frontiers in Neuroanatomy

ORIGINAL RESEARCH
published: 07 April 2016
doi: 10.3389/fnana.2016.00037

# Large-Scale Simulations of Plastic Neural Networks on Neuromorphic Hardware

*James C. Knight*[1]*, *Philip J. Tully*[2, 3, 4], *Bernhard A. Kaplan*[5], *Anders Lansner*[2, 3, 6] and *Steve B. Furber*[1]

[1] *Advanced Processor Technologies Group, School of Computer Science, University of Manchester, Manchester, UK,* [2] *Department of Computational Biology, Royal Institute of Technology, Stockholm, Sweden,* [3] *Stockholm Brain Institute, Karolinska Institute, Stockholm, Sweden,* [4] *Institute for Adaptive and Neural Computation, School of Informatics, University of Edinburgh, Edinburgh, UK,* [5] *Department of Visualization and Data Analysis, Zuse Institute Berlin, Berlin, Germany,* [6] *Department of Numerical analysis and Computer Science, Stockholm University, Stockholm, Sweden*

OPEN ACCESS

**Edited by:**
*Wolfram Schenck, University of Applied Sciences Bielefeld, Germany*

**Reviewed by:**
*Guy Elston, Centre for Cognitive Neuroscience, Australia*
*Thomas Pfeil, Ruprecht-Karls-University Heidelberg, Germany*

***Correspondence:**
*James C. Knight*
*james.knight@manchester.ac.uk*

**Received:** *30 November 2015*
**Accepted:** *18 March 2016*
**Published:** *07 April 2016*

**Citation:**
*Knight JC, Tully PJ, Kaplan BA, Lansner A and Furber SB (2016) Large-Scale Simulations of Plastic Neural Networks on Neuromorphic Hardware. Front. Neuroanat. 10:37. doi: 10.3389/fnana.2016.00037*

SpiNNaker is a digital, neuromorphic architecture designed for simulating large-scale spiking neural networks at speeds close to biological real-time. Rather than using bespoke analog or digital hardware, the basic computational unit of a SpiNNaker system is a general-purpose ARM processor, allowing it to be programmed to simulate a wide variety of neuron and synapse models. This flexibility is particularly valuable in the study of biological plasticity phenomena. A recently proposed learning rule based on the Bayesian Confidence Propagation Neural Network (BCPNN) paradigm offers a generic framework for modeling the interaction of different plasticity mechanisms using spiking neurons. However, it can be computationally expensive to simulate large networks with BCPNN learning since it requires multiple state variables for each synapse, each of which needs to be updated every simulation time-step. We discuss the trade-offs in efficiency and accuracy involved in developing an event-based BCPNN implementation for SpiNNaker based on an analytical solution to the BCPNN equations, and detail the steps taken to fit this within the limited computational and memory resources of the SpiNNaker architecture. We demonstrate this learning rule by learning temporal sequences of neural activity within a recurrent attractor network which we simulate at scales of up to $2.0 \times 10^4$ neurons and $5.1 \times 10^7$ plastic synapses: the largest plastic neural network ever to be simulated on neuromorphic hardware. We also run a comparable simulation on a Cray XC-30 supercomputer system and find that, if it is to match the run-time of our SpiNNaker simulation, the super computer system uses approximately 45× more power. This suggests that cheaper, more power efficient neuromorphic systems are becoming useful discovery tools in the study of plasticity in large-scale brain models.

**Keywords: SpiNNaker, learning, plasticity, digital neuromorphic hardware, Bayesian confidence propagation neural network (BCPNN), event-driven simulation, fixed-point accuracy**

## 1. INTRODUCTION

Motor, sensory and memory tasks are composed of sequential elements and are therefore thought to rely upon the generation of temporal sequences of neural activity (Abeles et al., 1995; Seidemann et al., 1996; Jones et al., 2007). However it remains a major challenge to learn such functionally meaningful dynamics within large-scale models using biologically plausible synaptic and neural plasticity mechanisms. Using SpiNNaker, a neuromorphic hardware platform for simulating large-scale spiking neural networks, and BCPNN, a plasticity model based on Bayesian inference, we demonstrate how temporal sequence learning could be achieved through modification of recurrent cortical connectivity and intrinsic excitability in an attractor memory network.

Spike-Timing-Dependent Plasticity (Bi and Poo, 1998) (STDP) inherently reinforces temporal causality which has made it a popular choice for modeling temporal sequence learning (Dan and Poo, 2004; Caporale and Dan, 2008; Markram et al., 2011). However, to date, all large-scale neural simulations using STDP (Morrison et al., 2007; Kunkel et al., 2011) have been run on large cluster machines or supercomputers, both of which consume many orders of magnitude more power than the few watts required by the human brain. Mead (1990) suggested that the solution to this huge gap in power efficiency was to develop an entirely new breed of "neuromorphic" computer architectures inspired by the brain. Over the proceeding years, a number of these neuromorphic architectures have been built with the aim of reducing the power consumption and execution time of large neural simulations.

Large-scale neuromorphic systems have been constructed using a number of approaches: NeuroGrid (Benjamin et al., 2014) and BrainScaleS (Schemmel et al., 2010) are built using custom analog hardware; True North (Merolla et al., 2014) is built using custom digital hardware and SpiNNaker (Furber et al., 2014) is built from software programmable ARM processors.

Neuromorphic architectures based around custom hardware, especially the type of sub-threshold analog systems which Mead (1990) proposed, have huge potential to enable truly low-power neural simulation, but inevitably the act of casting algorithms into hardware requires some restrictions to be accepted in terms of connectivity, learning rules, and control over parameter values. As an example of these restrictions, of the large-scale systems previously mentioned, only BrainScaleS supports synaptic plasticity in any form implementing both short-term plasticity and pair-based STDP using a dedicated mixed-mode circuit.

As a software programmable system, SpiNNaker will require more power than a custom hardware based system to simulate a model of a given size (Stromatias et al., 2013). However this software programmability gives SpiNNaker considerable flexibility in terms of the connectivity, learning rules, and ranges of parameter values that it can support. The neurons and synapses which make up a model can be freely distributed between the cores of a SpiNNaker system until they fit within memory; and the CPU and communication overheads taken in advancing the simulation can be handled within a single simulation time step.

This flexibility has allowed the SpiNNaker system to be used for the simulation of large-scale cortical models with up to $5.0 \times 10^4$ neurons and $5.0 \times 10^7$ synapses (Sharp et al., 2012, 2014); and various forms of synaptic plasticity (Jin et al., 2010; Diehl and Cook, 2014; Galluppi et al., 2015; Lagorce et al., 2015). In the most recent of these papers, Galluppi et al. (2015) and Lagorce et al. (2015) demonstrated that Sheik et al.'s (2012) model of the learning of temporal sequences from audio data can be implemented on SpiNNaker using a voltage-gated STDP rule. However, this model only uses a small number of neurons and Kunkel et al.'s (2011) analysis suggests that STDP alone cannot maintain the multiple, interconnected stable attractors that would allow spatio-temporal sequences to be learnt within more realistic, larger networks. This conclusion adds to growing criticism of simple STDP rules regarding their failure to generalize over experimental observations (see e.g., Lisman and Spruston, 2005, 2010; Feldman, 2012 for reviews).

We address some of these issues by implementing spike-based BCPNN (Tully et al., 2014)—an alternative to phenomenological plasticity rules which exhibits a diverse range of mechanisms including Hebbian, neuromodulated, and intrinsic plasticity—all of which emerge from a network-level model of probabilistic inference (Lansner and Ekeberg, 1989; Lansner and Holst, 1996). BCPNN can translate correlations at different timescales into connectivity patterns through the use of locally stored synaptic traces, enabling a functionally powerful framework to study the relationship between structure and function within cortical circuits. In Sections 2.1–2.3, we describe how this learning rule can be combined with a simple point neuron model as the basis of a simplified version of Lundqvist et al.'s (2006) cortical attractor memory model. In Sections 2.4, 2.5, we then describe how this model can be simulated efficiently on SpiNNaker using an approach based on a recently proposed event-driven implementation of BCPNN (Vogginger et al., 2015). We then compare the accuracy of our new BCPNN implementation with previous non-spiking implementations (Sandberg et al., 2002) and demonstrate how the attractor memory network can be used to learn and replay spatio-temporal sequences (Abbott and Blum, 1996). Finally, in Section 3.3, we show how an anticipatory response to this replay behavior can be decoded from the neurons' sub-threshold behavior which can in turn be used to infer network connectivity.

## 2. MATERIALS AND METHODS

### 2.1. Simplified Cortical Microcircuit Architecture

We constructed a network using connectivity based on a previously proposed cortical microcircuit model (Lundqvist et al., 2006) and inspired by the columnar structure of neocortex (Mountcastle, 1997). The network consists of $N_{HC}$ hypercolumns arranged in a grid where each hypercolumn consists of 250 inhibitory basket cells and 1000 excitatory pyramidal cells evenly divided into 10 minicolumns. Within each hypercolumn, the pyramidal cells send AMPA-mediated connections to the basket cells with a connection probability of 10 % and a weight of 0.4 nA (defined as a postsynaptic current (PSC) amplitude). The basket cells then send GABAergic connections back to the pyramidal cells with a connection probability of 10 % and a weight of 2 nA. The basket cells are also recurrently connected through GABAergic connections, again with a connection probability of 10 % and a connection weight of 2 nA. The functional outcome of this local connectivity (excitatory to inhibitory and vice versa) is to enable winner-take-all (WTA) dynamics within each hypercolumn. While the strength of the local synapses remains fixed, all pyramidal cells in the network are also recurrently connected to each other through global AMPA and NMDA connections using plastic BCPNN synapses (see Section 2.2): also with a connection probability of 10 %. All connections in the network have distance-dependent synaptic delays such that, between two cells located in hypercolumns $H_{xy}^{pre}$ and $H_{xy}^{post}$, the delay is calculated based on the Euclidean distance between the grid coordinates of the hypercolumns (meaning that all local connections have delays of 1 ms):

$$t_d^{H_{xy}^{pre} H_{xy}^{post}} = \frac{d_{norm}\sqrt{\left(H_x^{post} - H_x^{pre}\right)^2 + \left(H_y^{post} - H_y^{pre}\right)^2}}{V} + 1 \tag{1}$$

Where conduction velocity $V = 0.2\,\mathrm{mm\,ms^{-1}}$ and $d_{norm} = 0.75\,\mathrm{mm}$.

## 2.2. Synaptic and Intrinsic Plasticity Model

The spike-based BCPNN learning rule is used to learn the strengths of all global synaptic connections and the intrinsic excitabilities of all pyramidal cells in the network described in Section 2.1. The goal of the learning process is to estimate the probabilities of pre- and postsynaptic neurons firing ($P_i$ and $P_j$ respectively), along with the probability of them firing together ($P_{ij}$). Then, as Lansner and Holst (1996) describe, these probabilities can be used to calculate the synaptic strengths and intrinsic excitabilities of the network allowing it to perform Bayesian inference. Tully et al. (2014) developed an approach for estimating these probabilities based on pre- and postsynaptic spike trains ($S_i$ and $S_j$ respectively), defined as summed Dirac delta functions $\delta(\cdot)$ where $t_{i,j}^f$ represent the times of spikes:

$$S_i(t) = \sum_{t_i^f} \delta(t - t_i^f) \qquad S_j(t) = \sum_{t_j^f} \delta(t - t_j^f) \tag{2}$$

These spike trains are then smoothed using exponentially weighted moving averages to calculate the $Z$ traces:

$$\tau_{z_i}\frac{dZ_i}{dt} = \frac{S_i}{f_{max}\Delta t} - Z_i \qquad \tau_{z_j}\frac{dZ_j}{dt} = \frac{S_j}{f_{max}\Delta t} - Z_j \tag{3}$$

Here, the maximum allowed firing rate $f_{max}$ and spike duration $\Delta t = 1\,\mathrm{ms}$ combine with the lowest attainable probability estimate $\epsilon = \frac{1000}{f_{max}\tau_p}$ introduced in Equation (5) to maintain a linear mapping from neuronal spike rates to probabilities. For more details on the Bayesian transformation entailed by these equations, see Tully et al. (2014). The $Z$ trace time constants $\tau_{z_i}$ and $\tau_{z_j}$ determine the time scale over which correlations can be detected and are inspired by fast biological processes such as $Ca^{2+}$ influx via NMDA receptors or voltage-gated $Ca^{2+}$ channels. The $Z$ traces are then fed into the $P$ traces, where a coactivity term is introduced:

$$\tau_p\frac{dP_i}{dt} = Z_i - P_i \quad \tau_p\frac{dP_{ij}}{dt} = Z_iZ_j - P_{ij} \quad \tau_p\frac{dP_j}{dt} = Z_j - P_j \tag{4}$$

The $P$ trace time constant $\tau_p$ models long-term memory storage events such as gene expression or protein synthesis. It can be set higher to more realistically match these slow processes, but since simulation time increases with higher $\tau_p$ values, in this work we keep them just long enough to preserve the relevant dynamics. Estimated levels of activity in the $P$ traces are then combined to compute a postsynaptic bias membrane current $I_{\beta_j}$ and synaptic weight between pre- and postsynaptic neurons $w_{ij}$:

$$I_{\beta_j} = \beta_{gain}\log(P_j + \epsilon) \quad w_{ij} = w_{gain}^{syn}\log\frac{P_{ij} + \epsilon^2}{(P_i + \epsilon)\left(P_j + \epsilon\right)} \tag{5}$$

Here, $\beta_{gain}$ is used to scale the BCPNN bias into an intrinsic input current to the neuron which is used to model an A-type K+ channel (Jung and Hoffman, 2009) or other channel capable of modifying the intrinsic excitability of a neuron (Daoudal and Debanne, 2003). Similarly, $w_{gain}^{syn}$ is used to scale the BCPNN weight into a current-based synaptic strength.

## 2.3. Neuronal Model

We model excitatory and inhibitory cells as IAF neurons with exponentially decaying PSCs (Liu and Wang, 2001; Rauch et al., 2003). The sub-threshold membrane voltage $V_m$ of these neurons evolves according to:

$$\tau_m\frac{dV_m}{dt} = -V_m + R_m\left(I_s + I_a + I_{\beta_j}\right) \tag{6}$$

The membrane time constant $\tau_m$ and capacitance $C_m$ determine the input resistance $R_m = \frac{\tau_m}{C_m}$ through which input currents from the afferent synapses ($I_s$), spike-frequency adaption mechanism ($I_a$) and the intrinsic input current from the BCPNN learning rule ($I_{\beta_j}$) – described in Section 2.2—are applied. When $V_m$ reaches the threshold $V_t$ a spike is emitted, $V_m$ is reset to $V_r$ and $\alpha$ is added to the adaption current $I_a$. We use Liu and Wang's (2001) model of spike-frequency adaption with the adaption time constant $\tau_a$:

$$\tau_a\frac{dI_a}{dt} = -I_a \tag{7}$$

The synaptic input current to postsynaptic neuron $j$ ($I_{s_j}$) is modeled as a sum of exponentially shaped PSCs from other presynaptic neurons in the network:

$$\tau_{syn}\frac{dI_{s_j}}{dt} = -I_{s_j} + \sum_{syn}\sum_{i=0}^{n} w_{ij}^{syn}\sum_{t_i^f}\delta(t - t_i^f) \tag{8}$$

$w_{ij}^{syn}$ indicates the weight of the connection between neurons $i$ and $j$ [where $syn \in$ (AMPA, GABA, NMDA) denotes the synapse type], $t_i^f$ represents the arrival time of spikes from presynaptic neuron $i$ (where there are $n$ neurons in the network), and $\tau_{syn}$ is the synaptic time constant.

## 2.4. Simulating Spiking Neural Networks on SpiNNaker

SpiNNaker is a digital neuromorphic architecture designed for the simulation of spiking neural networks. Although systems built using this architecture are available in sizes ranging from single boards to room-size machines, they all share the same basic building blocks—the SpiNNaker chip (Furber et al., 2014). Each of these chips is connected to its six immediate neighbors using a chip-level interconnection network with a hexagonal mesh topology. Each SpiNNaker chip contains 18 ARM cores connected to each other through a network-on-chip, and connected to an external network through a multicast router. Each core has two small tightly-coupled memories: 32 KiB for instructions and 64 KiB for data; and shares 128 MiB of off-chip SDRAM with the other cores on the same chip. Although this memory hierarchy is somewhat unusual, the lack of global shared memory means that many of the problems of simulating large spiking neural networks on a SpiNNaker system are shared with more typical distributed computer systems. Morrison et al. (2005) and Kunkel et al. (2012) developed a collection of approaches for mapping such networks onto large distributed systems in a memory-efficient manner while still obtaining supra-linear speed-up as the number of processors increases. The SpiNNaker neural simulation kernel employs a very similar approach where, as shown in **Figure 1**, each processing core is responsible for simulating between 100 and 1000 neurons and their afferent synapses. The neurons are simulated using a time-driven approach with their state held in the tightly-coupled data memory. Each neuron is assigned a 32 bit ID and, when a simulation step results in a spike, it sends a packet containing this ID to the SpiNNaker router. These "spike" packets are then routed across the network fabric to the cores that are responsible for simulating these synapses. Biological neurons have in the order of $10^3 - 10^4$ afferent synapses, so updating all of these every time step would be extremely computationally intensive. Instead, as individual synapses only receive spikes at relatively low rates, they can be updated only when they transfer a spike as long as their new state can be calculated from:

1. The synapse's previous state.
2. The time since the last spike was transferred.
3. Information available from the time-driven simulation of the postsynaptic neuron.

Using this event-driven approach on SpiNNaker is also advantageous as, due to their sheer number, synapses need to be stored in the off-chip SDRAM which has insufficient bandwidth for every synapse's parameters to be retrieved every simulation time step (Painkras and Plana, 2013). Instead, on receipt of a "spike" packet, a core retrieves the row of the connectivity matrix associated with the firing neuron from SDRAM. Each of these rows describes the parameters associated with the synapses connecting the firing neuron to those simulated on the core. Once a row is retrieved the weights are inserted into an input ring-buffer, where they remain until any synaptic delay has elapsed and they are applied to the neuronal input current.

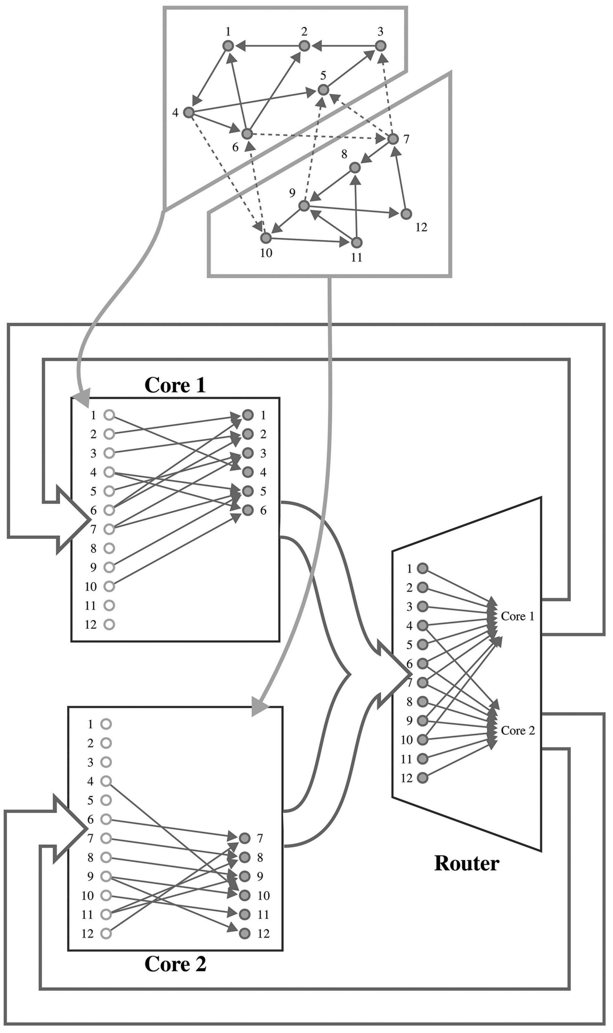

**FIGURE 1 | Mapping of a spiking neural network to SpiNNaker.** For example a network consisting of 12 neurons is distributed between two SpiNNaker cores. Each core is responsible for simulating six neurons (filled circles) and holds a list of afferent synapses (non-filled circles) associated with each neuron in the network. The SpiNNaker router routes spikes from firing neurons (filled circles) to the cores responsible for simulating the neurons to which they make efferent synaptic connections.

In addition to enabling large-scale simulations with static synapses, this event-driven approach can, in theory, be extended to handle any type of plastic synapse that meets the 3 criteria outlined above. However, simulating plastic synapses has additional overheads in terms of memory and CPU load—both of which are very limited resources on SpiNNaker. Several different approaches have been previously developed that aim to minimize

**Algorithm 1** Algorithmic Implementation of STDP

```
function processRow($t$)
    for each $j$ in $postSynapticNeurons$ do
        $history \leftarrow getHistoryEntries(j, t_{old}, t)$

        for each $(t_j, s_j)$ in $history$ do
            $w_{ij} \leftarrow applyPostSpike(w_{ij}, t_j, t_{old}, s_i)$

        $(t_j, s_j) \leftarrow getLastHistoryEntry(t)$
        $w_{ij} \leftarrow applyPreSpike(w_{ij}, t, t_j, s_j)$
        $addWeightToRingBuffer(w_{ij}, j)$

    $s_i \leftarrow addPreSpike(s_i, t, t_{old})$
    $t_{old} \leftarrow t$
```

memory usage (Jin et al., 2010), reduce CPU load (Diehl and Cook, 2014) or offload the processing of plastic synapses to dedicated cores (Galluppi et al., 2015). Morrison et al. (2007) also extended their work on distributed spiking neural network simulation to include synaptic plasticity, developing an algorithm for simulating plastic synapses in an event-driven manner, using a simplified model of synaptic delay to reduce CPU and memory usage. In this work, we combine elements of Diehl and Cook's (2014) and Morrison et al.'s (2007) approaches, resulting in **Algorithm 1** which is called whenever the connectivity matrix row associated with an incoming "spike" packet is retrieved from the SDRAM. As well as the weights of the synapses connecting the presynaptic neuron to the postsynaptic neurons simulated on the local core ($w_{ij}$), the row also has a header containing the time at which the presynaptic neuron last spiked ($t_{old}$) and its state at that time ($s_i$). The exact contents of the state depends on the plasticity rule being employed, but as only the times of presynaptic spikes are available at the synapse, the state often consists of one or more low-pass filtered versions of this spike train.

The algorithm begins by looping through each postsynaptic neuron ($j$) in the row and retrieving a list of the times ($t_j$) at which that neuron spiked between $t_{old}$ and $t$ and its state at that time ($s_j$). In the SpiNNaker implementation, these times and states are stored in a fixed-length circular queue located in the tightly-coupled data memory to which a new entry gets added whenever a local neuron fires. Next, the effect of the interaction between these postsynaptic spikes and the presynaptic spike that occurred at $t_{old}$ is applied to the synapse using the *applyPostSpike* function. The synaptic update is then completed by applying the effect of the interaction between the presynaptic spike that instigated the whole process and the most recent postsynaptic spike to the synapse using the *applyPreSpike* function before adding this weight to the input ring buffer. Finally, the header of the row is updated by calling the *addPreSpike* function to update $s_i$ and setting $t_{old}$ to the current time.

## 2.5. An Event-based, SpiNNaker Implementation of Bayesian Learning

Equations (3)–(5) cannot be directly evaluated within the event-driven synaptic processing scheme outlined in Section 2.4, but as they are simple first-order linear ODEs, they can be solved to obtain closed-form solutions for $Z(t)$ and $P(t)$. These equations then need only be evaluated when spikes occur. Vogginger et al. (2015) converted this resultant system of equations into a spike-response model (Gerstner and Kistler, 2002) which, as it only consists of linear combinations of $e^{\frac{-t}{\tau_z}}$ and $e^{\frac{-t}{\tau_p}}$, can be re-framed into a new set of new state variables $Z_i^*$, $Z_j^*$, $P_i^*$, $P_j^*$, and $P_{ij}^*$. These, like the state variables used in many STDP models are simply low-pass filtered versions of the spike-trains and can be evaluated when a spike occurs at time $t$:

$$Z_i^*(t) = Z_i^*(t^{last})e^{-\frac{\Delta t}{\tau_{z_i}}} + S_i(t) \quad P_i^*(t) = P_i^*(t^{last})e^{-\frac{\Delta t}{\tau_p}} + S_i(t) \tag{9}$$

$Z^*$ and $P^*$ can now be stored in the pre and postsynaptic state ($s_i$ and $s_j$) and updated in the *addPreSpike* function called from algorithm 1; and when postsynaptic neurons fire. The correlation trace, $P_{ij}$ can similarly be re-framed in terms of a new state variable:

$$P_{ij}^*(t) = P_{ij}^*(t^{last})e^{-\frac{\Delta t}{\tau_p}} + S_i(t)Z_j^*(t) \tag{10}$$

$P_{ij}^*$ can now be stored alongside the synaptic weight $w_{ij}$ in each synapse and evaluated in the *applyPreSpike* and *applyPostSpike* functions called from algorithm 1. The final stage of the event-based implementation is to obtain the $P_i$, $P_j$ and $P_{ij}$ values required to evaluate (Equation 5) from the new state variables and thus obtain $w_{ij}$ and $\beta_j$.

$$P_i(t) = a_i \left(Z_i^*(t) - P_i^*(t)\right) \tag{11}$$

$$P_{ij}(t) = a_{ij} \left(Z_i^*(t)Z_j^*(t) - P_{ij}^*(t)\right) \tag{12}$$

With the following coefficients used for brevity:

$$\begin{aligned} \tau_{z_{ij}} &= \left(\frac{1}{\tau_{z_i}} + \frac{1}{\tau_{z_j}}\right)^{-1} \qquad a_i = \frac{1}{f_{max}\left(\tau_{z_i} - \tau_p\right)} \\ a_{ij} &= \frac{1}{{f_{max}}^2\left(\tau_{z_j} + \tau_{z_i}\right)\left(\tau_{z_{ij}} - \tau_p\right)} \end{aligned} \tag{13}$$

This approach makes implementing spike-based BCPNN on SpiNNaker feasible from an algorithmic point of view, but limitations of the SpiNNaker architecture further complicate the problem. The most fundamental of these limitations is that, as Moise (2012, p. 20) explains, for reasons of silicon area and energy efficiency, SpiNNaker has no hardware floating point unit. While floating point operations can be emulated in software, this comes at a significant performance cost meaning that performance-critical SpiNNaker software needs to instead use fixed-point arithmetic. Hopkins and Furber (2015) discussed the challenges of using fixed-point arithmetic for neural simulation on the SpiNNaker platform in detail but, in the context of this work, there are two main issues of particular importance. Firstly the range of fixed-point numeric representations is static so, to attain maximal accuracy, the optimal representation for storing each state variable must be chosen ahead of time. Vogginger et al.

(2015) investigated the use of fixed-point types for BCPNN as a means of saving memory and calculated that, in order to match the accuracy of a time-driven floating point implementation, a fixed-point format with 10 integer and 12 fractional bits would be required. However, not only is the model described in Section 2.2 somewhat different from the reduced modular model considered by Vogginger et al. (2015), but the ARM architecture only allows 8, 16, or 32 bit types to be natively addressed. Therefore, we re-evaluated these calculations for the SpiNNaker implementation and chose to use 16 bit types for two reasons:

1. In order to implement the *getLastHistoryEntry* and *getHistoryEntries* functions used in algorithm 1, each neuron needs to store a history of $Z^*_j$ and $P^*_j$ values in the tightly-coupled data memory, therefore minimizing the size of these variables is important.
2. The SpiNNaker CPU cores can perform multiplication operations on signed 16 bit values faster than it can on 32 bit values, allowing more spikes to be transferred each time-step.

Based on a total of 16 bit, the number of bits used for the integer and fractional parts of the fixed-point representation needs to be determined based on the range of the state variables. As all of the $Z^*$ and $P^*$ state variables are linear sums of exponential spike responses and $P^*$ has the largest time constant, it decays slowest meaning that it will reach the highest value. Therefore we can calculate the maximum value which our fixed-point format must be able to represent in order to handle a maximum spike frequency of $f_{max}$ as follows:

$$P^*_{max} = \frac{1}{1 - e^{-\frac{1}{f_{max} \times \tau_p}}} \tag{14}$$

In order to match the firing rates of pyramidal cells commonly observed in cortex, low values of the maximum firing rate ($f_{max}$, e.g., 20 or 50 Hz) are often used with the BCPNN model described in Section 2.2. On this basis, by using a signed fixed-point format with 6 integer and 9 fractional bits, if $f_{max} = 20$ Hz, traces with $\tau_p < 3.17$ s can be represented and, if $f_{max} = 50$ Hz, traces with $\tau_p < 1.27$ s can be represented.

The second problem caused by the lack of floating point hardware is that there is no standard means of calculating transcendental functions for fixed-point arithmetic. This means that the exponential and logarithm functions required to implement BCPNN must be implemented by other means. While it is possible to implement approximations of these functions using, for instance a Taylor series, the resultant functions are likely to take in the order of 100 CPU cycles to evaluate (Moise, 2012), making them too slow for use in the context of BCPNN where around ten of these operations will be performed every time a spike is transferred. Another approach is to use pre-calculated lookup tables (LUTs). These are particularly well suited to implementing functions such as $e^{\frac{-t}{\tau}}$ where $t$ is discretized to simulation time steps and, for small values of $\tau$, the function decays to 0 after only a small number of table entries. While the $\log(x)$ function has neither of these ideal properties, $x$ can be normalized into the form $x = y \times 2^n : n \in \mathbb{Z}, y \in [1, 2)$ so a LUT is only required to cover the interval $[1, 2)$ within which $\log(x)$ is relatively linear.

## 3. RESULTS

### 3.1. Validating BCPNN Learning on SpiNNaker with Previous Implementations

In this section we demonstrate that the implementation of BCPNN we describe in Section 2.5 produces connection weights and intrinsic excitabilities comparable to those learned by previous models. To do this we used the procedure developed by Tully et al. (2014) and the network described in **Table 1** to compare two neurons, connected with a BCPNN synapse, modeled using both our spiking BCPNN implementation and as abstract units with simple, exponentially smoothed binary activation patterns (Sandberg et al., 2002). We performed this comparison by presenting the neurons with five patterns of differing relative activations, each repeated for ten consecutive 200 ms trials. Correlated patterns meant both neurons were firing at $f_{max}$ Hz or $\epsilon$ Hz each trial; independent patterns meant uniform sampling of $f_{max}$ Hz and $\epsilon$ Hz patterns for both neurons in each trial; anti-correlated patterns meant one neuron fired at $f_{max}$ Hz and the other at $\epsilon$ Hz or vice-versa in each trial; both muted meant both neurons fired at $\epsilon$ Hz in all trials; and post muted meant uniform sampling of presynaptic neuron activity while the postsynaptic neuron fired at $\epsilon$ Hz in all trials.

As **Figure 2** shows, during the presentation of patterns in which both units are firing, the responses from the abstract model fall well within the standard deviation of the SpiNNaker model's responses, but as units are muted, the two models begin to diverge. Further investigation into the behavior of the individual state variables shows that this is due to the $P^*$ term of Equation (11) coming close to underflowing the 16 bit fixed-point format when a long time has passed since the last spike. This inaccuracy in the $P^*$ term is then further amplified when the weights and intrinsic excitabilities are calculated using (Equation 5) as for small values of $x$, $\log(x)$ approaches its vertical asymptote. The standard deviations visible in **Figure 2** reflect the fact that for the spiking learning rule, the realization of Poisson noise that determined firing rates was different for each trial, but with a rate modulation that was repeated across trials.

### 3.2. Learning Sequential Attractors using Spike-Based BCPNN

In this section we consider a functional use case of the the modular attractor network described in Section 2.1 involving learning temporal sequences of attractors. With asymmetrical BCPNN time constants, it was previously proposed that this network could self-organize spontaneously active sequential attractor trajectories (Tully et al., 2014). We built a suitable network using the neuron and plasticity models described in Sections 2.2, 2.3; and the parameters listed in **Table 2**. Using this network we employed a training regime—a subset of which is shown in **Figure 3A**—in which we repeatedly stimulated all cells in a mutually exclusive sequence of minicolumns for 50 training epochs. Each minicolumn was stimulated for 100 ms, such that the neurons within it fired at an average rate of $f_{max}$ Hz. During training we disabled the term in Equation (8) that incorporates input from the plastic AMPA and NMDA synapses meaning that, while the weights were learned online, the dynamics of the network did not disturb the training regime. A recall phase

**TABLE 1 | Model description of the BCPNN validation network.**

**(A) Model summary**

| | |
|---|---|
| Populations | Presynaptic, postsynaptic, presynaptic input, postsynaptic input |
| Connectivity | One-to-one |
| Neuron model | Leaky integrate-and-fire with exponential-shaped synaptic current inputs and spike-frequency adaption (Liu and Wang, 2001) |
| Synapse model | Current-based with exponential-shaped PSCs |
| Plasticity | BCPNN AMPA synapses |
| Input | Externally generated Poisson spike trains |
| Measurements | Intrinsic bias current and synaptic weights |

**(B) Populations**

| Name | Elements | Size |
|---|---|---|
| Presynaptic | Leaky IAF | 1 |
| Postsynaptic | Leaky IAF | 1 |
| Presynaptic input | External spike source | 1 |
| Postsynaptic input | External spike source | 1 |

**(C) Connectivity**

| Source | Target | Pattern | Weight |
|---|---|---|---|
| Presynaptic input | Presynaptic | One-to-one | 2 nA |
| Postsynaptic input | Postsynaptic | One-to-one | 2 nA |
| Presynaptic | Postsynaptic | One-to-one | Plastic |

**(D) Neuron and synapse model**

| | |
|---|---|
| Type | Leaky integrate-and-fire with exponential-shaped synaptic current inputs and spike-frequency adaption (Liu and Wang, 2001) as described in Section 2.3 |
| Parameters | $\tau_m = 10$ ms membrane time constant |
| | $C_m = 250$ pF membrane capacitance |
| | $V_t = -55.4$ mV threshold voltage |
| | $V_r = -70$ mV reset voltage |
| | $\alpha = 0.0$ nA adaption current (disabled) |
| | $\tau_{AMPA} = 2.5$ ms AMPA synapse time constant |

**(E) Plasticity**

| | |
|---|---|
| Type | BCPNN AMPA synapses as described in Section 2.2 |
| Parameters | $f_{max} = 50$ Hz maximum spiking frequency |
| | $\tau_{z_i} = 10$ ms presynaptic primary trace time constant |
| | $\tau_{z_j} = 10$ ms postsynaptic primary trace time constant |
| | $\tau_p = 1000$ ms probability trace time constant |
| | $w_{gain}^{syn} = 1$ nA weight gain |
| | $\beta_{gain} = 1$ nA intrinsic bias gain |

**(F) Input**

| Type | Description |
|---|---|
| Externally generated Poisson spike trains | As described in Section 3.1 |

*After* Nordlie et al. (2009).

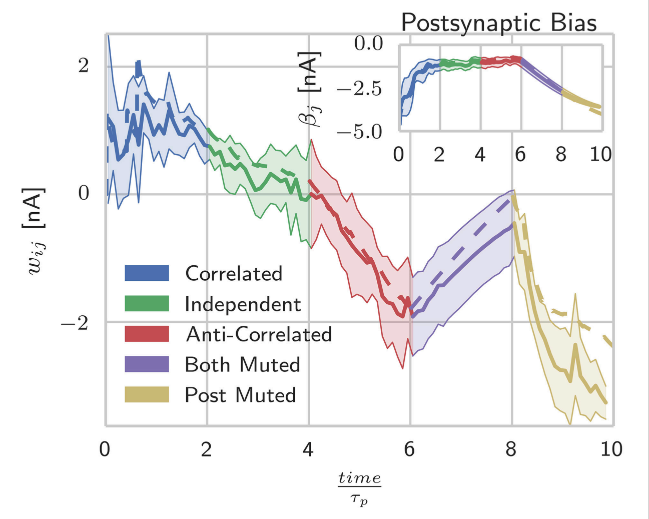

**FIGURE 2 | Spike-based BCPNN estimates abstract BCPNN for different input patterns.** Comparing weight and bias (inset) development under different protocols when using abstract (dotted) and SpiNNaker (solid) versions of the learning rule. SpiNNaker simulations were repeated 10 times and averaged, with standard deviations illustrated by the shaded regions.

followed this learning phase in which a 50 ms stimulus of $f_{max}$ Hz was applied to all cells in the first minicolumn of the learned sequence. During both the training and recall phases we provided background input to each cell in the network from an independent 65 Hz Poisson spike source. These Poisson spike sources are simulated on additional SpiNNaker cores to those used for the neural simulation algorithm described in Section 2.4.

We found that the training regime was able to produce the recurrent connectivity required to perform temporal sequence recall in the same serial order that patterns were presented during training as shown in **Figure 3B**. Each sequence element replayed as a learned attractor state that temporarily stifled the activity of all other cells in the network due to WTA and asymmetrically projected NMDA toward neurons of the subsequent sequence element, allowing a stable trajectory to form. Activity within attractor states was sharpened and stabilized by learned auto-associative AMPA connectivity; and sequential transitions were jointly enabled by neural adaptation and inter-pattern heteroassociation via NMDA synapses.

Because of the modular structure of the network described in Section 2.1, this temporal sequence learning can be performed using networks of varying scales by instantiating different number of hypercolumns and linearly scaling the $w_{gain}^{syn}$ parameter of the connections between them. By doing this, we investigated how the time taken to simulate the network on SpiNNaker scales with network size. **Figure 4** shows how these times are split between the training and testing phases; and how long is spent generating data on the host computer, transferring it to and from SpiNNaker and actually running the simulation. As the SpiNNaker simulation always runs at a fixed fraction of real-time (for this simulation 0.5×), the simulation time remains constant

TABLE 2 | Parameters for the modular attractor network.

**(A) Model summary**

| | |
|---|---|
| Populations and connectivity | Modular structure described in Section 2.1 |
| Neuron model | Leaky integrate-and-fire with exponential-shaped synaptic current inputs and spike-frequency adaption (Liu and Wang, 2001) |
| Synapse model | Current-based with exponential-shaped PSCs |
| Plasticity | BCPNN AMPA and NMDA synapses |
| Input | Externally generated Poisson spike trains and independent fixed-rate Poisson spike trains |
| Measurements | Spiking activity, membrane voltages, intrinsic bias current and synaptic weights |

**(B) Neuron and synapse model**

| | |
|---|---|
| Type | Leaky integrate-and-fire with exponential-shaped synaptic current inputs and spike-frequency adaption (Liu and Wang, 2001) as described in Section 2.3 |
| Parameters | $\tau_m = 20$ ms membrane time constant |
| | $C_m = 250$ pF membrane capacitance |
| | $V_t = -50$ mV threshold voltage |
| | $V_r = -70$ mV reset voltage |
| | $\alpha = 0.15$ nA adaption current |
| | $\tau_a = 300$ ms adaption time constant |
| | $\tau_{\mathrm{AMPA}} = 5$ ms AMPA synapse time constant |
| | $\tau_{\mathrm{GABA}} = 5$ ms GABA synapse time constant |
| | $\tau_{\mathrm{NMDA}} = 150$ ms NMDA synapse time constant |

**(C) Plasticity**

| | |
|---|---|
| Type | BCPNN AMPA synapses as described in Section 2.2 |
| Parameters | $f_{max} = 20$ Hz maximum spiking frequency |
| | $\tau_{z_i} = 5$ ms presynaptic primary trace time constant |
| | $\tau_{z_j} = 5$ ms postsynaptic primary trace time constant |
| | $\tau_p = 2000$ ms probability trace time constant |
| | $w_{gain}^{syn} = \frac{0.546}{N_{HC}}$ nA weight gain |
| Type | BCPNN NMDA synapses as described in Section 2.2 |
| Parameters | $f_{max} = 20$ Hz maximum spiking frequency |
| | $\tau_{z_i} = 5$ ms presynaptic primary trace time constant |
| | $\tau_{z_j} = 150$ ms postsynaptic primary trace time constant |
| | $\tau_p = 2000$ ms probability trace time constant |
| | $w_{gain}^{syn} = \frac{0.114}{N_{HC}}$ nA weight gain |
| | $\beta_{gain} = 0.05$ nA intrinsic bias gain |

**(D) Input**

| Type | Description |
|---|---|
| Externally generated Poisson spike trains | As described in Section 3.2 |
| Independent fixed-rate Poisson spike trains | As described in Section 2.1 |

*After Nordlie et al. (2009).*

as the network grows, but the times required to generate the data and to transfer it grow significantly, meaning that when $N_{HC} = 16$ ($2.0 \times 10^4$ neurons and $5.1 \times 10^7$ plastic synapses), the total simulation time is 146 min. However, the amount of time spent in several phases of the simulation is increased by limitations of the current SpiNNaker toolchain. 84 min is spent downloading the learned weight matrices and re-uploading them for the testing: A process that is only required because the changing of parameters (in this case, whether learning is enabled or not) mid-simulation is not currently supported. Additionally, the current implementation of the algorithm outlined in Section 2.4 only allows neurons simulated on one core to have afferent synapses with a single learning rule configuration. This means that we have to run the training regime twice with the same input spike trains, once for the AMPA synapses and once for the NMDA synapses: Doubling the time taken to simulate the training network.

Previous supercomputer simulations of modular attractor memory networks have often used more complex neuron models and connectivity (Lundqvist et al., 2010), making simulation times difficult to compare with our SpiNNaker simulation due to the simplifications we outlined in Section 2.1. In order to present a better comparison, we built a network model with the same connectivity as our SpiNNaker model and simulated it on a Cray XC-30 supercomputer system using NEST version 2.2 (Gewaltig and Diesmann, 2007) with the spike-based BCPNN implementation developed by Tully et al. (2014). NEST does not include the adaptive neuron model we described in Section 2.3 so we used the adaptive exponential model (Brette and Gerstner, 2005): a simple point neuron model with spike-frequency adaption.

As previously discussed SpiNNaker runs at a fixed-fraction of real-time so we distribute our NEST simulations across increasing numbers of Cray XC-30 compute nodes (each consisting of two 2.5 GHz Intel Ivy Bridge Xeon processors) until the simulation completed in the same time as those shown in **Figure 4** for our SpiNNaker simulations. **Table 3** shows the result of both these supercomputer simulations and a second set with the time taken for the mid-simulation downloading and re-uploading of weights—currently required by the SpiNNaker software—removed. Due to this redundant step and because NEST parallelizes the generation of simulation data across the compute nodes, at all three scales, our modular attractor network can be simulated using 2 compute nodes. However, if we remove the time spent downloading and re-uploading the weights, 9 compute nodes are required to match the run-time of the SpiNNaker simulation when $N_{HC} = 16$.

While a more in-depth measurement of power usage is out of the scope of this work, we can also derive approximate figures for the power usage of our simulations running on both systems based on the 1 W peak power usage of the SpiNNaker chip and the 30 kW power usage of a Cray XC-30 compute rack (Cray, 2013). While these figures ignore the power consumed by the host computer connected to the SpiNNaker system; the power consumed by the "blower" and storage cabinets connected to the Cray XC-30; and assume that all CPUs are running at peak power usage, they show that even in the worst case, SpiNNaker uses 45× less power than the Cray XC-30 and, if the limitations of the current SpiNNaker software are addressed, this can be improved to 200×.

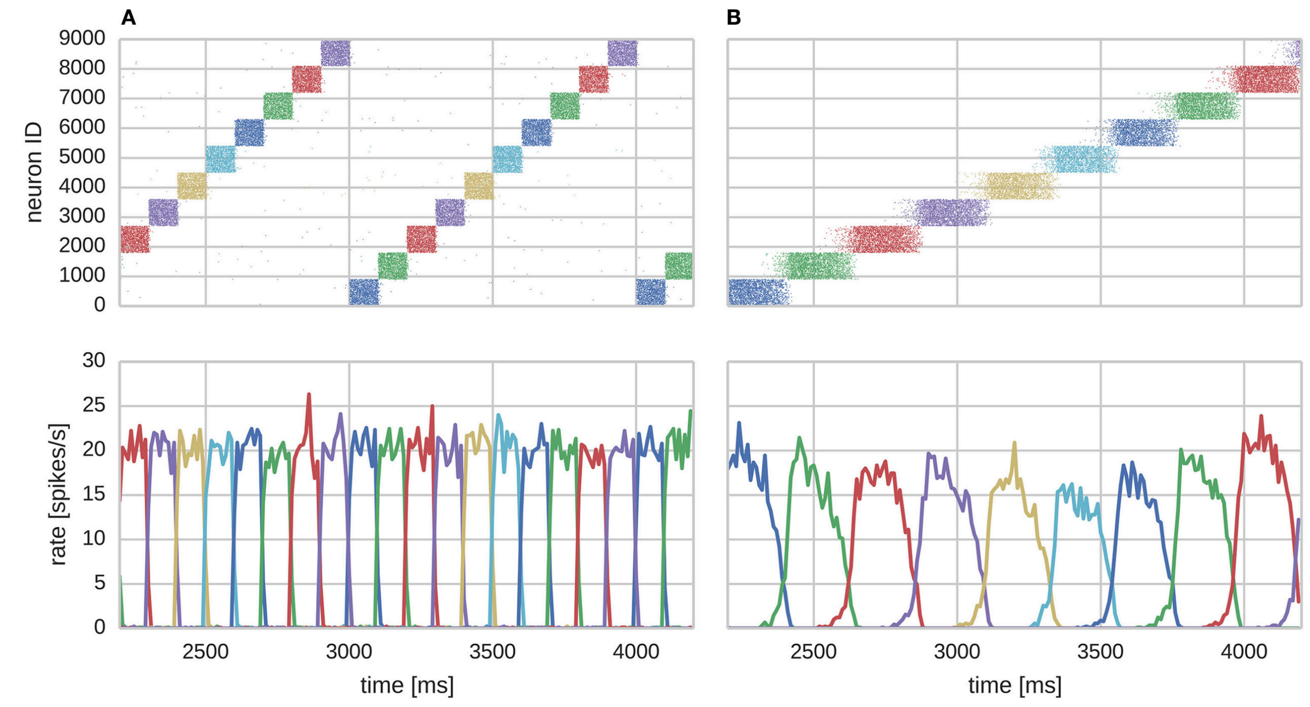

**FIGURE 3 | Learning sequential attractor states. (A)** Training. **(B)** Replay.

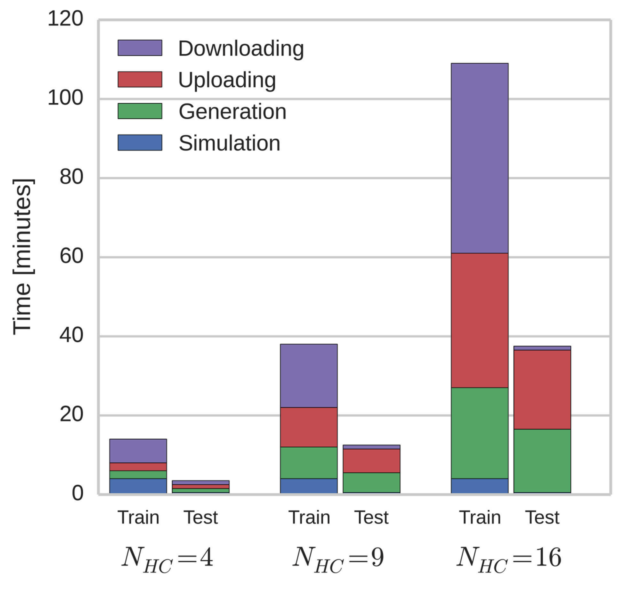

**FIGURE 4 | Total simulation time on SpiNNaker.**

## 3.3. Connectivity Patterns Show Different Signatures in Membrane Potentials

The purpose of this section is to study how learning parameters influence the resulting connectivity patterns and the effect of learned connectivity on membrane dynamics during sequence replay. For this purpose we vary two parameters of the learning rule that control the time window within which correlations are detected – $\tau_{z_i}$ on the pre- and $\tau_{z_j}$ on the postsynaptic side. The network is trained using the same regime described in Section 3.2 and two different configurations, one with $\tau_{z_i} = \tau_{z_j}$ on NMDA synapses, and one with $\tau_{z_i} \neq \tau_{z_j}$. If $\tau_{z_i}$ and $\tau_{z_j}$ are equal, the $Z_i$ and $Z_j$ traces evolve in the same manner, meaning that, as their dynamics propagate through the $P$ traces to the synaptic weights, the forward and reciprocal connections between minicolumns develop symmetrically as shown in the top row of **Figure 5**. However, when $\tau_{z_i} \neq \tau_{z_j}$, the $Z_i$ and $Z_j$ traces evolve differently and, as the bottom row of **Figure 5** shows, asymmetrical connections develop between minicolumns. It is important to note that the spiking activity during the training regime is the same in both configurations and the shape of the resulting connectivity results only from the learning time-constants $\tau_{z_i}$ and $\tau_{z_j}$.

In order to analyze the effect of the different learned connectivity patterns shown in **Figure 5**, we studied the impact of the two connectivity kernels on the subthreshold dynamics of neurons during sequence replay. As described in Section 3.2, after training, the trained sequence can be replayed by applying a 50 ms stimulus of $f_{max}$ Hz into all cells of the first minicolumn in the learned sequence. Later, in the sequence replay when the stimulus has been removed, we recorded the membrane potential of all of the neurons in the network and stored the point in time when the firing rate of the respective minicolumn was maximal. We then align the membrane potential traces to this point in time and average them over all cells in a minicolumn. Interestingly,

**TABLE 3 | Comparison of power usage of modular attractor network simulations running on SpiNNaker with simulations distributed across enough compute nodes of a Cray XC-30 system to match SpiNNaker simulation time.**

| Simulation | | SpiNNaker | | Cray XC-30 | |
|---|---|---|---|---|---|
| $N_{HC}$ | time [min] | # chips | Peak CPU power usage [W] | # compute nodes | Peak CPU power usage [W] |
| 4 | 17 | 6 | 6 | 2 | 938 |
| 9 | 50 | 12 | 12 | 2 | 938 |
| 16 | 146 | 21 | 21 | 2 | 938 |
| 4 | 9 | 6 | 6 | 4 | 1875 |
| 9 | 23 | 12 | 12 | 14[a] | 6563 |
| 16 | 62 | 21 | 21 | 9 | 4219 |

*Cray XC-30 power usage is based on the 30 kW power usage of an entire Cray XC-30 compute rack (Cray, 2013). SpiNNaker power usage is based on the 1 W peak power usage of the SpiNNaker chip (Furber et al., 2014).*

*Top: SpiNNaker simulation times include downloading of learned weights and re-uploading required by current software.*

*Bottom: Time taken to download learned weights, re-generate and re-upload model to SpiNNaker have been removed.*

[a]*We are unsure why more supercomputer compute nodes are required to match the SpiNNaker simulation times when $N_{HC} = 9$ than when $N_{HC} = 16$. We assume this is an artifact of the different scaling properties of the two simulators, but further investigation is outside of the scope of this work.*

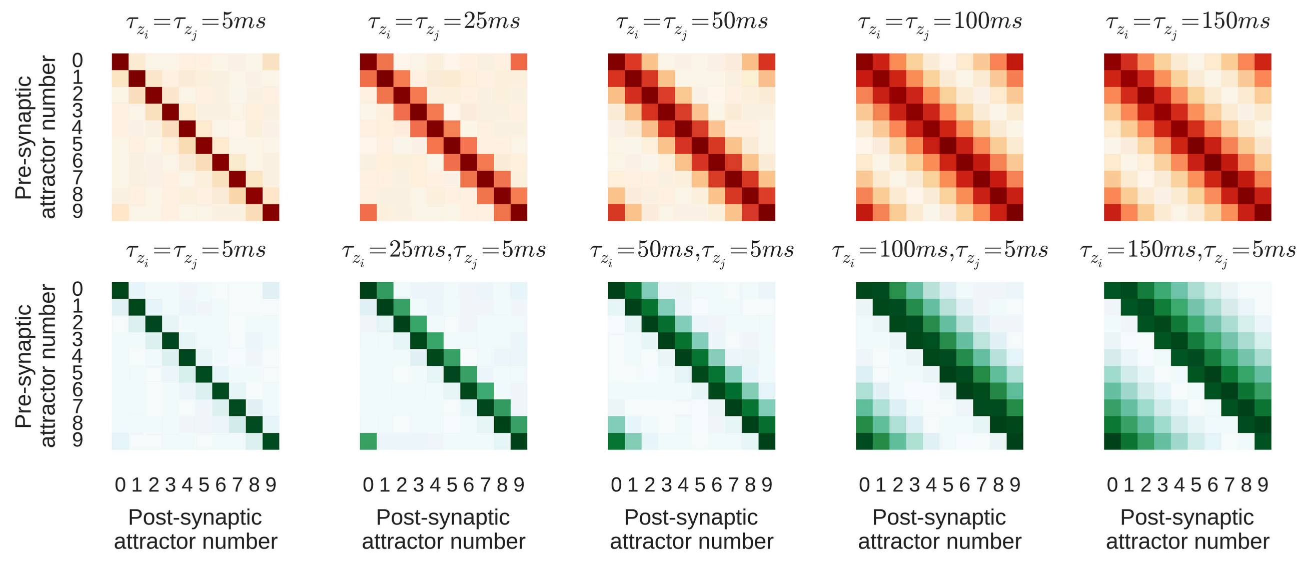

**FIGURE 5 | Average strength of NMDA connections between attractors resulting from different learning time constants.** Darker colors correspond to larger synaptic weights. $\tau_{z_i}$ increases from left-to-right. Top, red row: Symmetrical kernel with $\tau_{z_j} = \tau_{z_i}$. Bottom, green row: Asymmetrical kernel with $\tau_{z_j} = 5$ ms.

as **Figure 6** illustrates, these averaged and aligned membrane responses show different characteristics for the network models built on symmetric and asymmetric connectivity. Both network types show similar membrane characteristics before the sequence arrives at the minicolumn, but, the network with symmetric connectivity shows a significantly slower decrease in membrane potential after the sequence has passed. In contrast, the network with asymmetric connectivity shows a strong after-stimulus hyperpolarization due to the increased inhibitory input originating from minicolumns later in the sequence which get subsequently activated. The slower decrease in the mean membrane potential in the symmetric network can be explained by the excitatory projections in both directions of the sequence providing excitatory current flow to previously activated neurons. The implications of this experiment and interpretations of these different characteristics is discussed in Section 4.2.

## 4. DISCUSSION

The contribution of this study is threefold. Firstly, we have shown that BCPNN can be efficiently implemented within the constraints of the SpiNNaker neuromorphic architecture. Secondly, we have shown how BCPNN can be used in a functionally meaningful context to perform near real-time learning of temporal sequences within a large-scale modular attractor network—the largest plastic neural network ever to

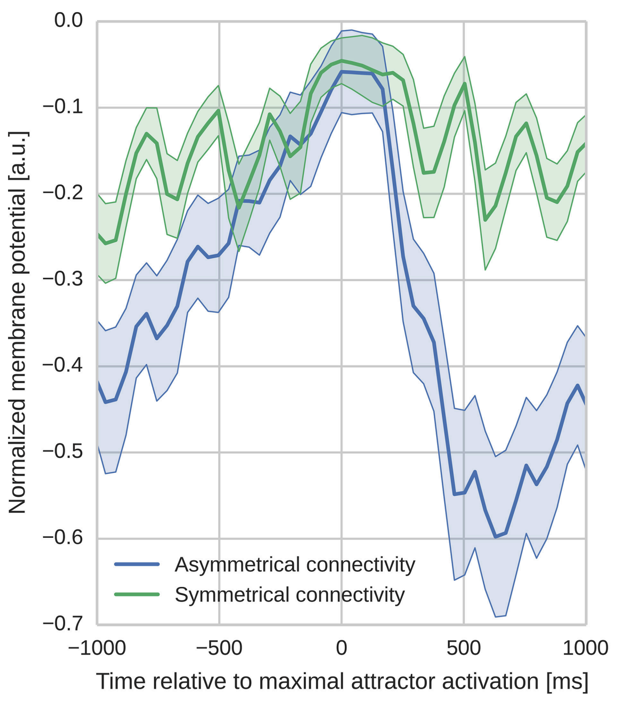

**FIGURE 6 | Aligned average membrane potentials during sequence replay for two different connectivities.** The membrane potentials have been recorded from all neurons in the trained network during sequence replay. These membrane voltages have then been averaged and aligned to the time of peak activity in the temporal domain (bold lines represent the mean, shaded areas represent the standard deviation). The y-axis has been normalized to improve visibility (0 corresponds to $V_t$ and −1 corresponds to the minimal membrane voltage in the sample). In the network with asymmetric connectivity the mean membrane response shows a pronounced drop after the peak response, whereas the network with symmetric connectivity does not. Oscillatory behavior originates from switches between discrete attractor states alternated by phases of inhibitory feedback.

be simulated on neuromorphic hardware. Finally, we have demonstrated the value of SpiNNaker as a tool for investigating plasticity within large-scale brain models by exploring how, by changing a single parameter in the BCPNN learning rule, both symmetric and asymmetric connectivity can be learned, which in turn influence underlying membrane potential characteristics.

## 4.1. Learning Temporal Sequences in Cortical Microcircuits

The total duration of temporal sequences is longer than the time courses of individual cellular or synaptic processes and therefore, such sequences are thought to be driven by circuit-level phenomena although the intricacies of this relationship have yet to be fully explored. The massively recurrent and long-range nature of cortical connectivity, taken together with the emergence of temporal sequences at fine scales and distributed over spatial areas, suggests the presence of generic cortical microcircuit mechanisms. The model presented here is modularly organized into hypercolumns, each implementing WTA dynamics (Douglas and Martin, 2004). This modular structure also allowed us to vary the number of hypercolumns the network contained without effecting its functionality (Djurfeldt et al., 2008). Such distributed systems generally exhibit an improved signal-to-noise ratio, error resilience, generalizability and a structure suitable for Bayesian calculations (McClelland et al., 1986; Barlow, 2001). Like their uniformly interconnected counterparts, they can also exhibit high variability in their spike train statistics (Litwin-Kumar and Doiron, 2012; Lundqvist et al., 2012). Moreover, due to their capacity to exhibit a rich repertoire of behaviorally relevant activity states, these topologies are also well suited for information processing and stimulus sensitivity (Lundqvist et al., 2010; Wang et al., 2011).

Previous investigations have shown that the attractors which emerge within such modular networks reproduce features of local UP states (Lundqvist et al., 2006). This observation remains consistent with the extension considered here since, *in vivo*, UP state onsets are accompanied by the sequential activation of cortical neurons (Luczak et al., 2007). This redundant neural coding scheme should not necessarily be viewed in terms of anatomical columns, but rather functional columns consisting of subgroups of neurons with similar receptive fields that are highly connected (Yoshimura et al., 2005) and co-active (Cossart et al., 2003). Similar stereotypical architectures have been used as building blocks for other unified mathematical frameworks of the neocortex (Johansson and Lansner, 2007; George and Hawkins, 2009; Bastos et al., 2012).

The dynamics of the model consists of attractors, whose activations produce self-sustaining spiking among groups of neurons spread across different hypercolumns. Activity within attractors is sharpened by the fast dynamics of the AMPA receptor, until the network transitions to a subsequent attractor due to neural adaptation and asymmetrical NMDA connectivity, both of which have longer time constants of activation. In this work we have shown how these dynamics could be learned using BCPNN, a learning rule which offers an alternative to phenomenological STDP rules that often require complementary mechanisms due to their prevailing instability (Kempter et al. 2001; Babadi and Abbott 2010; but see Gütig et al. 2003).

## 4.2. Sequence Anticipation and Asymmetric Connectivity as Observed in the Membrane Potential Dynamics

In both the symmetric and asymmetric networks, the stimulus-aligned mean membrane potential traces show a similar rise prior to sequence arrival which can be interpreted as a form of anticipation of the impending activity peak. By anticipation we mean the premature build-up of a neuronal response which is becoming increasingly similar to the response when the actual stimulus is present and represents the neural signature of expectation or prediction of future input. Anticipation is an important function of neural circuits and is observed not only in early sensory structures such as the retina (Berry et al., 1999; Hosoya et al., 2005; Vaney et al., 2012), but also in downstream structures and cortical areas (Rao and Ballard, 1999; Enns and Lleras, 2008) which are involved in more abstract cognitive tasks (Riegler, 2001; Butz et al., 2003). Anticipation can also be regarded as a form of prediction of future events: something which Bar (2007) and Bubic et al. (2010) argue is a fundamental function of the brain. This predictive capability can improve sensory perception (Yoshida and Katz, 2011; Rohenkohl et al., 2012) and is important for other modalities such as motor control and learning (Shadmehr et al., 2010; Schlerf et al., 2012). However, the connectivity which implements this predictive or anticipatory function, and the mechanisms which give rise to it, are not well understood. We believe that BCPNN learning helps fill this gap—as we discussed in Section 3.2, it can learn functional connectivity at a network scale and, as previously argued in this section, it exhibits anticipatory behavior.

We studied the network response by looking at the membrane potential dynamics prior to and after a stimulus and compared the response of two network connectivities trained with different learning parameters. As membrane potential dynamics are the result of a multitude of parameters, we constructed identical experimental settings in terms of input, to make sure that the differences in the membrane potential dynamics can be linked as closely as possible to the differences in the underlying connectivity. That is, the only major difference between the two settings is the characteristic shape of the connectivity (either being symmetric or asymmetric, see **Figure 5**) resulting from different learning parameters. Since the two models implement a different flow of recurrent excitation, the gain parameters in both networks have been adjusted so that both operate in a similar activity regime in order to enable a meaningful comparison of the temporal characteristics introduced by the connectivity shape. The voltage traces arising from the different network connectivities shown in **Figure 6** exhibit different post-stimulus characteristics during sequence replay, with a faster hyperpolarization happening in networks with asymmetric connectivity. Hence we propose that by aligning the average membrane voltage of a population of neurons—in a perceptual context, to its preferred stimulus and, in a task-related context, to its peak activity—and then analyzing the post-stimulus characteristics of this average voltage, the population's afferent connectivity can be inferred.

## 4.3. Asymmetric Connectivity Supports Motion Preference

In the context of visual perception of motion, asymmetric connectivity has been found to play an important role in direction-sensitive ganglion cells in the retina (Kim et al., 2008; Vaney et al., 2012). A previous study by Kaplan et al. (2013) proposed asymmetric connectivity as a means of extrapolating the trajectory of a moving stimulus in the absence of a stimulus: similar, in many respects, to the experiment presented here. Recently, this hypothesized tuning property-based connectivity has been confirmed by the observation of neuronal modules in mouse V1 that exhibit similar motion direction preference (Wertz et al., 2015). The model we present here uses a Hebbian-Bayesian rule to explain how such feature-selective connectivity between neurons tuned to similar stimulus features could arise. It could therefore serve not only as a framework for modeling observed connectivity patterns and helping to understand their functional implications, but also as a means of linking experimentally observed connectivity with earlier modeling studies (Kaplan et al., 2013) by explaining how asymmetric connectivity can arise through learning.

The question of how a preferred sequence direction could be learned and replayed is not only relevant for sensory systems, but also other systems where sequence learning, generation and replay are important (Luczak et al., 2015). We addressed this question by training networks with both symmetric and asymmetric connectivity using a single sequence direction. We then triggered sequence replay in both networks in a similar way to experiments by Gavornik and Bear (2014) and Xu et al. (2012) which studied sequence learning in early visual cortices. Models with symmetric connectivity can show sequence replay in both directions, not only in the trained one. The intuition being that if one were to employ the same training protocol described in Section 3.2, one could replay the sequence forwards or backwards by presenting a cue to the first or last attractor. Instead of being directed by asymmetrical connectivity, the preferred sequence trajectory would evolve according to adaptation. Hence, the direction of the sequence during training alone is not sufficient to create a preferred replay direction as observed in experiments (Xu et al., 2012). Instead, we argue that the asymmetric connectivity caused by a difference in the learning parameters, i.e., an unequal temporal correlation time window, is necessary to replay sequences in only the trained, and therefore preferred, direction.

## 4.4. Results in Context of Anatomical Data

The presented model addresses the question of how connectivity emerges at a cellular and network level in response to temporally varying stimuli. Through usage of different learning time constants, connectivity kernels of varying widths develop as shown in **Figure 5**. There exists a large body of anatomical evidence reporting regional variations in cortical circuitry in terms of structural features such as dendritic morphology and the density of dendritic spines (see e.g., Jacobs and Scheibel, 2002; Elston, 2003 for reviews). In the visual system the hierarchical organization of areas (Riesenhuber and Poggio, 1999) is reflected in their varying dendritic complexity. When compared to areas such as V1, V2, and V4 which respond to simpler visual features, areas associated with more complex functionality also exhibit more complex dendritic morphologies and have a higher number of connections (Jacobs and Scheibel, 2002; Elston and Fujita, 2014).

It stands to reason that the structural and electrophysiological differences observed in both pyramidal cells and interneurons influences activity on a cellular level (Spruston, 2008), shaping the way in which information is integrated and therefore the functional roles of both the individual cells and the entire circuit (Elston, 2003). These regional variations appear to be consistent across species and to change during development (see Elston and Fujita, 2014 for a recent review). Pyramidal cells in V1 reduce their dendritic complexity and those in inferotemporal and prefrontal areas grow larger dendritic structures over the first months and years of development. In the light of the presented model, these observations could lead to the interpretation that reducing the dendritic extent of pyramidal cells mirrors an improved perceptual precision in V1 as finer temporal correlations are detected (represented by short learning time constants $\tau_{z_{i,j}}$ and smaller dendritic extent as shown in the panels on the left side of **Figure 5**). In contrast, as more abstract associations are learned, pyramidal cells in higher areas grow more spines over larger dendritic territories. This allows these cells to integrate information from more diverse sources, requiring integration and association to occur over longer time scales (larger $\tau_{z_{i,j}}$). In this context, it is important to note that the learning time constants may not necessarily equal the synaptic time constants (which are determined by receptor and channel kinetics), but could vary depending on the area and with it the function or task to be learned.

Despite the fact that our model uses point neurons and thus does not directly represent the dendritic field, we argue that the learning time-constants determine a neuron's capability to integrate information over time which – given a topographic stimulus representation such as that seen in V1—could be linked to the size of the dendritic field of a neuron. Hence, the presented learning framework offers the possibility to study these arguments in more quantitative detail.

## 4.5. Scaling the Modular Attractor Model

In Section 3.2 we presented simulations of the modular attractor network model described in Section 2.1 with up to 16 hypercolumns, connected using sparse, random 10 % global connectivity. At this scale each pyramidal cell in the network receives $4.0 \times 10^3$ afferent excitatory synapses but—if the model were scaled up to, for example, the scale of the mouse neocortex with approximately $1.6 \times 10^7$ neurons (Braitenberg and Schüz, 2013)—each pyramidal cell would receive $1.3 \times 10^6$ afferent synapses. As we discussed in Section 4.4, pyramidal cell connectivity varies widely across the layers and areas of the neocortex. However, in this section we base our discussion of the scaling properties of our model on the assumption that each pyramidal cell receives $8.0 \times 10^3$ afferent synapses. This number is consistent with averages calculated across cortical layers and areas in mice (Braitenberg and Schüz, 2013), cats (Beaulieu and Colonnier, 1989) and humans (Pakkenberg et al., 2003). The reason this number is significantly lower than the one obtained by naïvely scaling our current model is because of the "patchy" nature of long-range cortical connectivity (Goldman and Nauta, 1977; DeFelipe et al., 1986; Gilbert and Wiesel, 1989; Bosking et al., 1997). Specifically, each pyramidal cell only connects to around 10, approximately hypercolumn-sized, clusters of neurons located within a radius of a few millimeters. Additionally, while each hypercolumn in our model contains 10 minicolumns, biological hypercolumns typically have closer to 100 (Mountcastle, 1997; Buxhoeveden and Casanova, 2002). This means that, because of the winner-take-all dynamics within each hypercolumn, while 10 % of neurons in our model are active at any given time, only 1 % would be active in a more realistic model.

As Sharp and Furber (2013) discuss, when simulating spiking neural networks on SpiNNaker, the majority of CPU time is spent within the event-driven synaptic processing stage, making the CPU load highly dependent on the rate of incoming *synaptic events* (a single spike innervating a single synapse). The combined effect of the more realistic global connectivity and sparser activity discussed in the previous paragraph would be to reduce the rate of incoming synaptic events by a factor of 5 when compared to our current model. This means that a model with more realistic connectivity could actually run faster than the current model on SpiNNaker - Potentially in biological real-time rather than the 0.5× real-time we use in this work.

However, as we discussed in Section 3.2, the time spent actually running simulations on SpiNNaker is often dwarved by the time spent generating simulation data on the host computer and transferring it to and from the SpiNNaker system. One way of reducing the time taken to generate the simulation data and upload it to SpiNNaker would be to perform some of the data generation on SpiNNaker itself. The most obvious target for this approach would be the generation of the connectivity matrices as, not only do these represent the bulk of the uploaded data, but they are typically defined probabilistically meaning that they could be generated based on a very small uploaded definition. While this approach would undoubtedly reduce the time taken to generate and upload the simulation data, even the 1 min currently taken to download the results at the end of the simulation would grow to several hours if the network was scaled up to the size of even a mouse's neocortex. These slow upload and download times are due to current simulations all having been run on single board SpiNNaker systems, connected to the host computer through a single ethernet link. While the theoretical bandwidth of this link is 100 Mbit $s^{-1}$, inefficiencies in the current SpiNNaker system software reduce the effective bandwidth to only a few MiB $s^{-1}$.

Not only is work currently underway to improve the bandwidth of the ethernet links, but in the case of large-scale network simulations running across multiple SpiNNaker boards, if the host computer is powerful enough and connected to the SpiNNaker system through a sufficiently fast network, data can be transferred to multiple SpiNNaker boards in parallel. Furthermore, if still more bandwidth is required, each SpiNNaker board also has several high-speed serial connectors which could be used for transferring data to and from SpiNNaker at the full 1 Gbit $s^{-1}$ bandwidth of the chip-level interconnect network. Together, the improvements to the scalability of the model discussed in this section would also act to further increase the power efficiency of SpiNNaker when compared to traditional super computer systems that we briefly discuss in Section 3.2.

## 4.6. Extensions of BCPNN on SpiNNaker and other Future Considerations

Since we have shown that BCPNN learning is possible on SpiNNaker, the implementation we describe in Section 2.5 could be extended to support spike-based reinforcement learning (Izhikevich, 2007) by adding an extra level of $E$ (i.e., "eligibility") traces with time constants between those of the $Z$ and $P$ traces (Tully et al., 2014). Representing downstream cellular processes that interact with increased intracellular $Ca^{2+}$ concentrations (Yagishita et al., 2014), $E$ traces propagate into the $P$ traces at a rate previously described as $\kappa$ (Tully et al., 2014). The $\kappa$ parameter models the delivery of delayed reward signals in the form of interactions with global neuromodulatory systems, which have been linked to the emergence of sequential activity (Gavornik and Bear, 2014; Ikegaya, 2004). Using this extended BCPNN model, the modular attractor memory model we describe in Section 2.1 could be extended to include basal ganglia input (Berthet et al., 2012), allowing it to switch between behavioral sequences when this might be a beneficial strategy for successful task completion (Ponzi and Wickens, 2010).

Similarly, Vogginger et al.'s (2015) original event-driven BCPNN model includes a set of $E^*$ state variables which are used to represent the components of the spike-response model arising from $E$ trace dynamics. Though omitted here, the SpiNNaker BCPNN implementation could be extended to include these traces at the cost of some extra computational cost, and the memory required to store an additional 16 bit trace with each synapse and with each entry in the postsynaptic history structure. In Section 3.1 we showed that by using a 16 bit fixed-point representation for the $Z^*$ and $P^*$ state variables, we can produce results comparable to previous floating-point implementations when both $\tau_p$ and $f_{max}$ are relatively small. However, this approach doesn't scale to the type of model described by Fiebig and Lansner (2014) where learning time constants span many orders of magnitude. In these situations, it may be necessary to use a 32 bit fixed-point representation for the $P^*$ traces, further increasing the memory and computational cost of the learning rule.

As spikes from neuromodulator-releasing populations can arrive at the synapse at any time, integrating spike-based reinforcement learning into an event-driven, distributed simulation requires incorporating the times of modulatory as well as postsynaptic spikes into algorithm 1. Because entire populations of neuromodulator-releasing neurons can affect the modulatory input received by a single synapse, the per-neuron history structure discussed in Section 2.4 is not a viable means of storing them. Potjans et al. (2010) extend Morrison et al.'s (2007) STDP algorithm to support neuromodulated learning by introducing "volume transmitter" populations which handle all the incoming modulatory input to a virtual "volume." These populations maintain a spike-history of all incoming modulatory spikes which they deliver to the synapses of neuronal populations within this volume, both at presynaptic spike times and after a fixed period so as to 'flush out' the spike-history data structure and allow it to be kept relatively small. This approach has the potential to map well to the SpiNNaker architecture and could be used as the basis of a future SpiNNaker implementation of spike-based reinforcement learning using BCPNN.

A benefit of the model proposed here is its robustness and flexibility. Non-sequential attractor networks without learning have previously been emulated on a neuromorphic microchip (Pfeil et al., 2013) and on a simulated version of the BrainScaleS system (Petrovici et al., 2014). Though not shown here, the connectivity required by these types of randomly hopping attractor networks can also be learned. Variations of this network run on supercomputers have been shown to account for disparate cognitive phenomena including perceptual rivalry and completion (Kaplan and Lansner, 2013); attentional blink (Lundqvist et al., 2006; Silverstein and Lansner, 2011); and diverse oscillatory regimes (Lundqvist et al., 2010). But our model was a reduced version of previous detailed ones insofar that we did not utilize Hodgkin-Huxley neurons with calcium-dependent potassium channels or regular spiking non-pyramidal cells; nor did we explicitly model connections among basket cells, saturating synapses, a Vm-dependent $Mg^{2+}$ blockade or short-term depression.

A problem not stressed by the aforementioned models is how the connectivity required for stable activity propagation might be learned (Wörgötter and Porr, 2005; Kunkel et al., 2011), despite the biochemical (Peters et al., 2014) and metabolic (Picard et al., 2013) changes accompanying learned sequential behaviors. Several promising approaches have been developed (Sussillo and Abbott, 2009; Laje and Buonomano, 2013; Hennequin et al., 2014), albeit with biological motivations driven more from the perspective of algorithmic optimization, rather than from bottom-up neural processing. Here, we have shown that activity could propagate through recurrent cortical microcircuits as a result of a probabilistic learning rule based on neurobiologically plausible time courses and dynamics. The model predicts that the interaction between several learning and dynamical processes constitute a compound mnemonic engram that can flexibly generate step-wise sequential increases of activity within pools of excitatory neurons. We have shown that this large-scale learning model can be efficiently simulated at scale using neuromorphic hardware and our simulations suggest that flexible systems such as SpiNNaker offer a promising tool for the study of collective dynamical phenomena emerging from the complex interactions occurring between individual neurons and synapses whose properties change over time.

## AUTHOR CONTRIBUTIONS

JK developed the SpiNNaker BCPNN implementation and performed the experiments. JK and BK analyzed the data. PT, AL, and JK developed the simplified modular attractor network architecture. JK, PT, BK, AL, and SF wrote the paper.

## ACKNOWLEDGMENTS

The design and construction of SpiNNaker was funded by EPSRC (the UK Engineering and Physical Sciences Research Council) under grants EP/G015740/1 and EP/G015775/1. The research was supported by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement 320689 and also by the EU Flagship Human Brain Project (FP7-604102) and the EU BrainScales project (EU-FP7-FET-269921). JK is supported by a Kilburn Studentship from the School of Computer Science at The University of Manchester. Additionally, we thank Andrew Mundy for creating **Figure 1**; Michael Hopkins, Patrick Camilleri, and Luis Plana for taking the time to read the manuscript and provide feedback; and our reviewers for their helpful and valuable feedback.

## REFERENCES

Abbott, L. F., and Blum, K. I. (1996). Functional significance of long-term potentiation for sequence learning and prediction. *Cereb. Cortex* 6, 406–416. doi: 10.1093/cercor/6.3.406

Abeles, M., Bergman, H., Gat, I., Meilijson, I., Seidemann, E., Tishby, N., et al. (1995). Cortical activity flips among quasi-stationary states. *Proc. Natl. Acad. Sci. U.S.A.* 92, 8616–8620. doi: 10.1073/pnas.92.19.8616

Babadi, B., and Abbott, L. F. (2010). Intrinsic stability of temporally shifted spike-timing dependent plasticity. *PLoS Comput. Biol.* 6:e1000961. doi: 10.1371/journal.pcbi.1000961

Bar, M. (2007). The proactive brain: using analogies and associations to generate predictions. *Trends Cogn. Sci.* 11, 280–289. doi: 10.1016/j.tics.2007.05.005

Barlow, H. (2001). Redundancy reduction revisited. *Network* 12, 241–253. doi: 10.1080/net.12.3.241.253

Bastos, A. M., Usrey, W. M., Adams, R. A., Mangun, G. R., Fries, P., and Friston, K. J. (2012). Canonical microcircuits for predictive coding. *Neuron* 76, 695–711. doi: 10.1016/j.neuron.2012.10.038

Beaulieu, C., and Colonnier, M. (1989). Number and size of neurons and synapses in the motor cortex of cats raised in different environmental complexities. *J. Comp. Neurol.* 289, 178–187. doi: 10.1002/cne.9028 90115

Benjamin, B. V., Gao, P., McQuinn, E., Choudhary, S., Chandrasekaran, A. R., Bussat, J. M., et al. (2014). Neurogrid: a mixed-analog-digital multichip system for large-scale neural simulations. *Proc. IEEE* 102, 699–716. doi: 10.1109/JPROC.2014.2313565

Berry, M. J. II, Brivanlou, I. H., Jordan, T. A., and Meister, M. (1999). Anticipation of moving stimuli by the retina. *Nature* 398, 334–338. doi: 10.1038/ 18678

Berthet, P., Hellgren-Kotaleski, J., and Lansner, A. (2012). Action selection performance of a reconfigurable basal ganglia inspired model with hebbian–bayesian go-nogo connectivity. *Front. Behav. Neurosci.* 6:65. doi: 10.3389/fnbeh.2012.00065

Bi, G.-Q., and Poo, M.-M. (1998). Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *J. Neurosci.* 18, 10464–10472.

Bosking, W. H., Zhang, Y., Schofield, B., and Fitzpatrick, D. (1997). Orientation selectivity and the arrangement of horizontal connections in tree shrew striate cortex. *J. Neurosci.* 17, 2112–2127.

Braitenberg, V., and Schüz, A. (2013). *Cortex: Statistics and Geometry of Neuronal Connectivity*. Berlin; Heidelberg; New York, NY: Springer Science & Business Media.

Brette, R., and Gerstner, W. (2005). Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *J. Neurophysiol.* 94, 3637–3642. doi: 10.1152/jn.00686.2005

Bubic, A., von Cramon, D. Y., and Schubotz, R. I. (2010). Prediction, cognition and the brain. *Front. Hum. Neurosci.* 4:25. doi: 10.3389/fnhum.2010.00025

Butz, M. V., Sigaud, O., and Gérard, P. (2003). "Internal models and anticipations in adaptive learning systems," in *Anticipatory Behavior in Adaptive Learning Systems*, eds M. V. Butz, O. Sigaud, and P. Gérard (Heidelberg; Berlin: Springer), 86–109. doi: 10.1007/978-3-540-45002-3_6

Buxhoeveden, D. P., and Casanova, M. F. (2002). The minicolumn hypothesis in neuroscience. *Brain* 125, 935–951. doi: 10.1093/brain/awf110

Caporale, N., and Dan, Y. (2008). Spike timing-dependent plasticity: a hebbian learning rule. *Annu. Rev. Neurosci.* 31, 25–46. doi: 10.1146/annurev.neuro.31.060407.125639

Cossart, R., Aronov, D., and Yuste, R. (2003). Attractor dynamics of network up states in the neocortex. *Nature* 423, 283–288. doi: 10.1038/nature 01614

Cray (2013). Cray XC30-ACø Supercomputer. Technical Report, Cray Inc.

Dan, Y., and Poo, M.-M. (2004). Spike timing-dependent plasticity of neural circuits. *Neuron* 44, 23–30. doi: 10.1016/j.neuron.2004.09.007

Daoudal, G., and Debanne, D. (2003). Long-term plasticity of intrinsic excitability: learning rules and mechanisms. *Learn. Mem.* 10, 456–465. doi: 10.1101/lm.64103

DeFelipe, J., Conley, M., and Jones, E. (1986). Long-range focal collateralization of axons arising from corticocortical cells in monkey sensory-motor cortex. *J. Neurosci.* 6, 3749–3766.

Diehl, P. U., and Cook, M. (2014). "Efficient Implementation of STDP Rules on SpiNNaker Neuromorphic Hardware," in *Neural Networks (IJCNN), The 2014 International Joint Conference on* (Beijing), 4288–4295. doi: 10.1109/IJCNN.2014.6889876

Djurfeldt, M., Lundqvist, M., Johansson, C., Rehn, M., Ekeberg, O., and Lansner, A. (2008). Brain-scale simulation of the neocortex on the IBM Blue Gene/L supercomputer. *IBM J. Res. Dev.* 52, 31–41. doi: 10.1147/rd.521.0031

Douglas, R. J., and Martin, K. A. (2004). Neuronal circuits of the neocortex. *Annu. Rev. Neurosci.* 27, 419–451. doi: 10.1146/annurev.neuro.27.070203.144152

Elston, G. N. (2003). Cortex, cognition and the cell: new insights into the pyramidal neuron and prefrontal function. *Cereb. Cortex* 13, 1124–1138. doi: 10.1093/cercor/bhg093

Elston, G. N., and Fujita, I. (2014). Pyramidal cell development: postnatal spinogenesis, dendritic growth, axon growth, and electrophysiology. *Front. Neuroanat.* 8:78. doi: 10.3389/fnana.2014.00078

Enns, J. T., and Lleras, A. (2008). What's next? new evidence for prediction in human vision. *Trends Cogn. Sci.* 12, 327–333. doi: 10.1016/j.tics.2008.06.001

Feldman, D. E. (2012). The spike-timing dependence of plasticity. *Neuron* 75, 556–571. doi: 10.1016/j.neuron.2012.08.001

Fiebig, F., and Lansner, A. (2014). Memory consolidation from seconds to weeks : a three-stage neural memory consolidation from seconds to weeks : a three-stage neural network model with autonomous reinstatement dynamics. *Front. Comput. Neurosci.* 8:64. doi: 10.3389/fncom.2014.00064

Furber, S. B., Galluppi, F., Temple, S., and Plana, L. A. (2014). The SpiNNaker project. *Proc. IEEE* 102, 652–665. doi: 10.1109/JPROC.2014.2304638

Galluppi, F., Lagorce, X., Stromatias, E., Pfeiffer, M., Plana, L. A., Furber, S. B., et al. (2015). A framework for plasticity implementation on the SpiNNaker neural architecture. *Front. Neurosci.* 8:429. doi: 10.3389/fnins.2014.00429

Gavornik, J. P., and Bear, M. F. (2014). Learned spatiotemporal sequence recognition and prediction in primary visual cortex. *Nat. Neurosci.* 17, 732–737. doi: 10.1038/nn.3683

George, D., and Hawkins, J. (2009). Towards a mathematical theory of cortical micro-circuits. *PLoS Comput. Biol.* 5:e1000532. doi: 10.1371/journal.pcbi.1000532

Gerstner, W., and Kistler, W. M. (2002). *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge: Cambridge University Press. doi: 10.1017/cbo9780511815706

Gewaltig, M.-O., and Diesmann, M. (2007). NEST (NEural simulation tool). *Scholarpedia* 2:1430. doi: 10.4249/scholarpedia.1430

Gilbert, C. D., and Wiesel, T. N. (1989). Columnar specificity of intrinsic horizontal and corticocortical connections in cat visual cortex. *J. Neurosci.* 9, 2432–2442.

Goldman, P. S., and Nauta, W. J. (1977). Columnar distribution of cortico-cortical fibers in the frontal association, limbic, and motor cortex of the developing rhesus monkey. *Brain Res.* 122, 393–413. doi: 10.1016/0006-8993(77)90453-x

Gütig, R., Aharonov, R., Rotter, S., and Sompolinsky, H. (2003). Learning input correlations through nonlinear temporally asymmetric hebbian plasticity. *J. Neurosci.* 23, 3697–3714.

Hennequin, G., Vogels, T. P., and Gerstner, W. (2014). Optimal control of transient dynamics in balanced networks supports generation of complex movements. *Neuron* 82, 1394–1406. doi: 10.1016/j.neuron.2014.04.045

Hopkins, M., and Furber, S. (2015). Accuracy and efficiency in fixed-point neural ODE solvers. *Neural Comput.* 27, 2148–2182. doi: 10.1162/NECO_a_00772

Hosoya, T., Baccus, S. A., and Meister, M. (2005). Dynamic predictive coding by the retina. *Nature* 436, 71–77. doi: 10.1038/nature03689

Ikegaya, Y. (2004). Synfire chains and cortical songs: temporal modules of cortical activity. *Science* 304, 559–564. doi: 10.1126/science.1093173

Izhikevich, E. M. (2007). Solving the distal reward problem through linkage of STDP and dopamine signaling. *Cereb. Cortex* 17, 2443–2452. doi: 10.1093/cercor/bhl152

Jacobs, B., and Scheibel, A. (2002). "Regional dendritic variation in primate cortical pyramidal cells," in *Cortical Areas: Unity Diversity*, eds A. Schuez and R. Miller (London; New York, NY: Tailor and Francis), 111–131. doi: 10.1201/9780203299296.pt2

Jin, X., Rast, A., Galluppi, F., Davies, S., and Furber, S. (2010). "Implementing spike-timing-dependent plasticity on SpiNNaker neuromorphic hardware," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*, Number 1 (Barcelona: IEEE), 1–8. doi: 10.1109/ijcnn.2010.5596372

Johansson, C., and Lansner, A. (2007). Towards cortex sized artificial neural systems. *Neural Netw.* 20, 48–61. doi: 10.1016/j.neunet.2006.05.029

Jones, L. M., Fontanini, A., Sadacca, B. F., Miller, P., and Katz, D. B. (2007). Natural stimuli evoke dynamic sequences of states in sensory cortical ensembles. *Proc. Natl. Acad. Sci. U.S.A.* 104, 18772–18777. doi: 10.1073/pnas.0705546104

Jung, S.-C., and Hoffman, D. A. (2009). Biphasic somatic a-type k+ channel downregulation mediates intrinsic plasticity in hippocampal ca1 pyramidal neurons. *PLoS ONE* 4:e6549. doi: 10.1371/journal.pone.0006549

Kaplan, B. A., and Lansner, A. (2013). A spiking neural network model of self-organized pattern recognition in the early mammalian olfactory system. *Front. Neural Circuits* 8:5. doi: 10.3389/fncir.2014.00005

Kaplan, B. A., Lansner, A., Masson, G. S., and Perrinet, L. U. (2013). Anisotropic connectivity implements motion-based prediction in a spiking neural network. *Front. Comput. Neurosci.* 7:112. doi: 10.3389/fncom.2013.00112

Kempter, R., Gerstner, W., and van Hemmen, J. L. (2001). Intrinsic stabilization of output rates by spike-based hebbian learning. *Neural Comput.* 13, 2709–2741. doi: 10.1162/089976601317098501

Kim, I.-J., Zhang, Y., Yamagata, M., Meister, M., and Sanes, J. R. (2008). Molecular identification of a retinal cell type that responds to upward motion. *Nature* 452, 478–482. doi: 10.1038/nature06739

Kunkel, S., Diesmann, M., and Morrison, A. (2011). Limits to the development of feed-forward structures in large recurrent neuronal networks. *Front. Comput. Neurosci.* 4:160. doi: 10.3389/fncom.2010.00160

Kunkel, S., Potjans, T. C., Eppler, J. M., Plesser, H. E., Morrison, A., and Diesmann, M. (2012). Meeting the memory challenges of brain-scale network simulation. *Front. Neuroinform.* 5:35. doi: 10.3389/fninf.2011.00035

Lagorce, X., Stromatias, E., Galluppi, F., Plana, L. A., Liu, S.-C., Furber, S. B., et al. (2015). Breaking the millisecond barrier on SpiNNaker: implementing asynchronous event-based plastic models with microsecond resolution. *Front. Neurosci.* 9:206. doi: 10.3389/fnins.2015.00206

Laje, R., and Buonomano, D. V. (2013). Robust timing and motor patterns by taming chaos in recurrent neural networks. *Nat. Neurosci.* 16, 925–933. doi: 10.1038/nn.3405

Lansner, A., and Ekeberg, Ö. (1989). A one-layer feedback artificial neural network with a bayesian learning rule. *Int. J. Neural Syst.* 1, 77–87. doi: 10.1142/S0129065789000499

Lansner, A., and Holst, A. (1996). A higher order Bayesian neural network with spiking units. *Int. J. Neural Syst.* 7, 115–128. doi: 10.1142/s0129065796000816

Lisman, J., and Spruston, N. (2005). Postsynaptic depolarization requirements for ltp and ltd: a critique of spike timing-dependent plasticity. *Nat. Neurosci.* 8, 839–841. doi: 10.1038/nn0705-839

Lisman, J., and Spruston, N. (2010). Questions about stdp as a general model of synaptic plasticity. *Front. Synaptic Neurosci.* 2:140. doi: 10.3389/fnsyn.2010.00140

Litwin-Kumar, A., and Doiron, B. (2012). Slow dynamics and high variability in balanced cortical networks with clustered connections. *Nat. Neurosci.* 15, 1498–1505. doi: 10.1038/nn.3220

Liu, Y. H., and Wang, X. J. (2001). Spike-frequency adaptation of a generalized leaky integrate-and-fire model neuron. *J. Comput. Neurosci.* 10, 25–45. doi: 10.1023/A:1008916026143

Luczak, A., Barthó, P., Marguet, S. L., Buzsáki, G., and Harris, K. D. (2007). Sequential structure of neocortical spontaneous activity *in vivo*. *Proc. Natl. Acad. Sci. U.S.A.* 104, 347–352. doi: 10.1073/pnas.0605643104

Luczak, A., McNaughton, B. L., and Harris, K. D. (2015). Packet-based communication in the cortex. *Nat. Rev. Neurosci.* 16, 745–755. doi: 10.1038/nrn4026

Lundqvist, M., Compte, A., and Lansner, A. (2010). Bistable, irregular firing and population oscillations in a modular attractor memory network. *PLoS Comput. Biol.* 6:e1000803. doi: 10.1371/journal.pcbi.1000803

Lundqvist, M., Herman, P., and Lansner, A. (2012). Variability of spike firing during theta-coupled replay of memories in a simulated attractor network. *Brain Res.* 1434, 152–161. doi: 10.1016/j.brainres.2011.07.055

Lundqvist, M., Rehn, M., Djurfeldt, M., and Lansner, A. (2006). Attractor dynamics in a modular network model of neocortex. *Network* 17, 253–276. doi: 10.1080/09548980600774619

Markram, H., Gerstner, W., and Sjöström, P. J. (2011). A history of spike-timing-dependent plasticity. *Front. Synaptic Neurosci.* 3:4. doi: 10.3389/fnsyn.2011.00004

McClelland, J. L., Rumelhart, D. E., and Hinton, G. E. (1986). *The Appeal of Parallel Distributed Processing*. Cambridge, MA: MIT Press.

Mead, C. (1990). Neuromorphic electronic systems. *Proc. IEEE* 78, 1629–1636. doi: 10.1109/5.58356

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. doi: 10.1126/science.1254642

Moise, M. (2012). *A Fixed Point Arithmetic Library for SpiNNaker*. PhD thesis, The University of Manchester, Manchester.

Morrison, A., Aertsen, A., and Diesmann, M. (2007). Spike-timing-dependent plasticity in balanced random networks. *Neural Comput.* 19, 1437–1467. doi: 10.1162/neco.2007.19.6.1437

Morrison, A., Mehring, C., Geisel, T., Aertsen, A. D., and Diesmann, M. (2005). Advancing the boundaries of high-connectivity network simulation with distributed computing. *Neural Comput.* 17, 1776–1801. doi: 10.1162/0899766054026648

Mountcastle, V. B. (1997). The columnar organization of the neocortex. *Brain* 120, 701–722. doi: 10.1093/brain/120.4.701

Nordlie, E., Gewaltig, M. O., and Plesser, H. E. (2009). Towards reproducible descriptions of neuronal network models. *PLoS Comput. Biol.* 5:e1000456. doi: 10.1371/journal.pcbi.1000456

Painkras, E., and Plana, L. (2013). SpiNNaker: a 1-W 18-core system-on-chip for massively-parallel neural network simulation. *IEEE J. Solid-State Circ.* 48, 1943–1953. doi: 10.1109/JSSC.2013.2259038

Pakkenberg, B., Pelvig, D., Marner, L., Bundgaard, M. J., Gundersen, H. J. G., Nyengaard, J. R., et al. (2003). Aging and the human neocortex. *Exp. Gerontol.* 38, 95–99. doi: 10.1016/S0531-5565(02)00151-1

Peters, A. J., Chen, S. X., and Komiyama, T. (2014). Emergence of reproducible spatiotemporal activity during motor learning. *Nature* 510, 263–267. doi: 10.1038/nature13235

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. doi: 10.1371/journal.pone.0108590. Available online at: http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0108590

Pfeil, T., Grübl, A., Jeltsch, S., Müller, E., Müller, P., Petrovici, M. A., et al. (2013). Six networks on a universal neuromorphic computing substrate. *Front. Neurosci.* 7:11. doi: 10.3389/fnins.2013.00011

Picard, N., Matsuzaka, Y., and Strick, P. L. (2013). Extended practice of a motor skill is associated with reduced metabolic activity in m1. *Nat. Neurosci.* 16, 1340–1347. doi: 10.1038/nn.3477

Ponzi, A., and Wickens, J. (2010). Sequentially switching cell assemblies in random inhibitory networks of spiking neurons in the striatum. *J. Neurosci.* 30, 5894–5911. doi: 10.1523/JNEUROSCI.5540-09.2010

Potjans, W., Morrison, A., and Diesmann, M. (2010). Enabling functional neural circuit simulations with distributed computing of neuromodulated plasticity. *Front. Comput. Neurosci.* 4:141. doi: 10.3389/fncom.2010.00141

Rao, R. P., and Ballard, D. H. (1999). Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nat. Neurosci.* 2, 79–87. doi: 10.1038/4580

Rauch, A., La Camera, G., Lüscher, H.-R., Senn, W., and Fusi, S. (2003). Neocortical pyramidal cells respond as integrate-and-fire neurons to *in vivo*–like input currents. *J. Neurophysiol.* 90, 1598–1612. doi: 10.1152/jn.00293.2003

Riegler, A. (2001). "The role of anticipation in cognition," in *AIP Conference Proceedings* (Liege: IOP Institute of Physics Publishing Ltd), 534–544. doi: 10.1063/1.1388719

Riesenhuber, M., and Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nat. Neurosci.* 2, 1019–1025. doi: 10.1038/14819

Rohenkohl, G., Cravo, A. M., Wyart, V., and Nobre, A. C. (2012). Temporal expectation improves the quality of sensory information. *J. Neurosci.* 32, 8424–8428. doi: 10.1523/JNEUROSCI.0804-12.2012

Sandberg, A., Lansner, A., Petersson, K. M., and Ekeberg, O. (2002). A Bayesian attractor network with incremental learning. *Network* 13, 179–194. doi: 10.1080/net.13.2.179.194

Schemmel, J., Bruderle, D., Grubl, 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* (Paris: IEEE), 1947–1950. doi: 10.1109/ISCAS.2010.5536970

Schlerf, J., Ivry, R. B., and Diedrichsen, J. (2012). Encoding of sensory prediction errors in the human cerebellum. *J. Neurosci.* 32, 4913–4922. doi: 10.1523/JNEUROSCI.4504-11.2012

Seidemann, E., Meilijson, I., Abeles, M., Bergman, H., and Vaadia, E. (1996). Simultaneously recorded single units in the frontal cortex go through sequences of discrete and stable states in monkeys performing a delayed localization task. *J. Neurosci.* 16, 752–768.

Shadmehr, R., Smith, M. A., and Krakauer, J. W. (2010). Error correction, sensory prediction, and adaptation in motor control. *Ann. Rev. Neurosci.* 33, 89–108. doi: 10.1146/annurev-neuro-060909-153135

Sharp, T., and Furber, S. (2013). "Correctness and performance of the SpiNNaker architecture," in *Neural Networks (IJCNN), The 2013 International Joint Conference on* (Dallas, TX). doi: 10.1109/ijcnn.2013.6706988

Sharp, T., Galluppi, F., Rast, A., and Furber, S. (2012). Power-efficient simulation of detailed cortical microcircuits on SpiNNaker. *J. Neurosci. Methods* 210, 110–118. doi: 10.1016/j.jneumeth.2012.03.001

Sharp, T., Petersen, R., and Furber, S. (2014). Real-time million-synapse simulation of rat barrel cortex. *Front. Neurosci.* 8:131. doi: 10.3389/fnins.2014.00131

Sheik, S., Coath, M., Indiveri, G., Denham, S. L., Wennekers, T., and Chicca, E. (2012). Emergent auditory feature tuning in a real-time neuromorphic VLSI system. *Front. Neurosci.* 6:17. doi: 10.3389/fnins.2012.00017

Silverstein, D. N., and Lansner, A. (2011). Is attentional blink a byproduct of neocortical attractors? *Front. Comput. Neurosci.* 5:13. doi: 10.3389/fncom.2011.00013

Spruston, N. (2008). Pyramidal neurons: dendritic structure and synaptic integration. *Nat. Rev. Neurosci.* 9, 206–221. doi: 10.1038/nrn2286

Stromatias, E., Galluppi, F., Patterson, C., and Furber, S. (2013). "Power analysis of large-scale, real-time neural networks on SpiNNaker," in *The 2013 International Joint Conference on Neural Networks (IJCNN)* (Dallas, TX), 1–8. doi: 10.1109/IJCNN.2013.6706927

Sussillo, D., and Abbott, L. F. (2009). Generating coherent patterns of activity from chaotic neural networks. *Neuron* 63, 544–557. doi: 10.1016/j.neuron.2009.07.018

Tully, P. J., Hennig, M. H., and Lansner, A. (2014). Synaptic and nonsynaptic plasticity approximating probabilistic inference. *Front. Synaptic Neurosci.* 6:8. doi: 10.3389/fnsyn.2014.00008

Vaney, D. I., Sivyer, B., and Taylor, W. R. (2012). Direction selectivity in the retina: symmetry and asymmetry in structure and function. *Nat. Rev. Neurosci.* 13, 194–208. doi: 10.1038/nrn3165

Vogginger, B., Schüffny, R., Lansner, A., Cederström, L., Partzsch, J., and Höppner, S. (2015). Reducing the computational footprint for real-time BCPNN learning. *Front. Neurosci.* 9:2. doi: 10.3389/fnins.2015.00002

Wang, S.-J., Hilgetag, C. C., and Zhou, C. (2011). Sustained activity in hierarchical modular neural networks: self-organized criticality and oscillations. *Front. Comput. Neurosci.* 5:30. doi: 10.3389/fncom.2011.00030

Wertz, A., Trenholm, S., Yonehara, K., Hillier, D., Raics, Z., Leinweber, M., et al. (2015). Single-cell–initiated monosynaptic tracing reveals layer-specific cortical network modules. *Science* 349, 70–74. doi: 10.1126/science.aab1687

Wörgötter, F., and Porr, B. (2005). Temporal sequence learning, prediction, and control: a review of different models and their relation to biological mechanisms. *Neural Comput.* 17, 245–319. doi: 10.1162/0899766053011555

Xu, S., Jiang, W., Poo, M.-M., and Dan, Y. (2012). Activity recall in a visual cortical ensemble. *Nat. Neurosci.* 15, 449–455. doi: 10.1038/nn.3036

Yagishita, S., Hayashi-Takagi, A., Ellis-Davies, G. C., Urakubo, H., Ishii, S., and Kasai, H. (2014). A critical time window for dopamine actions on the structural plasticity of dendritic spines. *Science* 345, 1616–1620. doi: 10.1126/science.1255514

Yoshida, T., and Katz, D. B. (2011). Control of prestimulus activity related to improved sensory coding within a discrimination task. *J. Neurosci.* 31, 4101–4112. doi: 10.1523/JNEUROSCI.4380-10.2011

Yoshimura, Y., Dantzker, J. L., and Callaway, E. M. (2005). Excitatory cortical neurons form fine-scale functional networks. *Nature* 433, 868–873. doi: 10.1038/nature03252

**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.