Neural Networks 133 (2021) 11–20

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Structural plasticity on an accelerated analog neuromorphic hardware system

Sebastian Billaudelle [a,*,1], Benjamin Cramer [a,*,1], Mihai A. Petrovici [a,c], Korbinian Schreiber [a], David Kappel [b], Johannes Schemmel [a,2], Karlheinz Meier [a,2]

[a] Kirchhoff-Institute for Physics, Heidelberg University, Germany
[b] Georg-August-University Göttingen, Germany
[c] Department of Physiology, University of Bern, Switzerland

ARTICLE INFO

Article history:
Received 4 March 2020
Received in revised form 17 July 2020
Accepted 28 September 2020
Available online 12 October 2020

Keywords:
Structural plasticity
Receptive fields
BrainScaleS
Spiking
Neural networks

ABSTRACT

In computational neuroscience, as well as in machine learning, neuromorphic devices promise an accelerated and scalable alternative to neural network simulations. Their neural connectivity and synaptic capacity depend on their specific design choices, but is always intrinsically limited. Here, we present a strategy to achieve structural plasticity that optimizes resource allocation under these constraints by constantly rewiring the pre- and postsynaptic partners while keeping the neuronal fan-in constant and the connectome sparse. In particular, we implemented this algorithm on the analog neuromorphic system BrainScaleS-2. It was executed on a custom embedded digital processor located on chip, accompanying the mixed-signal substrate of spiking neurons and synapse circuits. We evaluated our implementation in a simple supervised learning scenario, showing its ability to optimize the network topology with respect to the nature of its training data, as well as its overall computational efficiency.

1. Introduction

Experimental data shows that plasticity in the brain is not limited to changing only the strength of connections. The structure of the connectome is also continuously modified by removing and creating synapses (Bhatt et al., 2009; Grutzendler et al., 2002; Holtmaat & Svoboda, 2009; Xu et al., 2009; Zuo et al., 2005). Structural plasticity allows the nervous system to reduce its spatial and energetic footprint by limiting the number of fully expressed synaptic spine heads and maintaining sparsity (Knoblauch & Sommer, 2016). The lifetime of dendritic spines, involved at least in excitatory projections, varies dramatically (Trachtenberg et al., 2002).

The process of spine removal depends on the spine head size: smaller spines are removed while larger ones persist (Holtmaat et al., 2005, 2006). At the same time, new spines are continuously created. The spine volume also shows a strong correlation with the amplitude of the respective synaptic currents (Matsuzaki et al., 2001), hence suggesting a coupling of a connection's lifetime and its synaptic efficacy.

Neuromorphic devices implement novel computing paradigms by taking inspiration from the nervous system. With the prospect of solving shortcomings of existing architectures, they often also inherit some restrictions of their biological archetypes. The exact form and impact of these limitations depend on the overall design and architecture of a system. Ultimately however, all physical information processing systems, with neuromorphic ones making no exception, have to operate on finite resources.

For most neuromorphic systems, synaptic fan-in is – to various degrees – one of these limited resources. This applies to analog as well as digital platforms, especially when they implement fast on-chip memory. For example, TrueNorth and ODIN both allocate fixed memory regions for 256 synapses per neuron (Akopyan et al., 2015; Frenkel et al., 2018). Loihi imposes an upper limit of 4096 individual presynaptic partners per group of up to 1024 neurons located on a single core (Davies et al., 2018). In contrast, the digital neuromorphic multi-core platform SpiNNaker (Furber et al., 2013) allows to trade the number of synapses per neuron against overall network size or simulation performance. In general, digital systems often make use of time-multiplexed update logic, and hence can be designed to alleviate the issue of a limited fan-in by increasing memory size — albeit at the cost of prolonged simulation times.

Analog and mixed-signal systems mostly do not allow this trade-off, because their synapses are implemented physically,

* Corresponding authors.
E-mail addresses: sebastian.billaudelle@kip.uni-heidelberg.de (S. Billaudelle), benjamin.cramer@kip.uni-heidelberg.de (B. Cramer).

[1] Authors with equal contribution.

[2] Shared senior authorship.

https://doi.org/10.1016/j.neunet.2020.09.024

and therefore often constitute a fixed resource. Examples include DYNAP-SEL (providing 64 static synapses per neuron on four of its cores and 256 learning synapse circuits on a fifth core, Moradi et al., 2018; Thakur et al., 2018), Spikey (256 synapses per neuron, Schemmel et al., 2007), and BrainScaleS-1 (220 synapses per neuron, Schemmel et al., 2010). For this manuscript, we have used a prototype system of the BrainScaleS-2 architecture with 32 synapses per neuron. At full scale, a single application-specific integrated circuit (ASIC) features 256 synapses per neuron, with the additional option of merging multiple neuron circuits to larger logical entities in order to increase their overall fan-in Aamir, Müller et al. (2018), similarly to its predecessor BrainScaleS-1 (Schemmel et al., 2010).

The above list of neuromorphic systems is certainly not exhaustive, but it hints towards an ubiquitous issue of limited synaptic resources. A promising way to address this issue is to, once again, draw inspiration from the biological nervous system by supporting the emulation of sparse networks. In the field of machine learning, Denil et al. (2013) found that multi-layer networks in fact express many redundant connections for a wide range of common machine learning datasets. Consequently, new training schemes have been developed to incorporate the concept of sparsity during training (Bellec et al., 2017; Wen et al., 2016), allowing to compress large, fully connected networks into smaller, sparse representations without a significant loss in performance. In order to make neuromorphic devices amenable to such compressed models, it appears desirable to deeply anchor mechanisms for sparse networks and structural plasticity in the design of the systems themselves.

In this paper we present an efficient structural plasticity mechanism and an associated on-chip implementation thereof for the BrainScaleS-2 system, directly exploiting the synapse array's architecture. It leverages the fact that the network connectivity is partially defined and resolved within each synapse, which is enabled by local event filtering. This design choice promotes the mapping of sparse network graphs to the synapse matrix. The update algorithm is implemented on the embedded plasticity processor, which directly interfaces the synaptic memory through a vector unit. This near-memory design allows efficient parallel updates to the network's topology and weights.

Our approach enables fully local learning in a sparse connectome while inherently keeping the synaptic fan-in of a neuron constant. We further demonstrate its ability to optimize the network topology by forming clustered receptive fields and study its robustness with respect to sparsity constraints and choice of hyperparameters. While enabling an efficient, task-specific allocation of synaptic resources through learning, we also point out that our implementation of structural plasticity is computationally efficient in itself, requiring only a small overhead compared to the computation of, e.g., synaptic weight updates.

2. Methods

The BrainScaleS-2 architecture, which we discuss in Section 2.1, provides all features required to implement flexible on-chip plasticity rules, including our proposed mechanism for structural reconfiguration. Section 2.2 describes the algorithm for pruning and reassignment of synapses as well as an optimized implementation thereof. This structural plasticity scheme can be coupled with various weight dynamics. In this work, we employ a correlation-based weight update rule, which is described in Section 2.3. The combination of both is tested in a supervised classification task, as outlined in Section 2.4.

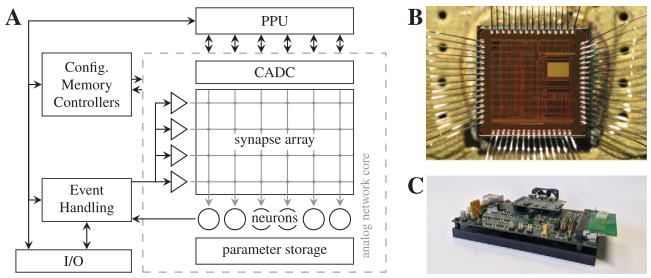

Fig. 1. BrainScaleS-2 prototype ASIC. (A) Block-level schematic. The analog neuromorphic core contains neuronal and synaptic circuits, which are accompanied by, inter alia, an analog parameter storage and the CADC for digitizing synaptic correlation data. It is surrounded by digital logic which interfaces the full-custom circuits and handles configuration data as well as spike traffic. The PPU is closely attached to the analog core, allowing it to access synaptic weights, address labels, and digitized correlation traces from the CADC. (B) Photograph of the BrainScaleS-2 prototype ASIC. (C) Experimental setup.

2.1. BrainScaleS-2 architecture

BrainScaleS-2 is a family of mixed-signal neuromorphic chips implemented in a 65 nm process (Fig. 1). It is centered around an analog neural network core implementing neuron and synapse circuits that behave similarly to their biological archetypes. State variables such as membrane potentials and synaptic currents are physically represented in the respective circuits and evolve in continuous time. Leveraging the intrinsic capacitances and conductances of the technology, time constants of neuron and synapse dynamics are rendered 1000 times smaller compared to typical values found in biology. This thousandfold acceleration facilitates the execution of time-consuming tasks, such as performing high-dimensional parameter sweeps, the investigation of learning and metalearning, or statistical computations requiring large volumes of data (Bohnstingl et al., 2019; Cramer et al., 2020).

The analog core features 32 silicon neurons[3] (Aamir, Stradmann et al., 2018) implementing leaky integrate-and-fire (LIF) dynamics $C_\mathrm{m}\dot{V}_\mathrm{m} = -g_\mathrm{l}(V_\mathrm{m} - E_\mathrm{l}) + I_\mathrm{syn}$, where V_m represents the membrane potential, C_m the membrane capacitance, g_l the leak conductance, and E_l the resting potential. Synaptic currents I_syn are modeled as superpositions of spike-triggered exponential kernels. The membrane is connected to a reset potential by a programmable conductance for a finite refractory period as soon as the membrane potential crosses a firing threshold V_th. All neurons are individually configurable via an on-chip analog parameter memory (Hock et al., 2013) and a set of digital control values.

Each neuron is associated with a column of 32 synapse circuits[3] (Friedmann et al., 2017), which receive their inputs from the chip's digital backend. Incoming events are tagged with addresses, which denote their presynaptic origins (Fig. 2). A 6 bit label is stored alongside the 6 bit weight in the synapse-local static random-access memory (SRAM). It allows to filter afferent spike trains by their addresses; only an event matching the locally stored label is forwarded to the postsynaptic neuron circuit. Each synapse also implements an analog circuit for measuring pairwise correlations between pre- and post-synaptic spike events (Friedmann et al., 2017), enabling access to various forms of learning rules based on nearest-neighbor spike-timing-dependent plasticity (STDP). The analog correlation traces are made accessible by the column-parallel analog-to-digital converter (CADC).

[3] Later versions of the system feature 512 neuron circuits with adaptive-exponential LIF dynamics and inter-compartmental conductances. Each neuron is connected to 256 synapse circuits. Support for conductance-based synapses is planned for future versions.

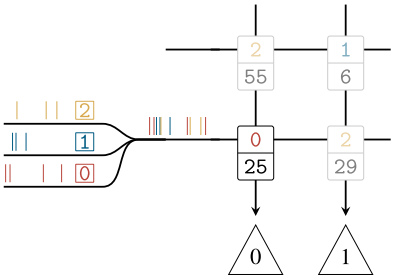

Fig. 2. Synaptic event filtering enables efficient structural plasticity. Events are identified with an address denoting their source (numbered and marked by color). Spike trains from different origins can be overlayed and injected into a single synapse row. Synapses filter afferent events by comparing the source address to a label stored in their local SRAM and forward only matching spikes to the postsynaptic neurons. Addresses and labels can be reconfigured by the PPU to implement weight dynamics and structural changes.

The versatility of the BrainScaleS-2 architecture is substantially augmented by the incorporation of a freely programmable embedded microprocessor (Friedmann et al., 2017), which enables the execution of custom programs interacting with the analog emulation of the neuro-synaptic dynamics. Together with the single instruction, multiple data (SIMD) vector unit, which is tightly coupled to the synapse array's SRAM controller and the CADC, it forms the PPU, which allows the efficient on-chip implementation of synaptic plasticity rules. Access to the chip-internal configuration bus further allows the processor to also reconfigure all other components of the neuromorphic system during experiment execution. The PPU can thus be used for a vast array of applications such as near-arbitrary learning rules, online circuit calibration, or the co-simulation of an environment capable of continuous interaction with the network running on the neuromorphic core. On the prototype system used in this work, the plasticity processor runs with a frequency of 100 MHz. Its SIMD unit operates in parallel on slices of 16 synapses.[4] Iterating row-wise across the synapse matrix, this parallel access lets plasticity algorithms scale $\sim \mathcal{O}(m)$ with the indegree m but $\sim \mathcal{O}(1)$ with the number of postsynaptic neurons.

A field-programmable gate array (FPGA) is used to interface the ASIC with a host computer. It also provides sequencing mechanisms for experiment control and spike handling. Our experiments were based on this paradigm. However, it was shown that the PPU can replace all of the FPGA's functionality during experiment runtime (Wunderlich et al., 2019), dramatically reducing the overall system's power consumption. In this case, the FPGA is only used for initial configuration as well as to read out and store observables for later analysis and visualization. This is an essential prerequisite for the scalability of the BrainScaleS-2 architecture.

2.2. Pruning and reassignment of synapses

We propose a mechanism and an optimized hardware implementation for structural plasticity inspired by two well-established biological observations. First, we assume that important, informative synapses have larger absolute weights. In our particular setting, this is achieved by Hebbian learning, augmented by slow unlearning, as outlined in Section 2.3, but this assumption holds for many other plasticity mechanisms as well (Frémaux & Gerstner, 2016; Mostafa, 2017; Oja, 1982; Urbanczik & Senn, 2014; Zenke & Ganguli, 2018). Second, we enable the network to manage its limited synaptic resources towards potentially improving its performance by removing weak synapses and creating new ones instead.

[4] Later versions of the system include multiple PPUs, which are clocked at frequencies of up to 400 MHz and feature vector registers capable of handling slices of 128 synapses.

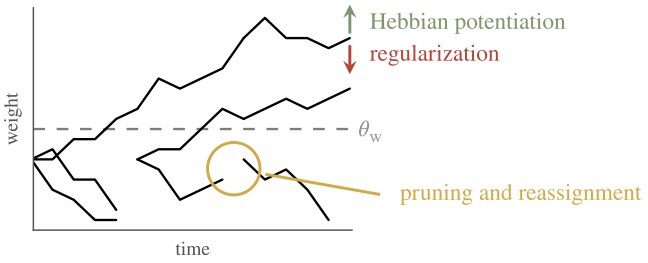

Fig. 3. Illustration of weight dynamics. The evolution of synaptic weights is governed by a Hebbian potentiation term and a regularizing force of opposing sign. A stochastic component in the weight update term leads to a random walk. Synapses with an efficacy below the pruning threshold θ_w are regularly reassigned to new receptors, allowing neurons to find more informative presynaptic partners, to which the connections can then be strengthened.

```
for row in 0 ... 31 do
  w ← synram_weights_read(row)
  w ← w + alpha * min(f_max,correlation_read(row))
  w ← w - beta * w * rates_read()
  w ← w + gamma * rng(-1,1)
  if w < theta_w then
    w ← w_init
    a ← rng(0,k)
    synram_labels_write(row,a)
  end if
  synram_weights_write(row,w)
end for
```

Algorithm 1: Plasticity algorithm including weight updates and structural reconfiguration. The update algorithm is executed by the on-chip PPU and is applied iteratively to the synapse rows. Synapses within a row are processed in parallel. The PPU supports SIMD vector instructions including arithmetic operations and access to the synaptic memory (`synram_weights_{read,write}()`, `synram_labels_write()`) and CADC data (`correlation_read()`). It has also access to the neuronal firing rates (`rates_read()`) and uniform pseudo-random number generators (`rng()`).

A synapse's eligibility for pruning is determined by the value of its weight: it is removed in case its efficacy falls below a threshold θ_w (Fig. 3). Whenever an afferent synapse is removed, the postsynaptic neuron replaces it with a connection to a randomly selected presynaptic partner, thus conserving its indegree. The newly created synapse is initialized with a low weight w_{init}. The pruning process takes place at a slower timescale than the network dynamics and weight updates, giving the synaptic weights time to develop and integrate over multiple update periods.

The implementation on BrainScaleS-2 exploits an in-synapse resolution of the connectome. Each event carries a label denoting its origin, allowing synapses to distinguish different sources. A synapse filters afferent spike trains by comparing this event address to the locally stored value and forwards only matching events to its postsynaptic neuron. Pruning and reassigning of synapses is implemented by remapping the label stored in the synapse-local SRAM, which effectively eliminates the previous connection.

As compared to other synaptic pruning and reassignment strategies, our algorithm and implementation of structural plasticity requires a particularly low overhead. The in-synapse definition of the connectome allows a purely local reassignment

mechanism which avoids global access patterns; for example, no reordering of routing tables is required, which can otherwise lead to increased computational complexity (Liu et al., 2018). At its core, reassignment only involves a single SRAM access. Also the evaluation of the pruning condition and the selection of a new presynaptic partner can be realized with just a few simple instructions (Algorithm 1).

2.3. Correlation-driven weight update algorithm

The synaptic reassignment algorithm described above is based on the assumption that high weights are assigned to informative synapses, which emerges through a manifold of learning strategies. In this work, we chose Hebbian weight dynamics, as on BrainScaleS-2 they allow a fully local implementation of weight updates and, furthermore, can be extended to form more advanced learning rules (Frémaux & Gerstner, 2016; Neftci et al., 2014). Here, the temporal evolution of the synaptic weights w_{ij}, which is illustrated in Fig. 3, obeys the following equations:

$$\Delta w_{ij} = \alpha \cdot f(S_i, S_j) - \beta \cdot \nu_i w_{ij} + \gamma \cdot \eta_{ij} \,, \tag{1}$$

$$f(S_i, S_j) = \min \left[f_{\max}, \sum_k \exp\left(- \frac{t_i^k - \max_l \left[t_j^l < t_i^k \right]}{\tau_{\text{STDP}}} \right) \right] . \tag{2}$$

The update rule Eq. (1) consists of three terms. The first term represents an implementation of STDP and depends on the post- and presynaptic spiketrains S_i and S_j, defined as vectors of ordered spike times t_i^k and t_j^l. The STDP kernel is exponential and positive for causal presynaptic spikes and zero for anti-causal ones, with a cutoff at a maximum value $f_{\max}$ Eq. (2). The second term implements homeostasis (by penalizing large postsynaptic firing rates) and forgetting (as an exponential decay). This regularizer encourages competition between the afferent synapses of a neuron. The third term induces exploration by means of a uniformly drawn random variable η_{ij} leading to an unbiased random walk, similar to the work by Kappel et al. (2015). On BrainScaleS-2, this stochastic component helps to overcome local minima induced by analog fixed-pattern noise and integer arithmetics. The three components are weighted with positive factors α, β, and γ, respectively.

All three contributions to the weight update rule can be mapped to specialized hardware components. The STDP-derived term is based on correlation traces. These observables are measured in analog synapse-local circuits and then digitized using the CADC (Section 2.1).

As stated above, the correlation values are capped. This is required to reduce the imbalance introduced by fixed-pattern deviations in the correlation measurement circuits' sensitivity, as some of these analog sensors might systematically detect stronger correlation values than others. This can lead to an overly strong synchronization of the respective receptor and label neurons, in turn resulting in a self-amplifying potentiation of the corresponding weight and a resulting dominance over the teacher spike train. In principle, a decrease of α could dampen such feedback, but the corresponding reduction of the exponential STDP kernel can be difficult to reconcile with fixed-point calculations of limited precision.

The homeostatic component requires access to the postsynaptic firing rates, which are read from spike counters via the on-chip configuration bus. Stochasticity is provided by an *xorshift* algorithm (Marsaglia et al., 2003) implemented in software.[5] The individual contributions are processed and accumulated on the embedded processor: using the SIMD vector unit, it is able to handle slices of 16 synapses in parallel.

[5] Later versions of the system feature hardware acceleration for the generation of pseudo-random numbers.

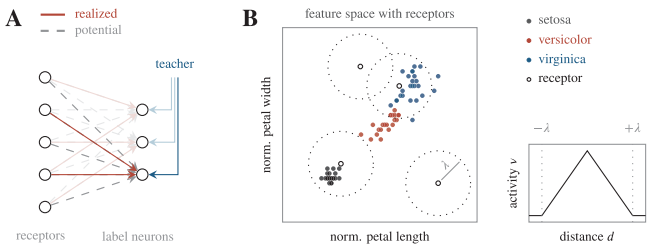

Fig. 4. Sparse network architecture and input encoding. (A) The two-layer network consists of a group of receptors and a label population. One teacher per label neuron ensures excitation of the correct labels during learning. The inputs project onto the label layer with a *potential* all-to-all connectivity (gray), but only a subset of synapses is realized (blue). (B) The receptors are uniformly distributed on the two-dimensional feature space, which is spanned by the petal widths and lengths of Iris flowers belonging to the three classes setosa, versicolor, and virginica. A receptor's activity is calculated from its Euclidean distance to a data point according to a triangular kernel with radius λ. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

2.4. Classification task

We applied the presented plasticity mechanism including structural reconfiguration to a two-layer network trained to perform a classification task. The network consisted of a group of spike sources in a receptor layer and a set of label neurons. These layers were set up such that every postsynaptic neuron could potentially receive input from any presynaptic partner in the receptor layer. Only a fixed fraction of these potential synapses was expressed at each point in time; the others were dormant, resulting in a sparse connectome. In addition to the feed-forward connections, label neurons were stimulated by teacher spike sources. These supervisory projections ensured excitation of a label neuron when an input belonging to their respective class was presented.

The network was trained on the Iris dataset (Fisher, 1936). Schmuker et al. (2014) already trained an early predecessor of the BrainScaleS-2 system on the same data, but used an off-chip preprocessing scheme based on principal component analysis to determine optimal receptor locations and static receptive fields. Here, we reduced the four-dimensional dataset to only two dimensions by selecting petal widths and lengths, renormalized to values between 0.2 and 0.8. The resulting two-dimensional feature space is shown in Fig. 4B. On this plane, n virtual receptors were placed at random locations drawn from a uniform distribution. These receptor neurons emitted Poisson-distributed spike trains with an instantaneous rate determined by their respective Euclidean distances d to a presented data point. The firing rate was calculated according to a triangular kernel $\nu_i(d) = \hat{\nu} \cdot \max(0, 1 - d/\lambda)$, with $\hat{\nu} = 50\,\text{kHz}$. This corresponds to a biologically plausible firing rate of 50 Hz, when taking the system's speedup into account. The radius λ of the receptors was scaled inversely with $\sqrt{n}$ to ensure a reasonable coverage of the feature space.

To impose a certain level of sparsity, we used the following procedure. Receptors were randomly grouped into m disjoint bundles of size k and each bundle was injected into a single synapse row. Within a bundle, each receptor was assigned a unique address. The sparsity of the connectome, defined as the ratio between the number of unrealized synapses and the number of potential synapses, was thus set to $1 - 1/k = 1 - m/n$. This setup allowed two degrees of freedom in the control of network sparsity (Fig. 7). Increasing the number of receptors n for a fixed synapse count m increased the bundle size k and thus the sparsity as well. On the other hand, for constant sparsity

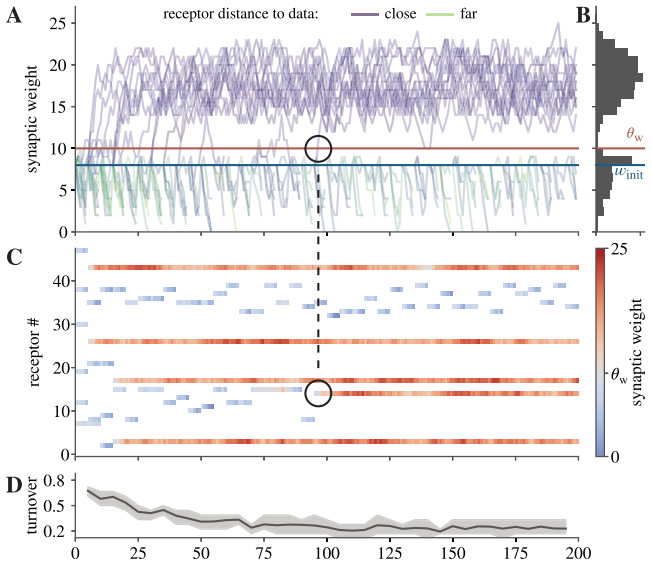

Fig. 5. Informative synapses emerge during training. (A) Exemplary evolution of realized afferent weights of the "setosa" label neuron during the course of a single experiment. The line color is determined by the average feature-space distance between the respective receptor and all "setosa" data points. Synapses that receive inputs from relevant receptors (i.e., those lying close to the features that are relevant for their postsynaptic label neuron) are strengthened towards values that lie above the pruning threshold θ_w. All other, less informative synapses remain below θ_w and are pruned at regular intervals of five epochs. For each pruned synapse, a new one is initialized at w_{init}, between the same label neuron and a previously unconnected receptor. (B) Distribution of synaptic weights during the last 50 epochs over 20 randomly initialized runs. Note that the histogram only takes into account realized synapses, which at all times are only 18 out of 144 potential ones. (C) Exemplary evolution of all synaptic weights between the receptor population and the "setosa" label neuron. At all times, only $n/k = 6$ synapses are realized. The transition from blue to red marks the pruning threshold θ_w. Note how gray/blue (subthreshold) and white (non-existent) states alternate, marking the pruning of weak synapses and re-initialization of new ones. One of these reassignments is highlighted and referenced to the corresponding threshold crossing in pane A. (D) Evolution of the turnover rate (fraction of pruned synapses per epoch) for the 20 runs. The solid line marks the mean and the gray area represent the 20 and 80 percentiles. As time progresses, the turnover rate converges to approximately 20%, indicating that all relevant receptors (on average five) have been found. The remaining "free" synapses (on average one) keep switching between all other receptors, but are pruned regularly as they are not informative for the respective class. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

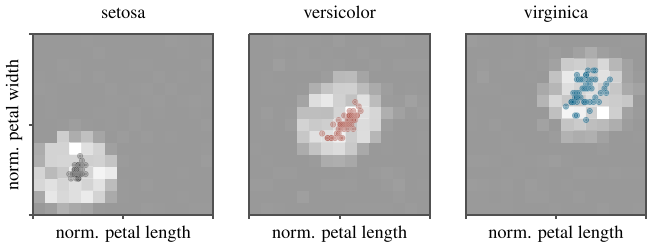

Fig. 6. Self-organized formation of receptive fields. The probability of synapse expression depends on the location of receptors in the feature space and the class of label neurons. Each square is shaded according to the probability for a label neuron to have formed a synapse with a receptor lying within that area (lighter for higher probability); estimated from the state at the end of training in 100 experiments with random initial conditions. The size of the three emerging clusters is determined by the receptor radius λ.

$1 - 1/k$, reducing the synapse count m incurred a reduction of the receptor count n. Disjoint bundles do impose restrictions on the realizable connectomes: First, it is not possible to project to a neuron from two presynaptic sources that both carry relevant information but by chance reside on the same bundle. Second, multapses, i.e., multiple projections from the same afferent to a single postsynaptic neuron, cannot be realized. These constraints can be circumvented by choosing overlapping bundles or injecting a single bundle into multiple synaptic rows.

The dataset, containing a total of 150 data points, was randomly divided into 120 training and 30 test samples to allow cross validation. Samples were presented to the network in random order. For each presented data point, the network's state was determined by a winner-take-all mechanism implemented in software, which compared the firing rates of the label neurons. Synaptic weights were updated according to Eq. (1) after each epoch. The pruning condition was evaluated regularly every five epochs.

3. Results

In this section, we describe experimental results of learning on the BrainScaleS-2 prototype using the plasticity rule and classification task outlined above. We evaluated the network's performance under varied sparsity constraints and performed sweeps on the hyperparameters to study the robustness of the learning algorithm and demonstrate its efficient use of limited synaptic resources. Moreover, we highlight the speed of our structural plasticity algorithm, especially in conjunction with its implementation on the BrainScaleS-2 system.

3.1. Self-configuring receptive fields

Depending on the nature of the data to be learned, i.e., the distribution of data points in the feature space, some receptors can be more informative than others (Fig. 4B). Our learning rule naturally selects the most informative receptors, thereby creating a topological order of the label neurons' receptive fields. This clustering of receptors is driven by the synaptic weight evolution as described by Eq. (1) (Fig. 3).

Fig. 5 shows this evolution during the course of an experiment. Starting from their initial values, synapses that contributed causally to the firing of their postsynaptic neurons were potentiated. After escaping the pruning threshold, they continued evolving until reaching an equilibrium with the homeostatic force. Weaker connections were regularly pruned and reassigned; the common initialization value manifests itself in a strongly pronounced peak.

The turnover rate, defined as the fraction of pruned synapses, also reflects the formation of receptive fields. As the receptors were randomly initialized at the beginning of the experiment, they did not reflect the spatial distribution of the dataset. This resulted in frequent pruning, indicated by a high turnover rate. Over time, a set of stable synapses was formed and the turnover rate gradually decreased.

The topology of the emergent connectome can be reconstructed from the synaptic labels. By repeating the experiment with varying seeds and therefore initial conditions, it is possible to calculate a probability density for a synapse to be expressed at a given point on the feature plane. This map closely resembles the distribution of the presented data (Fig. 6): the receptive fields of the respective label neurons cluster around the corresponding samples. The radius of these clusters is determined by the spread of the data as well as the support and shape of the receptors' kernels.

3.2. Increased network performance with structural plasticity

During the course of the training phase, the network's performance was repeatedly evaluated by presenting the test data

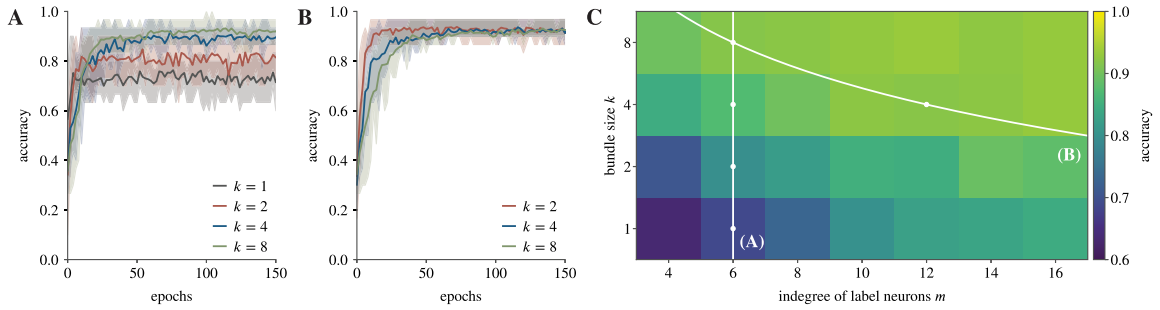

Fig. 7. Structural plasticity improves learning in sparse networks. (A) For a constant indegree m of the label neurons (equivalent with the number of synapse rows on the hardware), classification accuracy improves with larger k, as the neurons gain access to an increasing number of receptors $n = km$. (B) For a constant number of receptors n, structural plasticity can compensate for increased sparsity (reduced indegree m induced by a larger bundle size k) up to a certain degree. (C) Panels A and B can be embedded into a more extensive sweep over the number of the indegree m and bundle size k.

to the receptor layer. In this phase, the network's weights and connectome were frozen by disabling weight updates and structural modifications. To test the network's ability to generalize and reduce the impact of specific positioning of receptors or initial conditions, we trained and evaluated the network starting from 20 randomly drawn initial states.

The evolution of the network's accuracy can be observed in Fig. 7A. Starting from approximately chance level, the performance increased during training and converged to a stable value. In this specific experiment, we swept the bundle size k while keeping the number of utilized synapse rows m constant, resulting in a variable number of receptors $n = k \cdot m$. This corresponds to a scenario where the limited afferent synaptic resources per neuron are fully utilized and structural plasticity is required to expand the number of virtual presynaptic partners. For $k = 1$ the network was trained without structural reconfiguration and only had access to a small pool of receptors, resulting in a correspondingly low performance. As more receptors became available, the classification accuracy increased as well, up to 92.3% for structural plasticity with a bundle size of 8.

In a second sweep we kept the number of receptors n constant and varied the bundle size k. This resulted in a variable number of realized synapses m and hence different levels of sparsity. The classification accuracy's evolution for $k \in \{2, 4, 8\}$ is shown in Fig. 7B. The network achieved a performance of approximately 92% for all of the sparsity levels. In this experiment, we showed that learning with structural plasticity allows to reduce the utilization of synaptic resources while conserving the overall network performance. These results demonstrate that our learning algorithm enables a parsimonious utilization of hardware resources. The resulting pool of "free" synapses can then be used for other purposes, such as for the realization of deeper network structures. For larger receptor pools, we also note that learning converges more slowly, as the label neurons need more time to explore their respective receptive fields (Fig. 7A,B).

Both of the aforementioned experiments can be embedded into a more extensive sweep over receptor counts n and bundle sizes k. In Fig. 7C, the two experiments correspond to the two highlighted lines. Classification performance primarily depends on the count of available receptors — and to a much lesser extent on the amount of utilized hardware resources. For the employed classification task, only six synapses were sufficient to reach levels of accuracy otherwise only tangible with more than 32 synapses.

We established a baseline accuracy for this task by evaluating the network with artificially set up connectomes. For each bundle ($k = 8$, $m = 6$) and label neuron, we selected the receptor with the highest mean firing rate for all training data of the respective class. Analyzing the resulting receptive

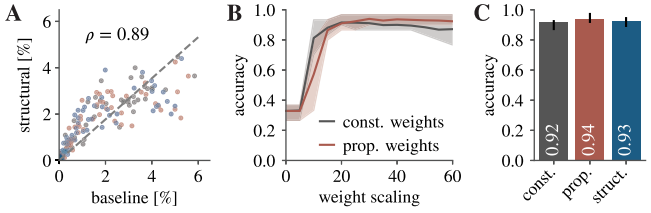

Fig. 8. Comparison of structural plasticity to a baseline estimate. Structural plasticity yields an accuracy comparable to the one of a network obtained by artificially choosing the most active receptors for each class. (A) A clear correlation between the synapse expression probabilities of the trained and the baseline networks can be observed. (B) The network was evaluated for the same set of synapses, but with a variable weight scaling factor. In one case the weights were configured homogeneously to a constant value, in the other they were additionally scaled with the receptors' mean activations. (C) Classification performance for structural plasticity is on par with the respective maxima from B, where the proportional scaling outperforms the constant one.

fields for multiple seeds and plotting the expression probabilities against the ones obtained through structural plasticity (cf. Fig. 6) shows a clear correlation (Fig. 8A). This indicates that the presented structural plasticity algorithm indeed establishes informative synapses. Due to the threshold-based pruning, which is essential for convergence to a stable connectome, receptors with the highest expression probabilities for the baseline selection were slightly underrepresented in the learnt structure. We estimated the baseline performance by considering two methods of assigning synaptic weights: First, we applied the same, constant weight value $w_{ij} = s$ for all established synapses, representing the case of pure structural plasticity and ignoring any weight dynamics. Second, we chose the weights as $w_{ij} = s \cdot \nu_j / \max_j(\nu_j)$, with the average receptor firing rate ν_j, demonstrating the combined effects of structural reconfiguration and Hebbian learning. We evaluated both cases with a varied scaling s (Fig. 8B) and selected the respective maxima for the performance comparison (Fig. 8C). Compared to these two baseline measures, which relied on global knowledge of the whole dataset and all receptor activities, the connectome that emerged through structural plasticity, only based on local information, yielded a comparable performance. For the two baseline results, a small but noticeable increase in accuracy is observable in case of Hebbian weight selection. It is noteworthy, that – despite this rather minor influence on the overall classification accuracy – STDP constitutes the driving force for the expression of meaningful synapses in the presented implementation of structural plasticity.

The network's performance depends on the selection of hyperparameters for the learning rule. Since the pruning condition is based on the synaptic weights, the selection of the pruning

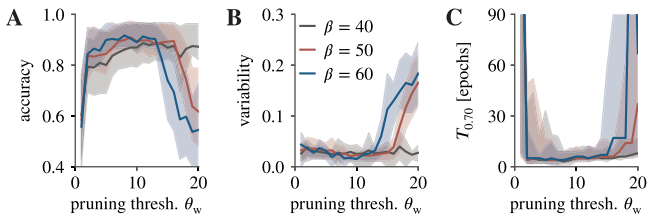

Fig. 9. Stability of network performance over a wide range of hyperparameters. We varied the pruning threshold θ_w and the regularization strength β, which both shape the steady-state weight distribution. For different aspects of learning performance, broad plateaus with respect to variations of these hyperparameter can be observed. Solid lines and shaded areas respectively denote mean and 20–80 percentiles, measured over 20 randomly initialized experiments. The plateaus mostly coincide for (A) classification accuracy after learning (average over the last 20 epochs), (B) variability of accuracy after learning (standard deviation over the last 20 epochs), and (C) number of epochs until an accuracy of 70% was reached.

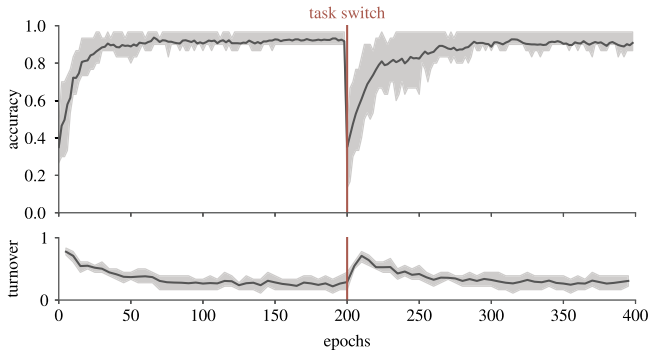

Fig. 10. Restoration of network performance after task switch. After training for 200 epochs, the receptor layer is randomly rearranged, leading to a mismatch in receptive fields. Ongoing structural plasticity unlearns the previously established connectome and quickly starts to again explore the input space. This process can be observed in an elevated turnover rate after the task switch, similar to the initial phase of the experiment.

threshold must take into account the distribution of learnt efficacies (Fig. 5). Thus, θ_w must be high enough to allow uninformative synapses to be pruned, but still low enough as to not affect previously found informative synapses. Fig. 9 displays different performance metrics as a function of the pruning threshold. These analyses are shown for a varied strength of the regularizing term β, as the weight distribution and scale depend on the balance of the positive Hebbian and this negative force. All three metrics exhibit broad plateaus of good performance, which coincide over a relatively wide range of θ_w.

3.3. On-the-fly adaptation to switching tasks

As demonstrated, structural plasticity enables learning in sparse networks by exploring the input space and forming informative receptive fields. So far we have considered experiments with a randomly initialized connectome and most importantly a homogeneous weight distribution. In another experiment, we tested the plasticity mechanism's ability to cope with a previously learned and therefore already structured weight distribution. We achieved this by abruptly changing the task during training. After 200 epochs, the receptors were moved to new, random locations, resulting in a misalignment of receptive fields and data points. The plasticity rule was executed continuously, before and after this task switch.

As shown in Fig. 10, the accuracy dropped to approximately chance level as the receptors were shuffled. This decline, however, was directly followed by a rapid increase of the turnover rate. The negative contribution of the regularization term outweighed the Hebbian forces, thereby resulting in decreasing synaptic efficacies. After a few epochs, most of the weights had fallen below θ_w and were eligible for pruning. This process allowed the network to successfully unlearn previous connections, thus rekindling exploration of the input space.

3.4. Fast and efficient hardware emulation

In our proposed implementation, structural reconfiguration only induces a small computational overhead. Synaptic pruning and reassignment is enabled by exploiting the synaptic filtering of spike events by their source address. Since the connectome is essentially defined by the address labels stored in the synapses' memory, it can also be reconfigured with local operations only.

The algorithm executed on the on-chip microprocessor can effectively be dissected into four steps (Alg. 1): accessing the synaptic weights, evaluation of the pruning condition, potential reassignment of the synaptic label, and a final write access to the synapse SRAM. The exact time required for executing the respective instructions depends on the neuromorphic system's architecture and the design of the plasticity processing unit. In general, memory access and the generation of pseudo-random numbers can be regarded as the most expensive operations. The former primarily depends on the system's design and can be optimized for low access times. Random number generation can also be sped up by implementing dedicated hardware accelerators.

Our implementation on BrainScaleS-2 is enabled by the PPU and its tight coupling to the neuromorphic core. Access to the synapse array as well as arithmetic operations are optimized by a parallel processing scheme. Performing a structural plasticity update on a single slice of 16 synapses takes approximately 110 clock cycles, which corresponds to 1.1 µs at a PPU clock frequency of 100 MHz (Fig. 11). This amounts to about seven clock cycles, or 69 ns, per synapse. In comparison, the Hebbian term, which is executed five times more often, requires approximately 3.8 µs for a slice or 240 ns per synapse. The regularizer and random walk take 69 ns and 97 ns per synapse, respectively. In our implementation, these terms were implemented separately and were not particularly optimized for performance. Sharing memory accesses or intermediate results between them would lead to an overall speedup of the plasticity mechanism.

The time spent on the generation of pseudo-random numbers, highlighted in Fig. 11, constitutes a significant portion for both the random walk and the pruning term. On the full-size BrainScaleS-2 system, hardware accelerators allow to reduce this contribution to a comparatively negligible 0.08 clock cycles per synapse[4].

Hence, our implementation of structural plasticity is doubly efficient. Not only can it effectively optimize the utilization of synaptic resources, but it can also achieve this at the cost of only a small overhead to the calculation of synaptic weight updates (Fig. 11).

The accelerated nature of the BrainScaleS-2 system also contributes to a rapid evaluation of plasticity schemes in general — and structural reconfiguration in particular. Emulating a single epoch of 24 biological seconds required a total of 137 ms on our system. Excluding the overhead induced by on-the-fly generation of input spike trains in Python, this number boils down to less than 50 ms, which corresponds to a speedup factor of about 500. As shown by Wunderlich et al. (2019), this overhead can be dramatically reduced by porting the experiment control from the host and FPGA to the PPU. This further allows to optimize the system's power consumption to below 60 mW, with only a weak dependence on the nature of ongoing network activity and plasticity (Wunderlich et al., 2019).

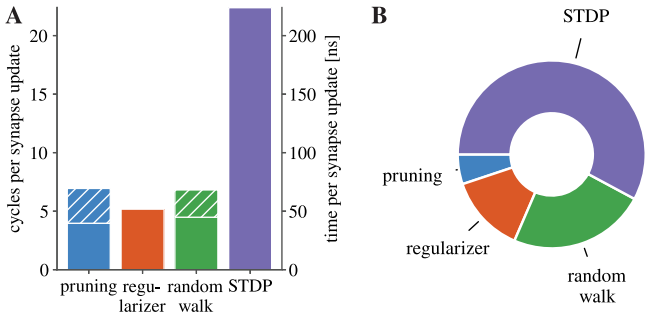

Fig. 11. Efficient mixed-signal implementation of structural plasticity. (A) Duration of a synapse update broken down into its four individual contributions, including structural reconfiguration. The hatched areas indicate the time spent on pseudo-random number generation. (B) Contributions of the individual terms to the overall update duration, taking into consideration that pruning and reassignment are executed five times less often than synaptic weight updates.

BrainScaleS-2 achieves its speedup by exploiting the quick emulation with above-threshold analog transistor circuits. The individual parametrization of each circuit allows to dramatically reduce fabrication-induced fixed-pattern variations (Aamir, Stradmann et al., 2018). We employed calibration routines to equilibrate the behavior of the synaptic correlation sensors as well as the neuron circuits. To assess the remaining variations, we investigated the transferability of the network's topology and weight information. For this purpose, we trained a population of three label neurons and then replicated the learnt connectome to four other groups of neurons and their associated synapse columns. Since each of these instances of the network exhibited its own intrinsic circuit variations, this allowed us to infer a measure of transferability of training result across systems. The accuracies acquired for these networks deviated by only 1.2% from the originally trained population. This shows that a calibrated system can be used for inference with weights learnt on a different setup. Nevertheless, learning can partially compensate for non-ideal calibration data (Wunderlich et al., 2019), stressing the value of on-chip training.

4. Discussion

We have presented a fully local, on-chip structural plasticity mechanism together with an efficient implementation on a prototype of the BrainScaleS-2 architecture. The algorithm allows to train a network with a sparse connectome, thereby utilizing synaptic resources more efficiently. We showcased this implementation in a supervised learning task with weight updates driven by Hebbian potentiation. For this classification task, it was possible to drastically increase the sparsity of the connectome without significant performance loss. Self-configuring receptive fields led to near-perfect accuracy and a better utilization of synaptic resources without prior knowledge of the input data.

Structural plasticity has already been successfully applied to networks with various topologies and learning paradigms (Bellec et al., 2017; Bogdan et al., 2018; Butz et al., 2009; George et al., 2017; Kappel et al., 2015). In addition to a more efficient handling of synaptic resources, it is assumed to also improve network performance (Roy et al., 2014; Spiess et al., 2016) and, in conjunction with non-linear multi-compartmental neuron models, memory capacity (Hussain & Basu, 2016; Poirazi & Mel, 2001). We expect our plasticity scheme to be applicable to many of these different network topologies and learning rules: The weight dynamics can be easily extended by additional terms, e.g. modulatory reward signals, as they were used by Wunderlich et al. (2019). Alternatively, the proposed implementation of pruning and reassignment could be combined with completely different weight update mechanisms. This would allow to alleviate the ubiquitous issue of limited fan-in for multi-layer networks, that have, on their own, already been demonstrated on BrainScaleS (Kungl et al., 2018; Schmitt et al., 2017). Furthermore, as the PPU is freely programmable, the structural plasticity mechanism itself can be extended by e.g. additional pruning criteria such as bookkeeping (Spiess et al., 2016), spatial information (Bogdan et al., 2018), or silent synapses (Roy & Basu, 2016). However, all of these additions should be considered regarding their impact on the algorithm's locality.

There already exist several implementations of structural plasticity for various neuromorphic platforms. Most of them were based on an on-the-fly adaptation of connectivity tables. Such generally very flexible strategies have been successfully demonstrated especially on digital systems (Bogdan et al., 2018; Yan et al., 2019), where the event handling per se is already centered around look-up tables. For such an approach, the ordering of connectivity lists is important to minimize look-up latencies, which introduces overhead for the removal and insertion of a synapse (Liu et al., 2018). Related strategies were proposed also for analog neuromorphic systems (Bhaduri et al., 2018; George et al., 2017; Spiess et al., 2016). These implementations were based on optimized look-up matrices, using a representation comparable to our on-chip synapse matrix, which were stored and evaluated on external FPGAs. In these cases, learning and rewiring were also executed off-chip. In contrast, BrainScaleS-2 provides a local, in-synapse definition of the sparse connectome. This allowed our efficient implementation of on-chip plasticity and rewiring.

All of the named approaches have to allocate memory besides the actual synaptic weights, as sparse matrices always require the annotation of the placement of non-zero elements. However, the additional increase in memory is outweighed by the overall gains due to the smaller network graphs. External look-up tables can often be stored in dynamic random-access memory (DRAM), which reduces their spatial footprint compared to SRAM-based implementations. The inherent access latencies can, however, be detrimental, especially for accelerated neuromorphic systems.

We note that the accelerated nature of the BrainScaleS-2 system is especially relevant in the context of modeling biological rewiring processes. In vivo, structural changes to the connectome typically take place on time scales of hours to days (Lamprecht & LeDoux, 2004), which allows synapses to process large amounts of information and evolve accordingly before being potentially pruned. This throughput of information – essentially spikes – per unit of time is directly contingent on the specific time constants of neuro-synaptic dynamics. Consequently, the acceleration factor of BrainScaleS-2 can also translate directly to a corresponding speedup of structural plasticity.

Our implementation scales well with growing system sizes, since it is fully based on synapse-local quantities. In particular, it profits directly from the parallel handling of synaptic updates. On large systems, this would especially benefit the more complex network structures and associated larger synapse arrays required when tackling more difficult tasks.

CRediT authorship contribution statement

Sebastian Billaudelle: Conceived the idea, designed, implemented, and executed the experiments. **Benjamin Cramer:** Conceived the idea, designed, implemented, and executed the experiments. **Mihai A. Petrovici:** Contributed to the experiment design, contributed the evaluation and interpretation of the results. **Korbinian Schreiber:** Peripheral hardware for the experiment setup. **Johannes Schemmel:** Designer and architect of the neuromorphic platform, conceived and implemented the synapse circuits.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors express their gratitude towards A. Baumbach, E. Müller, P. Spilger, and Y. Stradmann for their support and helpful discussions.

This work has received funding from the European Union Sixth Framework Programme (FP6/2002-2006) under grant agreement no. 15879 (FACETS), the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 604102 (HBP), 269921 (BrainScaleS) and 243914 (Brain-i-Nets) and the Horizon 2020 Framework Programme (H2020/2014-2020) under grant agreement no. 720270 and 785907 (HBP), as well as the Manfred Stärk Foundation.

References

Aamir, S. A., Müller, P., Kiene, G., Kriener, L., Stradmann, Y., Grübl, A., Schemmel, J., & Meier, K. (2018). A mixed-signal structured adex neuron for accelerated neuromorphic cores. *IEEE Transactions on Biomedical Circuits and Systems*, *12*(5), 1027–1037. http://dx.doi.org/10.1109/TBCAS.2018.2848203.

Aamir, S. A., Stradmann, Y., Müller, P., Pehle, C., Hartel, A., Grübl, A., Schemmel, J., & Meier, K. (2018). An accelerated lif neuronal network array for a large-scale mixed-signal neuromorphic architecture. *IEEE Transactions on Circuits and Systems. I. Regular Papers*, *65*(12), 4299–4312. http://dx.doi.org/10.1109/TCSI.2018.2840718.

Akopyan, F., Sawada, J., Cassidy, A., Alvarez-Icaza, R., Arthur, J., Merolla, P., Imam, N., Nakamura, Y., Datta, P., Nam, G., Taba, B., Beakes, M., Brezzo, B., Kuang, J. B., Manohar, R., Risk, W. P., Jackson, B., & Modha, D. S. (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*, [ISSN: 0278-0070] *34*(10), 1537–1557. http://dx.doi.org/10.1109/TCAD.2015.2474396.

Bellec, G., Kappel, D., Maass, W., & Legenstein, R. (2017). Deep rewiring: Training very sparse deep networks. arXiv preprint arXiv:1711.05136.

Bhaduri, A., Banerjee, A., Roy, S., Kar, S., & Basu, A. (2018). Spiking neural classifier with lumped dendritic nonlinearity and binary synapses: a current mode vlsi implementation and analysis. *Neural Computation*, *30*(3), 723–760.

Bhatt, D. H., Zhang, S., & Gan, W.-B. (2009). Dendritic spine dynamics. *Annual Review of Physiology*, *71*, 261–282.

Bogdan, P. A., Rowley, A. G., Rhodes, O., & Furber, S. B. (2018). Structural plasticity on the spinnaker many-core neuromorphic system. *Frontiers in Neuroscience*, *12*, 434.

Bohnstingl, T., Scherr, F., Pehle, C., Meier, K., & Maass, W. (2019). Neuromorphic hardware learns to learn. *Frontiers in Neuroscience*, *13*.

Butz, M., Woergoetter, F., & van Ooyen, A. (2009). Activity-dependent structural plasticity. *Brain Research Reviews*, *60*(2), 287–305.

Cramer, B., Stöckel, D., Kreft, M., Wibral, M., Schemmel, J., Meier, K., & Priesemann, V. (2020). Control of criticality and computation in spiking neuromorphic networks with plasticity. *Nature Communications*, *11*(1), 1–11.

Davies, M., Srinivasa, N., Lin, T., Chinya, G., Cao, Y., Choday, S. H., Dimou, G., Joshi, P., Imam, N., Jain, S., Liao, Y., Lin, C., Lines, A., Liu, R., Mathaikutty, D., McCoy, S., Paul, A., Tse, J., Venkataramanan, G., ... Wang, H. (2018). Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, [ISSN: 0272-1732] *38*(1), 82–99. http://dx.doi.org/10.1109/MM.2018.112130359.

Denil, M., Shakibi, B., Dinh, L., Ranzato, M., & De Freitas, N. (2013). Predicting parameters in deep learning. In *Advances in neural information processing systems* (pp. 2148–2156).

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, *7*(2), 179–188.

Frémaux, N., & Gerstner, W. (2016). Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules. *Frontiers in Neural Circuits*, *9*, 85.

Frenkel, C., Lefebvre, M., Legat, J.-D., & Bol, D. (2018). A 0.086-mm^2 12.7-pj/sop 64k-synapse 256-neuron online-learning digital spiking neuromorphic processor in 28-nm cmos. *IEEE Transactions on Biomedical Circuits and Systems*, *13*(1), 145–158.

Friedmann, S., Schemmel, J., Grübl, A., Hartel, A., Hock, M., & Meier, K. (2017). Demonstrating hybrid learning in a flexible neuromorphic hardware system. *IEEE Transactions on Biomedical Circuits and Systems*, [ISSN: 1932-4545] *11*(1), 128–142. http://dx.doi.org/10.1109/TBCAS.2016.2579164.

Furber, S. B., Lester, D. R., Plana, L. A., Garside, J. D., Painkras, E., Temple, S., & Brown, A. D. (2013). Overview of the spinnaker system architecture. *IEEE Transactions on Computers*, [ISSN: 0018-9340] *62*(12), 2454–2467. http://dx.doi.org/10.1109/TC.2012.142.

George, R., Indiveri, G., & Vassanelli, S. (2017). Activity dependent structural plasticity in neuromorphic systems. In *Biomedical circuits and systems conference (BioCAS), 2017 IEEE* (pp. 1–4). IEEE.

Grutzendler, J., Kasthuri, N., & Gan, W.-B. (2002). Long-term dendritic spine stability in the adult cortex. *Nature*, *420*(6917), 812.

Hock, M., Hartel, A., Schemmel, J., & Meier, K. (2013). An analog dynamic memory array for neuromorphic hardware. In *Circuit theory and design (ECCTD), 2013 European conference on* (pp. 1–4). http://dx.doi.org/10.1109/ECCTD.2013.6662229.

Holtmaat, A., & Svoboda, K. (2009). Experience-dependent structural synaptic plasticity in the mammalian brain. *Nature Reviews Neuroscience*, *10*(9), 647–658.

Holtmaat, A. J., Trachtenberg, J. T., Wilbrecht, L., Shepherd, G. M., Zhang, X., Knott, G. W., & Svoboda, K. (2005). Transient and persistent dendritic spines in the neocortex in vivo. *Neuron*, *45*(2), 279–291.

Holtmaat, A., Wilbrecht, L., Knott, G. W., Welker, E., & Svoboda, K. (2006). Experience-dependent and cell-type-specific spine growth in the neocortex. *Nature*, *441*(7096), 979.

Hussain, S., & Basu, A. (2016). Multiclass classification by adaptive network of dendritic neurons with binary synapses using structural plasticity. *Frontiers in Neuroscience*, *10*, 113.

Kappel, D., Habenschuss, S., Legenstein, R., & Maass, W. (2015). Synaptic sampling: A bayesian approach to neural network plasticity and rewiring. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems, Vol. 28* (pp. 370–378). Curran Associates, Inc.

Knoblauch, A., & Sommer, F. T. (2016). Structural plasticity, effectual connectivity, and memory in cortex. *Frontiers in Neuroanatomy*, *10*, 63.

Kungl, A. F., Schmitt, S., Klähn, J., Müller, P., Baumbach, A., Dold, D., Kugele, A., Gürtler, N., Müller, E., & Koke, C. (2018). Generative models on accelerated neuromorphic hardware. arXiv preprint arXiv:1807.02389.

Lamprecht, R., & LeDoux, J. (2004). Structural plasticity and memory. *Nature Reviews Neuroscience*, *5*(1), 45.

Liu, C., Bellec, G., Vogginger, B., Kappel, D., Partzsch, J., Neumärker, F., Höppner, S., Maass, W., Furber, S. B., & Legenstein, R. (2018). Memory-efficient deep learning on a spinnaker 2 prototype. *Frontiers in Neuroscience*, *12*.

Marsaglia, G. (2003). Xorshift rngs. *Journal of Statistical Software*, *8*(14), 1–6.

Matsuzaki, M., Ellis-Davies, G. C., Nemoto, T., Miyashita, Y., Iino, M., & Kasai, H. (2001). Dendritic spine geometry is critical for ampa receptor expression in hippocampal ca1 pyramidal neurons. *Nature Neuroscience*, *4*(11), 1086.

Moradi, S., Qiao, N., Stefanini, F., & Indiveri, G. (2018). A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (dynaps). *IEEE Transactions on Biomedical Circuits and Systems*, *12*(1), 106–122.

Mostafa, H. (2017). Supervised learning based on temporal coding in spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(7), 3227–3235.

Neftci, E., Das, S., Pedroni, B., Kreutz-Delgado, K., & Cauwenberghs, G. (2014). Event-driven contrastive divergence for spiking neuromorphic systems. *Frontiers in Neuroscience*, *7*, 272.

Oja, E. (1982). Simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, *15*(3), 267–273.

Poirazi, P., & Mel, B. W. (2001). Impact of active dendrites and structural plasticity on the memory capacity of neural tissue. *Neuron*, *29*(3), 779–796.

Roy, S., Banerjee, A., & Basu, A. (2014). Liquid state machine with dendritically enhanced readout for low-power, neuromorphic vlsi implementations. *IEEE Transactions on Biomedical Circuits and Systems*, *8*(5), 681–695.

Roy, S., & Basu, A. (2016). An online unsupervised structural plasticity algorithm for spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(4), 900–910.

Schemmel, J., Brüderle, D., Grübl, A., Hock, M., Meier, K., & Millner, S. (2010). A wafer-scale neuromorphic hardware system for large-scale neural modeling. In: *Proceedings of the 2010 IEEE international symposium on circuits and systems (ISCAS10)* (pp. 1947–1950).

Schemmel, J., Brüderle, D., Meier, K., & Ostendorf, B. (2007). Modeling synaptic plasticity within networks of highly accelerated i amp;f neurons. In *2007 IEEE international symposium on circuits and systems* (pp. 3367–3370). http://dx.doi.org/10.1109/ISCAS.2007.378289.

Schmitt, S., Klähn, J., Bellec, G., Grübl, A., Güttler, M., Hartel, A., Hartmann, S., Husmann, D., Husmann, K., & Jeltsch, S. (2017). Neuromorphic hardware in the loop: Training a deep spiking network on the brainscales wafer-scale system. In *2017 international joint conference on neural networks (IJCNN)* (pp. 2227–2234). IEEE.

Schmuker, M., Pfeil, T., & Nawrot, M. P. (2014). A neuromorphic network for generic multivariate data classification. *Proceedings of the National Academy of Sciences*, *111*(6), 2081–2086.

Spiess, R., George, R., Cook, M., & Diehl, P. U. (2016). Structural plasticity denoises responses and improves learning speed. *Frontiers in Computational Neuroscience*, *10*, 93.

Thakur, C. S. T., Molin, J., Cauwenberghs, G., Indiveri, G., Kumar, K., Qiao, N., Schemmel, J., Wang, R. M., Chicca, E., & Olson Hasler, J. (2018). Large-scale neuromorphic spiking array processors: A quest to mimic the brain. *Frontiers in Neuroscience*, *12*, 891.

Trachtenberg, J. T., Chen, B. E., Knott, G. W., Feng, G., Sanes, J. R., Welker, E., & Svoboda, K. (2002). Long-term in vivo imaging of experience-dependent synaptic plasticity in adult cortex. *Nature*, *420*(6917), 788.

Urbanczik, R., & Senn, W. (2014). Learning by the dendritic prediction of somatic spiking. *Neuron*, *81*(3), 521–528.

Wen, W., Wu, C., Wang, Y., Chen, Y., & Li, H. (2016). Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems* (pp. 2074–2082).

Wunderlich, T., Kungl, A. F., Müller, E., Hartel, A., Stradmann, Y., Aamir, S. A., Grübl, A., Heimbrecht, A., Schreiber, K., & Stöckel, D. (2019). Demonstrating advantages of neuromorphic computation: a pilot study. *Frontiers in Neuroscience*, *13*, 260.

Xu, T., Yu, X., Perlik, A. J., Tobin, W. F., Zweig, J. A., Tennant, K., Jones, T., & Zuo, Y. (2009). Rapid formation and selective stabilization of synapses for enduring motor memories. *Nature*, *462*(7275), 915–919.

Yan, Y., Kappel, D., Neumärker, F., Partzsch, J., Vogginger, B., Höppner, S., Furber, S., Maass, W., Legenstein, R., & Mayr, C. (2019). Efficient reward-based structural plasticity on a spinnaker 2 prototype. *IEEE Transactions on Biomedical Circuits and Systems*, *13*(3), 579–591.

Zenke, F., & Ganguli, S. (2018). Superspike: Supervised learning in multilayer spiking neural networks. *Neural Computation*, *30*(6), 1514–1541.

Zuo, Y., Lin, A., Chang, P., & Gan, W.-B. (2005). Development of long-term dendritic spine stability in diverse regions of cerebral cortex. *Neuron*, *46*(2), 181–189.