arXiv:1703.06043v2 [q-bio.NC] 3 Jul 2017

Pattern representation and recognition with accelerated analog neuromorphic systems

M. A. Petrovici[†‡] S. Schmitt[†] J. Klähn[†] D. Stöckel[†] A. Schroeder[†]
G. Bellec[‖] J. Bill[†] O. Breitwieser[†] I. Bytschok[†] A. Grübl[†] M. Güttler[†] A. Hartel[†] S. Hartmann[§] D. Husmann[†] K. Husmann[†]
S. Jeltsch[†] V. Karasenko[†] M. Kleider[†] C. Koke[†] A. Kononov[†] C. Mauch[†] E. Müller[†] P. Müller[†] J. Partzsch[§] T. Pfeil[†] S. Schiefer[§]
S. Scholze[§] A. Subramoney[‖] V. Thanasoulis[§] B. Vogginger[§] R. Legenstein[‖] W. Maass[‖] R. Schüffny[§] C. Mayr[§] J. Schemmel[†] K. Meier[†]
[†] Heidelberg University, Kirchhoff-Institute for Physics, Im Neuenheimer Feld 227, D-69120 Heidelberg
[‡] University of Bern, Department of Physiology, Bühlplatz 5, CH-3012 Bern
[§]Technische Universität Dresden, Chair of Highly-Parallel VLSI-Systems and Neuromorphic Circuits, D-01062 Dresden
[‖]Graz University of Technology, Institute for Theoretical Computer Science, A-8010 Graz

***Abstract*—Despite being originally inspired by the central nervous system, artificial neural networks have diverged from their biological archetypes as they have been remodeled to fit particular tasks. In this paper, we review several possibilites to reverse map these architectures to biologically more realistic spiking networks with the aim of emulating them on fast, low-power neuromorphic hardware. Since many of these devices employ analog components, which cannot be perfectly controlled, finding ways to compensate for the resulting effects represents a key challenge. Here, we discuss three different strategies to address this problem: the addition of auxiliary network components for stabilizing activity, the utilization of inherently robust architectures and a training method for hardware-emulated networks that functions without perfect knowledge of the system's dynamics and parameters. For all three scenarios, we corroborate our theoretical considerations with experimental results on accelerated analog neuromorphic platforms.**

I. Introduction

Artificial neural networks (ANNs) rank among the most successful classes of machine learning models, but are – superficial similarities to sensory processing pathways in cortex notwithstanding – difficult to map to biologically realistic spiking neural networks. Nevertheless, we argue that such a reverse mapping is worthwhile for two reasons. First, it could help us understand information processing in the brain – assuming that it follows similar computational principles. Second, it enables machine learning applications on fast, low-power neuromorphic architectures that are specifically developed to mimic biological neuro-synaptic dynamics. In this manuscript, we discuss several ways to answer what we consider to be a key challenge for neuromorphic architectures with analog components: Is it possible to design spiking architectures and training methods that are amenable to neuromorphic implementation and remain functionally performant despite substrate-inherent imperfections?

More specifically, we review three different approaches [1]–[3]. The first two are based on recent insights about how networks of spiking neurons can be constructed to sample from predefined joint probability distributions [4], [5]. When these distributions are learned from data, these networks automatically build an internal, generative model, which is then straightforward to use for pattern recognition and memory recall [6]. Practical problems arise when the hardware dynamics and parameter ranges are incompatible to the target specifications of the network, as these inevitably distort the sampled distribution. The first approach involves the addition of auxiliary network components in order to make it robust to hardware-induced distortions (Sec. II). The second one restricts the network topology in a way that endows it with immunity to some of these effects (Sec. III). We demonstrate the effectiveness of both these approaches on the Spikey neuromorphic system [7].

The third strategy maps traditional feedforward architectures, trained offline with a backpropagation algorithm, to a network of spiking neurons on the neuromorphic device (Sec. IV). Here, the key to good performance is an additional learning phase where parameters are trained on hardware in the loop, while using the abstract network description as an approximation for the parameter updates. We show how this approach can restore network functionality despite having incomplete knowledge about the gradient along which the parameters need to descend. These experiments are performed on the BrainScaleS neuromorphic system [8].

While our networks are small compared to those used in contemporary machine learning applications, they showcase the potential of using accelerated analog neuromorphic systems for pattern representation and recognition. In particular, the used neuromorphic systems operate 10^4 times faster than their biological archetypes, thereby significantly speeding up both training and practical application.

II. Fast sampling with spikes

Following [4], [5], neural network activity can be interpreted as sampling from an underlying probability distribution over binary random variables (RVs). The mapping from spikes to states $\boldsymbol{z} = (z_1, \ldots, z_k)$ is defined by

$$z_k^{(t)} = \begin{cases} 1 & \text{if } t_k^s < t < t_k^s + \tau_{\text{ref}} \ , \\ 0 & \text{otherwise} \ , \end{cases} \tag{1}$$

where t_k^s are spike times of the kth neuron and τ_{ref} its absolute refractory period (Fig. 1 A). When using leaky integrate-and-fire (LIF) neurons, Poisson background noise is used to achieve a high-conductance state, in which the stochastic

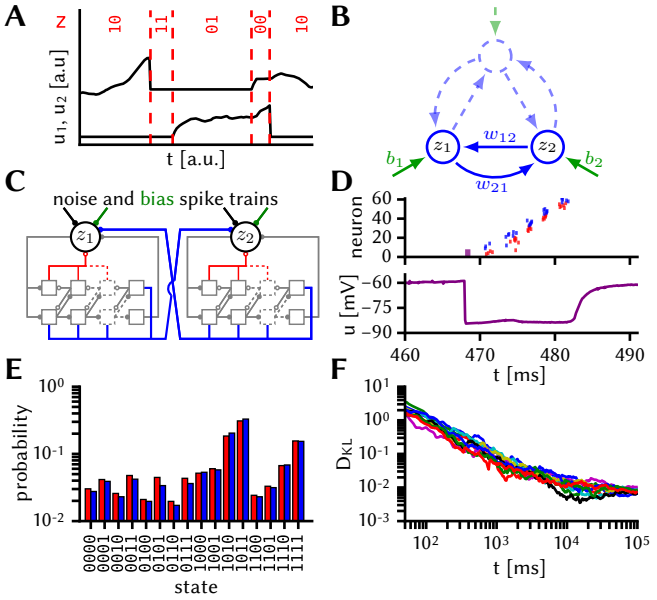

Fig. 1. Sampling with LIF neurons. **(A)** Exemplary membrane potential traces and mapping of refractory/non-refractory neuron states to states 1/0 of binary RVs. **(B)** Exemplary structure of a BM. A subset of 2 units (z_1, z_2) with biases (b_1, b_2) (green) and connected by weights $w_{12} = w_{21}$ (blue) is highlighted to exemplify the neuromorphic network structure in subplot C. **(C)** Sketch of sampling subnetworks representing binary RVs. Each subnetwork consists of a principal LIF neuron (black circle) and an associated synfire chain that implements refractoriness (red synapses), and coupling between sampling units (blue synapses). **(D)** Exemplary spike activity of a sampling unit and membrane potential of its PN. **(E)** Target (blue) vs. sampled (red) distribution on the Spikey chip. **(F)** Evolution of the Kullback-Leibler divergence between the sampled and the target distribution for multiple experimental runs. Time given in biological units.

response of a single neuron is well approximated by a logistic activation function

$$p(z_k = 1) = \sigma\left([\bar{u}_k - \bar{u}_k^0]/\alpha\right) \ , \tag{2}$$

where $\sigma(\cdot)$ is the logistic function and $\bar{u}_k$ represents the noise-free membrane potential of the kth neuron. The parameters $\bar{u}_k^0$ (bias parameter determining the inflection point) and α (slope) are controlled by the intensity of the background noise. With appropriate settings of synaptic weights w_{ij} and bias parameters $\bar{u}_k^0$, these networks can be trained to sample from Boltzmann distributions

$$p(\boldsymbol{z}) \propto \exp[-E(\boldsymbol{z})] = \exp\left[\boldsymbol{z}^T \boldsymbol{W} \boldsymbol{z}/2 + \boldsymbol{z}^T \boldsymbol{b}\right] \ , \tag{3}$$

where the weight matrix $\boldsymbol{W}$ and the bias vector $\boldsymbol{b}$ can be chosen freely. This enables the emulation of Boltzmann machines (BMs) with networks of LIF neurons (Fig. 1 B).

A core assumption of the neural sampling framework is that the membrane potential u_k of a neuron reflects the state $\boldsymbol{z}_{\setminus k}$ of all presynaptic neurons at any moment in time:

$$u_k(\boldsymbol{z}_{\setminus k}) = \textstyle\sum_{j \neq k}^{n} W_{kj} z_j + b_k \ . \tag{4}$$

In particular, this requires that all neurons instantaneously transmit their states (spikes) to all their postsynaptic partners. In any physical system, this assumption is necessarily violated to some degree, since signal transmission can never be instantaneous. In the particular case of accelerated neuromorphic hardware, synaptic transmission delays become even more problematic, as they can be in the same order of magnitude as the state-encoding refractory times themselves. Furthermore, the required equivalence between post-synaptic potential (PSP) durations and refractory states (1,4) can be violated if either of these are unstable. On Spikey, for example, refractory times have relative spike-to-spike variations $\sigma_{\tau_{\text{ref}}}/\tau_{\text{ref}}$ between 2 % and 20 %. These two kinds of timing mismatch pose a fundamental problem to the implementation of spiking BMs in accelerated analog substrates.

Here, we alleviate the issue of substrate-induced timing mismatches by using a recurrent network structure that represents each RV with a small subnetwork, called a sampling unit. The subnetworks are built such that refractory times can be well controlled and, in addition, intra-unit refractory states and inter-unit state communication across the network are inseparably coupled (Fig. 1 C).

Sampling units consist of a single principle neuron (PN) and a small synfire chain of excitatory (EPs) and inhibitory populations (IPs). The EPs of each stage project to both populations in the following stage, thereby relaying an activity pulse in the forward direction. The IPs project backwards, ensuring that neurons from previous stages only spike once. Additionally, all IPs and the last EP also project onto the PN with large weights. Therefore, after the PN elicits a spike, the IPs sequentially pull its membrane potential close to the inhibitory reversal potential, prohibiting it from firing as long as the synfire chain is active (Fig. 1 D). When the pulse has reached the final synfire stage, its EP pulls the PN's membrane potential back to its equilibrium value. The total duration of this pseudo-refractory period can then be controlled by the synfire chain length and parameters.

In addition to controlling refractoriness, the synfire chains also mediate the interaction between PNs. The connections from a synfire chain to other PNs simply mirror its connections to its own PN. This guarantees a match between effective interaction durations and pseudo-refractory periods. The correct synapse parameter settings (weights, time constants) are determined in an iterative training procedure [1].

The results of a hardware emulation can be seen in Fig. 1 E, F. A network of four sampling units was trained on Spikey to sample from a target Boltzmann distribution. After training, the network needs about 10^4 ms of biological time to achieve a good match between the sampled and the target distribution. Considering the hardware acceleration factor of 10^4, this happens in 1 ms of wall-clock time.

III. Robust Hierarchical Networks

As discussed in the previous section, sampling LIF networks are ostensibly sensitive to different types of hardware-induced timing mismatch. In this subsection, we discuss how a sampling network model can be made robust by imposing a hierarchy onto the network structure [2]. This is the equivalent of moving from general BMs to restricted BMs (RBMs). In addition to making their operation more robust, as we discuss below, this hierarchization has the distinct advantage of significantly speeding up training.

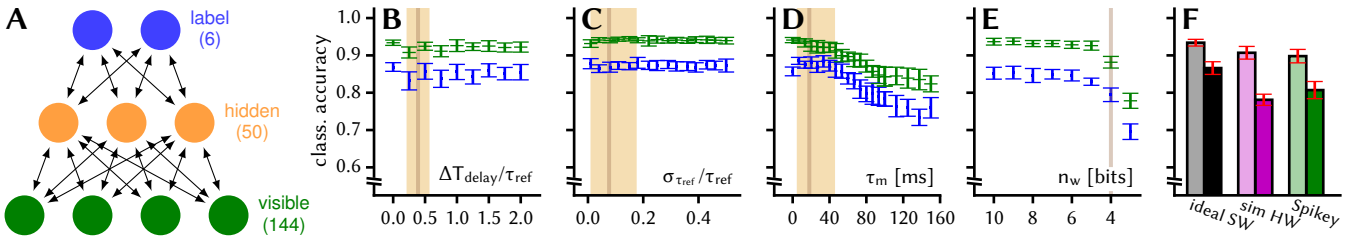

Fig. 2. Robustness from structure in hierarchical networks. **(A)** Hierarchical spiking network emulating an RBM. **(B)–(E)** Effects of hardware-induced distortions on the classification rate of the network. Each test image was presented for a duration of 1000 ms. Green: training data, blue: test data, brown: mean value and range of distortions measured on Spikey. Error bars represent trial-to-trial variations. **(B)** Synaptic transmission delays. **(C)** Spike-to-spike variability of refractory times. **(D)** Membrane time constant. **(E)** Synaptic weight discretization. **(F)** Comparison of classification rates in three scenarios: software simulation of the ideal, distortion-free case (black), software simulation of combined hardware-induced distortions as measured on Spikey (purple), hybrid emulation with the hidden layer on Spikey (green). Light colors for training data, dark colors for test data.

To emulate an RBM, we construct a hierarchical LIF network model with 3 layers: a visible layer representing the data, a hidden layer that learns particular motifs in the data and a label layer for classification (Fig. 2 A). The network was trained with a contrastive learning rule

$$\Delta W_{ij} \propto \langle z_i z_j \rangle_{\text{data}} - \langle z_i z_j \rangle_{\text{model}} \quad , \tag{5}$$

$$\Delta b_i \propto \langle z_i \rangle_{\text{data}} - \langle z_i \rangle_{\text{model}} \tag{6}$$

on a modified subset of the MNIST dataset ($\langle\cdot\rangle_{\text{data}}$ and $\langle\cdot\rangle_{\text{model}}$ represent expectation values when clamping training data and when the network samples freely, respectively). Due to hardware limitations, we used a small network and dataset (6 digits, 12×12 pixels, each with 20 training and 20 test samples) for this proof-of-principle experiment.

The specific influence of various hardware-induced distortion mechanisms were first studied in complementary software simulations. These simulations show that the classification accuracy of the network is essentially unaffected by the types of timing mismatch discussed above, even when their amplitudes are much larger than those measured on our neuromorphic substrate (Fig. 2 B, C). In order to facilitate a meaningful comparison with hardware experiments, two further distortion mechanisms were studied. An upper limit to the membrane conductance can prevent neurons from entering a high-conductance state, thereby distorting their activation functions away from their ideal logistic shape (2) and consequently modifying the sampled distribution. However, within the range achievable on Spikey, the effect on the classification accuracy remains small (Fig. 2 D). The largest effect (about 5.6 % regression in classification accuracy compared to ideal software simulations) stems from the discretization of synaptic weights, which have a resolution of 4 bits on Spikey (Fig. 2 E).

The robustness of this hierarchical architecture to timing mismatches is a consequence of both the training procedure and the information flow within the network. Training has the effect of creating a steep energy landscape $E(\boldsymbol{z})$ (3), for which deep energy minima, corresponding to particular learned digits, represent strong attractors, in which the system is placed during classification by clamping of the visible layer. Throughout the duration of such an attractor, visible neurons represent pixels of constant intensity encoded in their spiking probability, thereby entering a quasi-rate-based information representation regime. Therefore, the information they provide to the hidden layer is unaffected by temporal shifts or zero-mean noise. As they outnumber the hidden neurons 24:1, they effectively control the state of the hidden layer. The hidden layer neurons themselves are unaffected by timing mismatches because they are not interconnected. Second-order (hidden→label→hidden) lateral interactions are indeed distorted, but as they are mediated by only few label neurons, their relative strength is too weak to play a critical role.

These findings are corroborated by experiments on Spikey (Fig. 2 F). Due to the system's limitations, we used a hybrid approach, with the visible and label layers implemented in software and the hidden layer running on Spikey. In the ideal, undistorted case, the LIF network had a classification performance of 86.6 ± 1.7 % (93.4 ± 0.9 %) on the test (training) set. This was reduced to 78.1 ± 1.5 % (90.7 ± 1.7 %) when all distortive effects were simultaneously present in software simulations. In comparison, the hybrid emulation showed a performance of 80.7 ± 2.3 % (89.8 ± 1.8 %), which closely matched the software results within the trial-to-trial variability. We stress that this was a result of direct-to-hardware mapping, with no additional training to compensate for hardware-induced distortions (as compared to Sec. IV).

IV. In-the-loop training

In Sec. II, we used a training procedure based on (5,6) to optimize the hardware-emulated sampling network. Such simple contrastive learning rules can yield very good classification performance in networks of spiking neurons [6]. Another class of highly successful learning algorithms is based on error backpropagation. This, however, requires precise knowledge of the gradient of a cost function with respect to the network parameters, which is difficult to achieve on analog hardware. We propose a training method for hardware-emulated networks that circumvents this problem by using the cost function gradient with respect to the parameters of an ANN as an approximation of the true gradient with respect to the hardware parameters [3]. A similar method has previously been used for network training on a digital neuromorphic device [9].

Our training schedule consisted of two phases. In the first phase, an ANN was trained in software on a modified subset

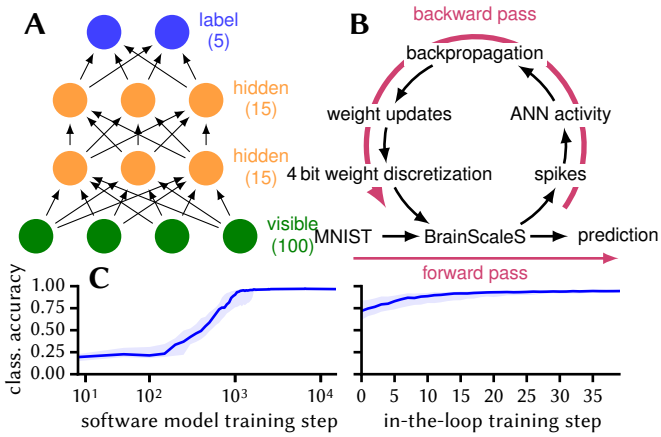

Fig. 3. In-the-loop training. **(A)** Structure of the feed-forward, rate-based deep spiking network. **(B)** Schematic of the training procedure with the hardware in the loop. **(C)** Classification accuracy over training step. Left: software training phase, right: hardware in-the-loop training phase.

of the MNIST dataset (5 digits, 10×10 pixels, with a total of 30690 training and 5083 test samples) using a simple cost function with regularization

$$C(\boldsymbol{W}) = \sum_{s\in S} \|\tilde{\boldsymbol{y}}_s - \hat{\boldsymbol{y}}_s)\|^2 + \sum_{kl} \tfrac{1}{2}\lambda W_{kl}^2 \qquad (7)$$

and backpropagation with momentum [10]

$$\Delta W_{kl} \leftarrow \eta \nabla_{W_{kl}} C(\boldsymbol{W}) + \gamma \Delta W_{kl} \ , \qquad (8)$$

$$W_{kl} \leftarrow W_{kl} - \Delta W_{kl} \ . \qquad (9)$$

Here, $\tilde{\boldsymbol{y}}_s$ and $\hat{\boldsymbol{y}}_s$ denote the target and network state of the label layer, respectively, and the sum runs over all samples within a minibatch S. The learned parameters were then translated to a feed-forward spiking neural network (Fig. 3 A). Here, the BrainScaleS wafer-scale system [8] was used for network emulation. Due to hardware imperfections, the ANN classification accuracy of 97 % dropped to $72\,^{+12}_{-10}$ % after mapping the network to the hardware substrate.

In the second training phase, the hardware-emulated network was trained in the loop (Fig. 3 B) for several iterations. Parameter updates were calculated using the same gradient descent rule as in the ANN, but the activation of all layers was measured on the hardware. The rationale behind this approach is that the activation function of an ANN unit is sufficiently similar to that of an LIF neuron to allow using the computed gradient as an approximation of the true hardware gradient. As seen in Fig. 3 C, this assumption is validated by the post-training performance of the hardware-emulated network: after 40 training iterations, the classification accuracy increased back to $95\,^{+1}_{-2}$ %.

V. Discussion

We have reviewed three strategies for emulating performant spiking network models in analog hardware. The proposed methods tackled the problems induced by substrate-inherent imperfections from different (and complementary) angles. The three strategies were implemented and evaluated with two different analog hardware systems.

An essential advantage of the employed neuromorphic platforms is provided by their accelerated dynamics. Despite possible losses in performance compared to precisely tunable software solutions, accelerated analog neuromorphic systems have the potential to vastly outperform classical simulations of neural networks in terms of both speed and energy consumption [3] – an invaluable advantage for online learning of complex, real world data sets. The network in Sec. II, for example, is already faster than equivalent software simulations (NEST 2.2.2 default build, single-threaded, Intel Core i7-2620M) by several orders of magnitude.

The studied networks serve as a proof of principle and are scalable to larger network sizes. Future research will have to address whether the results obtained for these small networks still hold as training tasks increase in complexity. Furthermore, the generative properties of the described hierarchical LIF networks remain to be studied. Another major step forward will be taken once training can take place entirely on the hardware, thereby rendering sequential reconfigurations between individual experiments unnecessary. Future generations of the used systems will feature on-board plasticity processor units, with early-stage experiments already showing promising results [11].

Acknowledgments

The first five authors contributed equally to this work. This research was supported by EU grants #269921 (BrainScaleS), #604102 and #720270 (Human Brain Project) and the Manfred Stärk Foundation.

References

[1] M. A. Petrovici, D. Stöckel, I. Bytschok, J. Bill, T. Pfeil, J. Schemmel, and K. Meier, "Fast sampling with neuromorphic hardware," Neural Information Processing Systems, Demonstration, 2015.

[2] M. A. Petrovici, A. Schroeder, O. Breitwieser, A. Grübl, J. Schemmel, and K. Meier, "Robustness from structure: Fast inference on a neuromorphic device with hierarchical LIF networks," IJCNN, 2017.

[3] S. Schmitt, J. Klähn, G. Bellec, A. Grübl, M. Güttler, A. Hartel *et al.*, "Neuromorphic hardware in the loop: Training a deep spiking network on the brainscales wafer-scale system," IJCNN, 2017.

[4] L. Buesing, J. Bill, B. Nessler, and W. Maass, "Neural dynamics as sampling: A model for stochastic computation in recurrent networks of spiking neurons," *PLoS Computational Biology*, vol. 7, no. 11, 2011.

[5] M. A. Petrovici, J. Bill, I. Bytschok, J. Schemmel, and K. Meier, "Stochastic inference with spiking neurons in the high-conductance state," *Physical Review E*, vol. 94, no. 4, 2016.

[6] L. Leng, M. A. Petrovici, R. Martel, I. Bytschok, O. Breitwieser *et al.*, "Spiking neural networks as superior generative and discriminative models," in *Cosyne Abstracts, Salt Lake City USA*, February 2016.

[7] T. Pfeil, A. Grübl, S. Jeltsch, E. Müller, P. Müller, M. A. Petrovici, M. Schmuker *et al.*, "Six networks on a universal neuromorphic computing substrate," *Frontiers in Neuroscience*, vol. 7, p. 11, 2013.

[8] J. Schemmel, D. Brüderle, A. Grübl, M. Hock, K. Meier, and S. Millner, "A wafer-scale neuromorphic hardware system for large-scale neural modeling," in *Proceedings of the 2010 IEEE ISCAS*, 2010, pp. 1947–1950.

[9] S. K. Esser, P. A. Merolla, J. V. Arthur, A. S. Cassidy, R. Appuswamy *et al.*, "Convolutional networks for fast, energy-efficient neuromorphic computing," *PNAS*, vol. 113, no. 41, pp. 11 441–11 446, 2016.

[10] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Networks*, vol. 12, no. 1, pp. 145–151, 1999.

[11] S. Friedmann, J. Schemmel, A. Grübl, A. Hartel *et al.*, "Demonstrating hybrid learning in a flexible neuromorphic hardware system," *IEEE Trans. on Biomed. Circ. and Syst.*, no. 99, pp. 1–15, 2016.