Benchmarking Deep Spiking Neural Networks on Neuromorphic Hardware

Christoph Ostrau(✉), Jonas Homburg, Christian Klarhorst, Michael Thies, and Ulrich Rückert

Technical Faculty, Bielefeld University, Bielefeld, Germany
costrau@techfak.uni-bielefeld.de

Abstract. With more and more event-based neuromorphic hardware systems being developed at universities and in industry, there is a growing need for assessing their performance with domain specific measures. In this work, we use the methodology of converting pre-trained non-spiking to spiking neural networks to evaluate the performance loss and measure the energy-per-inference for three neuromorphic hardware systems (BrainScaleS, Spikey, SpiNNaker) and common simulation frameworks for CPU (NEST) and CPU/GPU (GeNN). For analog hardware we further apply a re-training technique known as hardware-in-the-loop training to cope with device mismatch. This analysis is performed for five different networks, including three networks that have been found by an automated optimization with a neural architecture search framework. We demonstrate that the conversion loss is usually below one percent for digital implementations, and moderately higher for analog systems with the benefit of much lower energy-per-inference costs.

Keywords: Spiking neural networks · Neural architecture search · Benchmark

1 Introduction

Diverse event-based neuromorphic hardware systems promise the accelerated execution of so called spiking neural networks (SNN), also referred to as the third generation of neural networks [14]. The most prominent representatives of this class of hardware accelerators include the platforms Braindrop [16], BrainScaleS [22], DYNAPs [15], Loihi [5], SpiNNaker [8] and Truenorth [1]. With the diversity of hardware accelerators comes a problem for potential end-users: which platform is suited best for a given spiking neural network algorithm, possibly respecting inherent resource requirements for embedding in mobile robots or smart devices. Usually, this question is answered by evaluating a set of benchmarks on all qualified systems, which measure the state-of-the-art and quantify progress in future hardware generations (see e.g. [4])). Here, we face two major challenges with neuromorphic hardware. First, there is no universal interface to all hardware/software simulators despite some projects like PyNN [6]. Second,

I. Farkaš et al. (Eds.): ICANN 2020, LNCS 12397, pp. 610–621, 2020.
https://doi.org/10.1007/978-3-030-61616-8_49

there are quite a few promising network models and learning strategies, but still "the" algorithm for spiking neural networks is missing. One recent system overarching network is the cortical microcircuit model [2,13]. A follow-up publication [21] shows, how this benchmark has driven platform specific optimization that, in the end, improves the execution of various networks on the SpiNNaker platform confirming the value of benchmarks. However, it is also an example of a platform specific implementation to reach maximal performance on a given system.

One commonly agreed application for spiking neural networks is the conversion of conventionally trained artificial neural networks (ANN) to rate-based SNNs [7]. Although this is not using SNNs in their most efficient way, it is a pragmatic approach that is suitable to be ported to different accelerators, independent of their nature. In this work, we use this approach for evaluating five distinct networks, either defined by hardware restrictions, by already published work, or by employing neural architecture search (NAS) with Lamarck_ML [11] to optimize the network topology. We evaluate these networks on BrainScaleS, Spikey [20], and SpiNNaker as well as the CPU simulator NEST [9] and the CPU/GPU code-generation framework GeNN [25]. Furthermore, we use a retraining approach with neuromorphic hardware-in-the-loop (HIL) proposed in [23] to unlock the full potential of the analog neuromorphic hardware systems. Section 2 outlines the target systems, the software environment, and the used methods. Section 3 presents the results, including neuron parameter optimization, and accuracy along with energy measurements for all target platforms.

2 Methods

In the following we introduce all target systems and the software environment as well as the methodology followed.

2.1 Target Systems and Software

All target systems in this work support the simulation or emulation of leaky integrate-and-fire neurons with conductance-based synapses, although especially analog systems are limited to specific neuron models. **NEST** is a scaleable software simulator suited to simulate small as well as extensive networks on compute clusters. It is used in version 2.18 [12] executed with four threads on an Intel Core i7-4710MQ mobile processor. **GeNN** [25] is a code generation framework for the simulation of SNNs. In its current release version (4.2.1)[1], it supports generating code for a single-threaded CPU simulation or for graphics processing units (GPU) supporting NVIDIA CUDA. Networks are evaluated on a NVIDIA GeForce 1080 TI GPU; runtimes are measured for networks without recording any spikes due to the overhead of getting spikes back from GPU, which effectively stops the simulation at every time step and copies the data between GPU

[1] Here, we use the most recent GeNN from github (end of April 2020).

and CPU. For this publication we make use of single precision accuracy and all simulators use a time step of 1 ms. However, NEST is using an adaptive time-step to integrate the neuron model. The fully digital many-core architecture **SpiNNaker** [8] comes in two different sizes, which are both used in this work. The smaller SpiNN3 system is composed of four chips; the larger SpiNN5 board consists of 48 chips. A single chip comprises 18 ARM968 general purpose CPU cores, with each simulating up to 255 `IF_cond_exp` neurons. The system runs in real-time, simulating 1 ms of model time in 1 ms wall clock time. SpiNNaker is used with the latest released software version 5.1.0 using PyNN 0.9.4. Finally, we make use of two mixed-signal (analog neural circuits, digital interconnect) systems: First, the **Spikey** system [20] supports the emulation of 384 neurons with 256 synapses each. The emulated neuron model is subject to restricted parameter ranges (e.g. four bit weights, limited time constants) with some parameters prescribed by the hardware (e.g. the membrane capacitance). The system runs at a speedup of 10,000, therefore taking only 0.1 ms to emulate 1 ms of model time. Second, Spikey's successor **BrainScaleS** [22] shares many of Spikey's properties. Most notably is the now fully parameterizable neuron model, as well as the usage of wafer-scale integration, combining 384 accessible HICANN chips on a single wafer for a full system. Each chip implements 512 neuron circuits with 220 synapses each, where up to 64 circuits can be combined to form a single virtual neuron, allowing more robust emulations and a higher synapse fan-in.

While all of these platforms formally support the **PyNN** API [6], the supported API versions differ between simulators impeding the portability of code. **Cypress**[2] [24] is a C++ framework abstracting away these differences. For NEST, Spikey and SpiNNaker the framework makes use of their PyNN interfaces, however, for BrainScaleS and GeNN a lower-level C++ interface is used. Furthermore, the proposed networks studied below are part of the **S**piking **N**eural **A**rchitecture **B**enchmark **Suite**[3] (SNABSuite) [17,18], which also covers benchmarks like low-level synthetic characterizations and application-inspired (sub-)tasks with an associated framework for automated evaluation.

Energy measurements have been taken with a Ruideng UM25C power meter (SpiNNaker, Spikey), with a PeakTech 9035 for CPU simulations, or with the NVIDIA `smi` tool. There is no possibility for remote energy measurements on the BrainScaleS system. Thus, the values have been estimated from the number of pre-synaptic events using published data in [23].

2.2 Converting DNNs to SNNs

This work is based on the idea of [3,7], where a pre-trained artificial neural network is converted into a SNN. In this case, we train several multi-layer perceptrons that differ in size to classify MNIST handwritten digits. The training uses standard batch-wise gradient-descent in combination with error back-propagation. The conversion method exploits that the activation curve of a LIF

[2] https://github.com/hbp-unibi/cypress.

[3] The code for this and other work can be found at https://github.com/hbp-unibi/SNABSuite.

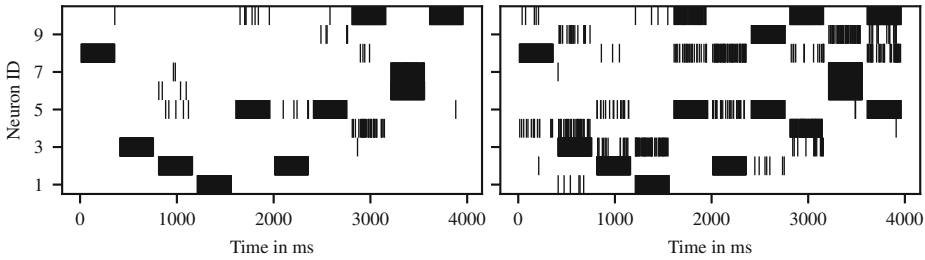

Fig. 1. Output spikes for converted networks. Left: Output spikes of a network that has been trained using a softmax layer as the last layer. Right: The same network trained with only ReLU activation functions.

neuron resembles the ReLU activation curve, such that float (analog) values of the ANN become spike rates in the SNN. All weights of the ANN are normalized to the maximal weight of the full network, and then scaled to a maximal value either given by restrictions of the hardware platform (e.g. 4 bit weights on Spikey/BrainScaleS) or determined by parameter optimization (see below for details). Similarly, other parameters of the SNN are found by extensive parameter tuning or are fixed due to hardware constraints. Neuron biases are not easily and efficiently mapped to SNNs, which is why we set all bias terms to zero in the training process of the ANN. In contrast to [7], we found that using a softmax layer as the last layer in the ANN for training does not necessarily decrease the performance of the SNN. However, using soft-max will lead to an increased number of spikes for all rejected classes (cf. Fig. 1).

As the Spikey platform is very limited in size and connectivity, the smallest and simplest network (referred to as *Spikey network*) consists of a single hidden layer with 100 neurons and no inhibitory connections. Spikey requires separation of excitation and inhibition at the neuron level and consists of two separate chips with limited connectivity between them. Thus, we only used positive weights and achieved the best performance using a hinge loss, which increases the weights for the winner neurons and decreases weights for the second place neuron only. Due to the acceleration factor of Spikey and BrainScaleS, communication bandwidth limits the usable spike rates. Too high rates (input as well as inter-neuron rates) will inevitably lead to spike loss that would reduce the performance of the network. This naturally restricts the parameter space to be evaluated. Still, there is a significant performance loss when applying the conversion process for analog systems. Perfect conversion requires that every synapse with the same weight and every neuron behaves in the same way, referring to identical activation curves. On analog systems, however, we have to deal with temporal noise perturbing the membrane voltage, trial-to-trial variation and analogmismatch between circuits [19]. As shown in [24], such a hardware network will perform at roughly 60–70% accuracy compared to a simulator, even after platform specific parameter tuning. [23] proposed to train the pre-trained neural network again while replacing the outputs of the ANN with spike rates recorded from hardware

employing back-propagation to train a device specific network. All details can be found in Fig. 7 of [23].

2.3 Neural Architecture Search (NAS)

Lamarck_ML[4] [11] is a modular and extensible Python library for application driven exploration of network architectures. This library allows to define a class of network architectures to be examined and operations to modify and combine those architectures. These definitions are then used by a search algorithm to explore and evaluate network architectures in order to maximize an objective function. For this work, the limitations of the neuromorphic hardware systems compared to state-of-the-art processing units are the leading motivation for the applied restrictions. The applied layer types are limited to fully connected layers which may be arranged in a nonsequential manner resulting in an acyclic directed graph structure. To preserve the structural information of a single neural network in the exploration process, a meta graph is created to contain the current network and the meta graph of the networks which were involved in creating it. This process is unbounded and accumulates structural information over several generations in the meta graph. To forget unprofitable information, the meta graph is designed to dismiss structural information that has not been used in the last five exploration steps. One exploration step consists of combining the meta graph of two network architectures and sampling a new path in this meta graph in order to create an improved architecture. A new architecture is created by sampling a path based on the quality of its best architecture and amending it with elements that have not been examined before.

The exploration procedure is performed by a genetic algorithm configured with a generation size of 36 network architectures of which 20 are selected based on an exponential ranking to create new architectures for the next generation. This next generation is created with an elitism replacement scheme that preserves the best two network architectures of the previous generation. In total 75 generations have been created in the NAS to find an architecture that achieves at least 97% evaluation accuracy. Above this threshold, an architecture is defined to be better if it requires less than 100 neurons for increasing the accuracy by 1%.

3 Results

The first two parts of this section present the parameter tuning process used for the converted SNNs. Details of four different networks are shown, the smallest one was defined by the restrictions of the Spikey platform, while the remaining networks were picked from the neural architecture search. The final part gathers the results for all networks including one model taken from literature.

[4] https://github.com/JonasDHomburg/LAMARCK_ML.

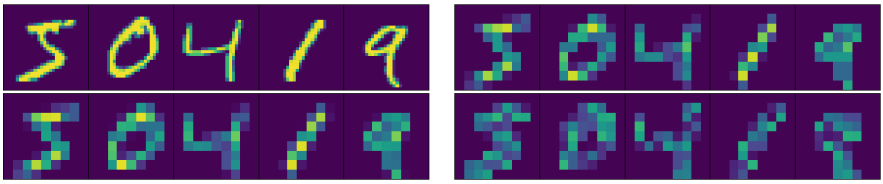

Fig. 2. Visualization of the down-scaled and converted images. The top left row shows the first five images of the MNIST training data set. The bottom left row shows down-scaled images using 3×3 average pooling. The top right row represents the conversion to spikes and back to analog values. The bottom right row shows differences between down-scaled and converted images scaled up by a factor of 10.

3.1 The Spikey Network and Parameter Optimization

This is the simplest network used in this work. As described above, it is motivated by the hardware restriction of the Spikey neuromorphic hardware system and uses a $89 \times 100 \times 10$ layout which requires images to be scaled down using 3×3 average pooling (cf. Fig. 2). These restrictions limit the test-accuracy of the pre-trained network to only 90.13%. This serves as the baseline for the following optimizations of the most relevant SNN parameters.

- The **maximal weight** determines the incoming activity per neuron. If chosen too high, the neuron operates in its non-linear and saturating range near the maximum output frequency.
- The **leakage/membrane time constant** describes the time window in which the neuron integrates incoming input. Too small values would require high frequencies for encoding analog values while higher numbers lead to saturation effects.
- The **sample presentation time** increases accuracy with higher values, which in turn require more energy and time.
- A higher **frequency range of input pixels** improves the pixel approximation accuracy, but is subject to saturation of neurons.

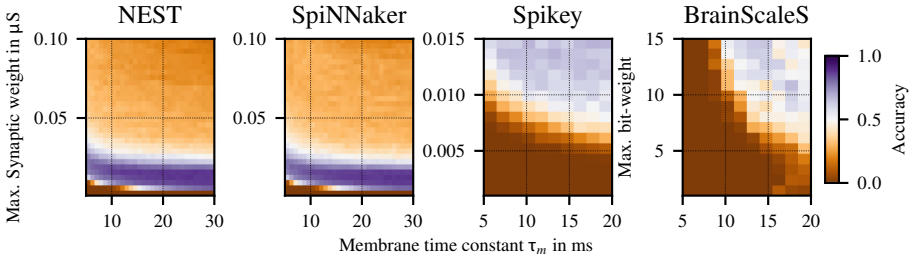

Fig. 3. Sweep over the maximal input frequency. Weights for BrainScaleS are set via low level digital weights (0 to 15).

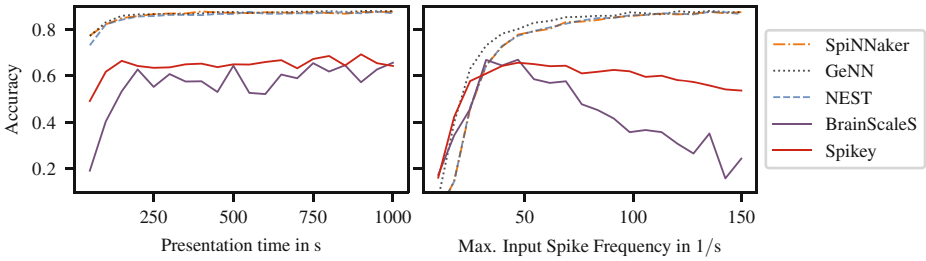

Fig. 4. Sweep over the sample presentation time (left) and the maximal input frequency (right)

Figure 3 shows parameter sweeps over the two most essential neuron parameters for the training set. The images show large areas of high relative accuracy for the analog platforms. On the simulated platforms, one can see the discussed effects of saturating neurons at high weights/time constants. Here, the area of high relative accuracy is rather narrow. Therefore, careful parameter tuning has to be done.

Taking a look at the most relevant conversion parameters, Fig. 4 shows the accuracy in relation to the sample presentation time and the maximal spike input frequency. First, simulating more than 200 ms will result in minor improvements only. Analog platforms converge a bit slower (which is partially caused by different neuron parameters used in the simulation), and the benefits of using presentation times larger than 200 ms are minor again. However, prolonged presentation times can cancel out some of the temporal noise on membrane voltages and synapses. Second, all platforms gain significantly from frequencies larger than 40 Hz. However, due to communication constraints in the accelerated analog platforms, the accuracy decreases for values above 60 Hz. Here, two bandwidth restrictions may play a major role: input spikes are inserted into the digital network using FPGAs. Any spike loss is usually reported by the respective software layer. However, on the wafer, there might be additional loss in the internal network, which is not reported. Output rates of hidden and ouput layers are a second source of potential spike loss which is only partially reported for the Spikey system (by monitoring spike buffers), but happens silently on the BrainScaleS system. The Spikey system reports full buffers for larger frequencies, which is why we assume that this is the major cause for spike loss on both systems.

To reach a high efficiency on larger systems, it is crucial to fully utilize them. Therefore, we used several parallel instances of the same network each classifying a separate portion of the data. In our setup this is controlled by choosing the batch size: a smaller batch size leads to more independent batches processed in parallel and thus effectively reduces processing time and energy per inference. This also avoids idle cores contributing to the energy calculation. These system-specific variations in batch size have negligible effects on the classification accuracy. On SpiNNaker, the hardware size and the required number of processor cores per network instance determine the parallelism. On GeNN, the

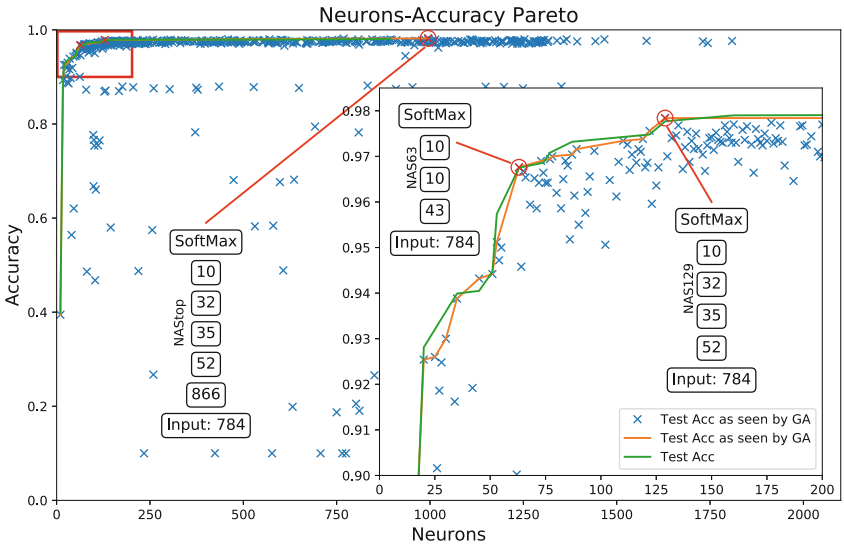

Fig. 5. Results of the optimization process. Highlighted are three candidates networks at the pareto front with their respective network layout.

working memory required to compile the GPU code is the determining factor. The latter is a limitation caused by using separate populations per layer, which could be merged to possibly lead to an increased parallelism of the networks, but not necessarily to increased efficiency. Only the Spikey system executes batches sequentially to avoid full spike buffers.

3.2 NAS Optimized Networks

The optimization process was driven by two major goals: to reach an accuracy larger than 97% and at the same time to reduce the network size in terms of the number of neurons. Results in Fig. 5 reveal, that this not necessarily leads to networks with a single hidden layer. Furthermore, the sequential neural networks outperformed all evaluated non-sequential architectures. We have chosen three candidates on the pareto-front for evaluation on neuromorphic hardware:

- the network with the highest evaluation accuracy (*NAStop*, 97.71%)
- the optimal network with the best trade-off (*NAS129*, 97.53%)
- a small network with still sufficient accuracy (*NAS63*, 96,76%)

3.3 Benchmark Results

Table 1 collects the results for all target platforms. Most striking is the energy efficiency of the analog platforms, which is two orders of magnitude higher compared to other simulators. Furthermore, HIL training recovers most of the

Table 1. Results from all converted networks. Highlighted are the best values per converted network. † Reduced number of neurons per core from its default 255 to 200 and × further reduced to 180 together with a slowed-down simulation (factor 2).

Platform	Accuracy in %	Conversion Loss in %	Wall clock time in ms	Energy per Inference in mJ	Batchsize
Spikey Network (ANN accuracy: 90.13%)					
Spikey	65.33	24.80	350	0.21	2500
Spikey HIL	84.99	5.14	350	0.21	100
BrainScaleS	61.65	28.43	900	0.33	10000
BrainScaleS HIL	83.87	6.56	900	0.36	10000
SpiNN3	88.41	1.72	264000	79	480
SpiNN5	88.40	1.73	23100	61	42
NEST	88.98	1.15	70542	316	2500
GeNN CPU	89.11	1.02	5070	10	10000
GeNN GPU	88.87	1.26	2623	21	100
NAS63 (ANN accuracy: 96,76%)					
SpiNN3	96.04	0.63	368500	109	670
SpiNN5	96.04	0.63	30800	80	56
NEST	96.37	0.30	217252	952	10000
GeNN CPU	96.29	0.38	16659	31	10000
GeNN GPU	96.32	0.35	17881	145	160
NAS129 (ANN accuracy: 97,53%)					
SpiNN3	96.86	0.67	458700	138	834
SpiNN5	97.25	0.28	38500	105	70
NEST	97.10	0.43	263134	1247	10000
GeNN CPU	97.42	0.11	20436	38	10000
GeNN GPU	97.34	0.19	18495	153	200
NAStop (ANN accuracy: 97,71%)					
SpiNN3†	96.80	0.91	918500	353	1670
SpiNN5†	97.42	0.29	82500	288	150
NEST	97.35	0.36	907869	4004	10000
GeNN CPU	97.53	0.18	96324	173	10000
GeNN GPU	97.51	0.20	21355	196	265
Network from [7] (ANN accuracy of 98.84%)					
SpiNN3×	97.83	1.01	2750000	1021	2500
SpiNN5†	98.77	0.07	104500	407	190
NEST	98.82	0.02	3061562	13869	10000
GeNN CPU	98.86	-0.02	314049	587	10000
GeNN GPU	98.85	-0.01	26632	293	280

conversion losses found for these platforms (despite the four bit weight accuracy). Larger networks have not been evaluated either due to size restrictions, or because combined spike rates of input pixels are too high to get any reasonable results. The SpiNNaker system, in both variants, performs on the same efficiency level as a CPU/GPU implementations although its technology is much older (130 nm vs. 22 nm CPU vs. 16 nm GPU). Furthermore, there is less than one percent loss in accuracy due to the conversion in almost all cases. However, for the large networks the system was performing at its limits, and we had to reduce the maximal number of neurons per core. Of course, this can be mitigated by further reducing the number of neurons per core or slowing down the system with respective negative impacts on the energy per inference. Interesting differences have been found for NEST: in some cases the accuracy is a bit lower, but the energy per inference is one order higher than for the GeNN CPU simulation. The latter is mainly due to the more accurate integrator employed by the NEST simulator (especially the adaptive time step in the integrator), which is also responsible for the significant energy gap between the two CPU simulators NEST and GeNN. Furthermore, the multi-threaded execution of NEST does not reduce the computation time compared to GeNN. With the increase of network complexity there is next to no increase in GPU execution time, indicating that despite parallelization of the networks, the GPU is still not utilized fully for the smaller networks (there are 3969-86,760 simultaneously simulated neurons for the GPU depending on the network). Still, for the larger networks, the GPU implementation is the fastest simulation available.

The last network in Table 1 is taken from [7], as the network weights are published within the respective repository. The layout is $784 \times 1200 \times 1200 \times 10$, and thus it is significantly larger. The results show that the SpiNN3 system still operates at its limits (as reported by the software stack) despite the used slow-down. The other platforms show nearly the same accuracy with next to no loss in the conversion process. Concerning the energy per inference, the larger SpiNNaker platform is slightly better than the CPU implementation, with the GPU being the most efficient platform.

4 Conclusion and Outlook

We have demonstrated the capability of all target platforms to simulate converted deep neural networks. The loss in the conversion process is negligible in many cases, and for analog platforms Spikey and BrainScaleS we successfully employed retraining to reach high accuracy. Furthermore, we calculated the used energy-per-inference, quantifying the efficiency vs. accuracy trade-off of analog platforms. The digital SpiNNaker platform is highly efficient if fully utilized despite the rather old chip manufacturing process, demonstrating the suitability for efficient large-scale simulations. If primarily simulation time at highest accuracy for not too large networks needs to be optimized, GeNN's GPU backend allow fast and efficient simulation of SNNs. The approach used in this work is not the most efficient way of using spiking neural networks. However,

the rate-coding applied here can be replaced with a more efficient time-to-first-spike (TTFS) encoding, using only a few spikes with much faster response times, which has recently been demonstrated on analog hardware [10]. Therefore, the results from this work must be seen as a conservative measure for the relative efficiency of SNNs on neuromorphic hardware. Furthermore, we did not make use of convolutional networks, because these currently cannot be mapped well to neuromorphic hardware. For the future of our benchmark suite we plan to include both: networks using TTFS encoding and convolutions. This will allow us to test more challenging data-sets with larger and more complex networks.

Funding/Acknowledgment. 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 Brain 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). Furthermore, we thank the Electronic Vision(s) group from Heidelberg University and Advanced Processor Technologies Research Group from Manchester University for access to their hardware systems and continuous support and James Knight from the University of Sussex for support regarding our GeNN implementation.

References

1. Akopyan, F., et al.: TrueNorth: design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip. IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. **34**(10), 1537–1557 (2015). https://doi.org/10.1109/TCAD.2015.2474396
2. van Albada, S.J., et al.: Performance comparison of the digital neuromorphic hardware SpiNNaker and the neural network simulation software NEST for a full-scale cortical microcircuit model. Front. Neurosci. **12**, 291 (2018). https://doi.org/10.3389/fnins.2018.00291
3. Cao, Y., Chen, Y., Khosla, D.: Spiking deep convolutional neural networks for energy-efficient object recognition. Int. J. Comput. Vision **113**(1), 54–66 (2014). https://doi.org/10.1007/s11263-014-0788-3
4. Davies, M.: Benchmarks for progress in neuromorphic computing. Nat. Mach. Intell. **1**(9), 386–388 (2019). https://doi.org/10.1038/s42256-019-0097-1
5. Davies, M., et al.: Loihi: a neuromorphic manycore processor with on-chip learning. IEEE Micro **38**(1), 82–99 (2018). https://doi.org/10.1109/MM.2018.112130359
6. Davison, A.P.: PyNN: a common interface for neuronal network simulators. Front. Neuroinform. **2**(January), 11 (2008). https://doi.org/10.3389/neuro.11.011.2008
7. Diehll, P.U., et al.: Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In: Proceedings of the International Joint Conference on Neural Networks 2015-September (2015). https://doi.org/10.1109/IJCNN.2015.7280696
8. Furber, S.B., et al.: Overview of the SpiNNaker system architecture. IEEE Trans. Comput. **62**(12), 2454–2467 (2013). https://doi.org/10.1109/TC.2012.142
9. Gewaltig, M.O., Diesmann, M.: NEST (neural simulation tool). Scholarpedia **2**(4), 1430 (2007)

10. Göltz, J., et al.: Fast and deep neuromorphic learning with time-to-first-spike coding (2019). https://doi.org/10.1145/3381755.3381770
11. Homburg, J.D., Adams, M., Thies, M., Korthals, T., Hesse, M., Rückert, U.: Constraint exploration of convolutional network architectures with neuroevolution. In: Rojas, I., Joya, G., Catala, A. (eds.) IWANN 2019. LNCS, vol. 11507, pp. 735–746. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-20518-8_61
12. Jordan, J., et al.: NEST 2.18.0 (2019). https://doi.org/10.5281/ZENODO.2605422
13. Knight, J.C., Nowotny, T.: GPUs outperform current HPC and neuromorphic solutions in terms of speed and energy when simulating a highly-connected cortical model. Front. Neurosci. **12**(December), 1–19 (2018). https://doi.org/10.3389/fnins.2018.00941
14. Maass, W.: Networks of spiking neurons: the third generation of neural network models. Neural Netw. **10**(9), 1659–1671 (1997). https://doi.org/10.1016/S0893-6080(97)00011-7
15. Moradi, S., et al.: A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs). IEEE Trans. Biomed. Circuits Syst. **12**(1), 106–122 (2018). https://doi.org/10.1109/TBCAS.2017.2759700
16. Neckar, A., et al.: Braindrop: a mixed-signal neuromorphic architecture with a dynamical systems-based programming model. Proc. IEEE **107**(1), 144–164 (2019). https://doi.org/10.1109/JPROC.2018.2881432
17. Ostrau, C., et al.: Comparing neuromorphic systems by solving sudoku problems. In: Conference Proceedings: 2019 International Conference on High Performance Computing & Simulation (HPCS). IEEE (2019). https://doi.org/10.1109/HPCS48598.2019.9188207
18. Ostrau, C., et al.: Benchmarking of neuromorphic hardware systems. In: Proceedings of the Neuro-Inspired Computational Elements Workshop. Association for Computing Machinery (ACM) (2020). https://doi.org/10.1145/3381755.3381772
19. Petrovici, M.A., et al.: Characterization and compensation of network-level anomalies in mixed-signal neuromorphic modeling platforms. PLoS ONE, **9**(10) (2014). https://doi.org/10.1371/journal.pone.0108590
20. Pfeil, T., et al.: Six networks on a universal neuromorphic computing substrate. Front. Neurosci. **7**(7 FEB), 11 (2013). https://doi.org/10.3389/fnins.2013.00011
21. Rhodes, O., et al.: Real-time cortical simulation on neuromorphic hardware. Philos. Trans. R. Soc. A: Math. Phys. Eng. Sci. **378**(2164), 20190160 (2020). https://doi.org/10.1098/rsta.2019.0160
22. Schemmel, J., et al.: A wafer-scale neuromorphic hardware system for large-scale neural modeling. In: Proceedings of 2010 IEEE International Symposium on Circuits and Systems, pp. 1947–1950 (2010). https://doi.org/10.1109/ISCAS.2010.5536970
23. Schmitt, S., et al.: Neuromorphic hardware in the loop: training a deep spiking network on the BrainScaleS wafer-scale system. In: 2017 International Joint Conference on Neural Networks (IJCNN), pp. 2227–2234. IEEE (2017). https://doi.org/10.1109/IJCNN.2017.7966125
24. Stöckel, A., et al.: Binary associative memories as a benchmark for spiking neuromorphic hardware. Front. Comput. Neurosci. **11**(August), 71 (2017). https://doi.org/10.3389/fncom.2017.00071
25. Yavuz, E., et al.: GeNN: a code generation framework for accelerated brain simulations. Sci. Rep. **6**(2015), 18854 (2016). https://doi.org/10.1038/srep18854