Benchmarking and Characterization of event-based Neuromorphic Hardware

Christoph Ostrau, Christian Klarhorst, Michael Thies, Ulrich Rückert

February 8, 2019

We present the modular framework *SNABSuite* (**S**piking **N**eural **A**rchitecture **B**enchmark **Suite**) for "black-box" benchmarking of neuromorphic hardware systems and spiking neural network software simulators. The motivation for having a coherent collection of benchmarks is twofold: first, benchmarks evaluated on different platforms provide measures for direct comparison of performance indicators (e.g. resource efficiency, quality of the result, robustness ...). By using the platforms as they are provided for possible end-users and evaluating selected performance indicators, benchmarks support the decision for or against a system based on use-case requirements. Second, benchmarks may reveal opportunities for effective improvements of a system and can contribute to future development. Systems like the Heidelberg BrainScaleS-project [2], IBM TrueNorth [3], the Manchester SpiNNaker chip [4] or the Intel Loihi platform [5] drive the evolution of neuromorphic hardware implementations, while comparable benchmarks and corresponding measures are still rare. The problem of "comparable" measures can be addressed in two ways: concerning application driven measures like classification accuracy it may be advantageous to implement these algorithms in a highly specialized manner to get the best result

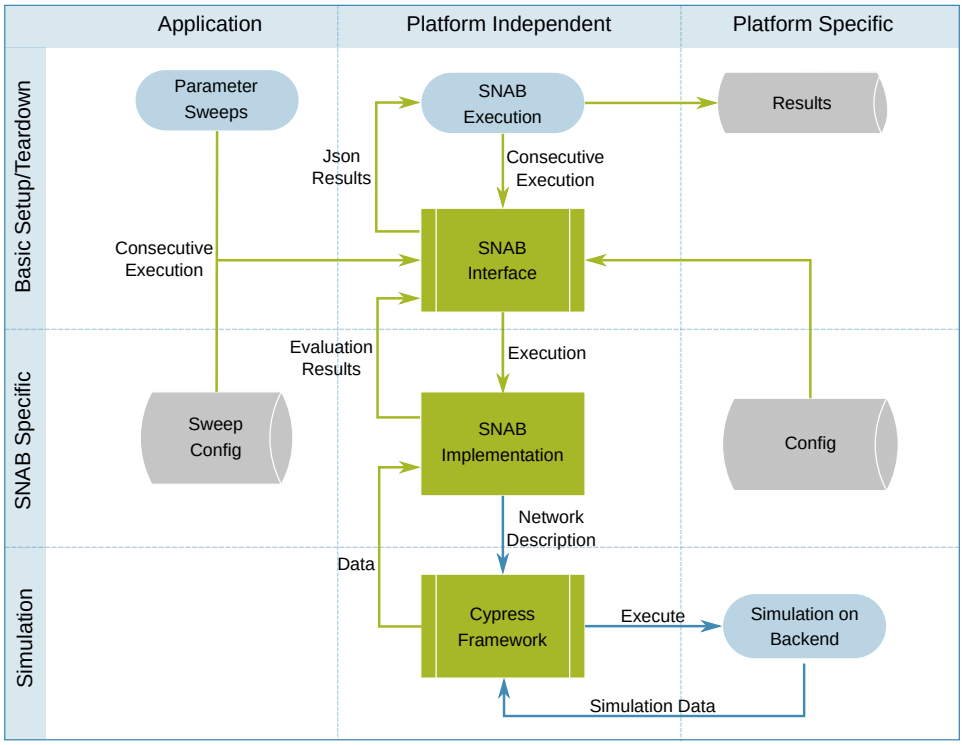

Figure 1: Flow chart of the benchmark framework *SNABSuite*. Blue arrows represent information flow of the Cypress framework (introduced in [1]), green arrows depict the flow of benchmark results and configuration data. The diagram distinguishes between platform-specific, benchmark-specific and fully reusable parts to highlight, which software components have to be provided when extending the suite.

on every hardware (see e.g. [6, 7, 8, 9]). This approach reveals what can be achieved on a specific backend, but will not necessarily uncover differences or deficiencies, as these are worked around in the specialization process. In addition, most of these benchmarks are published by the hardware maintainers themselves, and thus are meant to promote a certain platform, which in some cases, has been adapted to this specific task. In this case, results might not reflect the state of the platform that is visible to possible end-users. Despite these problems, most of these citations compare a single hardware platform to a software simulator, which renders platform-overarching comparisons complicated. One alternative is a "black-box" approach, where benchmark networks are developed independent of the hardware on which they will be executed.

Our benchmark framework, which is shown in Fig. 1, supports the second approach. Target platforms vary quite significantly in their implementation with the only commonality being the execution of spiking (event-based) neural networks. For this reason, SNABSuite only uses spiking networks for performance estimation. These networks are specified as an abstract network description, which is automatically converted to a backend specific implementation at runtime. Nevertheless, individual backend constraints need to be considered (like supported neuron models, neuron parameter restrictions and chip sizes). Hence, these benchmark parameters are factored out into external configuration files, which are consulted during the benchmark setup (compare Fig. 2). All benchmarks share a common interface, which allows embedding the benchmarks into different evaluation strategies (for e.g. parameter space exploration). This modular approach for providing a benchmark framework simplifies adding new benchmarks: the struggle of utilizing different backends with different interfaces is completely abstracted away while network description is handled in a declarative and intuitive way. Detailed information about the backends is only necessary for creating configuration files. Furthermore, new simulation platforms can be added without altering existing benchmarks or the core of the framework itself.

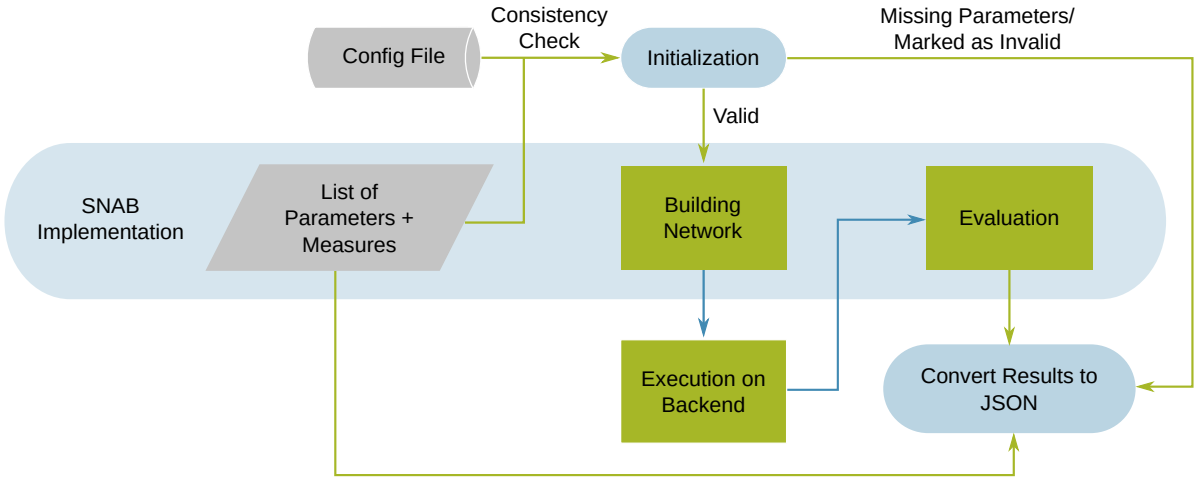

Figure 2: Flow chart for the execution of a single benchmark: after an initialization phase that checks the validity of configuration files, the network is constructed by the abstract Cypress framework. The network is executed and evaluated, before results are stored in a JSON-based database.

Benchmarks are partitioned in three different categories with the aim to cover a broad range of measures and to avoid too strong specialization of the benchmark suite (see Fig. 3). Thus, the application level is only one part of the suite: results from applications alone do not allow to extrapolate. Instead, we also go for sub-tasks and low-level benchmarks, which characterize the hardware/software platform and, as a result, allow to estimate the performance of an application, which is not (yet) part of the suite, on the platform under consideration. As discussed above, low-level benchmarks are referring to simulations based on a network description. For example, we measure the output/readout bandwidth by using a set of steadily spiking neurons. If the output rates of the network are varying largely between neurons it is most likely that a bandwidth bottleneck has been reached. This performance indicator provides an upper limit for the spike rates that can be expected on the target platform. For application inspired sub-tasks (compare Fig. 3), we use networks that perform elementary tasks to benchmark general performance of an archetypical network architecture on a given system. For example, the

Winner-Takes-All architecture is a commonly used paradigm for spiking network architectures (e.g. [10, 11]). A simple Winner-Takes-All network consisting of at least two competing populations (group of neurons), with only one population being active at a time, can be used to benchmark how good this architecture can be realized on the hardware. It requires sub-task specific quality measures to evaluate the performance. At last, the application level is for instance represented by an application from the domain of constraint-satisfaction problems: solving Sudokus. Here, we apply the Winner-Takes-All architecture and record, how many of 100 assorted Sudokus of varying difficulty were solved. These three proposed benchmarks are connected: the maximal output rate of neurons may be restricted, limiting the maximal activity of a winning population of neurons. This restrains the activity level difference of populations and thus might impede the detection of a winner. Furthermore, if the Winner-Takes-All network is imperfectly realized by e.g. dead or overly active neurons, winning probability is influenced, and hence a solution to the Sudoku problem might not be found. Without seeing the results for the Sudoku problem, we can already infer that Winner-Takes-All networks will yield a reduced performance on these particular backends.

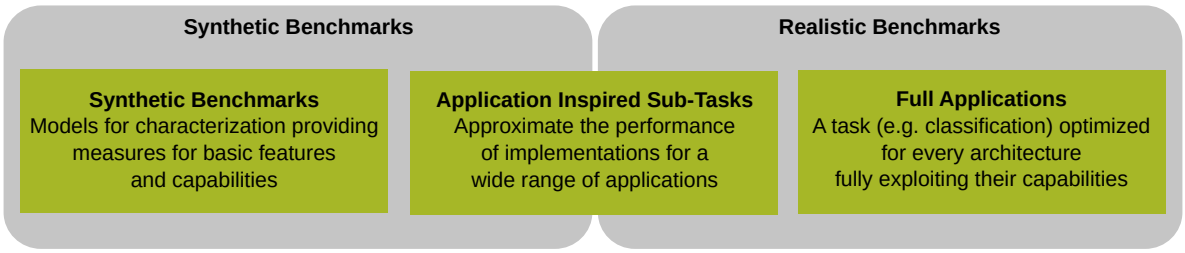

Figure 3: Benchmarks are separated into 3 categories.

The proposed approach provides the right level of configurability while still using the same network architecture and sets us in position to compare systems of very different nature, as e.g. the SpiNNaker and BrainScaleS systems. While the SpiNNaker system, which is a digital neuromorphic processor compound, provides some flexibility in models and architectures, the BrainScaleS system is a mixed-signal architecture with a single parameterized analog neuron and synapse model and a digital communication fabric. The behavior of these analog parts is affected by different sources of variations and noise [2], which have to be accounted for in the network design and/or parameter choice. Yet, benchmark measures have to be comparable across platforms, and thus the architecture of the network has to be fixed. This may result in some benchmarks not being applicable on certain platforms, or the implementation has to be adapted to the platform, which is then considered a different benchmark. Consequently, it is infeasible to provide one unified benchmark score covering all applications, instead benchmark results consist of a list of different measures which are interpreted individually.

Currently, our framework supports the above mentioned BrainScaleS system, the Spikey [12] and the SpiNNaker chip through their PyNN [13] interfaces. On the software simulator side, NEST [14] is supported and accessed via its PyNN [15] interface or more directly via its SLI interface [14]. Furthermore, an initial set of characterizations and benchmarks is ready for use and we will present first results.

The overall goal of this work is to provide a set of measures, which helps external users (including researchers from outside of neuromorphic computing) to select their target platform for a specific application. Our framework renders development, execution and reuse of benchmarks comparably simple, as well as extending of the benchmark suite. The application domain is not yet completely covered, but the set of benchmarks will be extended constantly, while we also add backends for new platforms as e.g. Loihi [5] and the GPU-accelerated GeNN [16] platforms.

Funding/Acknowledgments

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7) under grant agreement no 604102 and the EU's Horizon 2020 research and innovation programme under grant agreements No 720270 and 785907 (Human Brian Project, HBP). It has been further supported by the Cluster of Excellence Cognitive Interaction Technology "CITEC" (EXC 277) at Bielefeld University, which is funded by the German Research Foundation (DFG).

References

[1] Andreas Stöckel et al. "Binary associative memories as a benchmark for spiking neuromorphic hardware". In: *Frontiers in computational neuroscience* 11 (2017), p. 71.

[2] Mihai A. Petrovici et al. "Characterization and compensation of network-level anomalies in mixed-signal neuromorphic modeling platforms". In: *PloS one* 9.10 (2014), e108590.

[3] Andrew S. Cassidy et al. "Cognitive computing building block: A versatile and efficient digital neuron model for neurosynaptic cores". In: *Neural Networks (IJCNN), The 2013 International Joint Conference on*. IEEE. 2013, pp. 1–10.

[4] Eustace Painkras et al. "SpiNNaker: A 1-W 18-core system-on-chip for massively-parallel neural network simulation". In: *Solid-State Circuits, IEEE Journal of* 48.8 (2013), pp. 1943–1953.

[5] Mike Davies et al. "Loihi: A neuromorphic manycore processor with on-chip learning". In: *IEEE Micro* 38.1 (2018), pp. 82–99.

[6] Alan Diamond, Thomas Nowotny, and Michael Schmuker. "Comparing Neuromorphic Solutions in Action: Implementing a Bio-Inspired Solution to a Benchmark Classification Task on Three Parallel-Computing Platforms". In: *Frontiers in neuroscience* 9 (2015).

[7] James Courtney Knight and Thomas Nowotny. "GPUs outperform current HPC and neuromorphic solutions in terms of speed and energy when simulating a highly-connected cortical model". In: *Frontiers in neuroscience* 12 (2018), p. 941.

[8] Sacha J van Albada et al. "Performance comparison of the digital neuromorphic hardware SpiNNaker and the neural network simulation software NEST for a full-scale cortical microcircuit model". In: *Frontiers in neuroscience* 12 (2018).

[9] Timo Wunderlich et al. "Demonstrating Advantages of Neuromorphic Computation: A Pilot Study". In: *arXiv preprint arXiv:1811.03618* (2018).

[10] Raphaela Kreiser et al. "Pose Estimation and Map Formation with Spiking Neural Networks : towards Neuromorphic SLAM". In: (2018), pp. 2159–2166.

[11] Gabriel A. Fonseca Guerra and Steve B. Furber. "Using Stochastic Spiking Neural Networks on SpiNNaker to Solve Constraint Satisfaction Problems". In: *Frontiers in Neuroscience* 11.December (2017). URL: `http://journal.frontiersin.org/article/10.3389/fnins.2017.00714/full`.

[12] Thomas Pfeil et al. "Six networks on a universal neuromorphic computing substrate". In: *Frontiers in Neuroscience* 7 (2013), p. 11.

[13] Andrew Davison et al. "PyNN: a common interface for neuronal network simulators". In: *Frontiers In Neuroinformatics, 2009, 2, 11* (2009).

[14] Marc-Oliver Gewaltig and Markus Diesmann. "NEST (NEural Simulation Tool)". In: *Scholarpedia* 2.4 (2007), p. 1430.

[15] Jochen Martin Eppler et al. "PyNEST: a convenient interface to the NEST simulator". In: *Frontiers In Neuroinformatics, 2009, 2, 12* (2009).

[16] Esin Yavuz, James Turner, and Thomas Nowotny. "GeNN: a code generation framework for accelerated brain simulations". In: *Scientific reports* 6 (2016).