Accelerated physical emulation of Bayesian inference in spiking neural networks

Akos F. Kungl [1], **Sebastian Schmitt** [1], **Johann Klähn** [1], **Paul Müller** [1], **Andreas Baumbach** [1], **Dominik Dold** [1],
Alexander Kugele [1], Eric Müller [1], Christoph Koke [1], Mitja Kleider [1], Christian Mauch [1], Oliver Breitwieser [1],
Luziwei Leng [1], Nico Gürtler [1], Maurice Güttler [1], Dan Husmann [1], Kai Husmann [1], Andreas Hartel [1],
Vitali Karasenko [1], Andreas Grübl [1], Johannes Schemmel [1], Karlheinz Meier [1], Mihai A. Petrovici [1,2]

[1]Kirchhoff-Institute for Physics, Heidelberg University, 69120 Heidelberg, Germany
[2]Department of Physiology, University of Bern, 3012 Bern, Switzerland

2019 November 14th

Abstract

The massively parallel nature of biological information processing plays an important role for its superiority to human-engineered computing devices. In particular, it may hold the key to overcoming the von Neumann bottleneck that limits contemporary computer architectures. Physical-model neuromorphic devices seek to replicate not only this inherent parallelism, but also aspects of its microscopic dynamics in analog circuits emulating neurons and synapses. However, these machines require network models that are not only adept at solving particular tasks, but that can also cope with the inherent imperfections of analog substrates. We present a spiking network model that performs Bayesian inference through sampling on the BrainScaleS neuromorphic platform, where we use it for generative and discriminative computations on visual data. By illustrating its functionality on this platform, we implicitly demonstrate its robustness to various substrate-specific distortive effects, as well as its accelerated capability for computation. These results showcase the advantages of brain-inspired physical computation and provide important building blocks for large-scale neuromorphic applications.

1 Introduction

The aggressive pursuit of Moore's law in conventional computing architectures is slowly but surely nearing its end (Waldrop, 2016), with difficult-to-overcome physical effects, such as heat production and quantum uncertainty, representing the main limiting factor. The so-called von Neumann bottleneck between processing and memory units

represents the main cause, as it effectively limits the speed of these largely serial computation devices. The most promising solutions come in the form of massively parallel devices, many of which are based on brain-inspired computing paradigms (Indiveri et al., 2011; Furber, 2016), each with its own advantages and drawbacks.

Among the various approaches to such neuromorphic computing, one class of devices is dedicated to the physical emulation of cortical circuits: not only do they instantiate neurons and synapses that operate in parallel and independently of each other, but these units are actually represented by distinct circuits that emulate the dynamics of their biological archetypes (Mead, 1990; Indiveri et al., 2006; Schemmel et al., 2010; Jo et al., 2010; Pfeil et al., 2013; Qiao et al., 2015; Chang et al., 2016; Wunderlich et al., 2019). Some important advantages of this approach lie in their reduced power consumption and enhanced speed compared to conventional simulations of biological neuronal networks, which represent direct payoffs of replacing the resource-intensive numerical calculation of neuro-synaptic dynamics with the physics of the devices themselves.

However, such computation with analog dynamics, without the convenience of binarization, as used in digital devices, has a downside of its own: variability in the manufacturing process (fixed pattern noise) and temporal noise both lead to reduced controllability of the circuit dynamics. Additionally, one relinquishes much of the freedom permitted by conventional algorithms and simulations, as one is confined by the dynamics and parameter ranges cast into the silicon substrate. The main challenge of exploiting these systems therefore lies in designing performant network models using the available components while maintaining a degree of robustness towards the substrate-induced distortions. Just like for the devices themselves, inspiration for such models often comes from neuroscience, as the brain needs to meet similar demands.

With accumulating experimental evidence (Berkes et al., 2011; Pouget et al., 2013; Orbán et al., 2016; Haefner et al., 2016), the view of the brain itself as an analytical computation device has shifted. The stochastic nature of neural activity in vivo is being increasingly regarded as an explicit computational resource rather than a nuisance that needs to be dealt with by sophisticated error-correcting mechanisms or by averaging over populations. Under the assumption that stochastic brain dynamics reflect an ongoing process of Bayesian inference in continuous time, the output variability of single neurons can be interpreted as a representation of uncertainty. Theories of neural sampling (Buesing et al., 2011; Hennequin et al., 2014; Aitchison and Lengyel, 2016; Petrovici et al., 2016; Kutschireiter et al., 2017) provide an analytical framework for embedding this type of computation in spiking neural networks.

In this paper we describe the realization of neural sampling with networks of leaky integrate-and-fire neurons (Petrovici et al., 2016) on the BrainScaleS accelerated neuromorphic platform (Schemmel et al., 2010). With appropriate training, the variability of the analog components can be naturally compensated and incorporated into a functional network structure, while the network's ongoing dynamics make explicit use of the analog substrate's intrinsic acceleration for Bayesian inference (section 2.3). We demonstrate sampling from low-dimensional target probability distributions with randomly chosen parameters (section 3.1) as well as inference in high-dimensional spaces constrained by real-world data, by solving associated classification and constraint satisfaction problems (pattern completion, section 3.2). All network components are fully contained on the neuromorphic substrate, with external inputs only used for sensory evidence (visual data). Our work thereby contributes to the search for novel paradigms of information processing that can directly benefit from the features of neuro-inspired physical model systems.

2 Methods

2.1 The BrainScaleS system

BrainScaleS (Schemmel et al., 2010) is a mixed-signal neuromorphic system, realized in $180\,\mathrm{nm}$ CMOS technology, that emulates networks of spiking neurons. Each BrainScaleS wafer module consists of a $20\,\mathrm{cm}$ silicon wafer with 384 HICANN (High Input Count Analog Neural Network) chips, see fig. 1 A. On each chip, 512 analog circuits emulate the adaptive exponential integrate-and-fire (AdEx) model (Brette and Gerstner, 2005; Millner et al., 2010)
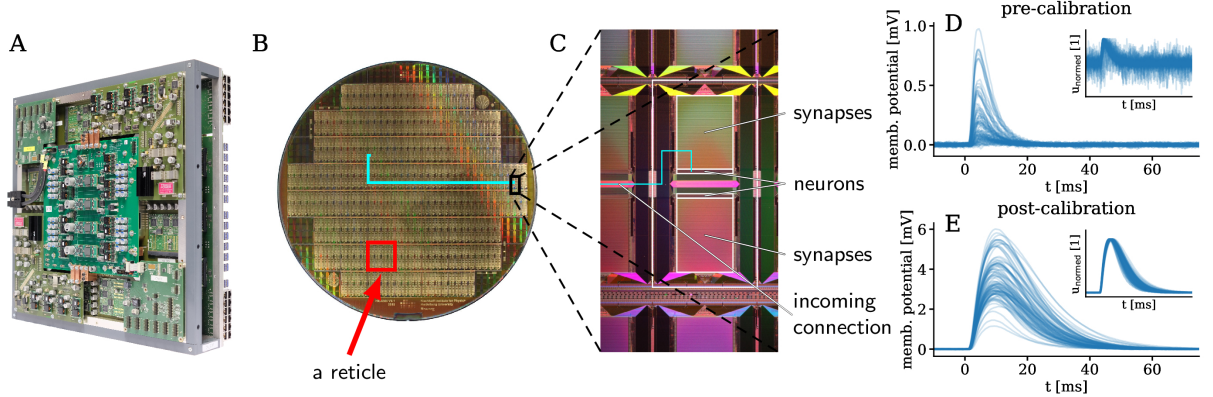
Figure 1: **(A)** Photograph of a fully assembled wafer module of the BrainScaleS system (dimensions: $50\,\mathrm{cm} \times 50\,\mathrm{cm} \times 15\,\mathrm{cm}$). One module hosts 384 HICANN chips on 48 reticles, with 512 physical neurons per chip and 220 synapse circuits per neuron. The wafer itself lies at the center of the module and is itself not visible. 48 FPGAs are responsible for I/O and experiment control. Support PCBs provide power supply for the on-wafer circuits as well as access to neuron membrane voltages. The connectors for inter-wafer (sockets resembling USB-A) and off-wafer/host connectivity (Gigabit-Ethernet sockets) are distributed over all four edges of the main PCB. Mechanical stability is provided by an aluminum frame. **(B)** The wafer itself is composed of 48 reticles (e.g., red rectangle), each containing 8 HICANN chips (e.g., black rectangle, enlarged in C). Inter-reticle connectivity is added in a post-processing step. **(C)** On a single HICANN chip, the largest area is occupied by the two synapse matrices which instantiate connections to the neurons positioned in the neuron array. **(D-E)** Postsynaptic potentials (PSPs) measured on 100 different neuron membranes using the same parameter settings before **(D)** and after **(E)** calibration. The insets show the height-normalized PSPs. The calibration serves two purposes. First, it provides a translation rule between the desired neuron parameters and the technical parameters set on the hardware. In this case, it brings the time constants τ_{mem} and τ_{syn} close to the target of $8\,\mathrm{ms}$, as evidenced by the small spread of the normalized PSPs. Second, in the absence of such a translation rule, it sets the circuits to their correct working points. Here, this happens for the synaptic weights: after calibration, PSP heights are, on average closer to the target working point of $3\,\mathrm{mV}$, but they remain highly diverse due to the variability of the substrate. For more details see Schmitt et al. (2017). The PSPs are averaged over 375 presynaptic spikes and smoothed with a Savitzky-Golay filter (Savitzky and Golay, 1964) to eliminate readout noise. The time-constants are given in the biological domain, but they are 10^4 faster on the system.

of spiking neurons with conductance-based synapses. The dynamics evolve with an acceleration factor of 10^4 with respect to biological time, i.e., all specific time constants (synaptic, membrane, adaptation) are approximately 10^4 times smaller than typical corresponding values found in biology (Schemmel et al., 2010; Petrovici et al., 2014). To preserve compatibility with related literature (Petrovici et al., 2016; Schmitt et al., 2017; Leng et al., 2018; Dold et al., 2019), we refer to system parameters in the biological domain unless specified otherwise, e.g., a membrane time constant given as $10\,\mathrm{ms}$ is actually accelerated to $1\,\mathrm{\mu s}$ on the chip.

The parameters of the neuron circuits are stored in analog memory cells (floating gates) with $10\,\mathrm{bit}$ resolution, and the synaptic weights are stored in $4\,\mathrm{bit}$ SRAM (Schemmel et al., 2010). The analog memory cells are similar to the ones in Lande et al. (1996), and they are described in Loock (2006) and Millner (2012).

Spike events are transported digitally and can reach all other neurons on the wafer with the help of an additional redistribution layer that instantiates an on-wafer circuit-switched network (Zoschke et al., 2017) (fig. 1 B).

Because of mismatch effects (fixed-pattern noise) inherent to the substrate, the response to incoming stimuli varies from neuron to neuron (fig. 1 D). In order to bring all neurons into the desired regime and reduce the neuron-to-neuron response variability, we employ a standard calibration procedure that is performed only once, during the commissioning of the system (Schmitt et al., 2017; Petrovici et al., 2017b). Nevertheless, even after calibration, a significant degree of diversity persists (fig. 1 E). The emulation of functional networks that do not rely on population averaging therefore requires appropriate training algorithms (section 3.2).
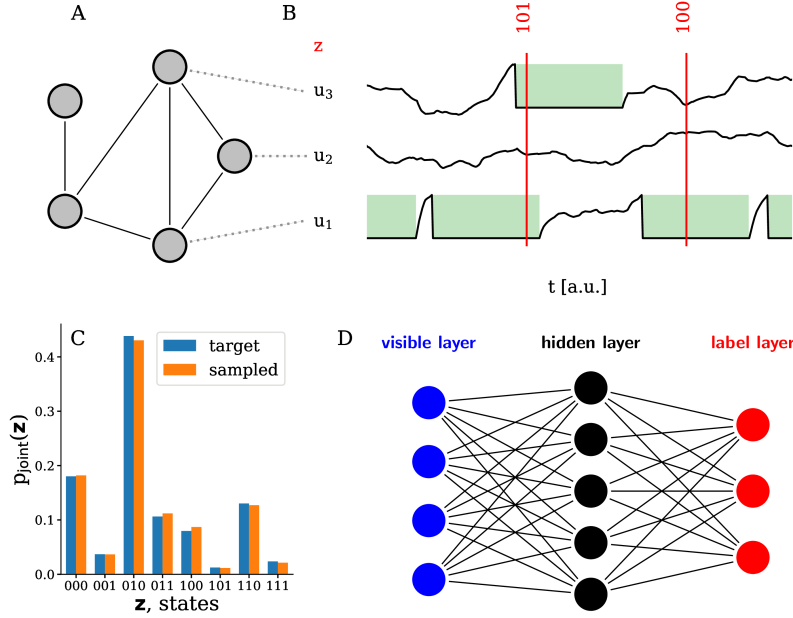
Figure 2: **Sampling with leaky integrate-and-fire (LIF) neurons**. **(A)** Schematic of a spiking sampling network (SSN) with 5 neurons. Each line represents two reciprocal synaptic connections with equal weights. **(B)** Example membrane potentials of three neurons in the network. Following a spike, the refractory mechanism effectively clamps the membrane potential to the reset value for a duration τ_{ref}. During this time, the RV corresponding to that neuron is in the state $z = 1$ (marked in green). At any point in time, the state sampled by the network can therefore be constructed directly from its output spikes and the refractory time τ_{ref} of the neurons. **(C)** Probability distribution sampled by an SSN with three neurons as compared to the target distribution. **(D)** Based on this framework (Petrovici et al., 2016), hierarchical sampling networks can be built, which can be trained on real-world data. Each line represents a reciprocal connection (two synapses) between the connected neurons.

2.2   Sampling with leaky integrate-and-fire neurons

The theory of sampling with leaky integrate-and-fire neurons (Petrovici et al., 2016) describes a mapping between the dynamics of a population of neurons with conductance-based synapses (equations given in table 1) and a Markov-chain Monte Carlo sampling process from an underlying probability distribution over binary random variables (RVs). Each neuron in such a sampling network corresponds to one of these RVs: if the k-th neuron has spiked in the recent past and is currently refractory, then it is considered to be in the *on-state* $z_k = 1$, otherwise it is in the *off-state* $z_k = 0$ (fig. 2 A, B). With appropriate synaptic parameters, such a network can approximately sample from a Boltzmann distribution defined by

$$p^*(\boldsymbol{z}) = \frac{1}{Z} \exp\left(\frac{1}{2}\boldsymbol{z}^T \boldsymbol{W} \boldsymbol{z} + \boldsymbol{z}^T \boldsymbol{b}\right) \ , \tag{1}$$

where Z is the partition sum, $\boldsymbol{W}$ a symmetric, zero-diagonal effective weight matrix and b_i the effective bias of the i-th neuron.

In the original model, each neuron receives excitatory and inhibitory Poisson input. This plays two important roles: it transforms a deterministic LIF neuron into a stochastic firing unit and induces a high-conductance state, with an effective membrane time constant that is much smaller than other time constants in the system: $\tau_{\mathrm{eff}} \gg \tau_{\mathrm{syn}}, \tau_{\mathrm{ref}}$ - (see, e.g., Destexhe et al., 2003; Petrovici, 2016), which symmetrizes the neural activation function, as explained in the following. The activation function of an LIF neuron without noise features a sharp onset, but only a slow converge to its maximum value, hence being highly asymmetric around the point of $50\,\%$ activity. Background Poisson noise

smears out the onset of the activation function, while the reduced membrane time constant accelerates the convergence to the maximum, making the activation function more symmetric and thus more similar to a logistic function, which is a prerequisite for this form of sampling. For the explicit derivation see Petrovici et al. (2016) and Petrovici (2016). A mapping of this activation function to the abovementioned logistic function $1/[1 + \exp(-x)]$ provides the translation from the dimensionless weights and biases of the target distribution to the corresponding biological parameters of the spiking network (Petrovici, 2016).

Although different in their dynamics, such sampling spiking networks (SSNs) function similarly to (deep) Boltzmann machines (Hinton et al., 1984), which makes them applicable to the same class of machine learning problems (Leng et al., 2018). Training can be done using an approximation of wake-sleep algorithm (Hinton et al., 1995; Hinton, 2012), which implements maximum-likelihood learning on the training set:

$$\Delta b_i = \eta(\langle z_i \rangle^* - \langle z_i \rangle) , \tag{2}$$

$$\Delta W_{ij} = \eta(\langle z_i z_j \rangle^* - \langle z_i z_j \rangle) , \tag{3}$$

where $\langle \cdot \rangle$ and $\langle \cdot \rangle^*$ represent averages over the sampled (model or sleep phase) and target (data or wake phase) distribution, respectively, and η is the learning rate.

In order to enable a fully-contained neuromorphic emulation on the BrainScaleS system, the original model had to be modified. The changes in the network structure, noise generation mechanism and learning algorithm are described in section 2.3.

For low-dimensional, fully specified target distributions, we used the Kullback-Leibler divergence (DKL, Kullback and Leibler, 1951) as a measure of discrepancy between the sampled (p) and the target (p^*) distributions:

$$D_{\mathrm{KL}}(p \parallel p^*) = - \sum_{z_i \in \Omega} p(z_i) \ln \left(\frac{p(z_i)}{p^*(z_i)} \right) \tag{4}$$

This was done in part to preserve comparability with previous studies (Buesing et al., 2011; Petrovici et al., 2015, 2016), but also because the DKL is the natural loss function for maximum likelihood learning. For visual datasets, we used the error rate (ratio of misclassified images in the test set) for discriminative tasks and the mean squared error (MSE) between reconstruction and original image for pattern completion tasks. The MSE is defined as

$$\mathrm{MSE} = \frac{1}{N_{\mathrm{pixels}}} \sum_{k=1}^{N_{\mathrm{pixels}}} \left(z_k^{\mathrm{data}} - z_k^{\mathrm{recon}} \right)^2 , \tag{5}$$

where z_k^{data} is the reference data value, z_k^{recon} is the model reconstruction and the sum goes over the N_{pixels} pixels to be reconstructed by the SSN.

2.3 Experimental setup

The physical emulation of a network model on an analog neuromorphic substrate is not as straightforward as a software simulation, as it needs to comply with the constraints imposed by the emulating device. Often, it may be tempting to fine-tune the hardware to a specific configuration that fits one particular network, e.g., by selecting specific neuron and synapse circuits that operate optimally given a particular set of network parameters, or by manually tweaking individual hardware parameters after the network has been mapped and trained on the substrate. Here, we explicitly refrained from any such interventions in order to guarantee the robustness and scalability of our results.

All experiments were carried out on a single module of the BrainScaleS system using a subset of the available HICANN chips. The network setup was specified in the BrainScaleS-specific implementation of PyNN (Davison et al., 2009) and the standard calibration (Schmitt et al., 2017) was used to set the analog parameters. The full setup consisted of two main parts: the SSN and the source of stochasticity.
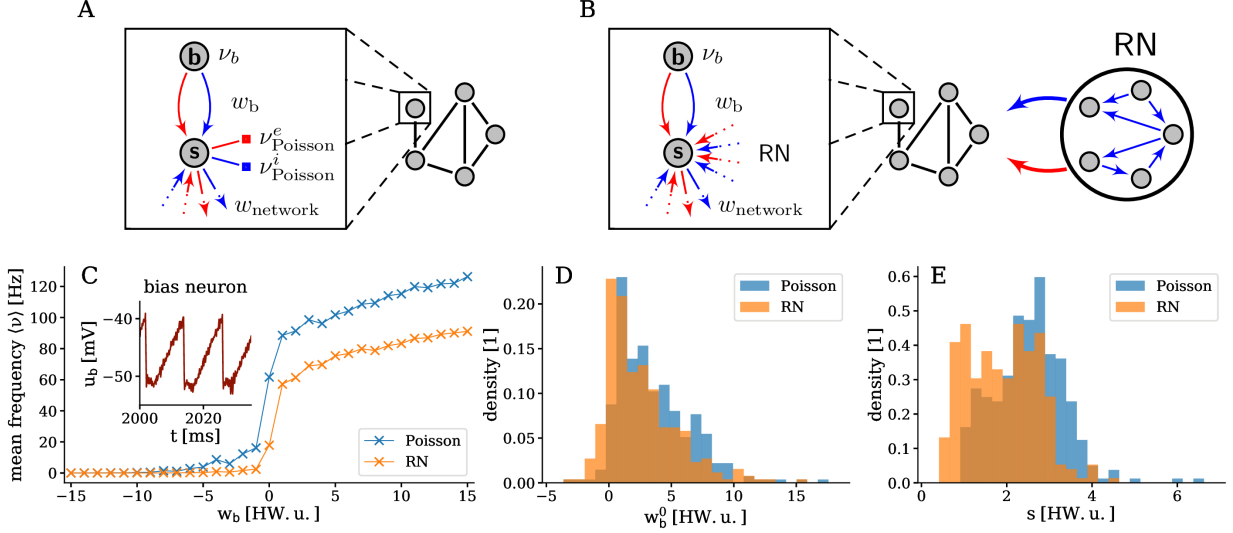
Figure 3: **Experimental setup.** Each sampling unit is instantiated by a pair of neurons on the hardware. The bias neuron ⓑ is configured with a suprathreshold leak potential and generates a regular spike train that impinges on the sampling neuron ⓢ, thereby serving as a bias, controlled by w_b. **(A)** As a benchmark, we provided each sampling neuron with private, off-substrate Poisson spike sources. **(B)** Alternatively, in order to reduce the I/O load, the noise was generated by a random network (RN). The RN consisted of randomly connected inhibitory neurons with $E_{\text{leak}} > V_{\text{thresh}}$. Connections were randomly assigned, such that each sampling neuron received a fixed number of excitatory and inhibitory presynaptic partners (table 1). **(C)** Exemplary activation function (mean firing frequency) of a single sampling neuron with Poisson noise and with an RN as a function of the bias weight. The standard deviation of the the trial-to-trial variability is on the order of $0.1\,\text{Hz}$ for both activation functions, hence the error bars are to small to be shown. The inset shows the membrane trace of the corresponding bias neuron. **(D-E)** The figures show histograms over all neurons in a sampling network on a calibrated BrainScaleS system. The width s and the midpoint w_{b}^{0} of the activation functions with Poisson noise and with an RN are calculated by fitting the logistic function $\langle \nu \rangle = \nu_0 / \{1 + \exp[-(w_{\text{b}} - w_{\text{b}}^{0})/s]\}$ to the data.

In the original sampling model (Petrovici et al., 2016), in order to affect biases, the wake-sleep algorithm (eq. (2)) requires access to at least one reversal potential (E_{l}, E_{exc}, or E_{inh}), which are all controlled by analog memory cells. Given that rewriting analog memory cells is both less precise and slower than rewriting the SRAM cells controlling the synaptic weights, we modified our SSNs to implement biases by means of synaptic weights. To this end, we replaced individual sampling neurons by sampling units, each realized using two hardware neurons (fig. 3 A, B). Like in the original model, a sampling neuron was set up to encode the corresponding binary RV. Each sampling neuron was accompanied by a bias neuron set up with a suprathreshold leak potential that ensured regular firing (fig. 3 C, inset). Each bias neuron projected to its target sampling neuron with both an excitatory and an inhibitory synapse (with independent weights), thus inducing a controllable offset of the sampling neuron's average membrane potential. Because excitatory and inhibitory inputs are routed through different circuits for each neuron, two types of synapses were required to allow the sign of the effective bias to change during training. For larger networks, in order to optimize the allocation of hardware resources, we shared the use of bias neurons among multiple sampling neurons (connected via distinct synapses). Similarly, in order to allow sign switches during training, connections between sampling neurons were implemented by pairs of synapses (one excitatory and one inhibitory) as well.

The dynamics of the sampling neurons were rendered stochastic in two different ways. The first setup served as a benchmark and represented a straightforward implementation of the theoretical model from (Petrovici et al., 2016), with Poisson noise generated on the host computer and fed in during the experiment (fig. 3 A). In the second setup, we used the spiking activity of a sparse recurrent random network (RN) of inhibitory neurons, instantiated on the same wafer, as a source of noise (fig. 3 B). For a more detailed study of sampling-based Bayesian inference with noise generated

6

by deterministic networks, we refer to (Jordan et al., 2017). The mutual inhibition ensured a relatively constant (sub)population firing rate with suitable random statistics that can replace the ideal Poisson noise in our application. Projections from the RN to the SSN were chosen as random and sparse; this resulted in weak, but non-zero shared-input correlations. The remaining correlations are compensated by appropriate training; the Hebbian learning rule (eq. (2)) changes the weights and biases in the network such that they cancel the input correlations induced by the RN activity (Bytschok et al., 2017; Dold et al., 2019). Hence, the same plasticity rule simultaneously addresses three issues: the learning procedure itself, the compensation of analog variability in neuronal excitability, and the compensation of cross-correlations in the input coming from the background network. This allowed the hardware-emulated RN to replace the Poisson noise required by the theoretical model.

With these noise-generating mechanisms, the activation function of the neurons, defined by the firing rate as a function of the bias weight w_b, took on an approximately logistic shape, as required by the sampling model (fig. 3 C). Due mainly to the variability of the hardware circuits and the precision of the analog parameters, the exact shape of this activation function varied significantly between neurons (fig. 3 D-E). Effectively, this means that initial weights and biases were set randomly, but also that the effective learning rates were different for each neuron. However, as we show below, this did not prevent the training procedure from converging to a good solution. This robustness with respect to substrate variability represents an important result of this work. The used neuron parameters are shown in table 2 and a summary of the used networks is given in table 4. Our largest experiment a network of 609 neurons with 208 sampling neurons, 1 bias neuron and 400 neurons in the RN (table 4 C) used hardware resources on 28 HICANN chips distributed over 7 reticles. Each of these functional neurons was realized by combining four of the 512 neuronal compartments ("denmems") available on each HICANN, in order to reduce variability in their leak potentials and membrane time constants; for details see (Schemmel et al., 2010).

To train the networks on a neuromorphic substrate without embedded plasticity, we used a training concept often referred to as in-the-loop training (Schmuker et al., 2014; Esser et al., 2016; Schmitt et al., 2017). With the setup discussed above, the only parameters changed during training were digital, namely the synaptic weights between sampling neurons and the weights between bias and sampling neurons. This allowed us to work with a fixed set of analog parameters, which significantly amplified the precision and speed of reconfiguration during learning, as compared to having used the analog storage instead. The updates of the digital parameters (synaptic weights) were calculated on the host computer based on the wake-sleep algorithm (eq. (2)) but using the spiking activity measured on the hardware. During the iterative procedure, the values of the weights were saved and updated as a double precision floating point variable, followed by (deterministic) discretization in order to comply with the single-synapse weight resolution of 4 bits. The learning parameters are given in table 3. Clamping (i.e. forcing neurons into state 1 or 0 with strong excitatory or inhibitory input) was done by injecting regular spike trains with $100\,\mathrm{Hz}$ frequency from the host through 5 synapses simultaneously, excitatory for $z_k = 1$ and inhibitory for $z_k = 0$. These multapses (multiple synapses connecting two neurons) were needed to exceed the upper limit of single synaptic weights and thus ensure proper clamping.

3 Results

3.1 Learning to approximate a target distribution

The experiments described in this section serve as a general benchmark for the ability of our hardware-emulated SSNs and the associated training algorithm to approximate fully specified target Boltzmann distributions. The viability of our proposal to simultaneously embed deterministic RNs as sources of pseudo-stochasticity is tested by comparing the sampling accuracy of RN-driven SSNs to the case where noise is injected from the host as perfectly uncorrelated Poisson spike trains.

Target distributions p^* over 5 RVs were chosen by sampling weights and biases from a Beta distribution centered around zero: $b_i, w_{ji} \sim 2[\text{Beta}(0.5, 0.5) - 0.5]$. Similarly to previous studies (Petrovici et al., 2016; Jordan et al., 2017), by giving preference to larger absolute values of the target distribution's parameters, we thereby increased the probability of instantiating rougher, more interesting energy landscapes. The initial weights and biases of the network were sampled from a uniform distribution over the possible hardware weights. Due to the small size of the state space, the "wake" component of the wake-sleep updates could be calculated analytically as $\langle z_i z_j \rangle = p^*(z_i = 1, z_j = 1)$ and $\langle z_i \rangle = p^*(z_i = 1)$ by explicit marginalization of the target distribution over non-relevant RVs.

For training, we used 500 iterations with 1×10^5 ms sampling time per iteration. Afterwards, the parameter configuration that produced the lowest $D_{\text{KL}}(p \parallel p^*)$ was tested in a longer (5×10^5 ms) experiment. To study the ability of the trained networks to perform Bayesian inference, we clamped two of the five neurons to fixed values $(z_1, z_2) = (0, 1)$ and compared the sampled conditional distribution to the target conditional distribution. Results for one of these target distributions are shown in fig. 4.

On average, with Poisson noise, the training showed fast convergence during the first 20 iterations, followed by fine-tuning and full convergence within 200 iterations. As expected, the convergence of the setups using RNs was significantly slower due to the need to overcome the additional background correlations, but they were still able to achieve similar performance (fig. 4 A).

In both setups, during the test run, the trained SSNs converged to the target distribution following an almost identical power law, which indicates similar mixing properties (fig. 4 B). For longer sampling durations ($\gg 10 \times 10^3$ ms), the systematic deviations from the target distributions become visible and the $D_{\text{KL}}(p \parallel p^*)$ reaches the same plateau at approximately $D_{\text{KL}}(p \parallel p^*) \approx 2 \times 10^{-2}$ as observed during training. Figure 4 C and D respectively show the sampled joint and marginal distributions after convergence. These observations remained consistent across a set of 20 different target distributions (see fig. 4 E for a representative selection).

Similar observations hold for the inference experiments. Due to the smaller state space, convergence happened faster (fig. 4 E). The corresponding joint and marginal distributions are shown in fig. 4 F and G, respectively. The lower accuracy of these distributions is mainly because of the asymmetry of the effective synaptic weights caused by the variability of the substrate, towards which the learning algorithm is agnostic. The training took 5×10^2 s wall-clock time, including the pure experiment runtime, the initialization of the hardware and the calculation of the updates on the host computer (total turn-over time of the training). This corresponds to a speed-up factor of 100 compared to the equivalent 5×10^4 s of biological real time. While the nominal 10^4 speed-up remained intact for the emulation of network dynamics, the total speed-up factor was reduced due to the overhead imposed by network (re)configuration and I/O between the host and the neuromorphic substrate.

We carried out the same experiments as described previously with 20 different samples for the weights and the biases of the target distribution. In fig. 5 we show the final DKLs after training to represent a target distribution both with Poisson noise and with the activity of a random network. The experiments were repeated 10 times for each sample. Median learning results remained consistent across target distributions, with the variability reflecting the difficulty of the problem (discrepancies between LIF and Glauber dynamics become more pronounced for larger weights and biases). Variability across trials for the same target distribution is due to the trial-to-trial variability of the analog parameter storage (floating gates), due to the inherent stochasticity in the learning procedure (sampling accuracy in an update step), as well as due to systematic discrepancies between the effective pre-post and post-pre interaction strengths between sampling units, which are themselves a consequence of the aforementioned floating gate variability.

3.2 Learning from data

In order to obtain models of labeled data, we trained hierarchical SSNs analogously to restricted Boltzmann machines (RBMs). Here, we used two different datasets: a reduced version of the MNIST (LeCun et al., 1998) and the fashion MNIST (Xiao et al., 2017) datasets, which we abbreviate as rMNIST and rFMNIST in the following. The images were
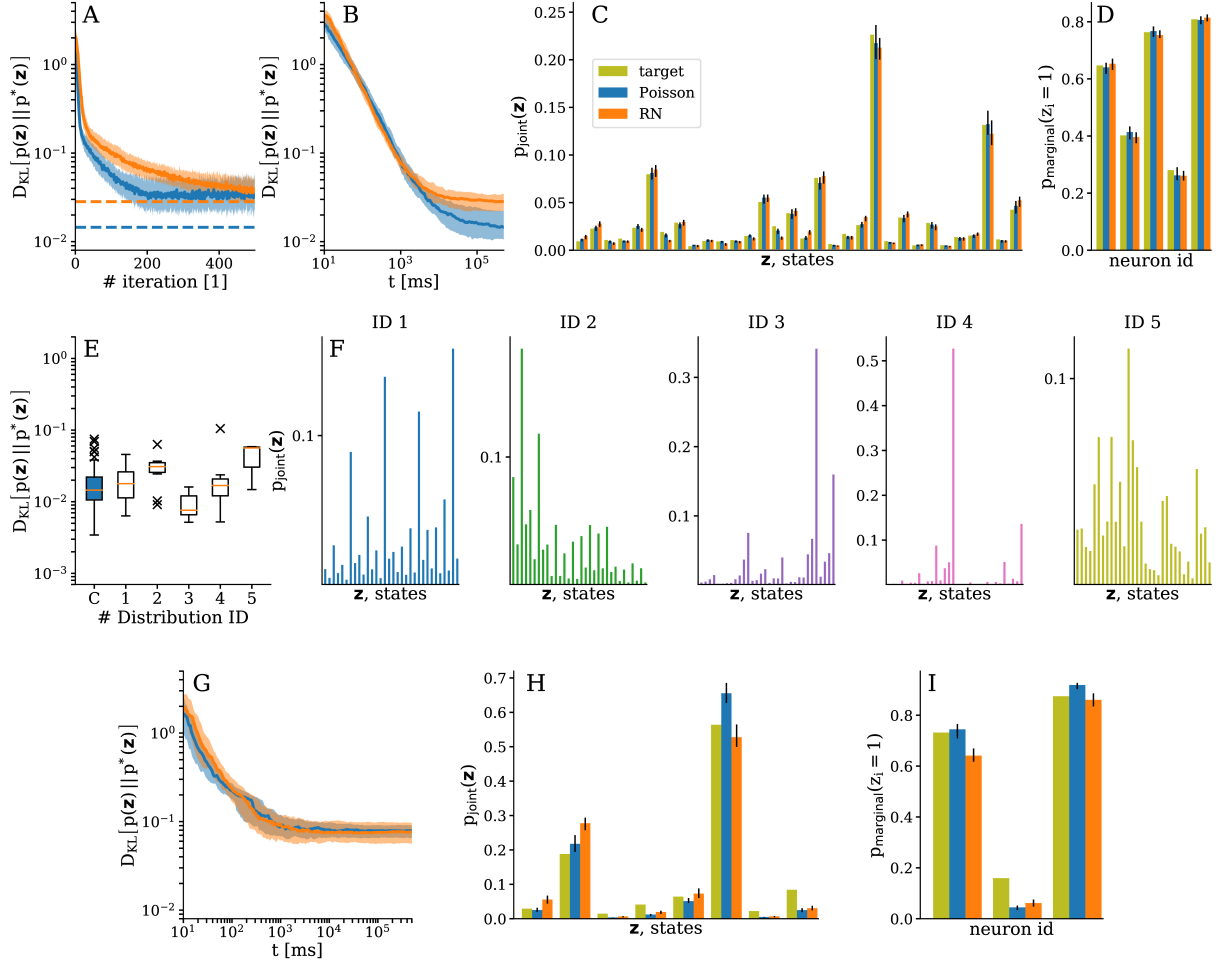
Figure 4: **Emulated SSNs sampling from target Boltzmann distributions.** Sampled distributions are depicted in blue for setups with Poisson noise and in orange for setups using RNs. Target distributions shown in dark yellow. Data was gathered from 150 runs with random initializations. Median values are shown as dark colors and interquartile ranges as either light colors or error bars. **(A)** Improvement of sampled distributions during training. The observed variability after convergence (during the plateau) is not due to noise in the system, but rather a consequence of the weight discretization: when the ideal (target) weights lie approximately mid-way between two consecutive integer values on the hardware, training leads to oscillations between these values. The parameter configuration showing the best performance during a training run – which, due to the abovementioned oscillations, was not necessarily the one in the final iteration – was chosen as the end result of the training phase. Averages of these results are shown as dashed lines. **(B)** Convergence of sampled distributions for the trained SSNs. **(C)** and **(D)** Sampled joint and marginal distributions of the trained SSNs after 5×10^5 ms, respectively. **(E)** Consistency of training results for different target distributions using Poisson noise. Here, we show a representative selection of 6 distributions with 10 independent runs per distribution. The box highlighted in blue corresponds to the target distribution used in the other panels of fig. 4. The data is plotted following the traditional box-and-whiskers scheme: the orange line represents the median, the box represents the interquartile range, the whiskers represent the full data range and the $\times$ represent the far outliers. **(F)** Target distributions corresponding to the last five box-and-whiskers plots in **(E)**. **(G)** Convergence of conditional distributions for the trained SSNs. **(H)** and **(I)** Sampled conditional joint and marginal distributions of the trained SSNs after 5×10^5 ms, respectively.
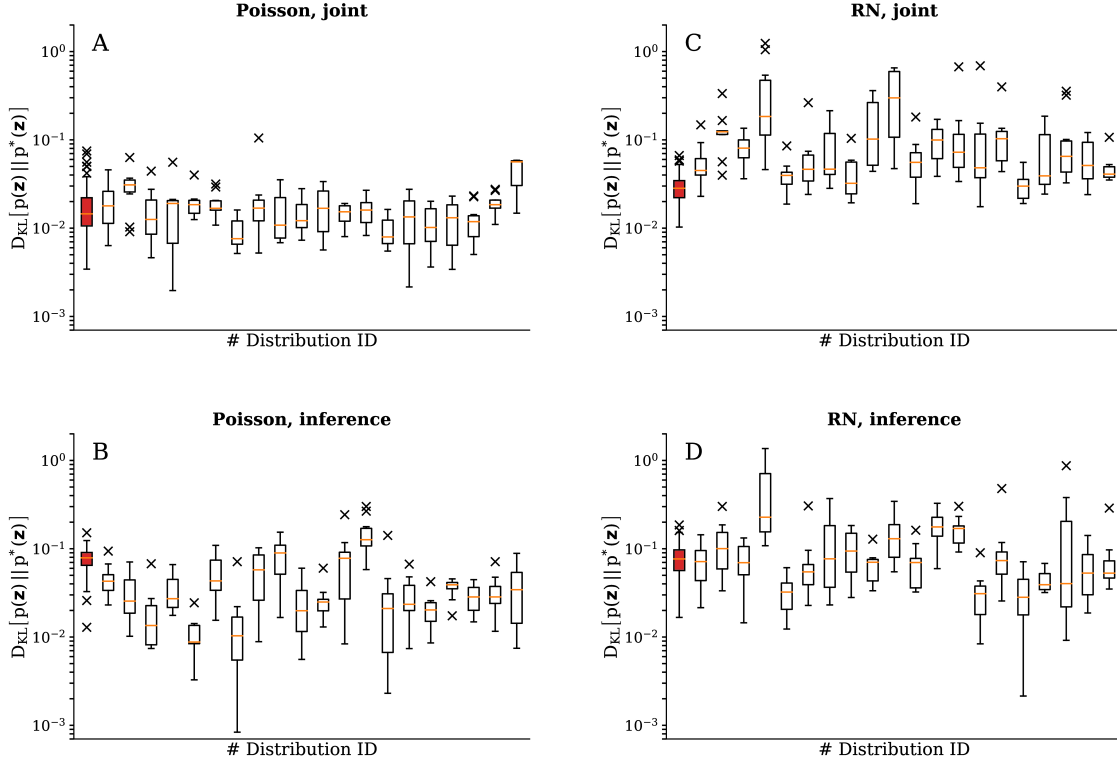
Figure 5: **Emulated SSNs sampling from different target Boltzmann distributions.** The figure shows the results of experiments identical to the ones in section 3.1 for 20 different target distributions with 10 repetitions for each sample. We show the $D_{\mathrm{KL}}(p \parallel p^*)$ of the test-run after training for **(A)** the joint distributions with Poisson noise, **(B)** the inference experiment with Poisson noise, **(C)** the joint distributions with a random background network and **(C)** the inference experiment with a random background network. The data is plotted following the traditional box-and-whiskers scheme: the orange line represents the median, the box represents the interquartile range, the whiskers represent the full data range and the $\times$ represent the far outliers. In each subplot the leftmost data (highlighted in red) corresponds to the distribution shown in fig. 4.

first reduced with nearest-neighbor resampling (`misc.imresize` function in the SciPy library (Jones et al., 2001–)) and then binarized around the median gray value over each image. We used all images from the original datasets (approx. 6000 per class) from 4 classes (0, 1, 4, 7) for rMNIST and 3 classes (**T**-shirts, **Tr**ousers, **S**neakers) for rFMNIST (fig. 6 A-B). The emulated SSNs consisted of 3 layers, with 144 visible, 60 hidden and either 4 label units for rMNIST or 3 for rFMNIST.

Pre-training was done on simulated classical RBMs using the CAST algorithm (Salakhutdinov, 2010). The pre-training provided a starting point for training on the hardware in order to accelerate the convergence of the in-the-loop training procedure. We use the performance of these RBMs in software simulations using Gibbs sampling as a reference for the results obtained with the hardware-emulated SSNs. After pre-training, we mapped these RBMs to approximately equivalent SSNs on the hardware, using an empirical translation factor based on an average activation function (fig. 3 C) to calculate the initial hardware synaptic weights from weights and biases of the RBMs. Especially for rMNIST, this resulted in a significant deterioration of the classification performance (fig. 6 C). After mapping, we continued training using the wake-sleep algorithm, with the hardware in the loop. While in the previous task it was possible to calculate the data term explicitly, it now had to be sampled as well. In order to ensure proper clamping, the synapses from the hidden to the label layer and from the hidden layer to the visible layer were turned off during the wake phase.
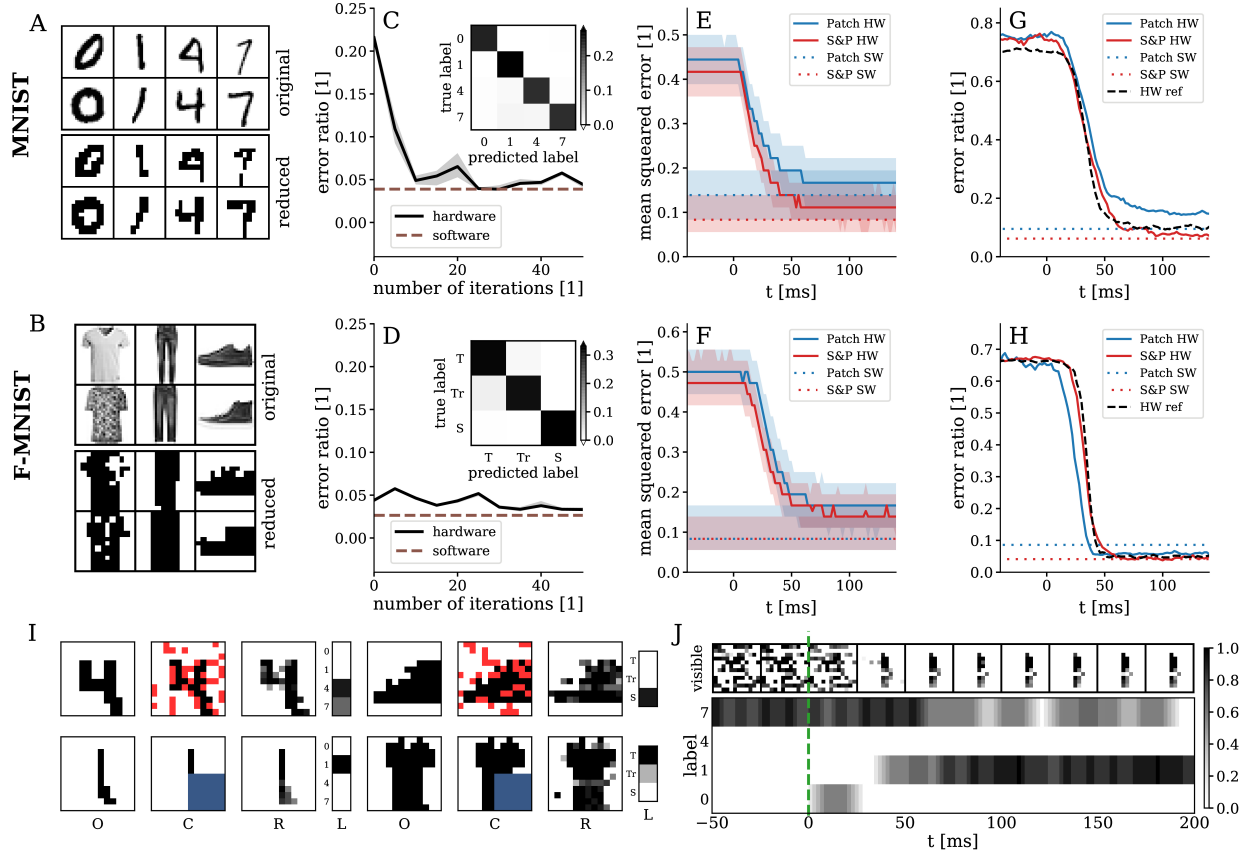
Figure 6: **Behavior of hierarchical SSNs trained on data.** Top row: rMNIST; middle row: rFMNIST; bottom row: exemplary setups for the partial occlusion scenarios. **(A-B)** Exemplary images from the rMNIST (A) and rFMNIST (B) datasets used for training and comparison to their MNIST and FMNIST originals. **(C-D)** Training with the hardware in the loop after translation of pre-trained parameters. Confusion matrices after training shown as insets. Performance of the reference RBMs shown as dashed brown lines. Results are given as median and interquartile values over 10 test runs. **(E-F)** Pattern completion and **(G-H)** error ratio of the inferred label for partially occluded images (blue: patch; red: salt&pepper). Solid lines represent median values and shaded areas show interquartile ranges over 250 test images per class. Performance of the reference RBMs shown as dashed lines. As a reference, we also show the error ratio of the SNNs on unoccluded images in (G) and (H). **(I)** Snapshots of the pattern completion experiments: O - original image, C - clamped image (red and blue pixels are occluded), R - response of the visible layer, L - response of the label layer. **(J)** Exemplary temporal evolution of a pattern completion experiment with patch occlusion. For better visualization of the activity in the visible layer in (J) and (I), we smoothed out its discretized response to obtain grayscale pixel values, by convolving its state vector with a box filter of $10\,\mathrm{ms}$ width.

The SSNs were tested for both their discriminative and their generative properties. For classification, the visible layer was clamped to images from the test set (black pixels correspond to $z_k = 1$ and white pixels to $z_k = 0$). Each image was presented for 500 biological milliseconds, which corresponds to $50\,\mu s$ wall-clock time. The neuron in the label layer with the highest firing rate was interpreted as the label predicted by the model. The spiking activity of the neurons was read out directly from the hardware, without additional off-chip post-processing. For both datasets, training was able to restore the performance lost in the translation of the abstract RBM to the hardware-emulated SSN. The emulated SSNs achieved error rates of $4.45^{+0.12}_{-0.36}\%$ on rMNIST and $3.32^{+0.27}_{-0.04}\%$ on rFMNIST. These values are close to the ones obtained by the reference RBMs: $3.89^{+0.10}_{-0.02}\%$ on rMNIST and $2.645^{+0.002}_{-0.010}\%$ on rFMNIST (fig. 6 C-D, confusion matrices shown as insets).

The gross wall-clock time needed to classify the 4125 images in the rMNIST test set was $10\,s$ ($2.4\,ms$ per image, $210\times$ speed-up). For the 3000 images in the rFMNIST test set, the emulation ran for $9.4\,s$ ($3.1\,ms$ per image; $160\times$ speed-up). This subsumes the runtime of the BrainScaleS software stack, hardware configuration and the network emulation. The runtime of the software-stack includes the translation from a PyNN-based network description to a corresponding hardware configuration. As before, the difference between the nominal acceleration factor and the effective speed-up stems from the I/O and initialization overhead of the hardware system.

To test the generative properties of our emulated SSNs, we set up two scenarios requiring them to perform pattern completion. For each class, 250 incomplete images were presented as inputs to the visible layer. For each image, 25% of visible neurons received no input, with the occlusion following two different schemes: salt&pepper (upper row in fig. 6 I) and patch (lower row in fig. 6 I). Each image was presented for $500\,ms$. In order to remove any initialization bias resulting from preceding images, random input was applied to the visible layer between consecutive images.

Reconstruction accuracy was measured using the mean squared error (MSE) between the reconstructed and original occluded pixels. For binary images, as in our case, the MSE reflects the average ratio of mis-reconstructed to total reconstructed pixels. Simultaneously, we also recorded the classification accuracy on the partially occluded images. After stimulus onset, the MSE converged from chance level ($\approx 50\%$) to its minimum ($\approx 10\%$) within $50\,ms$ (fig. 6 E-F). Given an average refractory period of $\approx 10\,ms$ (fig. 3 C), this suggests that the network was able to react to the input with no more than 5 spikes per neuron. For all studied scenarios, the reconstruction performance of the emulated SSNs closely matched the one achieved by the reference RBMs. Examples of image reconstruction are shown in fig. 6 I-J for both datasets and occlusion scenarios. The classification performance deteriorated only slightly compared to non-occluded images and also remained close to the performance of the reference RBMs (fig. 6 G-H). The temporal evolution of the classification error closely followed that of the MSE.

As a further test of the generative abilities of our hardware-emulated SSNs, we recorded the images produced by the visible layer during guided dreaming. In this task, the visible and hidden layers of the SSN evolved freely without external input, while the label layer was periodically clamped with external input such that exactly one of the label neurons was active at any time (enforced one-hot coding). In a perfect model, this would cause the visible layer to sample only from configurations compatible with the hidden layer, i.e., from images corresponding to that particular class. Between the clamping of consecutive labels, we injected 100ms random input to visible layer to facilitate the changing of the image. The SSNs were able to generate varied and recognizable pictures, within the limits imposed by the low resolution of the visible layer (fig. 7). For rMNIST, all used classes appeared in correct correspondence to the clamped label. For rFMNIST, images from the class "Sneakers" were not always triggered by the corresponding guidance from the label layer, suggesting that the learned modes in the energy landscape are too deep, and sneakers too dissimilar to T-shirts and Trousers, to allow good mixing during guided dreaming.

4   Discussion

This manuscript presents the first scalable demonstration of sampling-based probabilistic inference with spiking networks on a highly accelerated analog neuromorphic substrate. We trained fully connected spiking networks to
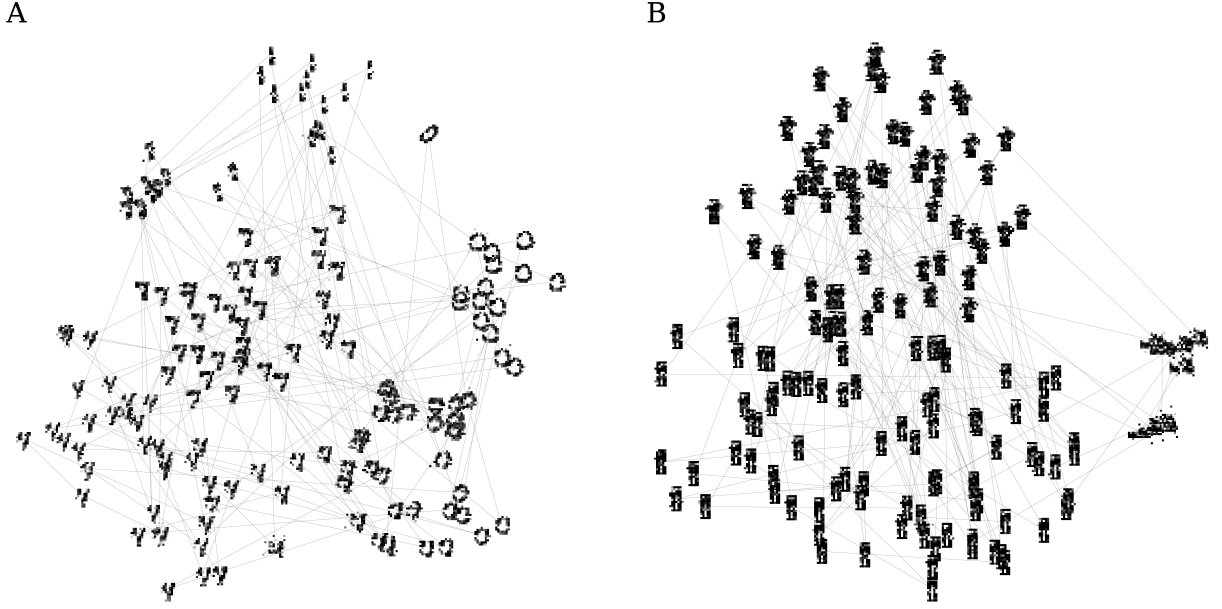
A

B

Figure 7: **Generated images during guided dreaming.** The visible state space, along with the position of the generated images within it, was projected to two dimensions using t-SNE (Maaten and Hinton, 2008). The thin lines connect consecutive samples. **(A)** rMNIST; **(B)** rFMNIST.

sample from target distributions and hierarchical spiking networks as discriminative and generative models of higher-dimensional input data. Despite the inherent variability of the analog substrate, we were able to achieve performance levels comparable to those of software simulations in several benchmark tasks, while maintaining a significant overall acceleration factor compared to systems that operate in biological real time. Importantly, by co-embedding the generation of stochasticity within the same substrate, we demonstrated the viability of a fully embedded neural sampling model with significantly reduced demands on off-substrate I/O bandwidth. Having a fully embedded implementation allows the runtime of the experiments to scale as $\mathcal{O}(1)$ with the size of the emulated network; this is inherent to the nature of physical emulation, for which wall-clock runtime only depends on the emulated time in the biological reference frame. In the following, we address the limitations of our study, point out links to related work and discuss its implications within the greater context of computational neuroscience and bio-inspired AI.

4.1 Limitations and constraints

The most notable limitation imposed by the current commissioning state of the BrainScaleS system was on the size of the emulated SSNs. At the time of writing, due to limited software flexibility, system assembly and substrate yield, the usable hardware real-estate was reduced to a patchy and non-contiguous area, thereby strongly limiting the maximum connectivity between different locations within this area. In order to limit synapse loss to small values (below $2\,\%$), we restricted ourselves to using a small but contiguous functioning area of the wafer, which in turn limited the maximum size of our SSNs and noise-generating RNs. Ongoing improvements in post-production and assembly, as well as in the mapping and routing software, are expected to enhance on-wafer connectivity and thereby automatically increase the size of emulable networks, as the architecture of our SSNs scales naturally to such an increase in hardware resources.

To a lesser extent, the sampling accuracy was also affected by the limited precision of hardware parameter control. The writing of analog parameters exhibits significant trial-to-trial variability; in any given trial, this leads to a heterogeneous substrate, which is known to reduce the sampling accuracy (Probst et al., 2015). Most of this variability is compensated during learning, but the $4\,\mathrm{bit}$ resolution of the synaptic weights and the imperfect symmetry in the effective weight matrix due to analog variability of the synaptic circuits ultimately limit the ability of the SSN to approximate target

13

distributions. This leads to the "jumping" behavior of the $D_{\mathrm{KL}}(p \parallel p^*)$ in the final stages of learning (fig. 4 A). In smaller networks, synaptic weight resolution is a critical performance modifier (Petrovici et al., 2017b). However, the penalty imposed by a limited synaptic weight resolution is known to decrease for larger deep networks with more and larger hidden layers, both spiking and non-spiking (Courbariaux et al., 2015; Petrovici et al., 2017a). Furthermore, the successor system (BrainScaleS-2, Aamir et al., 2016) is designed with a 6-bit weight resolution.

In the setup we used shared bias neurons for several neurons in the sampling network. This helped us save hardware resources, thus allowing the emulation of larger functional networks. Such bias neuron sharing is expected to introduce some small amount of temporal correlations between the sampling neurons. However, this effect was too small to observe in our experiments for several reasons. First, the high firing rate of the bias neurons helped smooth out the bias voltage induced into the sampling neurons. Second, the different delays and spike timing jitter on the hardware reduces such cross-correlations. Third, other dominant limitations overshadow the effect of shared bias neurons. In any case, the used training procedure inherently compensates for excess cross-correlations, thus effectively removing any distortions to the target distribution that this effect might introduce (Bytschok et al., 2017; Dold et al., 2019).

In the current setup, our SSNs displayed limited mixing abilities. During guided dreaming, images from one of the learned classes were more difficult to generate (fig. 7). Restricted mixing due to deep modes in the energy landscape carved out by contrastive learning is a well-known problem for classical Boltzmann machines, which is usually alleviated by computationally costly annealing techniques (Salakhutdinov, 2010; Desjardins et al., 2010; Bengio et al., 2013). However, the fully-commissioned BrainScaleS system will feature embedded short-term synaptic plasticity (Schemmel et al., 2010), which has been shown to promote mixing in spiking networks (Leng et al., 2018) while operating purely locally, at the level of individual synapses.

Currently, the execution speed of emulation runs is dominated by the I/O overhead, which in turn is mostly spent on setting up the experiment. This leads to the classification (section 3.2) of one image taking $2.4\,\mathrm{ms}$ to $3.9\,\mathrm{ms}$, whereas the pure network runtime is merely $50\,\mu\mathrm{s}$. A streamlining of the software layer that performs this setup is expected to significantly reduce this discrepancy.

The synaptic learning rule was local and Hebbian, but updates were calculated on a host computer using an iterative in-the-loop training procedure, which required repeated stopping, evaluation and restart of the emulation, thereby reducing the nominal acceleration factor of 10^4 by two orders of magnitude. By utilizing on-chip plasticity, as available, for example, on the BrainScaleS-2 successor system (Friedmann et al., 2017; Wunderlich et al., 2019), this laborious procedure becomes obsolete and the accelerated nature of the substrate can be exploited to its fullest extent.

4.2 Relation to other work

This study builds upon a series of theoretical and experimental studies of sampling-based probabilistic inference using the dynamics of biological neurons. The inclusion of refractory times was first considered in (Buesing et al., 2011). An extension to networks of leaky integrate-and-fire neurons and a theoretical framework for their dynamics and statistics followed in (Petrovici et al., 2013) and (Petrovici et al., 2016). The compensation of shared-input correlations through inhibitory feedback and learning was discussed in (Jordan et al., 2017), (Bytschok et al., 2017) and (Dold et al., 2019), inspired by the early study of asynchronous irregular firing in (Brunel, 2000) and by preceding correlation studies in theoretical (Tetzlaff et al., 2012) and experimental (Pfeil et al., 2016) work.

Previous small-scale studies of sampling on accelerated mixed-signal neuromorphic hardware include (Petrovici et al., 2015, 2017b,a). An implementation of sampling with spiking neurons and its application to the MNIST dataset was shown in (Pedroni et al., 2016) using the fully digital, real-time TrueNorth neuromorphic chip (Merolla et al., 2014).

We stress two important differences between (Pedroni et al., 2016) and this work. First, the nature of the neuromorphic substrate: the TrueNorth system is fully digital and calculates neuronal state updates numerically, in contrast to the physical-model paradigm instantiated by BrainScaleS. In this sense, TrueNorth emulations are significantly closer to

classical computer simulations on parallel machines: updates of dynamical variables are precise and robustness to variability is not an issue; however TrueNorth typically runs in biological real time (Merolla et al., 2014; Akopyan et al., 2015), which is 10.000 times slower than BrainScaleS. Second, the nature of neuron dynamics: the neuron model used in (Pedroni et al., 2016) is an intrinsically stochastic unit that sums its weighted inputs, thus remaining very close to classical Gibbs sampling and Boltzmann machines, while our approach considers multiple additional aspects of its biological archetype (exponential synaptic kernels, leaky membranes, deterministic firing, stochasticity through synaptic background, shared-input correlations etc.). Moreover, our approach uses fewer hardware neuron units to represent a sampling unit, enabling a more parsimonious utilization of the neuromorphic substrate.

4.3 Conclusion

In this work we showed how sampling-based Bayesian inference using hierarchical spiking networks can be robustly implemented on a physical model system despite inherent variability and imperfections. Underlying neuron and synapse dynamics are deterministic and close to their biological archetypes, but with much shorter time constants, hence the intrinsic acceleration factor of 10^4 with respect to biology. The entire architecture – sampling network plus background random network – was fully deterministic and entirely contained on the neuromorphic substrate, with external communication used only to represent input patterns and labels. Considering the deterministic nature of neurons in vitro (Mainen and Sejnowski, 1995; Reinagel and Reid, 2002; Toups et al., 2012), such an architecture also represents a plausible model for neural sampling in cortex (Jordan et al., 2017; Dold et al., 2019).

We demonstrated sampling from arbitrary Boltzmann distributions over binary random variables, as well as generative and discriminative properties of networks trained with visual data. The framework can be extended to sampling from arbitrary probability distributions over binary random variables, as it was shown in software simulations (Probst et al., 2015). For such networks, the two abovementioned computational tasks (pattern completion and classification) happen simultaneously, as they both require the calculation of conditional distributions, which is carried out implicitly by the network dynamics. Both during learning and for the subsequent inference tasks, the setup benefitted significantly from the fast intrinsic dynamics of the substrate, achieving a net speedup of 100 to 210 compared to biology.

We view these results as a contribution to the nascent, but expanding field of applications for biologically inspired physical-model systems. They demonstrate the feasibility of such devices for solving problems in machine learning, as well as for studying biological phenomena. Importantly, they explicitly addresses the search for robust computational models that are able to harness the strengths of these systems, most importantly their speed and energy efficiency. The proposed architecture scales naturally to substrates with more neuronal real-estate and can be used for a wide array of tasks that can be mapped to a Bayesian formulation, such as constraint satisfaction problems (Jonke et al., 2016; Fonseca Guerra and Furber, 2017), prediction of temporal sequences (Sutskever and Hinton, 2007), movement planning (Taylor and Hinton, 2009; Alemi et al., 2015), simulation of solid-state systems (Edwards and Anderson, 1975) and quantum many-body problems (Carleo and Troyer, 2017; Czischek et al., 2018).

Conflict of Interest Statement

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Author Contributions

Akos F. Kungl (AFK), Andreas Baumbach (AB), Dominik Dold (DD), Luziwei Leng (LL), Sebastian Schmitt (SS), Paul Müller (PM) and Mihai A. Petrovici (MAP) designed the study. AFK conducted the experiments and the evaluations. Nico Gürtler (NG) contributed to the evaluations and provided software support for the evaluation. AFK wrote the initial manuscript. Eric Müller (ECM), Christian Mauch (CM), Johann Klähn (JK), Sebastian Schmitt (SS), Kai Husmann

(KH) and Oliver Breitwieser (OB) supported experiment realization; ECM coordinated the software development for the neuromorphic systems. Alexander Kugele (AK), Christoph Koke (CK) and Mitja Kleider (MK) contributed with characterization, calibration testing and debugging of the system. Andreas Grübl (AG), Dan Husmann (DH), Maurice Güttler (MG) were responsible for system assembly. AG did the digital front- and back-end implementation. Vitali Karasenko (VK) provided FPGA firmware and supported system commissioning. Johannes Schemmel (JS) is the architect and lead designer of the neuromorphic platform. MAP, Karlheinz Meier (KM), JS, SS and ECM provided conceptual and scientific advice. All authors contributed to the final manuscript.

Supplemental Data

Two videos can be found in the online version of the article that exemplify fully-embedded sampling from a target distribution and fully-embedded pattern completion on the BrainScaleS-1.

References

Aamir, S. A., Müller, P., Hartel, A., Schemmel, J., and Meier, K. (2016). A highly tunable 65-nm cmos lif neuron for a large scale neuromorphic system. In *European Solid-State Circuits Conference, ESSCIRC Conference 2016: 42nd* (IEEE), 71–74

Aitchison, L. and Lengyel, M. (2016). The hamiltonian brain: efficient probabilistic inference with excitatory-inhibitory neural circuit dynamics. *PLoS computational biology* 12, e1005186

Akopyan, F., Sawada, J., Cassidy, A., Alvarez-Icaza, R., Arthur, J., Merolla, P., et al. (2015). Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34, 1537–1557

Alemi, O., Li, W., and Pasquier, P. (2015). Affect-expressive movement generation with factored conditional restricted boltzmann machines. In *2015 International Conference on Affective Computing and Intelligent Interaction (ACII)* (IEEE), 442–448

Bengio, Y., Mesnil, G., Dauphin, Y., and Rifai, S. (2013). Better mixing via deep representations. In *International conference on machine learning*. 552–560

Berkes, P., Orbán, G., Lengyel, M., and Fiser, J. (2011). Spontaneous cortical activity reveals hallmarks of an optimal internal model of the environment. *Science* 331, 83–87

Brette, R. and Gerstner, W. (2005). Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *Journal of neurophysiology* 94, 3637–3642

Brunel, N. (2000). Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons. *Journal of computational neuroscience* 8, 183–208

Buesing, L., Bill, J., Nessler, B., and Maass, W. (2011). Neural dynamics as sampling: a model for stochastic computation in recurrent networks of spiking neurons. *PLoS computational biology* 7, e1002211

Bytschok, I., Dold, D., Schemmel, J., Meier, K., and Petrovici, M. A. (2017). Spike-based probabilistic inference with correlated noise. In *BMC Neuroscience 2017* (Organization for Computational Neurosciences), vol. 18, P200

Carleo, G. and Troyer, M. (2017). Solving the quantum many-body problem with artificial neural networks. *Science* 355, 602–606

Chang, Y.-F., Fowler, B., Chen, Y.-C., Zhou, F., Pan, C.-H., Chang, T.-C., et al. (2016). Demonstration of synaptic behaviors and resistive switching characterizations by proton exchange reactions in silicon oxide. *Scientific reports* 6, 21268

Courbariaux, M., Bengio, Y., and David, J.-P. (2015). Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*. 3123–3131

Czischek, S., Gärttner, M., and Gasenzer, T. (2018). Quenches near ising quantum criticality as a challenge for artificial neural networks. *Physical Review B* 98, 024311

Davison, A. P., Brüderle, D., Eppler, J. M., Kremkow, J., Muller, E., Pecevski, D., et al. (2009). Pynn: a common interface for neuronal network simulators. *Frontiers in neuroinformatics* 2, 11

Desjardins, G., Courville, A., Bengio, Y., Vincent, P., and Delalleau, O. (2010). Parallel tempering for training of restricted boltzmann machines. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (MIT Press Cambridge, MA), 145–152

Destexhe, A., Rudolph, M., and Paré, D. (2003). The high-conductance state of neocortical neurons in vivo. *Nature reviews neuroscience* 4, 739

Dold, D., Bytschok, I., Kungl, A. F., Baumbach, A., Breitwieser, O., Senn, W., et al. (2019). Stochasticity from function — why the bayesian brain may need no noise. *Neural Networks* 119, 200–213

Edwards, S. F. and Anderson, P. W. (1975). Theory of spin glasses. *Journal of Physics F: Metal Physics* 5, 965

Esser, S. K., Merolla, P. A., Arthur, J. V., Cassidy, A. S., Appuswamy, R., Andreopoulos, A., et al. (2016). From the cover: Convolutional networks for fast, energy-efficient neuromorphic computing. *Proceedings of the National Academy of Sciences of the United States of America* 113, 11441

Fonseca Guerra, G. A. and Furber, S. B. (2017). Using stochastic spiking neural networks on spinnaker to solve constraint satisfaction problems. *Frontiers in neuroscience* 11, 714

Friedmann, S., Schemmel, J., Grübl, A., Hartel, A., Hock, M., and Meier, K. (2017). Demonstrating hybrid learning in a flexible neuromorphic hardware system. *IEEE transactions on biomedical circuits and systems* 11, 128–142

Furber, S. (2016). Large-scale neuromorphic computing systems. *Journal of neural engineering* 13, 051001

Haefner, R. M., Berkes, P., and Fiser, J. (2016). Perceptual decision-making as probabilistic inference by neural sampling. *Neuron* 90, 649–660

Hennequin, G., Aitchison, L., and Lengyel, M. (2014). Fast sampling for bayesian inference in neural circuits. *arXiv preprint arXiv:1404.3521*

Hinton, G. E. (2012). A practical guide to training restricted boltzmann machines. In *Neural networks: Tricks of the trade* (Springer). 599–619

Hinton, G. E., Dayan, P., Frey, B. J., and Neal, R. M. (1995). The" wake-sleep" algorithm for unsupervised neural networks. *Science* 268, 1158–1161

Hinton, G. E., Sejnowski, T. J., and Ackley, D. H. (1984). *Boltzmann machines: Constraint satisfaction networks that learn* (Carnegie-Mellon University, Department of Computer Science Pittsburgh, PA)

Indiveri, G., Chicca, E., and Douglas, R. J. (2006). A vlsi array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity. *IEEE transactions on neural networks* 17

Indiveri, G., Linares-Barranco, B., Hamilton, T. J., Van Schaik, A., Etienne-Cummings, R., Delbruck, T., et al. (2011). Neuromorphic silicon neuron circuits. *Frontiers in neuroscience* 5, 73

Jo, S. H., Chang, T., Ebong, I., Bhadviya, B. B., Mazumder, P., and Lu, W. (2010). Nanoscale memristor device as synapse in neuromorphic systems. *Nano letters* 10, 1297–1301

[Dataset] Jones, E., Oliphant, T., Peterson, P., et al. (2001–). SciPy: Open source scientific tools for Python

Jonke, Z., Habenschuss, S., and Maass, W. (2016). Solving constraint satisfaction problems with networks of spiking neurons. *Frontiers in neuroscience* 10, 118

Jordan, J., Petrovici, M. A., Breitwieser, O., Schemmel, J., Meier, K., Diesmann, M., et al. (2017). Stochastic neural computation without noise. *arXiv preprint arXiv:1710.04931*

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics* 22, 79–86

Kutschireiter, A., Surace, S. C., Sprekeler, H., and Pfister, J.-P. (2017). Nonlinear bayesian filtering and learning: a neuronal dynamics for perception. *Scientific reports* 7, 8722

Lande, T. S., Ranjbar, H., Ismail, M., and Berg, Y. (1996). An analog floating-gate memory in a standard digital technology. In *Proceedings of fifth international conference on microelectronics for neural networks* (IEEE), 271–276

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86, 2278–2324

Leng, L., Martel, R., Breitwieser, O., Bytschok, I., Senn, W., Schemmel, J., et al. (2018). Spiking neurons with short-term synaptic plasticity form superior generative networks. *Scientific reports* 8, 10651

Loock, J.-P. (2006). Evaluierung eines floating gate analogspeichers für neuronale netze in single-poly umc 180nm cmos-prozess. *Diploma thesis (in German), University of Heidelberg, HD-KIP-06-47*

Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research* 9, 2579–2605

Mainen, Z. F. and Sejnowski, T. J. (1995). Reliability of spike timing in neocortical neurons. *Science* 268, 1503–1506

Mead, C. (1990). Neuromorphic electronic systems. *Proceedings of the IEEE* 78, 1629–1636

Merolla, P. A., Arthur, J. V., Alvarez-Icaza, R., Cassidy, A. S., Sawada, J., Akopyan, F., et al. (2014). A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* 345, 668–673

Millner, S. (2012). *Development of a multi-compartment neuron model emulation*. Ph.D. thesis

Millner, S., Grübl, A., Meier, K., Schemmel, J., and Schwartz, M.-O. (2010). A VLSI implementation of the adaptive exponential integrate-and-fire neuron model. In *Adv Neur In*, eds. J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta. vol. 23, 1642–1650

Orbán, G., Berkes, P., Fiser, J., and Lengyel, M. (2016). Neural variability and sampling-based probabilistic representations in the visual cortex. *Neuron* 92, 530–543

Pedroni, B. U., Das, S., Arthur, J. V., Merolla, P. A., Jackson, B. L., Modha, D. S., et al. (2016). Mapping generative models onto a network of digital spiking neurons. *IEEE transactions on biomedical circuits and systems* 10, 837–854

Petrovici, M. A. (2016). *Form Versus Function: Theory and Models for Neuronal Substrates* (Springer)

Petrovici, M. A., Bill, J., Bytschok, I., Schemmel, J., and Meier, K. (2013). Stochastic inference with deterministic spiking neurons. *arXiv preprint arXiv:1311.3211*

Petrovici, M. A., Bill, J., Bytschok, I., Schemmel, J., and Meier, K. (2016). Stochastic inference with spiking neurons in the high-conductance state. *Physical Review E* 94, 042312

Petrovici, M. A., Schmitt, S., Klähn, J., Stöckel, D., Schroeder, A., Bellec, G., et al. (2017a). Pattern representation and recognition with accelerated analog neuromorphic systems. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)* (IEEE), 1–4

Petrovici, M. A., Schroeder, A., Breitwieser, O., Grübl, A., Schemmel, J., and Meier, K. (2017b). Robustness from structure: Inference with hierarchical spiking networks on analog neuromorphic hardware. In *Neural Networks (IJCNN), 2017 International Joint Conference on* (IEEE), 2209–2216

Petrovici, M. A., Stöckel, D., Bytschok, I., Bill, J., Pfeil, T., Schemmel, J., et al. (2015). Fast sampling with neuromorphic hardware. In *Advances in Neural Information Processing Systems (NIPS)*. vol. 28

Petrovici, M. A., Vogginger, B., Müller, P., Breitwieser, O., Lundqvist, M., Muller, L., et al. (2014). Characterization and compensation of network-level anomalies in mixed-signal neuromorphic modeling platforms. *PloS one* 9, e108590

Pfeil, T., Grübl, A., Jeltsch, S., Müller, E., Müller, P., Petrovici, M. A., et al. (2013). Six networks on a universal neuromorphic computing substrate. *Frontiers in neuroscience* 7, 11

Pfeil, T., Jordan, J., Tetzlaff, T., Grübl, A., Schemmel, J., Diesmann, M., et al. (2016). Effect of heterogeneity on decorrelation mechanisms in spiking neural networks: A neuromorphic-hardware study. *Physical Review X* 6, 021023

Pouget, A., Beck, J. M., Ma, W. J., and Latham, P. E. (2013). Probabilistic brains: knowns and unknowns. *Nature neuroscience* 16, 1170

Probst, D., Petrovici, M. A., Bytschok, I., Bill, J., Pecevski, D., Schemmel, J., et al. (2015). Probabilistic inference in discrete spaces can be implemented into networks of lif neurons. *Frontiers in computational neuroscience* 9, 13

Qiao, N., Mostafa, H., Corradi, F., Osswald, M., Stefanini, F., Sumislawska, D., et al. (2015). A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses. *Frontiers in neuroscience* 9, 141

Reinagel, P. and Reid, R. C. (2002). Precise firing events are conserved across neurons. *Journal of Neuroscience* 22, 6837–6841

Salakhutdinov, R. (2010). Learning deep boltzmann machines using adaptive mcmc. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. 943–950

Savitzky, A. and Golay, M. J. (1964). Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry* 36, 1627–1639

Schemmel, J., Brüderle, D., Grübl, A., Hock, M., Meier, K., and Millner, S. (2010). A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *Circuits and systems (ISCAS), proceedings of 2010 IEEE international symposium on* (IEEE), 1947–1950

Schmitt, S., Klähn, J., Bellec, G., Grübl, A., Guettler, M., Hartel, A., et al. (2017). Neuromorphic hardware in the loop: Training a deep spiking network on the brainscales wafer-scale system. In *Neural Networks (IJCNN), 2017 International Joint Conference on* (IEEE), 2227–2234

Schmuker, M., Pfeil, T., and Nawrot, M. P. (2014). A neuromorphic network for generic multivariate data classification. *Proceedings of the National Academy of Sciences* 111, 2081–2086

Sutskever, I. and Hinton, G. (2007). Learning multilevel distributed representations for high-dimensional sequences. In *Artificial intelligence and statistics*. 548–555

Taylor, G. W. and Hinton, G. E. (2009). Factored conditional restricted boltzmann machines for modeling motion style. In *Proceedings of the 26th annual international conference on machine learning* (ACM), 1025–1032

Tetzlaff, T., Helias, M., Einevoll, G. T., and Diesmann, M. (2012). Decorrelation of neural-network activity by inhibitory feedback. *PLoS computational biology* 8, e1002596

Toups, J. V., Fellous, J.-M., Thomas, P. J., Sejnowski, T. J., and Tiesinga, P. H. (2012). Multiple spike time patterns occur at bifurcation points of membrane potential dynamics. *PLoS computational biology* 8, e1002615

Waldrop, M. M. (2016). The chips are down for Moore's law. *Nature News* 530, 144

Wunderlich, T., Kungl, A. F., Müller, E., Hartel, A., Stradmann, Y., Aamir, S. A., et al. (2019). Demonstrating advantages of neuromorphic computation: A pilot study. *Frontiers in Neuroscience* 13, 260

[Dataset] Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms

Zoschke, K., Güttler, M., Böttcher, L., Grübl, A., Husmann, D., Schemmel, J., et al. (2017). Full wafer redistribution and wafer embedding as key technologies for a multi-scale neuromorphic hardware cluster. *EPTC 2017*

Tables

Table 1: **Description of the neuron and synapse model.** The variables are described including their numerical values in the experiment in table 2.

Type	Leaky integrate-and-fire (LIF), conductance based synapse, exponential kernel
Subthreshold dynamics	Subthreshold dynamics $[t \notin [t_{\mathrm{sp}}, t_{\mathrm{sp}} + \tau_{\mathrm{ref}})]$: $C_{\mathrm{m}}(\mathrm{d}/\mathrm{d}t)u(t) = -g_{\mathrm{l}}[u(t) - E_{\mathrm{leak}}] - g_{\mathrm{syn}}^{\mathrm{inh}}(t)[u(t) - E_{\mathrm{inh}}] - g_{\mathrm{syn}}^{\mathrm{exc}}(t)[u(t) - E_{\mathrm{exc}}]$
Reset and refractoriness	$[t \in [t_{\mathrm{sp}}, t_{\mathrm{sp}} + \tau_{\mathrm{ref}})]$: $u(t) = V_{\mathrm{reset}}$ This model was emulated on the BrainScaleS system (Schemmel et al., 2010)
Spiking	If $u(t)$ crosses V_{thresh} from below at $t = t_{\mathrm{sp}}$, neuron emits a spike with timestamp t_{sp}
Synapse dynamics	For each presynaptic spike at t_{sp} : $g_{\mathrm{syn}}(t) = J \exp[-(t - t_{\mathrm{sp}} - d)/(\tau_{\mathrm{syn}})]\theta(t - t_{\mathrm{sp}} - d)$ where J is the synaptic weight, d the synaptic delay and θ the Heaviside function This model was emulated on the BrainScaleS system (Schemmel et al., 2010)

Table 2: **Neuron parameters.** Parameters of the network setup specified in table 1. The analog parameters are shown as specified in the software setup and not as realized on the hardware. For details on the calibration procedure see, e.g., (Schmitt et al., 2017). *Legend:* * the calibration of the membrane time constant was not available at the time of this work, and the corresponding technical parameter was set to the smallest available value instead (fastest possible membrane dynamics for each neuron).

A		Sampling neuron
Name	*Value*	*Description*
V_{reset}	$-35\,\text{mV}$	reset potential
E_{leak}	$-20\,\text{mV}$	resting potential
V_{thresh}	$-20\,\text{mV}$	threshold potential
E_{inh}	$-100\,\text{mV}$	inhibitory reversal potential
E_{exc}	$60\,\text{mV}$	excitatory reversal potential
τ_{ref}	$4\,\text{ms}$	refractory time
τ_{mem}	ca. $7\,\text{ms}$	membrane time constant*
C_{mem}	$0.2\,\text{nF}$	membrane capacity
$\tau_{\text{syn}}^{\text{exc}}$	$8\,\text{ms}$	excitatory synaptic time constant
$\tau_{\text{syn}}^{\text{inh}}$	$8\,\text{ms}$	inhibitory synaptic time constant

B		Bias neuron
Name	*Value*	*Description*
V_{reset}	$-30\,\text{mV}$	reset potential
E_{leak}	$60\,\text{mV}$	resting potential
V_{thresh}	$-20\,\text{mV}$	threshold potential
E_{inh}	$-100\,\text{mV}$	inhibitory reversal potential
E_{exc}	$60\,\text{mV}$	excitatory reversal potential
τ_{ref}	$1.5\,\text{ms}$	refractory time
τ_{mem}	ca. $7\,\text{ms}$	membrane time constant*
C_{mem}	$0.2\,\text{nF}$	membrane capacity
$\tau_{\text{syn}}^{\text{exc}}$	$5\,\text{ms}$	excitatory synaptic time constant
$\tau_{\text{syn}}^{\text{inh}}$	$5\,\text{ms}$	inhibitory synaptic time constant

C		Neurons of the random network
Name	*Value*	*Description (all analog)*
V_{reset}	$-60\,\text{mV}$	reset potential
E_{leak}	$-10\,\text{mV}$	resting potential
V_{thresh}	$-20\,\text{mV}$	threshold potential
E_{inh}	$-100\,\text{mV}$	inhibitory reversal potential
E_{exc}	$60\,\text{mV}$	excitatory reversal potential
τ_{ref}	$4\,\text{ms}$	refractory time
τ_{mem}	ca. $7\,\text{ms}$	membrane time constant*
C_{mem}	$0.2\,\text{nF}$	membrane capacity
$\tau_{\text{syn}}^{\text{exc}}$	$8\,\text{ms}$	excitatory synaptic time constant
$\tau_{\text{syn}}^{\text{inh}}$	$8\,\text{ms}$	inhibitory synaptic time constant

D		Synapse
Name	*Value*	*Description*
w_{bias}	[0,15]	synaptic bias weight in hardware values (digital)
w_{network}	[0,15]	synaptic network weight in hardware values (digital)
d	on the order of $1\,\text{ms}$ (uncalibrated)	synaptic delay, estimated in (Schemmel et al., 2010)

Table 3: **Parameters for learning.** We did not carry any systematic hyper-parameter optimization. Note that the used learning parameters in the experiments in section 3.1 are not directly comparable because the different statistics of the background noise (Poisson or random network) correspond to different effective learning rates.

Experiment	Learning rate	Momentum factor	minibatch-size	Initial $(\mathbf{W}, \mathbf{b})$
target distribution, Poisson	1.0	0.6	-	$\mathcal{U}(-15, 15)$
target distribution, random network	0.5	0.6	-	$\mathcal{U}(-15, 15)$
rMNIST	0.4	0.6	7/class	pre-trained
rFMNIST	0.4	0.6	7/class	pre-trained

Table 4: **Network parameters.** Parameters are shown for the three different cases described in the manuscript: **A** Target Boltzmann distribution, Poisson noise. **B** Target Boltzmann distribution, random network for stochasticity. **C** Learning from data, random network for stochasticity. Note that the *in-degree*, sometimes also referred to as a *fan-in factor*, represents a neuron's number of pre-synaptic partners coming from some specific population.

A	**Probability distribution with Poisson Noise**	
Name	*Value*	*Description*
N_{s}	5	number of sampling neurons
N_{b}	1	number of bias neurons
N_{r}	0	number of random neurons
K_{RN}	-	within-population in-degree of neurons in the random network
K_{noise}	-	in-degree of sampling neurons from the random network
w_{RN}	-	synaptic weights in the random network in hardware units
$\nu_{\mathrm{Poisson}}^{\mathrm{e/i}}$	300 Hz	Poisson frequency to sampling neurons per synapse type
B	**Probability distribution with random network**	
Name	*Value*	*Description*
N_{s}	5	number of sampling neurons
N_{b}	1	number of bias neurons
N_{r}	200	number of random neurons
K_{RN}	20	within-population in-degree of neurons in the random network
K_{noise}	15	in-degree of sampling neurons from the random network
w_{RN}	10	synaptic weights in the random network in hardware units
$\nu_{\mathrm{Poisson}}^{\mathrm{e/i}}$	-	Poisson frequency to sampling neurons per synapse type
C	**High-dimensional dataset**	
Name	*Value*	*Description*
N_{s}	$\{207, 208\}$	number of sampling neurons, { rFMNIST, rMNIST }
N_{b}	1	number of bias neurons
N_{r}	400	number of random neurons
K_{RN}	20	within-population in-degree of neurons in the random network
K_{noise}	15	in-degree of sampling neurons from the random network
w_{RN}	10	synaptic weights in the random network in hardware units
$\nu_{\mathrm{Poisson}}^{\mathrm{e/i}}$	-	Poisson frequency to sampling neurons per synapse type