.370501	75.85	na	9.2	11.0	13.7000
CNV	ZCU104-FINN_INT4_12.50%	na	2	na	0.120	na	16666.667	na	133.644603	75.85	na	9.2	11.0	13.3500
CNV	ZCU104-FINN_INT4_12.50%	na	4	na	0.208	na	19138.756	na	153.467483	75.85	na	9.2	11.0	13.3000
CNV	ZCU104-FINN_INT4_12.50%	na	8	na	0.368	na	21739.130	na	174.319040	75.85	na	9.2	11.0	13.5500
CNV	ZCU104-FINN_INT4_12.50%	na	16	na	0.688	na	23357.664	na	187.297540	75.85	na	9.2	11.0	13.2250
CNV	ZCU104-FINN_INT4_12.50%	na	32	na	1.312	na	24242.424	na	194.392144	75.85	na	9.2	11.0	13.1500
CNV	ZCU104-FINN_INT4_12.50%	na	64	na	2.560	na	24719.969	na	198.221422	75.85	na	9.2	11.0	13.1750
CNV	ZCU104-FINN_INT4_12.50%	na	128	na	5.120	na	24960.998	na	200.154156	75.85	na	9.2	11.0	13.2750
CNV	ZCU104-FINN_INT4_12.50%	na	256	na	10.240	na	25085.742	na	201.154437	75.85	na	9.2	11.0	13.4000
CNV	ZCU104-FINN_INT4_12.50%	na	512	na	20.480	na	25146.113	na	201.638533	75.85	na	9.2	11.0	13.5000
CNV	ZCU104-FINN_INT4_12.50%	na	10000	na	396.730	na	25206.016	na	202.118876	75.85	na	9.2	11.0	14.3475
CNV	ZCU104-FINN_INT2_12.50%	na	1	0.05	0.086	na	23255.814	na	186.480838	73.64	na	9.2	11.0	13.2250
CNV	ZCU104-FINN_INT2_12.50%	na	2	na	0.126	na	31746.032	na	254.561145	73.64	na	9.2	11.0	12.8750
CNV	ZCU104-FINN_INT2_12.50%	na	4	na	0.200	na	40201.006	na	322.358842	73.64	na	9.2	11.0	13.1500
CNV	ZCU104-FINN_INT2_12.50%	na	8	na	0.328	na	48484.848	na	388.784287	73.64	na	9.2	11.0	12.7750
CNV	ZCU104-FINN_INT2_12.50%	na	16	na	0.592	na	53962.900	na	432.711011	73.64	na	9.2	11.0	12.7500
CNV	ZCU104-FINN_INT2_12.50%	na	32	na	1.120	na	57296.330	na	459.440706	73.64	na	9.2	11.0	12.6750
CNV	ZCU104-FINN_INT2_12.50%	na	64	na	2.176	na	59095.106	na	473.864508	73.64	na	9.2	11.0	12.6250
CNV	ZCU104-FINN_INT2_12.50%	na	128	na	4.224	na	60037.524	na	481.421453	73.64	na	9.2	11.0	12.6750
CNV	ZCU104-FINN_INT2_12.50%	na	256	na	8.448	na	60512.942	na	485.233676	73.64	na	9.2	11.0	12.7250
CNV	ZCU104-FINN_INT2_12.50%	na	512	na	16.896	na	60757.090	na	487.191419	73.64	na	9.2	11.0	12.6750

Table A.12: Level-3 - Inference results MLP TX2

NN_Topology	hw_quant_prun	Op mode	batch	lat-sys	lat-comp	fps-system	fps-comp	tp-system	tp-comp	top1	top5 [%]	Base_Pwr_W	Idle_Pwr_W	Full_Pwr_W
MLP	GPU_FP16_100%	maxp	1	1.03	0.698266	976.168	1466.41	19.551669	29.370726	97.30	na	1.8	4.7	9.881
MLP	GPU_FP16_100%	maxp	2	1.25	0.739980	1635.780	2785.55	32.763038	55.791781	97.30	na	1.8	4.7	9.903
MLP	GPU_FP16_100%	maxp	4	1.61	0.766234	2515.970	5392.60	50.392363	108.008385	97.30	na	1.8	4.7	10.262
MLP	GPU_FP16_100%	maxp	8	2.65	1.161670	3093.660	7027.46	61.962916	140.752996	97.30	na	1.8	4.7	9.645
MLP	GPU_FP16_100%	maxp	16	4.05	1.248630	4112.450	13008.00	82.368261	260.537232	97.30	na	1.8	4.7	11.359
MLP	GPU_FP16_100%	maxp	32	6.61	1.201560	5044.330	29751.90	101.032886	595.900805	97.30	na	1.8	4.7	10.959
MLP	GPU_FP16_100%	maxp	64	11.72	1.370560	5688.890	56369.00	113.942778	1129.014701	97.30	na	1.8	4.7	11.215
MLP	GPU_FP16_100%	maxp	128	23.07	2.305000	5885.060	61409.30	117.871867	1229.966870	97.30	na	1.8	4.7	12.142
MLP	GPU_FP32_100%	maxp	1	1.14	0.819372	877.464	1231.27	17.574726	24.661107	97.31	na	1.8	4.7	10.895
MLP	GPU_FP32_100%	maxp	2	1.38	0.894066	1376.340	1991.41	27.566714	39.885951	97.31	na	1.8	4.7	10.915
MLP	GPU_FP32_100%	maxp	4	1.85	0.958605	2115.700	4023.29	42.375355	80.582475	97.31	na	1.8	4.7	11.193
MLP	GPU_FP32_100%	maxp	8	2.70	1.204900	3002.930	6766.27	60.145685	135.521622	97.31	na	1.8	4.7	11.915
MLP	GPU_FP32_100%	maxp	16	4.24	1.252610	4129.030	13126.40	82.700342	262.908666	97.31	na	1.8	4.7	12.021
MLP	GPU_FP32_100%	maxp	32	6.90	1.519840	4697.250	24259.70	94.081220	485.897531	97.31	na	1.8	4.7	12.271
MLP	GPU_FP32_100%	maxp	64	13.30	2.768870	4899.520	25678.30	98.132486	514.310671	97.31	na	1.8	4.7	12.950
MLP	GPU_FP32_100%	maxp	128	24.10	3.442500	5475.940	39802.50	109.677602	797.204273	97.31	na	1.8	4.7	12.366
MLP	GPU_FP16_50%	maxp	1	0.67	0.335562	1526.080	3009.65	8.881786	17.516163	97.46	na	1.8	4.7	8.718
MLP	GPU_FP16_50%	maxp	2	0.86	0.356855	2386.950	5676.31	13.892049	33.036124	97.46	na	1.8	4.7	8.650
MLP	GPU_FP16_50%	maxp	4	1.20	0.354211	3482.990	11532.10	20.271002	67.116822	97.46	na	1.8	4.7	8.868
MLP	GPU_FP16_50%	maxp	8	2.01	0.509430	4179.590	15901.40	24.325214	92.546148	97.46	na	1.8	4.7	8.795
MLP	GPU_FP16_50%	maxp	16	3.34	0.527234	5044.330	30770.20	29.358001	179.082564	97.46	na	1.8	4.7	9.686
MLP	GPU_FP16_50%	maxp	32	6.21	0.628219	5505.380	62458.10	32.041312	363.506142	97.46	na	1.8	4.7	9.003
MLP	GPU_FP16_50%	maxp	64	11.48	0.898875	5851.430	94587.10	34.055323	550.496922	97.46	na	1.8	4.7	8.679
MLP	GPU_FP16_50%	maxp	128	21.56	1.126880	6168.670	141671.00	35.901659	824.525220	97.46	na	1.8	4.7	9.883
MLP	GPU_FP32_50%	maxp	1	0.65	0.334122	1560.980	2952.86	9.084904	17.185645	97.46	na	1.8	4.7	9.321
MLP	GPU_FP32_50%	maxp	2	0.86	0.377549	2398.130	5282.05	13.957117	30.741531	97.46	na	1.8	4.7	9.532
MLP	GPU_FP32_50%	maxp	4	1.17	0.380160	3482.990	10581.10	20.271002	61.582002	97.46	na	1.8	4.7	9.563
MLP	GPU_FP32_50%	maxp	8	2.26	0.660625	4047.430	15793.70	23.556043	91.919334	97.46	na	1.8	4.7	10.240
MLP	GPU_FP32_50%	maxp	16	3.51	0.749047	4807.510	30638.50	27.979708	178.316070	97.46	na	1.8	4.7	10.356
MLP	GPU_FP32_50%	maxp	32	6.10	0.790344	5446.810	57302.70	31.700434	333.501714	97.46	na	1.8	4.7	10.371
MLP	GPU_FP32_50%	maxp	64	11.67	1.198060	5720.670	66823.30	33.294299	388.911606	97.46	na	1.8	4.7	11.478
MLP	GPU_FP32_50%	maxp	128	21.95	1.454250	6095.240	103759.00	35.474297	603.877380	97.46	na	1.8	4.7	11.120
MLP	GPU_FP16_25%	maxp	1	0.59	0.278191	1750.430	3848.74	3.259301	7.166354	97.44	na	1.8	4.7	7.334
MLP	GPU_FP16_25%	maxp	2	0.73	0.259730	2820.940	8504.14	5.252590	15.834709	97.44	na	1.8	4.7	7.364
MLP	GPU_FP16_25%	maxp	4	1.09	0.287484	3792.590	15125.30	7.061803	28.163309	97.44	na	1.8	4.7	7.284
MLP	GPU_FP16_25%	maxp	8	1.79	0.307859	4697.250	29380.50	8.746279	54.706491	97.44	na	1.8	4.7	8.149
MLP	GPU_FP16_25%	maxp	16	3.31	0.363625	5171.720	56844.70	9.629743	105.844831	97.44	na	1.8	4.7	8.354
MLP	GPU_FP16_25%	maxp	32	6.15	0.526344	5919.080	115863.00	11.021327	215.736906	97.44	na	1.8	4.7	8.161
MLP	GPU_FP16_25%	maxp	64	11.14	0.661312	5988.300	145166.00	11.150215	270.299092	97.44	na	1.8	4.7	8.740
MLP	GPU_FP16_25%	maxp	128	21.25	0.800375	6243.900	220832.00	11.626142	411.189184	97.44	na	1.8	4.7	8.066
MLP	GPU_FP32_25%	maxp	1	0.51	0.222530	1980.660	4836.44	3.687989	9.005451	97.44	na	1.8	4.7	8.068
MLP	GPU_FP32_25%	maxp	2	0.81	0.227563	2968.120	9610.87	5.526639	17.895440	97.44	na	1.8	4.7	7.972
MLP	GPU_FP32_25%	maxp	4	1.09	0.235293	4063.490	18903.80	7.566218	35.198876	97.44	na	1.8	4.7	8.158
MLP	GPU_FP32_25%	maxp	8	1.98	0.282313	4923.080	29516.00	9.166775	54.958792	97.44	na	1.8	4.7	8.505
MLP	GPU_FP32_25%	maxp	16	3.26	0.514938	5120.000	56496.60	9.533440	105.196669	97.44	na	1.8	4.7	8.612
MLP	GPU_FP32_25%	maxp	32	5.95	0.532937	5688.890	105938.00	10.592713	197.256556	97.44	na	1.8	4.7	8.547
MLP	GPU_FP32_25%	maxp	64	11.18	0.692438	5953.490	134489.00	11.085398	250.418518	97.44	na	1.8	4.7	8.318
MLP	GPU_FP32_25%	maxp	128	21.70	0.872125	6168.670	193866.00	11.486064	360.978492	97.44	na	1.8	4.7	9.199
MLP	GPU_FP16_12.50%	maxp	1	0.57	0.278080	1759.450	3864.97	1.177072	2.585665	97.15	na	1.8	4.7	6.661
MLP	GPU_FP16_12.50%	maxp	2	0.75	0.285041	2716.180	7639.51	1.817124	5.110832	97.15	na	1.8	4.7	6.510
MLP	GPU_FP16_12.50%	maxp	4	1.10	0.290605	3849.620	15091.20	2.575396	10.096013	97.15	na	1.8	4.7	6.643
MLP	GPU_FP16_12.50%	maxp	8	1.64	0.306844	4785.050	29828.10	3.201198	19.954999	97.15	na	1.8	4.7	7.353
MLP	GPU_FP16_12.50%	maxp	16	3.12	0.342703	5657.460	71794.20	3.784841	48.030320	97.15	na	1.8	4.7	7.652
MLP	GPU_FP16_12.50%	maxp	32	5.61	0.317250	6023.530	121789.00	4.029742	81.476841	97.15	na	1.8	4.7	7.543
MLP	GPU_FP16_12.50%	maxp	64	9.11	0.463625	6206.060	192120.00	4.151854	128.528280	97.15	na	1.8	4.7	7.886
MLP	GPU_FP16_12.50%	maxp	128	21.57	0.665250	6360.250	278034.00	4.255007	186.004746	97.15	na	1.8	4.7	8.382
MLP	GPU_FP32_12.50%	maxp	1	0.60	0.216087	2019.720	4963.28	1.351193	3.320434	97.15	na	1.8	4.7	7.411
MLP	GPU_FP32_12.50%	maxp	2	0.87	0.287818	3002.930	9745.51	2.008960	6.519746	97.15	na	1.8	4.7	7.265
MLP	GPU_FP32_12.50%	maxp	4	1.22	0.369457	4129.030	19384.00	2.762321	12.967896	97.15	na	1.8	4.7	7.309
MLP	GPU_FP32_12.50%	maxp	8	1.69	0.237187	5019.610	36867.70	3.358119	24.664491	97.15	na	1.8	4.7	7.732
MLP	GPU_FP32_12.50%	maxp	16	3.19	0.410703	5688.890	72480.20	3.805867	48.489254	97.15	na	1.8	4.7	7.861
MLP	GPU_FP32_12.50%	maxp	32	5.71	0.453156	5851.430	140466.00	3.914607	93.971754	97.15	na	1.8	4.7	7.759
MLP	GPU_FP32_12.50%	maxp	64	10.87	0.591562	6095.240	173530.00	4.077716	116.091570	97.15	na	1.8	4.7	7.935
MLP	GPU_FP32_12.50%	maxp	128	21.53	0.684250	6320.990	264873.00	4.228742	177.200037	97.15	na	1.8	4.7	8.310

Table A.13: Level-3 - Inference results MLP NCS

NN_Topology	hw_quant_prun	Op mode	batch	lat-sys	lat-comp	fps-system	fps-comp	tp-system	tp-comp	top1	top5 [%]	Base_Pwr_W	Idle_Pwr_W	Full_Pwr_W
MLP	NCS_FP16_100%	na	1	4.57	2.89257	218.893	345.713	4.384208	6.924286	98.86	na	0.53	1.2	1.794
MLP	NCS_FP16_100%	na	2	8.09	4.68459	247.308	426.931	4.953332	8.551001	98.86	na	0.53	1.2	1.949
MLP	NCS_FP16_100%	na	4	14.84	8.11821	269.491	492.719	5.397635	9.868669	98.86	na	0.53	1.2	1.946
MLP	NCS_FP16_100%	na	8	28.30	14.80380	282.686	540.400	5.661918	10.823672	98.86	na	0.53	1.2	2.084
MLP	NCS_FP16_100%	na	16	54.64	27.49350	292.848	581.956	5.865453	11.655997	98.86	na	0.53	1.2	2.136
MLP	NCS_FP16_100%	na	32	107.39	53.05720	297.970	603.122	5.968041	12.079931	98.86	na	0.53	1.2	2.135
MLP	NCS_FP16_100%	na	64	212.26	104.43500	301.514	612.823	6.039024	12.274232	98.86	na	0.53	1.2	2.186
MLP	NCS_FP16_100%	na	128	421.84	206.00700	303.434	621.337	6.077480	12.444759	98.86	na	0.53	1.2	2.212
MLP	NCS_FP16_50%	na	1	3.53	1.93663	283.339	516.360	1.649033	3.005215	98.75	na	0.53	1.2	1.809
MLP	NCS_FP16_50%	na	2	5.93	2.76217	337.009	724.068	1.961392	4.214076	98.75	na	0.53	1.2	1.838
MLP	NCS_FP16_50%	na	4	10.65	4.33873	375.488	921.928	2.185340	5.365621	98.75	na	0.53	1.2	1.864
MLP	NCS_FP16_50%	na	8	20.03	7.23535	399.446	1105.680	2.324776	6.435058	98.75	na	0.53	1.2	1.933
MLP	NCS_FP16_50%	na	16	38.83	13.33140	412.007	1200.170	2.397881	6.984989	98.75	na	0.53	1.2	1.926
MLP	NCS_FP16_50%	na	32	75.78	24.59050	422.289	1301.310	2.457722	7.573624	98.75	na	0.53	1.2	1.956
MLP	NCS_FP16_50%	na	64	149.82	47.95140	427.188	1334.680	2.486234	7.767838	98.75	na	0.53	1.2	1.985
MLP	NCS_FP16_50%	na	128	297.84	93.36540	429.768	1370.960	2.501250	7.978987	98.75	na	0.53	1.2	2.010
MLP	NCS_FP16_25%	na	1	3.23	1.66639	309.240	600.099	0.575805	1.117384	98.49	na	0.53	1.2	1.620
MLP	NCS_FP16_25%	na	2	5.24	2.12441	381.922	941.439	0.711139	1.752959	98.49	na	0.53	1.2	1.710
MLP	NCS_FP16_25%	na	4	9.33	3.07935	428.685	1298.980	0.798211	2.418701	98.49	na	0.53	1.2	1.744
MLP	NCS_FP16_25%	na	8	17.48	5.06905	457.574	1578.210	0.852003	2.938627	98.49	na	0.53	1.2	1.779
MLP	NCS_FP16_25%	na	16	33.60	8.64809	476.212	1850.120	0.886707	3.444923	98.49	na	0.53	1.2	1.816
MLP	NCS_FP16_25%	na	32	66.10	16.10990	484.102	1986.350	0.901398	3.698584	98.49	na	0.53	1.2	1.835
MLP	NCS_FP16_25%	na	64	129.49	30.16850	494.267	2121.420	0.920325	3.950084	98.49	na	0.53	1.2	1.837
MLP	NCS_FP16_25%	na	128	256.77	58.25070	498.496	2197.400	0.928200	4.091559	98.49	na	0.53	1.2	1.877
MLP	NCS_FP16_12.50%	na	1	3.16	1.61570	316.199	618.928	0.211537	0.414063	97.95	na	0.53	1.2	1.548
MLP	NCS_FP16_12.50%	na	2	5.08	1.97230	393.474	1014.050	0.263234	0.678399	97.95	na	0.53	1.2	1.628
MLP	NCS_FP16_12.50%	na	4	8.87	2.67093	450.727	1497.600	0.301536	1.001894	97.95	na	0.53	1.2	1.670
MLP	NCS_FP16_12.50%	na	8	16.44	4.18684	486.767	1910.750	0.325647	1.278292	97.95	na	0.53	1.2	1.699
MLP	NCS_FP16_12.50%	na	16	31.79	7.11847	503.270	2247.670	0.336688	1.503691	97.95	na	0.53	1.2	1.711
MLP	NCS_FP16_12.50%	na	32	62.05	12.94530	515.686	2471.930	0.344994	1.653721	97.95	na	0.53	1.2	1.735
MLP	NCS_FP16_12.50%	na	64	123.00	24.00580	520.318	2666.020	0.348093	1.783567	97.95	na	0.53	1.2	1.719
MLP	NCS_FP16_12.50%	na	128	243.20	45.77660	526.317	2796.190	0.352106	1.870651	97.95	na	0.53	1.2	1.764

Table A.14: Level-3 - Inference results MLP U96 A53

NN_Topology	hw_datatype_prun	Op mode	batch	lat-sys	lat-comp	fps-system	fps-comp	tp-system	tp-comp	top1	top5 [%]	Base_Pwr_W	Idle_Pwr_W	Full_Pwr_W
MLP	U96-Quadcore A53_INT4_100%	na	2	na	101.382	na	19.727368	na	0.395119	98.77	na	9.2	14.3	15.314286
MLP	U96-Quadcore A53_INT4_100%	na	4	na	102.873	na	38.882894	na	0.778785	98.77	na	9.2	14.3	15.320663
MLP	U96-Quadcore A53_INT4_100%	na	8	na	100.440	na	79.649542	na	1.595301	98.77	na	9.2	14.3	15.320353
MLP	U96-Quadcore A53_INT4_100%	na	16	na	124.383	na	128.634942	na	2.576429	98.77	na	9.2	14.3	15.321807
MLP	U96-Quadcore A53_INT4_100%	na	32	na	170.250	na	187.958884	na	3.764628	98.77	na	9.2	14.3	15.310865
MLP	U96-Quadcore A53_INT4_100%	na	64	na	253.227	na	252.737662	na	5.062083	98.77	na	9.2	14.3	15.320570
MLP	U96-Quadcore A53_INT4_100%	na	128	na	434.834	na	294.365206	na	5.895841	98.77	na	9.2	14.3	15.322039
MLP	U96-Quadcore A53_INT2_100%	na	2	na	100.792	na	19.842845	na	0.397432	98.75	na	9.2	14.3	15.323613
MLP	U96-Quadcore A53_INT2_100%	na	4	na	101.204	na	39.524129	na	0.791629	98.75	na	9.2	14.3	15.334228
MLP	U96-Quadcore A53_INT2_100%	na	8	na	100.627	na	79.501525	na	1.592336	98.75	na	9.2	14.3	15.327745
MLP	U96-Quadcore A53_INT2_100%	na	16	na	125.092	na	127.905861	na	2.561826	98.75	na	9.2	14.3	15.311869
MLP	U96-Quadcore A53_INT2_100%	na	32	na	170.559	na	187.618361	na	3.757808	98.75	na	9.2	14.3	15.309146
MLP	U96-Quadcore A53_INT2_100%	na	64	na	253.599	na	252.366926	na	5.054657	98.75	na	9.2	14.3	15.318727
MLP	U96-Quadcore A53_INT2_100%	na	128	na	435.264	na	294.074401	na	5.890016	98.75	na	9.2	14.3	15.312922
MLP	U96-Quadcore A53_INT4_50%	na	2	na	26.779	na	74.685388	na	0.434669	98.62	na	9.2	14.3	15.353571
MLP	U96-Quadcore A53_INT4_50%	na	4	na	26.904	na	148.676777	na	0.865299	98.62	na	9.2	14.3	15.359184
MLP	U96-Quadcore A53_INT4_50%	na	8	na	27.285	na	293.201393	na	1.706432	98.62	na	9.2	14.3	15.370554
MLP	U96-Quadcore A53_INT4_50%	na	16	na	33.679	na	475.073488	na	2.764928	98.62	na	9.2	14.3	15.363093
MLP	U96-Quadcore A53_INT4_50%	na	32	na	47.012	na	680.677274	na	3.961542	98.62	na	9.2	14.3	15.362243
MLP	U96-Quadcore A53_INT4_50%	na	64	na	74.628	na	857.586965	na	4.991156	98.62	na	9.2	14.3	15.363118
MLP	U96-Quadcore A53_INT4_50%	na	128	na	129.957	na	984.941173	na	5.732358	98.62	na	9.2	14.3	15.360483
MLP	U96-Quadcore A53_INT2_50%	na	2	na	26.684	na	74.951282	na	0.436216	98.49	na	9.2	14.3	15.357660
MLP	U96-Quadcore A53_INT2_50%	na	4	na	27.236	na	146.864444	na	0.854751	98.49	na	9.2	14.3	15.358208
MLP	U96-Quadcore A53_INT2_50%	na	8	na	26.752	na	299.043062	na	1.740431	98.49	na	9.2	14.3	15.367722
MLP	U96-Quadcore A53_INT2_50%	na	16	na	33.614	na	475.992146	na	2.770274	98.49	na	9.2	14.3	15.367203
MLP	U96-Quadcore A53_INT2_50%	na	32	na	46.919	na	682.026471	na	3.969394	98.49	na	9.2	14.3	15.364860
MLP	U96-Quadcore A53_INT2_50%	na	64	na	74.500	na	859.060403	na	4.999732	98.49	na	9.2	14.3	15.367707
MLP	U96-Quadcore A53_INT2_50%	na	128	na	129.915	na	985.259593	na	5.734211	98.49	na	9.2	14.3	15.365400
MLP	U96-Quadcore A53_INT4_25%	na	2	na	7.828	na	255.493102	na	0.475728	98.29	na	9.2	14.3	15.348214
MLP	U96-Quadcore A53_INT4_25%	na	4	na	7.848	na	509.683996	na	0.949032	98.29	na	9.2	14.3	15.348087
MLP	U96-Quadcore A53_INT4_25%	na	8	na	7.853	na	1018.718961	na	1.896855	98.29	na	9.2	14.3	15.346164
MLP	U96-Quadcore A53_INT4_25%	na	16	na	10.063	na	1589.983106	na	2.960549	98.29	na	9.2	14.3	15.354819
MLP	U96-Quadcore A53_INT4_25%	na	32	na	14.395	na	2222.994095	na	4.139215	98.29	na	9.2	14.3	15.356949
MLP	U96-Quadcore A53_INT4_25%	na	64	na	23.355	na	2740.312567	na	5.102462	98.29	na	9.2	14.3	15.353874
MLP	U96-Quadcore A53_INT4_25%	na	128	na	41.156	na	3110.117601	na	5.791039	98.29	na	9.2	14.3	15.352365
MLP	U96-Quadcore A53_INT2_25%	na	2	na	7.754	na	257.931390	na	0.480268	98.04	na	9.2	14.3	15.338248
MLP	U96-Quadcore A53_INT2_25%	na	4	na	7.841	na	510.139013	na	0.949879	98.04	na	9.2	14.3	15.337409
MLP	U96-Quadcore A53_INT2_25%	na	8	na	7.850	na	1019.108280	na	1.897580	98.04	na	9.2	14.3	15.338295
MLP	U96-Quadcore A53_INT2_25%	na	16	na	10.115	na	1581.809194	na	2.945329	98.04	na	9.2	14.3	15.342816
MLP	U96-Quadcore A53_INT2_25%	na	32	na	14.387	na	2224.230208	na	4.141517	98.04	na	9.2	14.3	15.338731
MLP	U96-Quadcore A53_INT2_25%	na	64	na	23.176	na	2761.477390	na	5.141871	98.04	na	9.2	14.3	15.336141
MLP	U96-Quadcore A53_INT2_25%	na	128	na	41.228	na	3104.686136	na	5.780926	98.04	na	9.2	14.3	15.336937
MLP	U96-Quadcore A53_INT4_12.50%	na	2	na	2.536	na	788.643533	na	0.527603	97.54	na	9.2	14.3	15.403571
MLP	U96-Quadcore A53_INT4_12.50%	na	4	na	2.555	na	1565.557730	na	1.047358	97.54	na	9.2	14.3	15.405612
MLP	U96-Quadcore A53_INT4_12.50%	na	8	na	2.608	na	3067.484663	na	2.052147	97.54	na	9.2	14.3	15.409585
MLP	U96-Quadcore A53_INT4_12.50%	na	16	na	3.405	na	4698.972100	na	3.143612	97.54	na	9.2	14.3	15.420983
MLP	U96-Quadcore A53_INT4_12.50%	na	32	na	5.066	na	6316.620608	na	4.225819	97.54	na	9.2	14.3	15.395697
MLP	U96-Quadcore A53_INT4_12.50%	na	64	na	8.577	na	7461.816486	na	4.991955	97.54	na	9.2	14.3	15.393603
MLP	U96-Quadcore A53_INT4_12.50%	na	128	na	14.996	na	8535.609496	na	5.710323	97.54	na	9.2	14.3	15.393147
MLP	U96-Quadcore A53_INT2_12.50%	na	2	na	2.599	na	769.526741	na	0.514813	96.85	na	9.2	14.3	15.392657
MLP	U96-Quadcore A53_INT2_12.50%	na	4	na	2.577	na	1552.192472	na	1.038417	96.85	na	9.2	14.3	15.393918
MLP	U96-Quadcore A53_INT2_12.50%	na	8	na	2.566	na	3117.692907	na	2.085737	96.85	na	9.2	14.3	15.393484
MLP	U96-Quadcore A53_INT2_12.50%	na	16	na	3.458	na	4626.951995	na	3.095431	96.85	na	9.2	14.3	15.396590
MLP	U96-Quadcore A53_INT2_12.50%	na	32	na	5.027	na	6365.625622	na	4.258604	96.85	na	9.2	14.3	15.398132
MLP	U96-Quadcore A53_INT2_12.50%	na	64	na	8.325	na	7687.687688	na	5.143063	96.85	na	9.2	14.3	15.397998
MLP	U96-Quadcore A53_INT2_12.50%	na	128	na	15.070	na	8493.696085	na	5.682283	96.85	na	9.2	14.3	15.399641

Table A.15: Level-3 - Inference results MLP ZCU104 BISMO

NN_Topology	hw_datatype_prun	Op mode	batch	lat-sys	lat-comp	fps-system	fps-comp	tp-system	tp-comp	top1	top5 [%]	Base_Pwr_W	Idle_Pwr_W	Full_Pwr_W
MLP	ZCU104-BISMO_INT4_100%	na	2	na	3.259364	na	613.616569	na	12.290126	98.77	na	9.2	14.3	15.225
MLP	ZCU104-BISMO_INT4_100%	na	4	na	5.631140	na	710.335702	na	14.227314	98.77	na	9.2	14.3	15.325
MLP	ZCU104-BISMO_INT4_100%	na	8	na	10.265632	na	779.299316	na	15.608586	98.77	na	9.2	14.3	15.300
MLP	ZCU104-BISMO_INT4_100%	na	16	na	19.714788	na	811.573525	na	16.255006	98.77	na	9.2	14.3	15.475
MLP	ZCU104-BISMO_INT4_100%	na	32	na	38.386224	na	833.632399	na	16.696823	98.77	na	9.2	14.3	15.175
MLP	ZCU104-BISMO_INT4_100%	na	64	na	76.340167	na	838.352895	na	16.791370	98.77	na	9.2	14.3	15.300
MLP	ZCU104-BISMO_INT4_100%	na	128	na	154.567620	na	828.116523	na	16.586346	98.77	na	9.2	14.3	15.300
MLP	ZCU104-BISMO_INT2_100%	na	2	na	2.147294	na	931.404959	na	18.655110	98.75	na	9.2	14.3	15.175
MLP	ZCU104-BISMO_INT2_100%	na	4	na	3.365645	na	1188.479510	na	23.804056	98.75	na	9.2	14.3	15.425
MLP	ZCU104-BISMO_INT2_100%	na	8	na	5.782697	na	1383.437545	na	27.708871	98.75	na	9.2	14.3	15.550
MLP	ZCU104-BISMO_INT2_100%	na	16	na	10.675147	na	1498.808400	na	30.019633	98.75	na	9.2	14.3	15.350
MLP	ZCU104-BISMO_INT2_100%	na	32	na	20.320172	na	1574.789820	na	31.541465	98.75	na	9.2	14.3	15.175
MLP	ZCU104-BISMO_INT2_100%	na	64	na	39.836965	na	1606.548089	na	32.177552	98.75	na	9.2	14.3	15.400
MLP	ZCU104-BISMO_INT2_100%	na	128	na	79.756498	na	1604.884909	na	32.144240	98.75	na	9.2	14.3	15.225
MLP	ZCU104-BISMO_INT4_50%	na	2	na	1.190105	na	1680.524129	na	9.780650	98.62	na	9.2	14.3	15.275
MLP	ZCU104-BISMO_INT4_50%	na	4	na	1.907534	na	2096.948426	na	12.204240	98.62	na	9.2	14.3	15.200
MLP	ZCU104-BISMO_INT4_50%	na	8	na	3.340349	na	2394.959329	na	13.938663	98.62	na	9.2	14.3	15.475
MLP	ZCU104-BISMO_INT4_50%	na	16	na	6.277260	na	2548.882793	na	14.834498	98.62	na	9.2	14.3	15.375
MLP	ZCU104-BISMO_INT4_50%	na	32	na	12.228749	na	2616.784432	na	15.229685	98.62	na	9.2	14.3	15.350
MLP	ZCU104-BISMO_INT4_50%	na	64	na	23.566854	na	2715.678554	na	15.805249	98.62	na	9.2	14.3	15.400
MLP	ZCU104-BISMO_INT4_50%	na	128	na	46.917617	na	2728.186302	na	15.878044	98.62	na	9.2	14.3	15.400
MLP	ZCU104-BISMO_INT2_50%	na	2	na	0.849005	na	2355.697897	na	13.710162	98.49	na	9.2	14.3	15.350
MLP	ZCU104-BISMO_INT2_50%	na	4	na	1.227849	na	3257.728513	na	18.959980	98.49	na	9.2	14.3	15.225
MLP	ZCU104-BISMO_INT2_50%	na	8	na	2.001101	na	3997.798612	na	23.267188	98.49	na	9.2	14.3	15.375
MLP	ZCU104-BISMO_INT2_50%	na	16	na	3.562573	na	4491.135663	na	26.138410	98.49	na	9.2	14.3	15.400
MLP	ZCU104-BISMO_INT2_50%	na	32	na	6.650998	na	4811.308017	na	28.001813	98.49	na	9.2	14.3	15.325
MLP	ZCU104-BISMO_INT2_50%	na	64	na	12.741983	na	5022.766080	na	29.232499	98.49	na	9.2	14.3	15.400
MLP	ZCU104-BISMO_INT2_50%	na	128	na	25.079286	na	5103.813562	na	29.704195	98.49	na	9.2	14.3	15.400
MLP	ZCU104-BISMO_INT4_25%	na	2	na	0.488890	na	4090.899793	na	7.617255	98.29	na	9.2	14.3	15.350
MLP	ZCU104-BISMO_INT4_25%	na	4	na	0.733966	na	5449.841498	na	10.147605	98.29	na	9.2	14.3	15.375
MLP	ZCU104-BISMO_INT4_25%	na	8	na	1.242979	na	6436.147982	na	11.984108	98.29	na	9.2	14.3	15.225
MLP	ZCU104-BISMO_INT4_25%	na	16	na	2.257038	na	7088.935974	na	13.199599	98.29	na	9.2	14.3	15.325
MLP	ZCU104-BISMO_INT4_25%	na	32	na	4.331067	na	7388.479559	na	13.757349	98.29	na	9.2	14.3	15.400
MLP	ZCU104-BISMO_INT4_25%	na	64	na	8.464694	na	7560.816729	na	14.078241	98.29	na	9.2	14.3	15.375
MLP	ZCU104-BISMO_INT4_25%	na	128	na	16.747152	na	7643.090598	na	14.231435	98.29	na	9.2	14.3	15.550
MLP	ZCU104-BISMO_INT2_25%	na	2	na	0.378882	na	5278.693924	na	9.828928	98.04	na	9.2	14.3	15.350
MLP	ZCU104-BISMO_INT2_25%	na	4	na	0.511098	na	7826.287718	na	14.572548	98.04	na	9.2	14.3	15.325
MLP	ZCU104-BISMO_INT2_25%	na	8	na	0.783952	na	10204.712920	na	19.001175	98.04	na	9.2	14.3	15.275
MLP	ZCU104-BISMO_INT2_25%	na	16	na	1.346831	na	11879.742850	na	22.120081	98.04	na	9.2	14.3	15.400
MLP	ZCU104-BISMO_INT2_25%	na	32	na	2.448887	na	13067.161250	na	24.331054	98.04	na	9.2	14.3	15.375
MLP	ZCU104-BISMO_INT2_25%	na	64	na	4.681727	na	13670.169150	na	25.453855	98.04	na	9.2	14.3	15.325
MLP	ZCU104-BISMO_INT2_25%	na	128	na	9.032107	na	14171.665590	na	26.387641	98.04	na	9.2	14.3	15.225
MLP	ZCU104-BISMO_INT4_12.50%	na	2	na	0.221656	na	9022.990580	na	6.036381	97.54	na	9.2	14.3	15.375
MLP	ZCU104-BISMO_INT4_12.50%	na	4	na	0.326347	na	12256.884690	na	8.199856	97.54	na	9.2	14.3	15.350
MLP	ZCU104-BISMO_INT4_12.50%	na	8	na	0.519242	na	15407.065260	na	10.307327	97.54	na	9.2	14.3	15.250
MLP	ZCU104-BISMO_INT4_12.50%	na	16	na	0.928151	na	17238.574330	na	11.532606	97.54	na	9.2	14.3	15.775
MLP	ZCU104-BISMO_INT4_12.50%	na	32	na	1.779600	na	17981.563840	na	12.029666	97.54	na	9.2	14.3	15.425
MLP	ZCU104-BISMO_INT4_12.50%	na	64	na	3.351983	na	19093.175590	na	12.773334	97.54	na	9.2	14.3	15.400
MLP	ZCU104-BISMO_INT4_12.50%	na	128	na	6.721276	na	19044.002950	na	12.740438	97.54	na	9.2	14.3	15.400
MLP	ZCU104-BISMO_INT2_12.50%	na	2	na	0.177743	na	11252.188550	na	7.527714	96.85	na	9.2	14.3	15.375
MLP	ZCU104-BISMO_INT2_12.50%	na	4	na	0.237272	na	16858.317640	na	11.278215	96.85	na	9.2	14.3	15.400
MLP	ZCU104-BISMO_INT2_12.50%	na	8	na	0.352492	na	22695.544100	na	15.183319	96.85	na	9.2	14.3	15.350
MLP	ZCU104-BISMO_INT2_12.50%	na	16	na	0.588134	na	27204.670770	na	18.199925	96.85	na	9.2	14.3	15.375
MLP	ZCU104-BISMO_INT2_12.50%	na	32	na	1.049068	na	30503.274480	na	20.406691	96.85	na	9.2	14.3	15.400
MLP	ZCU104-BISMO_INT2_12.50%	na	64	na	1.952913	na	32771.555480	na	21.924171	96.85	na	9.2	14.3	15.375
MLP	ZCU104-BISMO_INT2_12.50%	na	128	na	3.816102	na	33542.080370	na	22.439652	96.85	na	9.2	14.3	15.400

Table A.16: Level-3 - Inference results MLP ZCU104 FINN

NN_Topology	hw_datatype_prun	Op mode	batch	lat-sys	lat-comp	fps-system	fps-comp	tp-system	tp-comp	top1	top5 [%]	Base_Pwr_W	Idle_Pwr_W	Full_Pwr_W
MLP	ZCU104-FINN_INT2_100%	na	1	na	0.023	na	43478.261	na	870.826090	98.75	na	9.2	13.71	13.9525
MLP	ZCU104-FINN_INT2_100%	na	2	na	0.036	na	55555.556	na	1112.722231	98.75	na	9.2	13.71	13.9050
MLP	ZCU104-FINN_INT2_100%	na	4	na	0.061	na	65573.770	na	1313.377039	98.75	na	9.2	13.71	13.9400
MLP	ZCU104-FINN_INT2_100%	na	8	na	0.111	na	72072.072	na	1443.531530	98.75	na	9.2	13.71	13.9325
MLP	ZCU104-FINN_INT2_100%	na	16	na	0.210	na	76190.476	na	1526.019044	98.75	na	9.2	13.71	13.9875
MLP	ZCU104-FINN_INT2_100%	na	32	na	0.408	na	78431.373	na	1570.901970	98.75	na	9.2	13.71	13.9000
MLP	ZCU104-FINN_INT2_100%	na	64	na	0.806	na	79404.467	na	1590.392070	98.75	na	9.2	13.71	14.2775
MLP	ZCU104-FINN_INT2_100%	na	128	na	1.600	na	80000.000	na	1602.320000	98.75	na	9.2	13.71	14.3650
MLP	ZCU104-FINN_INT2_100%	na	256	na	3.190	na	80250.784	na	1607.342953	98.75	na	9.2	13.71	14.5150
MLP	ZCU104-FINN_INT2_100%	na	512	na	6.369	na	80389.386	na	1610.119012	98.75	na	9.2	13.71	14.9500
MLP	ZCU104-FINN_INT4_50%	na	1	na	0.014	na	71428.571	na	415.714283	98.62	na	9.2	14.20	14.4750
MLP	ZCU104-FINN_INT4_50%	na	2	na	0.018	na	105263.158	na	612.631580	98.62	na	9.2	14.20	14.3400
MLP	ZCU104-FINN_INT4_50%	na	4	na	0.024	na	153846.154	na	895.384616	98.62	na	9.2	14.20	14.4100
MLP	ZCU104-FINN_INT4_50%	na	8	na	0.040	na	186046.512	na	1082.790700	98.62	na	9.2	14.20	14.3850
MLP	ZCU104-FINN_INT4_50%	na	16	na	0.080	na	213333.333	na	1241.599998	98.62	na	9.2	14.20	14.4325
MLP	ZCU104-FINN_INT4_50%	na	32	na	0.128	na	226950.355	na	1320.851066	98.62	na	9.2	14.20	14.5125
MLP	ZCU104-FINN_INT4_50%	na	64	na	0.256	na	235294.118	na	1369.411767	98.62	na	9.2	14.20	14.6400
MLP	ZCU104-FINN_INT4_50%	na	128	na	0.512	na	239252.336	na	1392.448596	98.62	na	9.2	14.20	14.5875
MLP	ZCU104-FINN_INT4_50%	na	256	na	1.024	na	241737.488	na	1406.912180	98.62	na	9.2	14.20	14.9800
MLP	ZCU104-FINN_INT4_50%	na	512	na	2.048	na	242884.250	na	1413.586335	98.62	na	9.2	14.20	15.1125
MLP	ZCU104-FINN_INT2_50%	na	1	na	0.008	na	125000.000	na	727.500000	98.49	na	9.2	12.76	13.0200
MLP	ZCU104-FINN_INT2_50%	na	2	na	0.011	na	181818.182	na	1058.181819	98.49	na	9.2	12.76	12.9975
MLP	ZCU104-FINN_INT2_50%	na	4	na	0.015	na	266666.667	na	1552.000002	98.49	na	9.2	12.76	13.0050
MLP	ZCU104-FINN_INT2_50%	na	8	na	0.023	na	347826.087	na	2024.347826	98.49	na	9.2	12.76	12.9400
MLP	ZCU104-FINN_INT2_50%	na	16	na	0.039	na	410256.410	na	2387.692306	98.49	na	9.2	12.76	12.9325
MLP	ZCU104-FINN_INT2_50%	na	32	na	0.072	na	444444.444	na	2586.666664	98.49	na	9.2	12.76	13.0200
MLP	ZCU104-FINN_INT2_50%	na	64	na	0.138	na	463768.116	na	2699.130435	98.49	na	9.2	12.76	13.0575
MLP	ZCU104-FINN_INT2_50%	na	128	na	0.269	na	475836.431	na	2769.368028	98.49	na	9.2	12.76	13.0375
MLP	ZCU104-FINN_INT2_50%	na	256	na	0.531	na	482109.228	na	2805.875707	98.49	na	9.2	12.76	13.1300
MLP	ZCU104-FINN_INT2_50%	na	512	na	1.056	na	484848.485	na	2821.818183	98.49	na	9.2	12.76	13.1850
MLP	ZCU104-FINN_INT4_25%	na	1	na	0.005	na	200000.000	na	372.400000	98.29	na	9.2	14.20	14.4225
MLP	ZCU104-FINN_INT4_25%	na	2	na	0.008	na	285714.286	na	532.000001	98.29	na	9.2	14.20	14.3275
MLP	ZCU104-FINN_INT4_25%	na	4	na	0.012	na	400000.000	na	744.800000	98.29	na	9.2	14.20	14.3250
MLP	ZCU104-FINN_INT4_25%	na	8	na	0.016	na	470588.235	na	876.235294	98.29	na	9.2	14.20	14.3200
MLP	ZCU104-FINN_INT4_25%	na	16	na	0.032	na	533333.333	na	993.066666	98.29	na	9.2	14.20	14.3525
MLP	ZCU104-FINN_INT4_25%	na	32	na	0.064	na	561403.509	na	1045.333334	98.29	na	9.2	14.20	14.3350
MLP	ZCU104-FINN_INT4_25%	na	64	na	0.128	na	576576.577	na	1073.585586	98.29	na	9.2	14.20	14.4150
MLP	ZCU104-FINN_INT4_25%	na	128	na	0.256	na	589861.751	na	1098.322580	98.29	na	9.2	14.20	14.4675
MLP	ZCU104-FINN_INT4_25%	na	256	na	0.512	na	595348.837	na	1108.539534	98.29	na	9.2	14.20	14.5300
MLP	ZCU104-FINN_INT4_25%	na	512	na	1.024	na	598130.841	na	1113.719626	98.29	na	9.2	14.20	14.5950
MLP	ZCU104-FINN_INT2_25%	na	1	na	0.002	na	500000.000	na	931.000000	98.04	na	9.2	13.86	14.1300
MLP	ZCU104-FINN_INT2_25%	na	2	na	0.003	na	666666.667	na	1241.333334	98.04	na	9.2	13.86	14.0100
MLP	ZCU104-FINN_INT2_25%	na	4	na	0.004	na	1000000.000	na	1862.000000	98.04	na	9.2	13.86	13.9800
MLP	ZCU104-FINN_INT2_25%	na	8	na	0.007	na	1142857.143	na	2128.000000	98.04	na	9.2	13.86	14.0225
MLP	ZCU104-FINN_INT2_25%	na	16	na	0.010	na	1600000.000	na	2979.200000	98.04	na	9.2	13.86	14.0600
MLP	ZCU104-FINN_INT2_25%	na	32	na	0.019	na	1684210.526	na	3135.999999	98.04	na	9.2	13.86	14.0650
MLP	ZCU104-FINN_INT2_25%	na	64	na	0.036	na	1777777.778	na	3310.222223	98.04	na	9.2	13.86	14.0050
MLP	ZCU104-FINN_INT2_25%	na	128	na	0.068	na	1882352.941	na	3504.941176	98.04	na	9.2	13.86	14.1025
MLP	ZCU104-FINN_INT2_25%	na	256	na	0.134	na	1910447.761	na	3557.253731	98.04	na	9.2	13.86	14.1450
MLP	ZCU104-FINN_INT2_25%	na	512	na	0.265	na	1932075.472	na	3597.524529	98.04	na	9.2	13.86	14.1825
MLP	ZCU104-FINN_INT4_12.50%	na	1	na	0.003	na	333333.330	na	222.999998	97.54	na	9.2	14.20	14.6500
MLP	ZCU104-FINN_INT4_12.50%	na	2	na	0.004	na	500000.000	na	334.500000	97.54	na	9.2	14.20	14.6500
MLP	ZCU104-FINN_INT4_12.50%	na	4	na	0.006	na	666666.670	na	446.000002	97.54	na	9.2	14.20	14.6500
MLP	ZCU104-FINN_INT4_12.50%	na	8	na	0.009	na	888888.890	na	594.666667	97.54	na	9.2	14.20	14.6500
MLP	ZCU104-FINN_INT4_12.50%	na	16	na	0.016	na	1000000.000	na	669.000000	97.54	na	9.2	14.20	14.6500
MLP	ZCU104-FINN_INT4_12.50%	na	32	na	0.029	na	1103448.280	na	738.206899	97.54	na	9.2	14.20	14.6500
MLP	ZCU104-FINN_INT4_12.50%	na	64	na	0.055	na	1163636.360	na	778.472725	97.54	na	9.2	14.20	14.6500
MLP	ZCU104-FINN_INT4_12.50%	na	128	na	0.109	na	1174311.930	na	785.614681	97.54	na	9.2	14.20	14.6500
MLP	ZCU104-FINN_INT4_12.50%	na	256	na	0.216	na	1185185.190	na	792.888892	97.54	na	9.2	14.20	14.6500
MLP	ZCU104-FINN_INT4_12.50%	na	512	na	0.429	na	1193473.190	na	798.433564	97.54	na	9.2	14.20	14.6500
MLP	ZCU104-FINN_INT2_12.50%	na	1	na	0.002	na	1000000.000	na	669.000000	96.85	na	9.2	13.85	14.1825
MLP	ZCU104-FINN_INT2_12.50%	na	2	na	0.002	na	1000000.000	na	669.000000	96.85	na	9.2	13.85	14.0675
MLP	ZCU104-FINN_INT2_12.50%	na	4	na	0.003	na	1333333.333	na	892.000000	96.85	na	9.2	13.85	14.1000
MLP	ZCU104-FINN_INT2_12.50%	na	8	na	0.004	na	2000000.000	na	1338.000000	96.85	na	9.2	13.85	14.0200
MLP	ZCU104-FINN_INT2_12.50%	na	16	na	0.006	na	2666666.667	na	1784.000000	96.85	na	9.2	13.85	14.0600
MLP	ZCU104-FINN_INT2_12.50%	na	32	na	0.010	na	3200000.000	na	2140.800000	96.85	na	9.2	13.85	14.0000
MLP	ZCU104-FINN_INT2_12.50%	na	64	na	0.019	na	3368421.053	na	2253.473684	96.85	na	9.2	13.85	14.0450
MLP	ZCU104-FINN_INT2_12.50%	na	128	na	0.035	na	3657142.857	na	2446.628571	96.85	na	9.2	13.85	14.0550
MLP	ZCU104-FINN_INT2_12.50%	na	256	na	0.068	na	3764705.882	na	2518.588235	96.85	na	9.2	13.85	14.0750
MLP	ZCU104-FINN_INT2_12.50%	na	512	na	0.133	na	3849624.060	na	2575.398496	96.85	na	9.2	13.85	14.1200

Table A.17: Level-3 - Inference results ResNet50 TPU

hw_quant_prun_net	Op mode	batch	lat-sys	lat-comp	fps-system	fps-comp	tp-system	tp-comp	top1	top5 [%]	Base_Pwr_W	Idle_Pwr_W	Full_Pwr_W
EdgeTPU_INT8_100%_RN50V15	fast	1	na	40.44151	10.552	24.727	86.5264	202.7614	na	na	na	na	1.1900
EdgeTPU_INT8_100%_RN50V15	fast	1	na	40.58504	10.589	24.640	86.8298	202.0480	na	na	na	na	1.4900
EdgeTPU_INT8_100%_RN50V15	slow	1	na	42.35792	10.075	23.608	82.6150	193.5856	na	na	na	na	0.9623
EdgeTPU_INT8_100%_RN50V15	slow	1	na	41.69559	7.111	23.983	58.3102	196.6606	na	na	na	na	1.0200

Table A.18: Level-3 - Inference results ResNet50 NCS

hw_datatype_prun_net	Op mode	batch	lat-sys	lat-comp	fps-system	fps-comp	tp-system	tp-comp	top1	top5 [%]	Base_Pwr_W	Idle_Pwr_W	Full_Pwr_W
NCS_FP16_100%_RN50	na	1	59.0438	56.3189	16.9366	17.7560	130.750552	137.076320	75.172	92.01	0.53	1.2	1.873
NCS_FP16_100%_RN50	na	2	116.196	110.7510	17.2123	18.0586	132.878956	139.412392	75.172	92.01	0.53	1.2	1.899
NCS_FP16_100%_RN50	na	4	231.346	220.9780	17.3239	18.1367	133.740508	140.015324	75.172	92.01	0.53	1.2	1.925
NCS_FP16_100%_RN50	na	8	461.168	440.2860	17.3811	18.2055	134.182092	140.546460	75.172	92.01	0.53	1.2	1.925
NCS_FP16_100%_RN50	na	16	921.238	878.9500	17.4019	18.2391	134.342668	140.805852	75.172	92.01	0.53	1.2	1.975
NCS_FP16_100%_RN50	na	32	1840.64	1755.2800	17.4192	18.2663	134.476224	141.015836	75.172	92.01	0.53	1.2	2.000
NCS_FP16_100%_RN50	na	64	3673.91	3510.8800	17.4542	18.2647	134.746424	141.003484	75.172	92.01	0.53	1.2	1.975
NCS_FP16_100%_RN50	na	128	7340.08	7019.1000	17.4726	18.2716	134.888472	141.056752	75.172	92.01	0.53	1.2	2.000

Table A.19: Level-3 - Inference results ResNet50 ZCU104 DPU

hw_quant_prun_net	Op mode	batch	lat-sys	lat-comp	fps-system	fps-comp	tp-system	tp-comp	top1	top5 [%]	Base_Pwr_W	Idle_Pwr_W	Full_Pwr_W
ZCU104-DPU_INT8_100%_RN50	na	1	17.96	14.9622	42.0869	66.8486	324.910868	516.071192	72.53	90.85	9.2	na	21.4111
ZCU104-DPU_INT8_100%_RN50	na	2	19.69	16.6310	101.8770	120.6120	786.490440	931.124640	72.53	90.85	9.2	na	25.7835
ZCU104-DPU_INT8_100%_RN50	na	3	25.37	22.4568	117.4590	133.3310	906.783480	1029.315320	72.53	90.85	9.2	na	26.3943
ZCU104-DPU_INT8_100%_RN50	na	4	33.46	30.4797	118.9440	131.8410	918.247680	1017.812520	72.53	90.85	9.2	na	26.5316
ZCU104-DPU_INT8_100%_RN50	na	5	40.28	37.2594	122.2610	138.2530	943.854920	1067.313160	72.53	90.85	9.2	na	26.8179
ZCU104-DPU_INT8_100%_RN50	na	6	47.95	45.2650	122.6650	136.8370	946.973800	1056.381640	72.53	90.85	9.2	na	26.8338
ZCU104-DPU_INT8_100%_RN50	na	7	56.74	53.6118	122.1980	135.7300	943.368560	1047.835600	72.53	90.85	9.2	na	26.8572
ZCU104-DPU_INT8_100%_RN50	na	8	64.88	62.2966	122.8050	135.2470	948.054600	1044.106840	72.53	90.85	9.2	na	26.8942

Table A.20: Level-3 - Inference results GoogLetNet TX2

hw_datatype_prun_net	Op mode	batch	lat-sys	lat-comp	fps-system	fps-comp	tp-system	tp-comp	top1	top5 [%]	Base_Pwr_W	Idle_Pwr_W	Full_Pwr_W
TX2_FP16_100%_GNv1	maxp	1	9.93	6.16337	99.8245	169.3380	312.450685	530.027940	66.928	87.832	1.8	4.7	8.07
TX2_FP16_100%_GNv1	maxp	2	17.06	10.61970	108.3600	192.3630	339.166800	602.096190	66.928	87.832	1.8	4.7	8.28
TX2_FP16_100%_GNv1	maxp	4	30.77	18.94080	120.1170	212.5030	375.966210	665.134390	66.928	87.832	1.8	4.7	8.47
TX2_FP16_100%_GNv1	maxp	8	59.85	35.75120	126.5290	224.5700	396.035770	702.904100	66.928	87.832	1.8	4.7	8.61
TX2_FP16_100%_GNv1	maxp	16	112.43	68.79580	135.8450	232.7940	425.194850	728.645220	66.928	87.832	1.8	4.7	8.69
TX2_FP16_100%_GNv1	maxp	32	234.31	135.18900	135.4680	236.7000	424.014840	740.871000	66.928	87.832	1.8	4.7	8.35
TX2_FP16_100%_GNv1	maxp	64	507.56	267.32100	124.8020	239.2780	390.630260	748.940140	66.928	87.832	1.8	4.7	8.07
TX2_FP16_100%_GNv1	maxp	128	1,090.82	532.17900	115.8900	240.6990	362.735700	753.387870	66.928	87.832	1.8	4.7	7.74
TX2_FP32_100%_GNv1	maxp	1	14.66	10.24650	67.5150	100.0330	211.321950	313.103290	66.956	87.844	1.8	4.7	9.15
TX2_FP32_100%_GNv1	maxp	2	25.75	18.04680	77.2597	112.3490	241.822861	351.652370	66.956	87.844	1.8	4.7	9.37
TX2_FP32_100%_GNv1	maxp	4	47.16	32.91710	84.5094	121.9370	264.514422	381.662810	66.956	87.844	1.8	4.7	9.56
TX2_FP32_100%_GNv1	maxp	8	84.80	62.40340	88.8812	128.5750	278.198156	402.439750	66.956	87.844	1.8	4.7	9.71
TX2_FP32_100%_GNv1	maxp	16	174.47	121.70700	91.7645	131.8400	287.222885	412.659200	66.956	87.844	1.8	4.7	9.75
TX2_FP32_100%_GNv1	maxp	32	338.86	238.44700	94.2997	134.2740	295.158061	420.277620	66.956	87.844	1.8	4.7	9.77
TX2_FP32_100%_GNv1	maxp	64	716.89	473.49800	88.4895	135.1560	276.972135	423.038280	66.956	87.844	1.8	4.7	9.50
TX2_FP32_100%_GNv1	maxp	128	1,501.33	938.02400	84.7963	136.3190	265.412419	426.678470	66.956	87.844	1.8	4.7	9.16
TX2_FP16_100%_RN50	maxp	1	16.86	12.51590	58.9353	81.6541	454.980516	630.369652	75.142	92.118	1.8	4.7	9.38
TX2_FP16_100%_RN50	maxp	2	28.82	21.01600	69.0865	96.1356	533.347780	742.166832	75.142	92.118	1.8	4.7	9.59
TX2_FP16_100%_RN50	maxp	4	53.55	39.20770	74.5107	102.3050	575.222604	789.794600	75.142	92.118	1.8	4.7	9.71
TX2_FP16_100%_RN50	maxp	8	102.85	75.83760	77.4877	105.7200	598.205044	816.158400	75.142	92.118	1.8	4.7	9.81
TX2_FP16_100%_RN50	maxp	16	198.03	145.37300	80.7253	110.2530	623.199316	851.153160	75.142	92.118	1.8	4.7	9.79
TX2_FP16_100%_RN50	maxp	32	384.42	283.94400	83.0158	112.5910	640.881976	869.202520	75.142	92.118	1.8	4.7	9.79
TX2_FP16_100%_RN50	maxp	64	750.35	564.20200	85.2552	113.3900	658.170144	875.370800	75.142	92.118	1.8	4.7	9.81
TX2_FP16_100%_RN50	maxp	128	1,505.03	1122.97000	85.9710	114.0330	663.696120	880.334760	75.142	92.118	1.8	4.7	9.86
TX2_FP32_100%_RN50	maxp	1	26.60	22.11170	37.4611	45.8165	289.199692	353.703380	75.148	92.114	1.8	4.7	10.54
TX2_FP32_100%_RN50	maxp	2	45.71	37.93000	43.6618	53.0236	337.069096	409.342192	75.148	92.114	1.8	4.7	10.83
TX2_FP32_100%_RN50	maxp	4	86.50	72.20740	46.1865	55.4962	356.559780	428.430664	75.148	92.114	1.8	4.7	10.86
TX2_FP32_100%_RN50	maxp	8	167.74	140.40100	47.6634	57.0089	367.961448	440.108708	75.148	92.114	1.8	4.7	11.01
TX2_FP32_100%_RN50	maxp	16	323.77	270.67100	49.3899	59.1311	381.290028	456.492092	75.148	92.114	1.8	4.7	11.13
TX2_FP32_100%_RN50	maxp	32	630.79	529.39800	50.6655	60.4808	391.137660	466.911776	75.148	92.114	1.8	4.7	11.15
TX2_FP32_100%_RN50	maxp	64	1,237.70	1046.48000	51.6155	61.1055	398.471660	471.734460	75.148	92.114	1.8	4.7	11.24
TX2_FP32_100%_RN50	maxp	128	2,566.91	2080.78000	49.8588	61.5141	384.909936	474.888852	75.148	92.114	1.8	4.7	10.88

Table A.21: Level-3 - Inference results GoogLeNet U96 DPU

hw_datatype_prun_net	Op mode	batch	lat-sys	lat-comp	fps-system	fps-comp	tp-system	tp-comp	top1	top5 [%]	Base_Pwr_W	Idle_Pwr_W	Full_Pwr_W
Ultra96-DPU_INT8_100%_GNv1	na	1	19.83	16.7913	50.4470	59.5752	157.899110	186.470376	69.41	89.31	2.5	6.8	8.02899
Ultra96-DPU_INT8_100%_GNv1	na	2	33.41	29.9483	59.9339	66.5095	187.593107	208.174735	69.41	89.31	2.5	6.8	8.23871
Ultra96-DPU_INT8_100%_GNv1	na	3	50.72	47.3482	59.6774	na	186.790262	na	69.41	89.31	2.5	6.8	8.33854
Ultra96-DPU_INT8_100%_GNv1	na	4	66.76	63.2061	59.7523	na	187.024699	na	69.41	89.31	2.5	6.8	8.23847
Ultra96-DPU_INT8_100%_GNv1	na	5	83.64	81.1080	59.5355	na	186.346115	na	69.41	89.31	2.5	6.8	8.22945
Ultra96-DPU_INT8_100%_GNv1	na	6	101.58	99.3080	57.3676	na	179.560588	na	69.41	89.31	2.5	6.8	8.22359
Ultra96-DPU_INT8_100%_GNv1	na	7	118.88	116.4740	58.8160	na	184.094080	na	69.41	89.31	2.5	6.8	8.49968
Ultra96-DPU_INT8_100%_GNv1	na	8	138.01	129.9520	59.4669	62.0088	186.131397	194.087544	69.41	89.31	2.5	6.8	8.23202
Ultra96-DPU_INT8_100%_RN50	na	1	42.35	39.2760	23.6524	25.4261	182.596528	196.289492	73.29	91.26	2.5	6.8	8.25072
Ultra96-DPU_INT8_100%_RN50	na	2	78.27	75.2752	25.5044	26.5641	196.893968	205.074852	73.29	91.26	2.5	6.8	8.35480
Ultra96-DPU_INT8_100%_RN50	na	3	117.24	114.3270	25.5303	26.2110	197.093916	202.348920	73.29	91.26	2.5	6.8	8.35110
Ultra96-DPU_INT8_100%_RN50	na	4	156.50	153.4290	25.5102	26.0135	196.938744	200.824220	73.29	91.26	2.5	6.8	8.35554
Ultra96-DPU_INT8_100%_RN50	na	5	195.55	192.3980	25.5336	26.0119	197.119392	200.811868	73.29	91.26	2.5	6.8	8.35559
Ultra96-DPU_INT8_100%_RN50	na	6	238.10	235.2840	25.1176	25.5712	193.907872	197.409664	73.29	91.26	2.5	6.8	8.36790
Ultra96-DPU_INT8_100%_RN50	na	7	277.54	275.0170	25.0032	25.6140	193.024704	197.740080	73.29	91.26	2.5	6.8	8.36065
Ultra96-DPU_INT8_100%_RN50	na	8	313.04	309.1250	25.3920	26.0528	196.026240	201.127616	73.29	91.26	2.5	6.8	8.36256
Ultra96-DPU_INT8_80%_RN50	na	1	38.72	35.6740	22.0335	27.9587	144.099090	182.849898	73.30	91.40	2.5	6.8	8.02220
Ultra96-DPU_INT8_80%_RN50	na	2	71.00	68.1823	28.1659	29.3968	184.204986	192.255072	73.30	91.40	2.5	6.8	8.28728
Ultra96-DPU_INT8_80%_RN50	na	3	106.40	103.5790	28.1043	28.9648	183.802122	189.429792	73.30	91.40	2.5	6.8	8.29845
Ultra96-DPU_INT8_80%_RN50	na	4	142.64	139.3070	28.1512	28.6587	184.108848	187.427898	73.30	91.40	2.5	6.8	8.31120
Ultra96-DPU_INT8_80%_RN50	na	5	177.23	174.5400	28.1485	28.7747	184.091190	188.186538	73.30	91.40	2.5	6.8	8.31911
Ultra96-DPU_INT8_80%_RN50	na	6	212.72	209.5120	28.0701	28.7418	183.578454	187.971372	73.30	91.40	2.5	6.8	8.32475
Ultra96-DPU_INT8_80%_RN50	na	7	248.48	244.8590	28.1002	28.7188	183.775308	187.820952	73.30	91.40	2.5	6.8	8.32831
Ultra96-DPU_INT8_80%_RN50	na	8	284.13	281.1500	28.1339	28.7024	183.995706	187.713696	73.30	91.40	2.5	6.8	8.33277
Ultra96-DPU_INT8_50%_RN50	na	1	29.87	26.9074	28.4839	37.2458	106.814625	139.671750	69.49	91.00	2.5	6.8	8.14657
Ultra96-DPU_INT8_50%_RN50	na	2	53.43	50.4831	37.2738	39.6200	139.776750	148.575000	69.49	91.00	2.5	6.8	8.27812
Ultra96-DPU_INT8_50%_RN50	na	3	80.15	77.4458	37.3879	38.6727	140.204625	145.022625	69.49	91.00	2.5	6.8	8.27480
Ultra96-DPU_INT8_50%_RN50	na	4	106.94	103.9280	37.3020	38.4609	139.882500	144.228375	69.49	91.00	2.5	6.8	8.28515
Ultra96-DPU_INT8_50%_RN50	na	5	134.06	130.7090	37.3153	38.6069	139.932375	144.775875	69.49	91.00	2.5	6.8	8.30613
Ultra96-DPU_INT8_50%_RN50	na	6	160.18	157.5860	37.2975	38.4986	139.865625	144.369750	69.49	91.00	2.5	6.8	8.29005
Ultra96-DPU_INT8_50%_RN50	na	7	186.78	184.1340	37.3479	38.4906	140.054625	144.339750	69.49	91.00	2.5	6.8	8.29093
Ultra96-DPU_INT8_50%_RN50	na	8	213.60	210.7190	37.2311	38.4158	139.616625	144.059250	69.49	91.00	2.5	6.8	8.29107
Ultra96-DPU_INT8_30.00%_RN50	na	1	24.75	21.7491	40.3083	45.9851	98.755335	112.663495	68.83	90.16	2.5	6.8	8.09695
Ultra96-DPU_INT8_30.00%_RN50	na	2	43.30	40.5016	46.1630	49.5801	113.099350	121.471245	68.83	90.16	2.5	6.8	8.25180
Ultra96-DPU_INT8_30.00%_RN50	na	3	65.10	62.6155	45.9442	48.1703	112.563290	118.017235	68.83	90.16	2.5	6.8	8.23384
Ultra96-DPU_INT8_30.00%_RN50	na	4	86.63	83.6273	46.0664	47.6412	112.862680	116.720940	68.83	90.16	2.5	6.8	8.23952
Ultra96-DPU_INT8_30.00%_RN50	na	5	108.52	105.4230	45.8889	47.8640	112.427805	117.266800	68.83	90.16	2.5	6.8	8.23772
Ultra96-DPU_INT8_30.00%_RN50	na	6	130.29	126.8330	45.7756	47.7383	112.150220	116.958835	68.83	90.16	2.5	6.8	8.24451
Ultra96-DPU_INT8_30.00%_RN50	na	7	151.77	149.9260	45.8651	47.7294	112.369495	116.937030	68.83	90.16	2.5	6.8	8.24192
Ultra96-DPU_INT8_30.00%_RN50	na	8	174.11	170.0950	45.6856	47.7208	111.929720	116.915960	68.83	90.16	2.5	6.8	8.23289

Table A.22: Level-3 - Inference results GoogLeNet ZCU102 DPU

hw_datatype_prun_net	Op mode	batch	lat-sys	lat-comp	fps-system	fps-comp	tp-system	tp-comp	top1	top5 [%]	Base_Pwr_W	Idle_Pwr_W	Full_Pwr_W
ZCU102-DPU_INT8_100%_GNv1	na	1	9.63	6.65570	103.4310	150.3910	323.739030	470.723830	69.49	89.26	20.0	29.0	31.45
ZCU102-DPU_INT8_100%_GNv1	na	2	9.87048	6.84872	202.4760	290.4970	633.749880	909.255610	69.49	89.26	20.0	29.0	33.90
ZCU102-DPU_INT8_100%_GNv1	na	3	10.0871	7.17587	296.5550	424.3120	928.217150	1328.096560	69.49	89.26	20.0	29.0	36.40
ZCU102-DPU_INT8_100%_GNv1	na	4	11.5514	8.61315	341.3540	462.2150	1068.438020	1446.732950	69.49	89.26	20.0	29.0	37.90
ZCU102-DPU_INT8_100%_GNv1	na	5	13.533	10.36270	364.8680	529.5730	1142.036840	1657.563490	69.49	89.26	20.0	29.0	38.90
ZCU102-DPU_INT8_100%_GNv1	na	6	15.7128	12.83240	371.2730	544.7020	1162.084490	1704.917260	69.49	89.26	20.0	29.0	37.40
ZCU102-DPU_INT8_100%_GNv1	na	7	18.3435	15.44560	377.0100	536.2010	1180.041300	1678.309130	69.49	89.26	20.0	29.0	38.50
ZCU102-DPU_INT8_100%_GNv1	na	8	20.8812	17.82240	379.1090	535.4910	1186.611170	1676.086830	69.49	89.26	20.0	29.0	39.00
ZCU102-DPU_INT8_100%_RN50	na	1	17.78	14.81680	43.7478	67.6427	337.733016	522.201644	72.53	90.85	20.0	29.0	31.80
ZCU102-DPU_INT8_100%_RN50	na	2	18.59	15.56740	107.3730	127.6570	828.919560	985.512040	72.53	90.85	20.0	29.0	34.40
ZCU102-DPU_INT8_100%_RN50	na	3	20.69	17.61070	120.6650	169.7440	931.533800	1310.423680	72.53	90.85	20.0	29.0	37.50
ZCU102-DPU_INT8_100%_RN50	na	4	24.62	21.52290	161.7640	183.8050	1248.818080	1418.974600	72.53	90.85	20.0	29.0	39.70
ZCU102-DPU_INT8_100%_RN50	na	5	29.96	27.04730	165.9870	193.0190	1281.419640	1490.106680	72.53	90.85	20.0	29.0	40.10
ZCU102-DPU_INT8_100%_RN50	na	6	35.50	32.48100	167.6360	193.7870	1294.149920	1496.035640	72.53	90.85	20.0	29.0	40.30
ZCU102-DPU_INT8_100%_RN50	na	7	41.59	38.55100	168.1320	190.7860	1297.979040	1472.867920	72.53	90.85	20.0	29.0	40.50
ZCU102-DPU_INT8_100%_RN50	na	8	47.61	44.40620	167.2920	191.9080	1291.494240	1481.529760	72.53	90.85	20.0	29.0	40.60

Table A.23: Level-3 - Inference results MobileNetv1 TPU

hw_datatype_prun_net	Op mode	batch	lat-sys	lat-comp	fps-system	fps-comp	tp-system	tp-comp	top1	top5 [%]	Base_Pwr_W	Idle_Pwr_W	Full_Pwr_W
TPU_INT8_100%_MNv1	slow	1	7.86	4.08249	127.256	244.949	145.07184	279.24186	69.5674	87.7058	0.253	0.253	0.462
TPU_INT8_100%_MNv1	fast	1	6.00	2.57047	166.533	389.034	189.84762	443.49876	69.5674	87.7058	0.253	0.253	0.532
TPU_INT8_100%_GNv1	slow	1	10.03	5.72131	99.741	174.785	312.18933	547.07705	69.2434	88.4458	0.253	0.253	0.463
TPU_INT8_100%_GNv1	fast	1	7.40	3.64852	135.087	274.084	422.82231	857.88292	69.2434	88.4458	0.253	0.253	0.538

Bibliography

[1] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," in *ICML2016*, 2016, pp. 173–182.

[2] G. Tagliavini, S. Mach, D. Rossi, A. Marongiu, and L. Benini, "A transprecision floating-point platform for ultra-low power computing," in *DATE 2018*. IEEE, 2018, pp. 1051–1056.

[3] A. C. I. Malossi, M. Schaffner, A. Molnos, L. Gammaitoni, G. Tagliavini, A. Emerson, A. Tomás, D. S. Nikolopoulos, E. Flamand, and N. Wehn, "The transprecision computing paradigm: Concept, design, and applications," in *DATE 2018*. IEEE, 2018, pp. 1105–1110.

[4] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[5] W. Sung, S. Shin, and K. Hwang, "Resiliency of deep neural networks under quantization," *arXiv preprint arXiv:1511.06488*, 2015.

[6] B. J. et al. (2017) gemmlowp: A small self-contained low-precision GEMM library. https://github.com/google/gemmlowp.

[7] A. Limited. (2017) Compute Library. https://developer.arm.com/technologies/compute-library.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[9] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[10] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. IEEE, 2014, pp. 10–14.

[11] M. Blott, T. B. Preußer, N. J. Fraser, G. Gambardella, K. O'Brien, Y. Umuroglu, M. Leeser, and K. Vissers, "FINN-R: An end-to-end deep-learning framework for fast exploration of quantized neural networks," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 2018.

[12] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, and D. S. Modha, "Learned step size quantization," *arXiv preprint arXiv:1902.08153*, 2019.

[13] D. Zhang, J. Yang, D. Ye, and G. Hua, "Lq-nets: Learned quantization for highly accurate and compact deep neural networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 365–382.

[14] M. Blott, T. B. Preußer, N. Fraser, G. Gambardella, K. O'Brien, Y. Umuroglu, and M. Leeser, "Scaling neural network performance through customized hardware architectures on reconfigurable logic," in *ICCD 2017*. IEEE, 2017, pp. 419–422.

[15] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *ISCA 2017*. ACM, 2017, pp. 1–12.

[16] D. Burger, "Microsoft unveils project brainwave for real-time ai," *Microsoft Research, Microsoft*, vol. 22, 2017.

[17] (2018) Active tpc benchmarks. http://www.tpc.org/information/benchmarks.asp.

[18] Y. Liu, H. Zhang, L. Zeng, W. Wu, and C. Zhang, "Mlbench: benchmarking machine learning services against human experts," *Proceedings of the VLDB Endowment*, vol. 11, no. 10, pp. 1220–1232, 2018.

[19] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne *et al.*, "The fair guiding principles for scientific data management and stewardship," *Scientific data*, vol. 3, 2016.

[20] (2020) Qutibench web portal. https://rcl-lab.github.io/QutibenchWeb.

[21] S. Williams, A. Waterman, and D. Patterson, "Roofline: An insightful visual performance model for multicore architectures," *Communications of the ACM*, vol. 52, no. 4, pp. 65–76, 2009.

[22] M. Blott, A. Vasilciuc, M. Leeser, and L. Doyle, "Evaluating theoretical baselines for ml benchmarking across different accelerators," *IEEE Design Test*, pp. 1–1, 2021.

[23] M. Blott, N. Fraser, G. Gambardella, L. Halder, J. Kath, Z. Neveu, Y. Umuroglu, A. Vasilciuc, M. Leeser, and L. Doyle, "Evaluation of optimized CNNs on heterogeneous accelerators using a novel benchmarking approach," *IEEE Transactions on Computers*, pp. 1–1, 2020.

[24] M. Blott, L. Halder, M. Leeser, and L. Doyle, "QuTiBench: Benchmarking neural networks on heterogeneous hardware," *ACM Journal of Emerging Technologies in Computing Systems (JETC) Special Issue*, 2018.

[25] M. Blott, J. Kath, L. Halder, Y. Umuroglu, N. Fraser, G. Gambardella, M. Leeser, and L. Doyle, "Evaluation of optimized CNNs on FPGA and non-FPGA based accelerators using a novel benchmarking approach," pp. 317–317, 2020.

[26] Y. Umuroglu, Y. Akhauri, N. J. Fraser, and M. Blott, "High-throughput DNN inference with LogicNets," in *2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 2020, pp. 238–238.

[27] M. Qasaimeh, K. Denolf, A. Khodamoradi, M. Blott, J. Lo, L. Halder, K. Vissers, J. Zambreno, and P. H. Jones, "Benchmarking vision kernels and neural network inference accelerators on embedded platforms," *Journal of Systems Architecture*, p. 101896, 2020.

[28] E. Giacoumidis, Y. Lin, M. Blott, and L. P. Barry, "Real-time machine learning based fiber-induced nonlinearity compensation in energy-efficient coherent optical networks," *APL Photonics*, vol. 5, no. 4, p. 041301, 2020.

[29] M. Kroes, L. Petrica, S. Cotofana, and M. Blott, "Evolutionary bin packing for memory-efficient dataflow inference acceleration on FPGA," *arXiv preprint arXiv:2003.12449*, 2020.

[30] G. Gambardella, J. Kappauf, M. Blott, C. Doehring, M. Kumm, P. Zipf, and K. Vissers, "Efficient error-tolerant quantized neural network accelerators," in *2019 IEEE International*

Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT). IEEE, 2019, pp. 1–6.

[31] Y. Umuroglu, D. Conficconi, L. Rasnayake, T. B. Preusser, and M. Själander, "Optimizing bit-serial matrix multiplication for reconfigurable computing," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 12, no. 3, pp. 1–24, 2019.

[32] Y. Yang, Q. Huang, B. Wu, T. Zhang, L. Ma, G. Gambardella, M. Blott, L. Lavagno, K. Vissers, J. Wawrzynek *et al.*, "Synetgy: Algorithm-hardware co-design for ConvNet accelerators on embedded FPGAs," in *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2019, pp. 23–32.

[33] V. Rybalkin, A. Pappalardo, M. G. Mohsin, G. Gambardella, N. Wehn, and M. Blott, "FINN-L: Library extensions and design trade-off analysis for variable precision LSTM networks on FPGA," *arXiv preprint https://arxiv.org/abs/1807.04093*, 2018.

[34] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers, "FINN: A framework for fast, scalable binarized neural network inference," in *ISFPGA 2017*. ACM, 2017, pp. 65–74.

[35] J. Faraone, G. Gambardella, N. Fraser, M. Blott, P. Leong, and D. Boland, "Customizing low-precision deep neural networks for FPGAs," in *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2018, pp. 97–973.

[36] J. Su, N. J. Fraser, G. Gambardella, M. Blott, G. Durelli, D. B. Thomas, P. H. Leong, and P. Y. Cheung, "Accuracy to throughput trade-offs for reduced precision neural networks on reconfigurable logic," in *Applied Reconfigurable Computing. Architectures, Tools, and Applications: 14th International Symposium, ARC 2018, Santorini, Greece, May 2-4, 2018, Proceedings*, vol. 10824. Springer, 2018, p. 29.

[37] J. Faraone, N. Fraser, M. Blott, and P. H. Leong, "Syq: Learning symmetric quantization for efficient deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4300–4309.

[38] T. B. Preußer, G. Gambardella, N. Fraser, and M. Blott, "Inference of quantized neural networks on heterogeneous all-programmable devices," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2018*. IEEE, 2018, pp. 833–838.

[39] J. Faraone, N. Fraser, G. Gambardella, M. Blott, and P. H. Leong, "Compressing low precision deep neural networks using sparsity-induced regularization in ternary networks," in *International Conference on Neural Information Processing*. Springer, 2017, pp. 393–404.

[40] N. J. Fraser, Y. Umuroglu, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers, "Scaling binarized neural networks on reconfigurable logic," in *PARMA DITAM2017*, 2017, pp. 25–30.

[41] Y. Umuroglu, N. J. Fraser, G. Gambardella, and M. Blott, "A C++ library for rapid exploration of binary neural networks on reconfigurable logic," *H2RC Archive*, 2016.

[42] "Xilinx whitepaper: The anatomy of an embedded machine learning accelerator," https://www.xilinx.com/support/documentation/white_papers/wp514-emerging-dnn.pdf, accessed: 2019-12-30.

[43] (2019) Xilinx whitepaper: Fpgas in the emerging dnn inference landscape. https://www.xilinx.com/support/documentation/white_papers/wp514-emerging-dnn.pdf.

[44] Hotchips'2018 (hc30t2): Architectures for accelerating deep neural nets. https://www.youtube.com/watch?v=ydsZ7A0FF0I&feature=youtu.be.

[45] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[46] B. C. Csáji, "Approximation with artificial neural networks," *Faculty of Sciences, Etvs Lornd University, Hungary*, vol. 24, p. 48, 2001.

[47] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS 2012*, 2012, pp. 1097–1105.

[48] D. Misra, "Mish: A self regularized non-monotonic neural activation function," *arXiv preprint arXiv:1908.08681*, 2019.

[49] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[50] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[51] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR 2015*, 2015, pp. 1–9.

[52] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *CVPR 2016*, 2016, pp. 2818–2826.

[53] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*. Springer, 2016, pp. 630–645.

[54] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[55] X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, and J. E. Gonzalez, "Skipnet: Learning dynamic routing in convolutional networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 409–424.

[56] M. Wortsman, A. Farhadi, and M. Rastegari, "Discovering neural wirings," in *Advances in Neural Information Processing Systems*, 2019, pp. 2684–2694.

[57] (2019) Artificial intelligence index report 2019. https://hai.stanford.edu/sites/default/files/ai_index_2019_report.pdf.

[58] F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer, "Densenet: Implementing efficient convnet descriptor pyramids," *arXiv preprint arXiv:1404.1869*, 2014.

[59] (2018) Deepbench. https://svail.github.io/DeepBench/.

[60] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, 2017.

[61] J. Yu, Y. Hu, X. Ning, J. Qiu, K. Guo, Y. Wang, and H. Yang, "Instruction driven cross-layer cnn accelerator with winograd transformation on fpga," in *2017 International Conference on Field Programmable Technology (ICFPT)*. IEEE, 2017, pp. 227–230.

[62] F. Iandola, M. Moskewicz, K. Ashraf, S. Han, W. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50× fewer parameters and < 1 MB model size," vol. abs/1602.07630, 2016.

[63] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training neural networks with weights and activations constrained to +1 or -1," in *CoRR*, vol. abs/1602.0, 2016.

[64] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606.06160*, 2016.

[65] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 525–542.

[66] W. Sung, S. Shin, and K. Hwang, "Resiliency of deep neural networks under quantization," *CoRR*, vol. abs/1511.0, 2015.

[67] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[68] A. Mishra, E. Nurvitadhi, J. J. Cook, and D. Marr, "Wrpn: Wide reduced-precision networks," *arXiv preprint arXiv:1709.01134*, 2017.

[69] M. Kim and P. Smaragdis, "Bitwise neural networks," *arXiv preprint arXiv:1601.06071*, 2016.

[70] H. Alemdar, V. Leroy, A. Prost-Boucle, and F. Pétrot, "Ternary neural networks for resource-efficient ai applications," in *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, 2017, pp. 2547–2554.

[71] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," *arXiv preprint arXiv:1612.01064*, 2016.

[72] B. Zhuang, C. Shen, M. Tan, L. Liu, and I. Reid, "Towards effective low-bitwidth convolutional neural networks," *In other words*, vol. 2, p. 2, 2017.

[73] Z. Cai, X. He, J. Sun, and N. Vasconcelos, "Deep learning with low precision by half-wave gaussian quantization," *arXiv preprint arXiv:1702.00953*, 2017.

[74] A. Mishra and D. Marr, "Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy," *arXiv preprint arXiv:1711.05852*, 2017.

[75] C. Ma, W. An, Y. Lei, and Y. Guo, "BV-CNNs: Binary volumetric convolutional networks for 3D object recognition," in *Proceedings of the British Machine Vision Conference 2017, BMVC*, 2017.

[76] W. Sun, H. Zhao, and Z. Jin, "An efficient unconstrained facial expression recognition algorithm based on stack-binarized auto-encoders and binarized neural networks," *Neurocomputing*, vol. 267, pp. 385–395, 2017.

[77] L. Lu, "Toward computation and memory efficient neural network acoustic models with binary weights and activations," *arXiv preprint arXiv:1706.09453*, 2017.

[78] S. Han, J. Kang, H. Mao, Y. Hu, X. Li, Y. Li, D. Xie, H. Luo, S. Yao, Y. Wang *et al.*, "Ese: Efficient speech recognition engine with sparse lstm on fpga," in *FPGA 2017*. ACM, 2017, pp. 75–84.

[79] S. Wu, G. Li, F. Chen, and L. Shi, "Training and inference with integers in deep neural networks," *arXiv preprint arXiv:1802.04680*, 2018.

[80] U. Köster, T. Webb, X. Wang, M. Nassar, A. K. Bansal, W. Constable, O. Elibol, S. Gray, S. Hall, L. Hornof *et al.*, "Flexpoint: An adaptive numerical format for efficient training of deep neural networks," in *NIPS 2017*, 2017, pp. 1742–1752.

[81] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaev, G. Venkatesh *et al.*, "Mixed precision training," *arXiv preprint arXiv:1710.03740*, 2017.

[82] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *ICML 2015*, 2015, pp. 1737–1746.

[83] S. M. S. Trimberger, "Three ages of fpgas: A retrospective on the first thirty years of fpga technology: This paper reflects on how moore's law has driven the design of fpgas through three epochs: the age of invention, the age of expansion, and the age of accumulation," *IEEE Solid-State Circuits Magazine*, vol. 10, no. 2, pp. 16–29, 2018.

[84] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *2011 38th Annual international symposium on computer architecture (ISCA)*. IEEE, 2011, pp. 365–376.

[85] (2019) Cerebras. https://www.cerebras.net/.

[86] (2019) Hotchips'2019 (hc31-k2): Dr. phillip wong (tsmc): What will the next node offer us? https://www.hotchips.org/hc31-keynotes-available-to-all/.

[87] (2019) Aspinity. https://www.aspinity.com/.

[88] A. C. Yüzügüler, F. Celik, M. Drumond, B. Falsafi, and P. Frossard, "Analog neural networks with deep-submicrometer nonlinear synapses," *IEEE Micro*, vol. 39, no. 5, pp. 55–63, 2019.

[89] (2019) D-wave. https://www.dwavesys.com/.

[90] (2019) Neuroblade. https://www.neuroblade.ai/.

[91] S. K. Essera, P. A. Merollaa, J. V. Arthura, A. S. Cassidya, R. Appuswamya, A. Andreopoulosa, D. J. Berga, J. L. McKinstrya, T. Melanoa, D. R. Barcha *et al.*, "Convolutional networks for fast energy-efficient neuromorphic computing," *Proc. Nat. Acad. Sci. USA*, vol. 113, no. 41, pp. 11 441–11 446, 2016.

[92] (2019) Intel cascade lake. https://en.wikipedia.org/wiki/Cascade_Lake_(microarchitecture).

[93] B. L. Durant, O. Giroux, M. Harris, and N. Stam. (2017, May) Inside volta: The world's most advanced data center gpu. https://devblogs.nvidia.com/inside-volta/.

[94] (2017, Dec.) Taking a deeper look at the amd radeon instinct gpus for deep learning. http://blog.exxactcorp.com/taking-deeper-look-amd-radeon-instinct-gpus-deep-learning/.

[95] (2019) Amd launches epyc rome, first 7nm cpu. https://www.hpcwire.com/2019/08/08/amd-launches-epyc-rome-first-7nm-cpu.

[96] D. Hardawar. (2018, Jan.) Amd's radeon vega gpu is headed everywhere, even to machine learning. https://www.engadget.com/2018/01/08/amd-radeon-vega-mobile/.

[97] D. Franklin. (2017, March) Nvidia jetson tx2 delivers twice the intelligence to the edge. https://devblogs.nvidia.com/jetson-tx2-delivers-twice-intelligence-edge/.

[98] (2019) Nvidia ax. https://www.nvidia.com/en-us/deep-learning-ai/products/agx-systems.

[99] (2019) Nvidia turing gpu architecture. https://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf.

[100] K. Sato *et al.* (2017) An in-depth look at google's first tensor processing unit (tpu). https://cloud.google.com/blog/big-data/2017/05/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu.

[101] P. Teich. (2018, May) Tearing apart google's tpu 3.0 ai coprocessor. https://www.nextplatform.com/2018/05/10/tearing-apart-googles-tpu-3-0-ai-coprocessor/.

[102] L. Armasu. (2016, April) Deep learning on a stick: Movidius' 'fathom' neural compute stick. https://www.tomshardware.com/news/movidius-fathom-neural-compute-stick, 31694.html.

[103] B. Moons, D. Bankman, L. Yang, B. Murmann, and M. Verhelst, "Binareye: An always-on energy-accuracy-scalable binary cnn processor with all memory on chip in 28nm cmos," in *Custom Integrated Circuits Conference (CICC), 2018 IEEE.* IEEE, 2018, pp. 1–4.

[104] K. Ando, K. Ueyoshi, K. Orimo, H. Yonekawa, S. Sato, H. Nakahara, M. Ikebe, T. Asai, S. Takamaeda-Yamazaki, T. Kuroda *et al.*, "Brein memory: A 13-layer 4.2 k neuron/0.8 m synapse binary/ternary reconfigurable in-memory deep neural network accelerator in 65 nm cmos," in *VLSI Circuits, 2017 Symposium on.* IEEE, 2017, pp. C24–C25.

[105] (2018) Blog: Unlocking the promise of approximate computing for on-chip ai accelerator. https://www.ibm.com/blogs/research/2018/06/approximate-computing-ai-acceleration/.

[106] (2018) Binarized neural network (bnn) accelerator ip. http://www.latticesemi.com/Products/DesignSoftwareAndIP/IntellectualProperty/IPCore/IPCores04/BNN.

[107] E. Chung, J. Fowers, K. Ovtcharov, M. Papamichael, A. Caulfield, T. Massengill, M. Liu, D. Lo, S. Alkalay, M. Haselman *et al.*, "Serving dnns in real time at datacenter scale with project brainwave," *IEEE Micro*, vol. 38, no. 2, pp. 8–20, 2018.

[108] A. Parashar, M. Rhu *et al.*, "Scnn: An accelerator for compressed-sparse convolutional neural networks," in *International Symposium on Computer Architecture (ISCA)*. IEEE, 2017, pp. 27–40.

[109] J. Albericio, P. Judd *et al.*, "Cnvlutin: Ineffectual-neuron-free deep neural network computing," *Computer Architecture News*, vol. 44, no. 3, pp. 1–13, 2016.

[110] S. Zhang, Z. Du *et al.*, "Cambricon-x: An accelerator for sparse neural networks," in *International Symposium on Microarchitecture*. IEEE Press, 2016, p. 20.

[111] Y. Umuroglu, L. Rasnayake, and M. Sjalander, "Bismo: A scalable bit-serial matrix multiplication overlay for reconfigurable computing," *arXiv preprint arXiv:1806.08862*, 2018.

[112] P. Judd, J. Albericio, T. Hetherington, T. M. Aamodt, and A. Moshovos, "Stripes: Bit-serial deep neural network computing," in *MICRO 2016*, 2016, pp. 1–12.

[113] B. J. Murphy. (2017, Jan.) Deep learning benchmarks of nvidia tesla p100 pcie, tesla k80, and tesla m40 gpus. https://www.microway.com/hpc-tech-tips/deep-learning-benchmarks-nvidia-tesla-p100-16gb-pcie-tesla-k80-tesla-m40-gpus/.

[114] NVIDIA. (2016) Tesla p40 inferencing accelerator. http://images.nvidia.com/content/pdf/tesla/184427-Tesla-P40-Datasheet-NV-Final-Letter-Web.pdf.

[115] Wikipedia. (2018) Amd rx vega series. https://en.wikipedia.org/wiki/AMD_RX_Vega_series.

[116] Y. Umuroglu and M. Jahre, "Streamlined deployment for quantized neural networks," *arXiv preprint arXiv:1709.04060*, 2017.

[117] (2017) Product brief: Myriadx: Enhanced visual intelligence at the network edge. https://uploads.movidius.com/1503874473-MyriadXVPU_ProductBriefaug25.pdf.

[118] (2018) Whitepaper: Deep learning on mppa manycore processor. http://www.kalrayinc.com/resources/.

[119] (2018) Arm's project trillium. https://www.arm.com/products/processors/machine-learning.

[120] Y. Chen, T. Chen, Z. Xu, N. Sun, and O. Temam, "DianNao family: Energy-efficient hardware accelerators for machine learning," *Communications of the ACM*, vol. 59, no. 11, pp. 105–112, 2016.

[121] S. Han, X. Liu *et al.*, "Eie: efficient inference engine on compressed deep neural network," in *International Symposium on Computer Architecture (ISCA)*. IEEE, 2016, pp. 243–254.

[122] D. J. Moss, S. Krishnan, E. Nurvitadhi, P. Ratuszniak, C. Johnson, J. Sim, A. Mishra, D. Marr, S. Subhaschandra, and P. H. Leong, "A customizable matrix multiplication framework for the intel harpv2 xeon+ fpga platform: A deep learning case study," in *FPGA 2018*. ACM, 2018, pp. 107–116.

[123] (2019) Avx-512. https://en.wikipedia.org/wiki/AVX-512.

[124] (2019) Deep learning server for ai research - nvidia dgx-1. https://www.nvidia.com/en-us/data-center/dgx.

[125] C. C. et al. (2018) Analysis of dawnbench, a time-to-accuracy machine learning performance benchmark. https://arxiv.org/abs/1806.01427.

[126] (2018) Benchmarking dnn processors. http://eyeriss.mit.edu/benchmarking.html.

[127] *ReQuEST '18: Proceedings of the 1st on Reproducible Quality-Efficient Systems Tournament on Co-designing Pareto-efficient Deep Learning*. New York, NY, USA: ACM, 2018.

[128] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[129] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *International Conference on Machine Learning*, 2016, pp. 1329–1338.

[130] R. Adolf, S. Rama, B. Reagen, G.-Y. Wei, and D. Brooks, "Fathom: Reference workloads for modern deep learning methods," in *IISWC2016*. IEEE, 2016, pp. 1–10.

[131] H. Zhu, M. Akrout, B. Zheng, A. Pelegris, A. Phanishayee, B. Schroeder, and G. Pekhimenko, "Tbd: Benchmarking and analyzing deep neural network training," *arXiv preprint arXiv:1803.06905*, 2018.

[132] J. d. F. Licht, S. Meierhans, and T. Hoefler, "Transformations of high-level synthesis codes for high-performance computing," *arXiv preprint arXiv:1805.08288*, 2018.

[133] K. Guo, S. Han, S. Yao, Y. Wang, Y. Xie, and H. Yang, "Software-hardware codesign for efficient neural network acceleration," *IEEE Micro*, vol. 37, no. 2, pp. 18–25, 2017.

[134] (2017) Collective knowledge. http://cknowledge.org/.

[135] Y. Gu, T. Wahl, M. Bayati, and M. Leeser, "Behavioral non-portability in scientific numeric computing," in *European Conference on Parallel Processing*. Springer, 2015, pp. 558–569.

[136] (2018) Open neural network exchange format. https://onnx.ai.

[137] (2018) Nnef. https://www.khronos.org/nnef.

[138] (2018) End to end deep learning compiler stack. https://tvm.ai.

[139] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *IJCV 2015*, vol. 115, no. 3, pp. 211–252, 2015.

[140] S. Thomas, C. Gohkale, E. Tanuwidjaja, T. Chong, D. Lau, S. Garcia, and M. B. Taylor, "Cortexsuite: A synthetic brain benchmark suite," in *Workload Characterization (IISWC), 2014 IEEE International Symposium on*. IEEE, 2014, pp. 76–79.

[141] T. Chen, Y. Chen, M. Duranton, Q. Guo, A. Hashmi, M. Lipasti, A. Nere, S. Qiu, M. Sebag, and O. Temam, "Benchnn: On the broad potential application scope of hardware neural network accelerators," in *Workload Characterization (IISWC), 2012 IEEE International Symposium on*. IEEE, 2012, pp. 36–45.

[142] J. Hauswald, Y. Kang, M. A. Laurenzano, Q. Chen, C. Li, T. Mudge, R. G. Dreslinski, J. Mars, and L. Tang, "Djinn and tonic: Dnn as a service and its implications for future warehouse scale computers," in *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2015, pp. 27–40.

[143] (2018) Kaggle is the place to do data science projects. https://www.kaggle.com/.

[144] A. Danalis, G. Marin, C. McCurdy, J. S. Meredith, P. C. Roth, K. Spafford, V. Tipparaju, and J. S. Vetter, "The scalable heterogeneous computing (shoc) benchmark suite," in *Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units*. ACM, 2010, pp. 63–74.

[145] (2018) Standard performance evaluation corporation. http://spec.org.

[146] J. D. McCalpin. (2018) Stream: Sustainable memory bandwidth in high performance computers. https://www.cs.virginia.edu/stream/.

[147] J.-H. Tao, Z.-D. Du, Q. Guo *et al.*, "Benchip: Benchmarking intelligence processors," *Journal of Computer Science and Technology*, vol. 33, no. 1, pp. 1–23, 2018.

[148] (2020) Mlmark, and eembc benchmark. https://www.eembc.org/mlmark/.

[149] (2019) chframework visualization for mlperfk. https://cknowledge.org/dashboard/mlperf.inference.

[150] (2018) Mlperf: A broad ml benchmark suite for measuring performance of ml software frameworks, ml hardware accelerators, and ml cloud platforms. https://mlperf.org/.

[151] (2019) Mlperf inference benchmark. https://arxiv.org/abs/1911.02549.

[152] (2016) Efficient implementation of neural network systems built on fpgas, and programmed with opencl. https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/solution-sheets/efficient_neural_networks.pdf.

[153] R. Kitchin, *The data revolution: Big data, open data, data infrastructures and their consequences*. Sage, 2014.

[154] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "Dbpedia: A nucleus for a web of open data," in *The semantic web*. Springer, 2007, pp. 722–735.

[155] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[156] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *arXiv preprint arXiv:1905.11946*, 2019.

[157] (2019) Dnndk. https://www.xilinx.com/support/documentation/user_guides/ug1327-dnndk-user-guide.pdf.

[158] (2019) Nvidia tensorrt. https://developer.nvidia.com/tensorrt.

[159] (2019) Openvino toolkit. https://docs.openvinotoolkit.org/.

[160] Efficientnet-edgetpu. https://github.com/tensorflow/tpu/tree/master/models/official/efficientnet/edgetpu.

[161] Y. Wu, M. Schuster, Z. Chen *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.