1

Benchmarking Neuromorphic Computing for Inference

Simon Narduzzi[1], Loreto Mateu[2], Petar Jokic[1], Erfan Azarkhish[1], and Andrea Dunbar[1]

[1]CSEM, Switzerland
[2]Fraunhofer IIS, Germany

Abstract

In the last decade, there has been significant progress in the IoT domain due to the advances in the accuracy of neural networks and the industrialization of efficient neural network accelerator ASICs. However, intelligent devices will need to be omnipresent to create a seamless consumer experience. To make this a reality, further progress is still needed in the low-power embedded machine learning domain. Neuromorphic computing is a technology suited to such low-power intelligent sensing. However, neuromorphic computing is hampered today by the fragmentation of the hardware providers and the difficulty of embedding and comparing the algorithms' performance. The lack of standard key performance indicators spanning across the hardware-software domains makes it difficult to benchmark different solutions for a given application on a fair basis. In this paper, we summarize the current benchmarking solutions used in both hardware and software for neuromorphic systems, which are in general applicable to low-power systems. We then discuss the challenges in creating a fair and user-friendly method to benchmark such systems, before suggesting a clear methodology that includes possible key performance indicators.

Keywords: neuromorphic, inference, accelerators, benchmarking, low power, IoT, ASIC, key performance indicators.

1.1 Introduction

The performance necessary for consumer uptake of IoT devices has not been achieved yet. Intelligent always-on edge devices and sensors powered by AI and running on ultra-low power devices require outstanding energy efficiencies, low latency (real-time), high-throughput, and uncompromised accuracy. Neuromorphic computing rises to the challenge; however, the neuromorphic computing landscape is fragmented with no universal Key Performance Indicators (KPI), and comparison on a fair basis remains illusive [1]. The landscape is complex: comparisons should consider various aspects such as industrial maturity, CMOS technology implications, arithmetic precision, silicon area, power consumption, and accuracy obtained from neural networks running on the devices. Comparing target use-cases has the advantage of looking at the system-wide requirements but adds additional complexity. For example, if we take into account the inference frequency, this affects the current leakage and active power, significantly impacting the mean power consumption of the system.

The most commonly accepted quantitative metrics for benchmarking neuromorphic hardware are TOPS (Tera Operations Per Second) for throughput, TOPS/W for energy efficiency, and TOPS/mm2 for area efficiency. Hardware metrics rarely take into account the algorithmic structure. For software, the performance of Machine Learning (ML) algorithms is usually defined for a given task. Their KPIs generally target the prediction performance in terms of reached objective (often accuracy). Until recently, the KPIs rarely accounted for algorithm complexity, the computational cost, or the structure which impacts its performance on a given hardware.

Moreover, these metrics are only applicable to traditional neural networks, such as Deep Neural Network (DNNs), while for Spiking Neural Networks (SNN), other metrics such as energy per synaptic operation for energy efficiency are used. Indeed, the very nature of these DNNs and SNNs prohibits a comparison based on standard NN parameters.

The main questions asked by end-users, system integrators, and sensor manufacturers are: what is the best solution for the application, and whether a given neuromorphic processor provides some advantages over the state-of-art microcontrollers. The inability to answer these questions thwarts the industrial interest. This white paper provides a brief guide to relevant metrics for fair benchmarking of neuromorphic inference accelerator ASICs, aiming to help compare different hardware approaches for various use-cases.

The paper is organized as follows: Section 1.2 provides an overview of the state-of-the-art benchmarking of inference accelerators at algorithm and hardware levels. Then we look specifically at the KPIs which are applicable to neuromorphic or power-sensitive applications, explaining what influences the metrics. Section 1.3 explains why combining KPIs for both hardware and algorithms is essential for fair benchmarking of neuromorphic computing. Finally, Section 1.4 summarizes and concludes the paper.

1.2 State-of-the-art in Benchmarking

Benchmarking of NNs inference performance for a task occurs at both the algorithm and hardware levels. The use-case provides the constraints and optimizations to be achieved through the combination of the ML model and the hardware. Currently, ML algorithms and hardware are usually benchmarked independently with their own metrics.

For ML algorithms, task-related metrics are the standard. Usually, the task-related metrics are independent of the nature of the ML model used, allowing the comparison between the algorithmic techniques used to perform the task: while the algorithm may change, the way to assess the performance of the algorithm on a certain task (e.g., image classification) remains the same. This methodology allows rapid development of deep learning techniques by comparing the performance of the algorithms on a given task. In order to target resource-limited IoT applications, metrics measuring the complexity of the model exist, such as the number of parameters, sparsity, depth, and (floating-point) operation counts, are taken into account. These KPIs are measurable via simulation of the model, and most of the current deep learning libraries now provide functions that report these KPIs.

On the other hand, hardware KPIs are extracted from the deployment platform while running a certain algorithmic model. They can be either simulated or computed by running the target application on the device. These KPIs usually include power consumption (estimation), latency, and memory metrics. In other words, they provide performance results of an ML algorithm for a certain use case on a specific hardware platform. This gives a good representation of how a single device works for a given use-case but makes benchmarking difficult. In the following sections, we present the current state-of-the-art solutions to benchmarking software and hardware with a focus on low-power devices. A summary of the standard KPIs is given in Table 1.1.

Table 1.1 Relevant KPIs for tasks, models and hardware domains. We also mention some combined KPIs to illustrate the inter-dependency of the domains.

	Metric	Definition	Unit
Task KPIs	Objective function	Determines and measures the goal of a given task	-
	(Balanced) Accuracy	Computes the ratio of correctly classified examples over the dataset (weighted by class occurrences)	%
	Precision	Computes the ratio of correctly classified samples per class	%
	True Positive Rate (TPR) / Sensitivity / Recall	Ratio of true positives over the total number of samples	%
	True Negative Rate (TNR) / Specificity	Ratio of true negatives over the total number of samples	%
	False Positive Rate (FPR)	Ratio of false positives over the total number of samples	%
	Mean average precision (mAP)	The mean of the average precision per sample	%
	F1-Score	Harmonic mean of the precision and recall	-
	Receiver operating characteristic (ROC)	Plot the true positive rate against the false positive rate of a class	-
	Area under the curve (AOC)	The area under the ROC curve	-
	Metric	**Definition**	**Unit**
Model KPIs	Complexity	Number of multiply-accumulate (multiply-additions) operations	MACs (MAdds)
		Number of (floating-point) operations, often assumed equivalent to 2x MACs	(FL)OPs
	Parameters	Total number of parameters in the model (weights, biases, etc.)	-
	Precision	Precision of the parameters and activations (floating point, integer, etc.)	bits
	Structure	Number and type of layers (and neurons)	
	Sparsity	Ratio of sparse values	%
	Maximum Activation	Maximum activation buffer of any layer in the network	
	Spike Count	Number of spikes emitted	-
	Spike rate / Spike Frequency	Rate at which the spikes are emitted (model level or neuron level)	Hz
	SynOps	Number of synaptic operations produced by the model at inference	-
	Metric	**Definition**	**Unit**
Hardware KPIs	(Idle/Peak) Power consumption	Power consumption of the system in idle/peak mode	Watts (W)
	(Peak) Number of operations	Number of operations per second (peak over short period of time)	TOPS (TOPs/s)
	Die size	Silicon area of the system	mm^2
	Memory size	On chip memory size	MB
	Memory bandwidth	Maximum data rate from memory (internal/external)	bit/s
	Energy per operation	Energy to perform one operation including necessary memory transfers	J/op
	Energy per synaptic operations	Energy to perform one synaptic operation including necessary memory transfers	J/SynOps
	Mean energy per spike	Energy required to execute one spike	J/Spike
	Energy efficiency	Tera operations per second per watt	TOPS/W
	Area efficiency	Tera operations per mm^2	$TOPS/mm^2$
	Precision	Precision of the parameters and activations (floating point, integer, etc.)	bits
	Maximum network size	Size constraints of network architecture (number of neurons/synapses, etc.)	
	Max. Core Frequency	Maximum frequency of the hardware	MHz
	Metric	**Definition**	**Unit**
Combined	Wake-up time	Time to load the model in memory and make it ready for inference	ms
	Latency	Time needed to perform inference	ms
	Inference rate / Throughput	Number of inferences per second	Inference/s
	Energy per inference	Energy consumed by the system when running an inference for a certain model	Joules (J)

1.2.1 Machine Learning

Machine learning techniques, and especially deep learning algorithms, are engineered iteratively for a given task's performance. ML algorithms are typically compared in terms of accuracy for a given task, such as segmentation or classification on a specified dataset. The task performance comparison is nowadays well established in the ML community. For classification tasks, accuracy, precision, recall, receiver operating characteristics (ROC), and area under the curve (AUC) are some of the most frequently used metrics. A typical example of a table is shown in Table 1.2. We refer the reader to [2, 3, 4] for a more detailed overview of relevant metrics in ML tasks.

In order to give fair comparison for different domains of deep learning, training and test datasets have been established. According to PapersWithCode [6], computer vision-related tasks have the largest number of datasets, with long-established quasi-standards such as CIFAR [7], ImageNet [8], and COCO [5]. Specific computer vision tasks have their own standard datasets, such as KITTI [10] for autonomous driving and FDDB [11] and WIDER Face [12] for face detection applications. Natural Language Processing (NLP) tasks are the second most popular tasks for machine learning, with near 2000 datasets comprising GLUE [13] and SQuAD [14] benchmarks. Audio, biomedical and physics-related tasks equally have their own datasets. It should be mentioned that other ML techniques also have their own equivalent dataset for example reinforcement learning (RL) tasks also have their own standard benchmarks e.g. OpenAI Gym [15] which contains a set of tasks to test reinforcement learning algorithms. Here the tasks take place in a virtual environment, and all the physics and interactions are handled by the environment.

Table 1.2 Accuracy (Acc) for different object detection settings on COCO test-dev. Adapted from [9].

Model	Acc	Acc_{50}	Acc_{75}	Acc_S	Acc_M	Acc_L
YOLOv2	21.6	44.0	19.2	5.0	22.4	35.5
SSD513	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet (ours)	39.1	59.1	42.3	21.8	42.7	50.2

The importance of the data set

The importance of the datasets can clearly be seen when looking at SNNs. Currently, the performance of SNNs does not reach DNN performance. Research in SNNs has focused on the structure of the network and learning algorithms rather than on task performance. Thus, the work used well-known datasets for DNNs and transformed them into event-based versions, such as MNIST-DVS, N-MNIST, and N-Caltech101[16]. Only recently, with the technology of event-based cameras, have SNN been applied to adapted datasets for various use-cases (e.g., DVS128[17] and TIDIGITS[18]). These new datasets will now allow us to see if SNNs can truly rival their DNN counterparts.

The standard ML benchmarking, as discussed above, usually focuses on accuracy. This means that the resources needed due to the underlying algorithm complexity, and thus power consumption, are ignored. In resource-constrained use cases such as those in edge ML, the models are designed to provide a computational advantage. For resource-constrained systems assessing the algorithmic performance on a target task, algorithms can be compared in terms of complexity, which determines the runtime constraints. In classical machine learning, there are well-established metrics for comparing the complexity of algorithms. For example, decision trees are defined by the number of nodes and depth of the tree [19]. NNs, on the other hand, are usually compared in terms of number of parameters or number of MAC operations [20, 21, 22]. We refer the reader to the survey by Hu et al. [23] for further discussion about model complexity. Table 1.3 shows a classic representation of results for an edge ML algorithm, taking into account the resources used:

In low power systems, the number of operations, multiply-accumulate (MAC), or multiply-add (MAD) are also used as an NN optimization parameter. The computation latency of an arithmetic block is also highly dependent

Table 1.3 Representation of resource-constrained KPIs, adapted from [20].

Network	mAP	Params	MAdds	CPU inference time
SSD300	23.2	36.1M	35.2B	-
SSD512	26.8	36.1M	99.5B	-
YOLOv2	21.6	50.7M	17.5B	-
MNetV1+SSDLite	22.2	5.1M	1.3B	270ms
MNetV2+SSDLite	22.1	4.3M	0.8B	200ms

on the precision used to represent the weights and activation of the NN (i.e., 8bit computations usually run at higher frequencies than for 32bits). For tiny devices, the type and number of layers of neural networks may be a metric of interest, as some hardware may be optimized for certain architectures: some platforms support separable convolutions, while others do not. The maximum supported activation size for a network layer can also be a limiting factor since some models might exceed this constraint for some embedded platforms.

Standard SNN topologies have also been compared using frameworks [24]. Among the metrics that can be used to compare SNN models, the type of neurons and synapses, the number of emitted spikes and synaptic operations, and the rate of the SNNs are the most often used.

It remains difficult, however, to compare cross-paradigm algorithms, especially when comparing deep learning with emerging paradigms like SNNs. While some efforts have been made to compare ANN and SNNs [25], a standard set of metrics has still to be defined.

1.2.2 Hardware

An increasing number of hardware evaluation tools aim at benchmarking ML applications directly on the hardware. For example, QuTiBench [37] presents a benchmarking tool that takes algorithmic optimization and co-design into account. The MLMark[27] benchmark targets ML applications running on MCUs at the edge. However, both QuTiBench and MLMark models are too large for tiny applications and require large memories, which are not available on tiny edge devices. TinyMLPerf [28] provides benchmarks for tiny systems based on imposed models and tasks, yielding the latency and speed-related KPIs. Submission of results using other network architectures is allowed in its open division. Further tools, like SMAUG [29], MAESTRO[30] and Aladdin[31], provide software solutions to emulate workloads on deep-learning accelerators using varying topologies.

The power consumption of edge ML processing hardware is of utmost interest as it directly impacts the battery lifetime of a system. Dynamic power dominates in most high-throughput applications, while leakage power is only significant in low duty cycle modes[32], where power gating, body biasing, and voltage scaling techniques are employed to reduce leakage. Peak power consumption corresponds to the maximum power consumption measured, which becomes relevant for battery- or energy harvesting-supplied applications.

The throughput metric indicates the number of operations that the hardware can perform per second, while latency is the time needed to perform an entire inference. Note that the peak throughput can usually not be reached for all network topologies, and latency does not directly scale with parallelization, as the peak throughput does[33]. Thus, latency is a combined HW/SW metric. It can be measured by running multiple inferences and afterward averaging the execution time. All parameters to run the inference should be loaded before measuring the inference time.

The CMOS technology employed for the hardware design impacts the die size and the area efficiency, and thus also directly determines its cost. Area efficiency provides a figure of merit between the throughput, limited by hardware resources and frequency, that can be achieved per area. On-chip memory size provides a raw estimation of the number of parameters of the NN that can be stored on the chip. In a multi-core architecture, usually, both the number of neurons and number of synapses per core are given.

Energy efficiency refers to the throughput that can be achieved per watt, which is equivalent to the number of operations per Joule. For obtaining this KPI, a NN is deployed to an inference accelerator, while execution time and power consumption are measured for performing inference. In the case of NNs, the multiply and accumulate (MAC) operation corresponds to two operations. Note that the bit precision of each operation directly impacts both the accuracy and the energy efficiency (e.g., 32bits float versus 8bits integer) and must therefore be carefully traded off. Energy per operation and energy per neuron are fair metrics if the bit resolution is provided since they are independent of the NN algorithm employed and therefore only hardware-related.

Some hardware only supports a limited number of layers and layer types with restricted dimensions. Others provide optimizations and specialized units. These optimizations, while not being directly comparable, have a strong impact on the hardware KPIs. Furthermore, power consumption is influenced by the core voltage supply, which depends on the CMOS technology used for the hardware design. Thus, the energy efficiency metric (TOPS/W) can be misleading unless all hardware restrictions are known. The same applies to other representations like GOPS/W. Typical display of performance in terms of OPS and associated power are presented in Table 1.4. and from these

Table 1.4 Typical display of performance comparison of neuromorphic hardware platforms, adapted from [34].

Accelerator	Type	Target application	Performance
NVIDIA Jetson Nano	GPU	Embedded	472 GOPS @ 5 – 10 W
Nvidia Jetson TX2	GPU	Edge	1,3 TOPS @ 7,5 W
NVIDIA Jetson AGX Xavier	GPU	Edge	30 TOPS @ 30 W
NVIDIA Drive AGX Pegasus	GPU	Automotive	320 TOPS
Intel Movidius Myriad 2 bzw. Myriad X	Chip	Embedded/Edge DL/Vision	4 TOPS @ 1 W (Myriad X)
MobilEye EyeQ4	Chip	Automotive	2.5 TOPS @ 3 W
GreenWaves GAP8	Chip	Battery powered AI	200 MOPS bis 8 GOPS @ <100mW
Canaan Kendryte K210	Chip	Embedded Vision & Audio	250 GOPS @ 300mW
Google Coral Edge TPU	Chip	Edge	4 TOPS @ <2,5W
Lattice sensAI Stack	Soft IP-Core	Embedded	<1 mW – 1 W
Videantis v-MP6000UDXM	Soft IP-Core	Embedded DL/Vision	<6,6 TOPS @ 400 MHz

Table 1.5 Recent display of performance comparison of neuromorphic hardware platforms, adapted from [35].

		Eyeriss	ENVISION	Thinker	UNPU	This work	
Technology		65nm	28nm	65nm	65nm	65nm	
Area		1176k gates (NAND-2)	1950k gates (NAND-2)	2950k gates (NAND-2)	4.0mm×4.0mm (Die Area)	2695k gates (NAND-2)	
On-chip SRAM (kB)		181.5	144	348	256	246	
Max Core Frequency (MHz)		200	200	200	200	200	
Bit Precision		16b	4b/8b/16b	8b/16b	1b-16b	8b	
Num. of MACs		168 (16b)	512 (8b)	1024 (8b)	13824 (bit-serial)	384 (8b)	
DNN Model		AlexNet	AlexNet	AlexNet	AlexNet	sparse AlexNet	sparse MobileNet
Batch Size		4	N/A	15	N/A	1	1
Core Frequency (MHz)		200	200	200	200	200	200
Bit Precision		16b	N/A	adaptive	8b	8b	8b
Inference/sec	(CONV only)	34.7	47	-	346	342.4	-
	(Overall)	-	-	254.3	-	278.7	1470.6
Inference/J	(CONV only)	124.8	1068.2	-	1097.5	743.4	-
	(Overall)	-	-	876.6	-	664.6	2560.3

terms the TOPS/W metric can be extrapolated. However, recent publications provide combined metrics as it is shown in Table 1.5.

Processing hardware is limited by the supported arithmetic precisions for parameters and activations, with the previously mentioned effects on accuracy. Some hardware implementations allow for several bit resolutions, allowing to dynamically trade-off throughput, memory needs, and accuracy. Generally, lower precisions lead to lower algorithmic accuracy.

1.3 Guidelines

Benchmarking of ML applications cannot be tackled as a standalone problem at the level of either only hardware or algorithms. A holistic view requires a wide range of expertise and domains. It requires a multidisciplinary and multidimensional approach considering, among other things, the hardware platform, the NN (model), and the use-case under evaluation. In order to make the right choices for building blocks, the system integrator needs to know the KPIs for a given use-case that different NNs will be able to deliver on different hardware platforms.

This section explains why a multidisciplinary approach combining both algorithms and hardware is needed to avoid drawing unfair and misleading conclusions and comparisons. In the following, we first describe what is unfair and fair benchmarking in Section 1.1, and then present a combined KPI approach and guidelines for benchmarking in sections 1.2 and 1.3.

1.3.1 Fair and Unfair Benchmarking

With the new generations of hardware accelerators, many optimizations in hardware try to co-optimize energy and performance, such as zero-skipping components, in-memory computing, and multi-core convolution units. However, it is sometimes unclear if these optimization features are correctly exploited when embedding complex deep learning models. This lack of transparency in the optimizations and embedding processes of the models results in sub-optimal deployments in the hardware. Furthermore, SDK documentation for a large number of accelerators is unclear or lacks critical content for high-level developers and data scientists to perform inference-time optimizations. This makes the embedding process and the subsequent measurements of the KPIs difficult.

Today, most models deployed on hardware are trained on GPU machines and deployed on target hardware platforms using their respective optimizations. The wide variety of optimizations employed in different hardware implementations [36,37] target specific use-cases, which might favor one or the other (benchmarking) algorithms (and the underlying layer types), further complicating fair benchmarking. Thus, there are hardware solutions that outperform others by orders of magnitude for specific tasks while providing poor performance in others. This type of benchmarking is unfair, as the models are not optimized and thus do not take advantage fully of each platform. Their KPIs are comparable, but the benchmarking is unfair with respect to the hardware, as a specially designed model for a particular platform could be more performant than another model deployed on another platform, see Figure 1.1a. This shows that use-case-agnostic benchmarking can be misleading. A platform might receive a low score with general benchmarks, while performing excellently for a hardware-tailored task.

In contrast, a fair benchmarking based on a defined use-case (independently of the model used) would exploit all the tools and optimizations provided by the constructor to exploit the hardware to its full potential. However, the results of the benchmarks can be challenging to compare, as the base model and optimizations are different between the compared hardware, see Figure 1.1b. If we compare with conventional benchmarking of processors, the benchmarks do not account for the underlying optimizations; a superscalar processor will be benchmarked against a non-superscalar processor using the same tests.

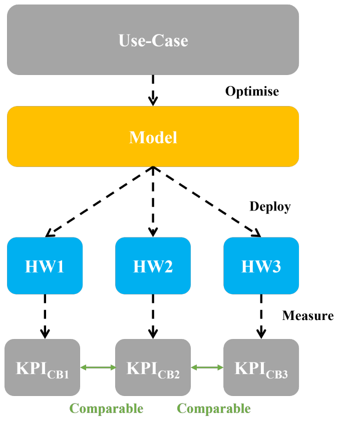
One particular aspect to take into account in the design of an inference accelerator is the selection of the CMOS technology and embedded non-volatile memory (eNVM). If eNVM is used for leveraging from the lack of power consumption for retaining the stored values after writing, the qualification of the memory by the foundry in the selected CMOS process is necessary for its industrialization and therefore a crucial criterion. The selection of the CMOS process has an impact on the cost and size of the inference accelerator IP that needs to be considered. Moreover, the CMOS process has also an impact on the active power and leakage power of the inference ASIC and needs to be part of the information provided for a fair comparison between inference accelerators fabricated in different CMOS processes.

There still remain challenges in the method of comparison. Benchmarking approaches for Von-Neumann architectures are relatively widespread and standardized [38, 39]. By contrast, clear benchmarking methodologies for non-Von-Neumann architectures do not exist yet, making them difficult to compare. In particular, neuromorphic circuit design is an emerging multidisciplinary challenge that is still in an exploratory phase making the comparison of the underlying hardware difficult due to its variety. Although many existing techniques report significantly reduced energy consumption figures, they still compare themselves to standard low-power microcontrollers.

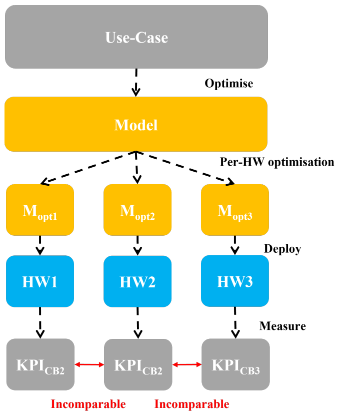
Benchmarking should be done at different stages and abstraction levels, considering various aspects such as the algorithm performance, the technical characteristics, the architectural parameters, and the flexibility and amenities hardware provides for a specific use-case. As of today, different KPI values can be obtained with the same algorithm and same hardware just by changing the use-case from always-on to event-based.

1.3.2 Combined KPIs and Approaches for Benchmarking

The application deployment KPIs are at the intersection of the performance indicators required by a given use-case, the model solving the task, and



(a) Unfair benchmarking



(b) Fair benchmarking

Figure 1.1 Benchmarking fairness. (a) Unfair benchmarking: the KPIs are comparable, but the benchmarked hardware platforms are not exploited to their full potential. (b) Fair benchmarking: the hardware platforms are exploited to their full potential, but the resulting combined KPIs (KPI$_{CB}$) are not comparable.

the hardware system on which the application is deployed, see Figure 1.2. Because of the large number of KPIs that can be reported, it is difficult to have an objective comparison between different platforms, as a platform can perform well on certain KPIs and poorly on others (e.g., simulating an SNN on a CNN accelerator). Furthermore, not all platforms report the same set of metrics and the metrics are not usually convertible to each other (e.g., energy consumption is not always relying only on MAC operations).

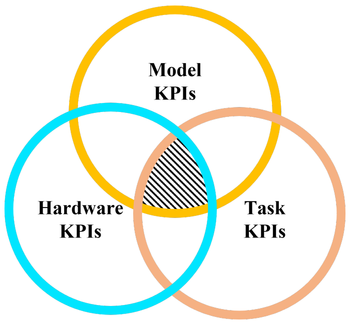


Figure 1.2 Combined KPIs for fair benchmarking

Some task-related metrics heavily depend on the use-case and application scenarios, and should be used only in these specific cases. For example, the performance of a keyword spotting algorithm should not be compared with the one of an object classification algorithm, even though both aim at high accuracy. For these reasons, a (small) set of KPIs are desirable which have the following properties:

- Orthogonality
- Reproducibility
- Objectiveness
- Use-case independence

To assess the performance of NN models running on hardware for a certain use-case, the KPIs should be combined, as shown in Table 1, to express the performance of the application on the hardware platform. In this regard, Fra et al. [40] have proposed a multi-metric approach taking into account: 1) accuracy, 2) number of parameters of the NN, 3) memory footprint in MB. These three metrics provide an overview of the NNs: which one provides better results in the classification task and which one has a smaller memory footprint. Further metrics which should now be taken into consideration are: 4) Energy consumption per inference, 5) the number of operations per second.

The resulting KPIs of the deployment could also contain an indicator about the flexibility of the hardware accelerator. For comparison in terms of flexibility, it is necessary to indicate the supported layer types, the supported bit resolution for inputs, parameters and activation functions, and the sizes of the kernel filters. By combining metrics that depend on the NN algorithm and the hardware, a fair comparison for a use-case can be achieved if the number of parameters of the NN is optimized and the dataset employed is the same.

1.3.3 Outlook : Use-case Based Benchmarking

A solution to the afore-mentioned challenge would be to propose a use-case-dependent benchmarking that does not rely at all on the model architectures of the given model. For an industrial setting, it is interesting to obtain high performance independently of the techniques used. What matters is that the application performs within the given constraints of the use-case.

A solution is illustrated in Figure 1.3. In this paradigm, a use-case would be defined by some target KPIs to reach, such as minimum accuracy

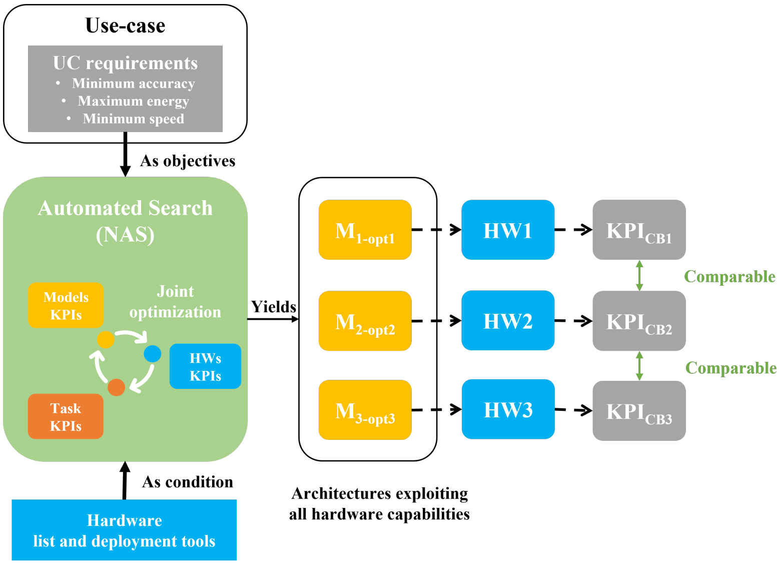


Figure 1.3 Benchmarking pipeline based on use-cases. An automated search finds the best possible model exploiting the performance offered by each target hardware platforms. The resulting combined KPIs are comparable.

and maximum energy. To benchmark the hardware, an automated search technique, such as Network Architecture Search (NAS), would try to find the model that fits the target hardware and then optimize the model further to improve the latency or memory use. This type of benchmarking would be use-case dependent and model agnostic, beside the meta-model composing the automated search. Such benchmarking method would output comparable (combined) KPIs, making the comparison of hardware and the selection of the best one possible. Of course, an extensive benchmarking suite covering several use-cases (audio-based, image-based, classification, regression, etc.) is necessary to ensure fairness across domains.

Following the methodology presented, there are some guidelines to follow in order to ensure that the extracted KPIs respect the properties presented in the section 1.2. In addition to measuring the combined KPIs, it is necessary to provide information on the entire deployment pipeline. Indeed, the KPIs related to the solved task, the (final) model deployed on the hardware, the characteristics of the hardware, and finally, the combined KPIs based on the previous information can be calculated.

The use-cases should be clearly defined and cover several machine learning tasks. Although the methodology can be applied to a single use-case to compare a few hardware platforms, the industrial application cases are generally broad. It is, therefore, preferable to select a neuromorphic platform that offers the best performance for a wide range of tasks. This can only be achieved with a benchmarking tool that is diversified in terms of the tasks to be solved.

The methodology also requires a complete software tool chain to have rapid and reproducible deployments of the NNs on the hardware. Quantization-aware training tools or even better hardware-aware training tools compatible with the target hardware platforms are beneficial. The efficient execution of algorithms does not only depend on the hardware architecture, like the processing resources, but equally on an efficient mapping strategy that schedules the hardware resources for high throughput and low power consumption. Depending on the architecture, algorithm-to-hardware compilers or on-board schedulers ensure this optimization.

Finally, adequate documentation about the hardware technology, the search algorithm used for benchmarking, the use-case realized by the benchmark, and the interpretation of the results provided by the benchmark is necessary to empower the user in its selection of the most suitable hardware platform.

1.4 Conclusion

In this paper, we have summarized the standard techniques for benchmarking NN accelerator hardware and ML software, in addition, we have specified the KPIs that are most relevant for resource aware inference. We have through example shown that, in ultra-low-power or neuromorphic systems, separating hardware and ML algorithms and use-case parameters leads to an ineffective means of comparison. Only when considering these three in a holistic manner, can system be benchmarked. Integrating KPIs that allow benchmarking at the system level in this way is complex. It is important to do this as the inability to benchmark the IoT systems today is reducing the uptake by industry. In this paper, we have proposed a benchmarking methodology based on use-cases where the ML algorithm is adapted to the hardware to allow fair comparison. Finally, we provide a guideline on what aspects are important to take into account while developing such benchmarking tool to ensure that the resulting KPIs are comparable.

Acknowledgements

This work is supported through the project ANDANTE. ANDANTE has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 876925. The JU receives support from the European Union's Horizon 2020 research and innovation programme and France, Belgium, Germany, Netherlands, Portugal, Spain, Switzerland. ANDANTE has also received funding from the German Federal Ministry of Education and Research (BMBF) under Grant No. 16MEE0116. The authors are responsible for the content of this publication.

References

[1] M. Davies. Benchmarks for progress in neuromorphic computing. *Nature Machine Intelligence*, 1(9):386–388, 2019.

[2] B. J. Erickson and F. Kitamura. Magician's corner: 9. performance metrics for machine learning models. *Radiology: Artificial Intelligence*, 3(3), 2021.

[3] A. Rácz, D. Bajusz, and K. Héberger. Multi-level comparison of machine learning classifiers and their performance metrics. *Molecules*, 24(15), 2019.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[5] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[6] https://paperswithcode.com. Website, 2021.

[7] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[9] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[10] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.

[11] V. Jain and E. Learned-Miller. Fddb: A benchmark for face detection in unconstrained settings. Technical report, UMass Amherst technical report, 2010.

[12] S. Yang, P. Luo, C.-C. Loy, and X. Tang. Wider face: A face detection benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5525–5533, 2016.

[13] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[14] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

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

[16] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in neuroscience*, 9:437, 2015.

[17] A. Amir, B. Taba, D. Berg, T. Melano, J. McKinstry, C. Di Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza, et al. A low power, fully event-based gesture recognition system. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7243–7252, 2017.

[18] J. Anumula, D. Neil, T. Delbruck, and S.-C. Liu. Feature representations for neuromorphic audio spike streams. *Frontiers in neuroscience*, 12:23, 2018.

[19] H. Buhrman and R. De Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288(1):21–43, 2002.

[20] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[21] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of*

the European conference on computer vision (ECCV), pages 116–131, 2018.

[22] M. Tan and Q. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[23] X. Hu, L. Chu, J. Pei, W. Liu, and J. Bian. Model complexity of deep learning: A survey. *Knowledge and Information Systems*, 63(10):2585–2619, 2021.

[24] S. R. Kulkarni, M. Parsa, J. P. Mitchell, and C. D. Schuman. Benchmarking the performance of neuromorphic and spiking neural network simulators. *Neurocomputing*, 447:145–160, 2021.

[25] S. Narduzzi, S. A. Bigdeli, S.-C. Liu, and L. A. Dunbar. Optimizing the consumption of spiking neural networks with activity regularization. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 61–65. IEEE, 2022.

[26] M. Blott. *Benchmarking Neural Networks on Heterogeneous Hardware*. PhD thesis, Trinity College, 2021.

[27] P. Torelli and M. Bangale. Measuring inference performance of machine-learning frameworks on edge-class devices with the mlmark benchmark. *Techincal Report. Available online: https://www. eembc. org/techlit/articles/MLMARK-WHITEPAPERFINAL-1. pdf (accessed on 5 April 2021)*, 2021.

[28] C. R. Banbury, V. J. Reddi, M. Lam, W. Fu, A. Fazel, J. Holleman, X. Huang, R. Hurtado, D. Kanter, A. Lokhmotov, et al. Benchmarking tinyml systems: Challenges and direction. *arXiv preprint arXiv:2003.04821*, 2020.

[29] S. Xi, Y. Yao, K. Bhardwaj, P. Whatmough, G.-Y. Wei, and D. Brooks. Smaug: End-to-end full-stack simulation infrastructure for deep learning workloads. *ACM Transactions on Architecture and Code Optimization (TACO)*, 17(4):1–26, 2020.

[30] H. Kwon, P. Chatarasi, M. Pellauer, A. Parashar, V. Sarkar, and T. Krishna. Understanding reuse, performance, and hardware cost of dnn dataflows: A data-centric approach using maestro. 2020.

[31] Y. S. Shao, B. Reagen, G.-Y. Wei, and D. Brooks. Aladdin: A pre-rtl, power-performance accelerator simulator enabling large design space exploration of customized architectures. In *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, pages 97–108. IEEE, 2014.

[32] F. Fallah and M. Pedram. Standby and active leakage current control and minimization in cmos vlsi circuits. *IEICE transactions on electronics*, 88(4):509–519, 2005.

[33] J. Hanhirova, T. Kämäräinen, S. Seppälä, M. Siekkinen, V. Hirvisalo, and A. Ylä-Jääski. Latency and throughput characterization of convolutional neural networks for mobile computer vision. In *Proceedings of the 9th ACM Multimedia Systems Conference*, pages 204–215, 2018.

[34] M. Breiling, R. Struharik, and L. Mateu. Machine learning: Elektronenhirn 4.0. 2019.

[35] Y.-H. Chen, T.-J. Yang, J. Emer, and V. Sze. Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(2):292–308, 2019.

[36] P. Jokic, E. Azarkhish, A. Bonetti, M. Pons, S. Emery, and L. Benini. A construction kit for efficient low power neural network accelerator designs. *arXiv preprint arXiv:2106.12810*, 2021.

[37] M. Blott, L. Halder, M. Leeser, and L. Doyle. Qutibench: Benchmarking neural networks on heterogeneous hardware. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 15(4):1–38, 2019.

[38] EMBCC ULPMark: https://www.eembc.org/ulpmark/. Website, 2021.

[39] EMBCC CoreMark: https://www.eembc.org/coremark/. Website, 2021.

[40] V. Fra, E. Forno, R. Pignari, T. Stewart, E. Macii, and G. Urgese. Human activity recognition: suitability of a neuromorphic approach for on-edge aiot applications. *Neuromorphic Computing and Engineering*, 2022.