Porting HTM Models to the Heidelberg Neuromorphic Computing Platform

Porting HTM Models to the Heidelberg Neuromorphic Computing Platform

Sebastian Billaudelle [1,*], Subutai Ahmad [2,†]

[1] Kirchhoff-Institute for Physics, Heidelberg, Germany
[2] Numenta, Inc., Redwood City, CA

ABSTRACT

Hierarchical Temporal Memory (HTM) is a computational theory of machine intelligence based on a detailed study of the neocortex. The Heidelberg Neuromorphic Computing Platform, developed as part of the Human Brain Project (HBP), is a mixed-signal (analog and digital) large-scale platform for modeling networks of spiking neurons. In this paper we present the first effort in porting HTM networks to this platform. We describe a framework for simulating key HTM operations using spiking network models. We then describe specific spatial pooling and temporal memory implementations, as well as simulations demonstrating that the fundamental properties are maintained. We discuss issues in implementing the full set of plasticity rules using Spike-Timing Dependent Plasticity (STDP), and rough place and route calculations. Although further work is required, our initial studies indicate that it should be possible to run large-scale HTM networks (including plasticity rules) efficiently on the Heidelberg platform. More generally the exercise of porting high level HTM algorithms to biophysical neuron models promises to be a fruitful area of investigation for future studies.

1 INTRODUCTION

The mammalian brain, particularly that of humans, is able to process diverse sensory input, learn and recognize complex spatial and temporal patterns, and generate behaviour based on context and previous experiences. While computers are efficient in carrying out numerical calculations, they fall short in solving cognitive tasks. Studying the brain and the neocortex in particular is an important step to develop new algorithms closing the gap between intelligent organisms and artificial systems. Numenta is a company dedicated to developing such algorithms and at the same time investigating the principles of the neocortex. Their Hierarchical Temporal Memory (HTM) models are designed to solve real world problems based on neuroscience results and theories.

Efficiently simulating large-scale neural networks in software is still a challenge. The more biophysical details a model features, the more computational ressources it requires. Different techniques for speeding up the execution of such implementations exist, e.g. by parallelizing calculations. Dedicated hardware platforms are also being developed. Digital neuromorphic hardware like the SpiNNaker platform often features highly parallelized processing architectures and optimized signal routing [Furber et al., 2014]. On the other hand, analog systems directly emulate the neuron's behavior in electronic microcircuits. The Hybrid Multi-Scale Facility (HMF) is a mixed-signal platform developed in the scopes of the BrainScaleS Project (BSS) and Human Brain Project (HBP).

In this paper we present efforts in porting HTM networks to the HMF. A framework for simulating HTMs based on spiking neural networks is introduced, as well as concrete network models for the HTM concepts spatial pooling and the temporal memory. We compare the behavior to software implementations in order to verify basic properties of the HTM networks. We discuss the overall applicability of these models on the target platform, the impact of synaptic plasticity, and connection routing considerations.

1.1 Hierarchical Temporal Memory

HTM represents a set of concepts and algorithms for machine intelligence based on neocortical principles [Hawkins et al., 2011]. It is designed to learn spatial as well as temporal patterns and generate predictions from previously seen sequences. It features continuous learning and operates on streaming data. An HTM network consists of one or multiple hierarchically arranged regions. The latter contain neurons organized in columns. The functional principle is captured in two algorithms which are laid out in detail in the original whitepaper [Hawkins et al., 2011]. Some of the mathematical properties of HTM are discussed in [Ahmad and Hawkins, 2015]. The following paragraphs are intended as an introductory overview and introduce the properties relevant to this work.

The *spatial pooler* is designed to map a binary input vector to a set of columns. By recognizing previously seen input data, it increases stability and reduces the system's susceptibility for noise. Its behaviour can be characterized by the following properties:

1. The columnar activity is sparse. Typically, 40 out of 2,048 colums are active, which is approximately a sparsity of 2 %. The number of active columns is constant in each time step and does not depend on the input sparsity.

[*] email: sebastian.billaudelle@kip.uni-heidelberg.de

[†] email: sahmad@numenta.com

2. The spatial pooler activates the k columns which receive the most input. In case of a tie between two columns, the active column is selected randomly, e.g. through structural advantages of certain cells compared to its neighbors.
3. Stimuli with low pairwise overlap counts are mapped to sparse columnar representations with low pairwise overlap counts, while high overlaps are projected onto representations with high overlap. Thus, similar input vectors lead to a similar columnar activation, while disjunct stimuli activate distinct columns.
4. A column must receive a minimum input (e.g. 15 bits) to become active.

The *temporal memory* operates on single cells within columns and further processes the spatial pooler's output. Temporal sequences are learned by the network and can be used for generating predictions and highlighting anomalies. Individual cells receive stimuli from other neurons on their distal dendrites. This lateral input provides a temporal context. By modifying a cell's distal connectivity, temporal sequences can be learned and predicted. The temporal memory's behavior can be summarized by the following:

1. Individual cells receive lateral input on their distal dendrites. In case a certain threshold is crossed, the cells enter a predictive (depolarized) state.
2. When a column becomes active due to proximal input, it activates only those cells that are in predictive state.
3. When a column with no predicted cells becomes active due to proximal input, all cells in the column become active. This phenomenon is referred to as columnar bursting.

1.2 Heidelberg Neuromorphic Computing Platform

The HMF is a hybrid platform consisting of a traditional high-performance cluster and a neuromorphic system. It is developed primarily at the Kirchhoff-Institute for Physics in Heidelberg and the TU Dresden while receiving funding from the BSS and HBP [HBP SP9 partners, 2014]. The platform's core is the wafer-scale integrated High Input Count Analog Neural Network (HICANN) chip as shown in Figure 1. Part of the chip's unique design is its mixed-signal architecture featuring analog neuron circuits and a digital communication infrastructure. Due to the short intrinsic time constants of the hardware neurons, the system operates on an accelerated timescale with a speed-up factor of 10×10^4 compared to biological real-time.

HICANN features 512 neurons or *dendritic membrane circuits*. Each circuit can be stimulated via 226 synapses on two synaptic inputs. As a default, the latter are configured for excitatory and inhibitory stimuli, respectively. However, they can be set up to represent e.g. two excitatory inputs with different synaptic time constants or reversal potentials. By connecting multiple dendritic membranes larger neurons with up to 14×10^3 synapses can be formed.

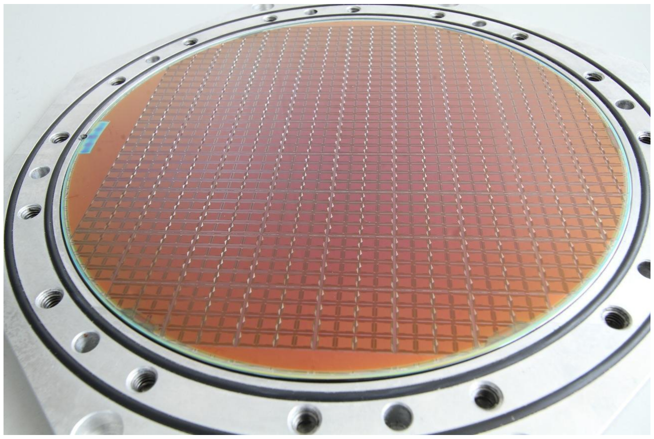

Fig. 1. A wafer containing 384 HICANN chips. The undiced wafer undergoes a custom post-processing step where additional metal layers are applied to establish inter-reticle connectivity and power distribution. (Photo courtesy of the Electronic Vision(s) group, Heidelberg.)

A single wafer contains 384 chips with 200×10^3 neurons and 45×10^6 synapses. Multiple wafers can be connected to form even larger networks. The BSS's infrastructure consists of six wafers and is being extended to 20 wafers for the first HBP milestone.

1.3 Spiking Neuron Model

There exist different techniques of varying complexity for simulating networks of spiking neurons. The reference implementation we use for HTM networks is based on first generation, binary neurons with discrete time steps [Numenta, Inc]. Third generation models, however, incorporate the concept of dynamic time and implement inter-neuron communication based individual spikes.

Starting from the original Hodgkin-Huxley equations [Hodgkin and Huxley, 1952], multiple spiking neuron models were developed that feature different levels of detail and abstraction. The HICANN chip implements Adaptive Exponential Integrate-and-Fire model (AdEx) neurons [Brette and Gerstner, 2005]. At its core, it represents a simple Leaky Integrate-and-Fire (LIF) model but features a detailed spiking behavior as well as spike-triggered and sub-threshold adaption. It was found to correctly predict approximately 96 % of the spike times of a Hodgkin-Huxley-type model neuron and about 90 % of the spikes recorded from a cortical neuron [Jolivet et al., 2008]. On the HMF and thus also in the following simulations, the neurons are paired with conductance-based synapses allowing for a fine-grained control of the synaptic currents and the implementation of e.g. shunting inhibition.

2 SPIKING NETWORK MODELS

Implementing neural network models for a neuromorphic hardware platform or dynamic software simulations requires an abstract network description defining the individual cell populations as well as the model's connectivity. For this work, our primary focus was on developing mechanistic and functional implementations of the software reference models while staying within the topological and parameter restrictions imposed by the hardware platform. A more detailed biophysical approach should begin with simulations of single HTM neurons and their dendritic properties before advancing to more complex systems, e.g. full networks.

In the following sections we describe spatial pooler and temporal memory models that incorporate basic HTM properties. These models are able to reproduce the fundamental behaviour of existing software implementations.

The simulations were set up in Python using the PyNN library [Davison et al., 2008]. Besides supporting a wide range of software simulators, this high-level interface is also supported by the HMF platform [Billaudelle, 2014a]. NEST was used as a simulation backend [Gewaltig and Diesmann, 2007]. To enable multiple synaptic time constants per neuron, a custom implementation of the AdEx model was written.

2.1 Spatial Pooler

At its core the spatial pooler resembles a k-Winners-Take-All (kWTA) network: k out of m columns are chosen to be active in each time step. In fact, kWTA networks have often been mentioned as an approximation for circuits naturally occurring in the neocortex [Felch and Granger, 2008]. Continuous-time and VLSI implementations of such systems have been discussed in the literature [Erlanson and Abu-Mostafa, 1991, Tymoshchuk, 2012, Maass, 2000]. In the implementation below we describe a novel approach to maintaining stable sparsity levels even with a large number of inputs.

The network developed for this purpose is presented in Figure 2. It follows a purely time-based approach and is designed for LIF neurons. It allows for very fast decision processes based on a single input event per source. Each column is represented by a single cell which accumulates feed-forward input from the spike sources. Here, the rise time of the membrane voltage decreases with the number of presynaptic events seen by the cell: cells receiving the most input will fire before the others. An inhibitory pool consisting of a single cell collects the network's activity. Low membrane and high synaptic time constants lead to a reliable summation of events. When a certain number of spikes have been collected – and thus the cell's threshold has been crossed – the pool strongly inhibits all cells of the network suppressing subsequent spike events.

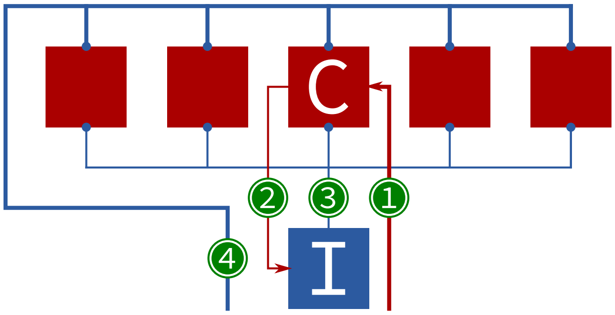

Fig. 2. Timing based implementation of the spatial pooler. Each column is represented by a single cell *C* and receives sparse input from the input vector ①. The columns become active when the number of connected active inputs crosses a threshold. The rise time of the membrane voltage highly depends on the number of coincident inputs: cells with more presynaptic activity will fire before those with less stimuli do. Inhibitory pool *I* accumulates the columnar spikes ② and in doing so acts as a counter. After a certain number of columns have become active, the pool will inhibit and shut down all columns preventing any further activity ③. To stabilize this kWTA model, all columns receive a subsampled feed-forward inhibition ④. This effectively prolongs the decision period for high input activity.

The model is extended by adding subtle feed-forward shunting inhibition. The inhibitory conductance increases with the overall input activity ν_{in}. With the reversal potential set to match the leakage potential, the conductance contributes to the leakage term

$$g_l' = g_l + g_{\text{inh}}(\nu_{\text{in}}).$$

thus increasing the membrane time constant. This effectively slows down the neurons' responses and thus prolongs the decision period of the network. With this inhibition, the resulting system is able to achieve stable sparsity levels with a large number of inputs, at the cost of slightly slower response times.

Tie situations between columns receiving the same number of presynaptic events are resolved by adding slight gaussian jitter to the weights of the excitatory feed-forward connections. This gives some columns structural advantages over other columns resulting in a slightly faster response to the same stimulus. By increasing the standard deviation σ_j of the jitter, the selection criterion can be blurred.

2.2 Temporal Memory

Similar to the spatial pooler, the temporal memory implementation was designed for fast reaction times

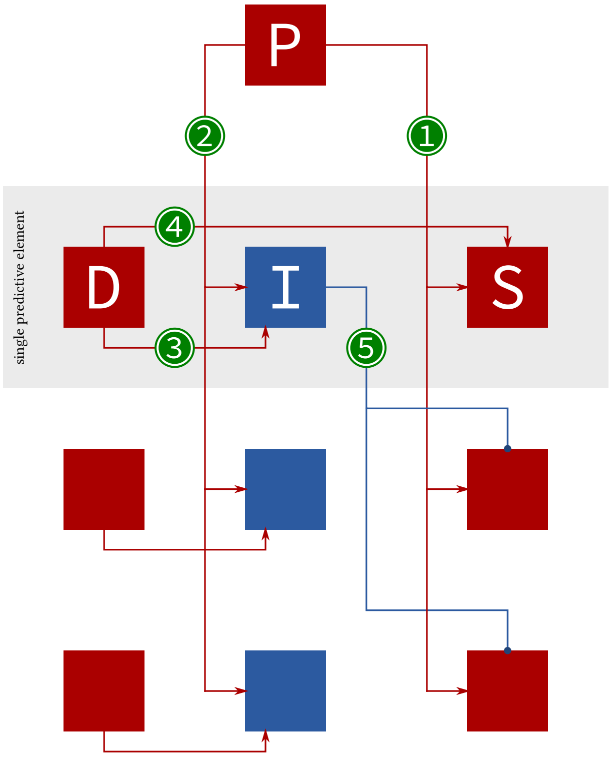

Fig. 3. Implementation of the temporal memory not including plasticity. Every HTM cell within a column is modeled with three individual LIF cells modeling different compartments (distal dendrites *D*, soma *S* and a lateral inhibition segment *I* – which is not biologically inspired). Per column, there exist multiple cell triples as well as one "head" cell *P* which participates in the columnar competition and collects proximal input for the whole column. Activity of this cell is forwarded to the individual soma cells of the column ①. Without a previous prediction, this results in all soma cells firing. However, the distal compartment sums over the input of the previous time step. When a threshold is reached, the inhibitory compartment as well as the soma are depolarized ③ ④. Together with proximal input ②, the inhibitory partition fires and inhibits all other cells in the column ⑤.

and spike-timing based response patterns. A complete network consists of m identical columns with n HTM cells each. Modelling these cells is a challenge in itself. A multicompartmental neuron model would represent the best fit. While a neuromorphic hardware chip implementing such a model is planned and first steps in that direction have already been taken [Millner, 2012], the current system does not provide this feature. Since HTM cells primarily depend on the active properties of a compartment, it can be modelled by a triple of individual LIF cells as shown in Figure 3.

A column collects proximal input using a single cell. In fact, this cell can be part of a spatial pooler network as presented in section 2.1. When the column becomes active, this cell emits a spike and excites both the neurons representing the HTM cells' somae as well as inhibitory cells. The inhibitory projection, however, is not strong enough to activate the target compartment alone. Instead, it only leads to a partial depolarization. The soma neuron, however, reaches the firing threshold for a single presynaptic event. This suffices as a columnar bursting mechanism (i.e. temporal memory property 3): without predictive input, all soma compartments will fire as a response to the proximal stimulus.

Distal input is processed for each cell individually by their dendritic segment compartments. A cell's dendritic segment receives input from other cells' somae. When its firing threshold is crossed, it partly depolarizes the inhibitory helper cell of the same triplet. This synaptic projection is set up with a relatively long synaptic time constant and a reversal potential matching the threshold voltage. This ensures that the predictive state is carried to the next time step and prohibits the cell from becoming active due to distal input alone. On proximal input, the already depolarized helper cell fires before the somatic compartments. The latter are then inhibited instantly, with the exception of the own triplet's soma. As described, this basic predictive mechanism fails when multiple cells are predicted, since the inhibitory compartments laterally inhibit every cell. The solution is to also depolarize the somatic compartments of predicted cells. In summary this mechanism satisfies temporal memory properties 1 and 2.

3 RESULTS

The network models presented in the previous section were simulated in software to investigate their behavior. In the following, respective experiments and their results are shown. Additionally, plasticity rules and topological requirements are discussed in respect of the HMF.

3.1 Network Simulations

The spatial pooler was analyzed for a network spanning 1,000 columns and an input vector of size 10,000. To speed up the simulation, the input connectivity was preprocessed in software by multiplying the stimulus vector to the connectivity matrix.

A first experiment was designed to verify the basic kWTA functionality. A random pattern was presented to the network. The number of active inputs per column – the input overlap score – can be visualized in a histogram as shown in Figure 5. By highlighting the columns activated by that specific stimulus, one can investigate the network's selection criteria. Complying with the requirements for a spatial pooler, only the rightmost bars – representing columns with the highest input counts – are highlighted. Furthermore, the model's capability to resolve ties between columns receiving the same input counts is demonstrated:

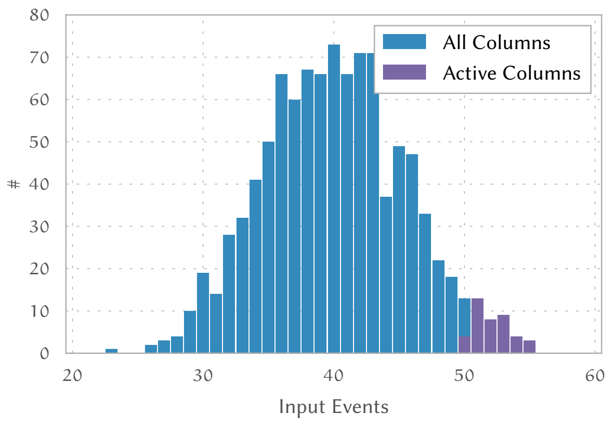

Fig. 5. Histogram showing the distribution of overlap scores individual columns receive. Columns activated by the spatial pooler network are highlighted. This confirms that only competitors with the highest input enter an active state. Furthermore, tie situations between columns with the same overlap score are resolved correctly.

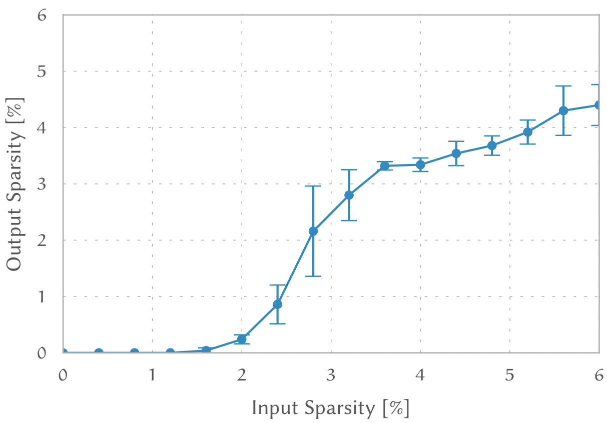

Fig. 6. The relative number of active columns is plotted against the input vector's sparsity. After a certain level of input sparsity is reached, columns start to enter active states. With higher presynaptic activity, columnar competition increases and the output sparsity reaches a plateau. The curve's exact course can be manipulated through the neurons' parameters as can the size of the plateau. Error bars indicate the standard deviation across five trials.

the bar at the decision boundary was not selected as a whole but only a few columns were picked. This verifies spatial pooler property 2.

In a second scenario, input vectors with varying sparsity were fed into the network, as shown in Figure 6. The number of active columns stays approximately constant across a wide range of input sparsity. Additionally the plot shows that columns must receive a minimum amount of input to become active at all. This verifies the underlaying kWTA approach as well as spatial pooler properties 1 and 4.

To verify the general functionality of a spatial pooler, expressed in property 3, a third experiment was set up. Input data sets with a variable overlap were generated starting from an initial random binary vector. For each stimulus, the overlap of the columnar activity with the initial dataset was calculated while sweeping the input's overlap. The resulting relation of input and output overlap scores is shown in Figure 7. Also included are the results of a similar experiment performed with a custom Python implementation of the spatial pooler directly following the original specification [Hawkins et al., 2011]. Multiple simulation runs all yielded results perfectly matching the reference data, thus verifying property 3.

The experiments have shown that the model presented in this section does fulfill the requirements for a spatial pooler and can be considered a solid kWTA implementation. The specific results of course depend on the individual network size and configuration. In this case, the network – most importantly the columnar neurons' time constants – was configured for a relatively short time step of $T = 50\,\text{ms}$. By choosing different parameter sets, the network can be tuned towards different operational scenarios, e.g. further increasing the model's stability.

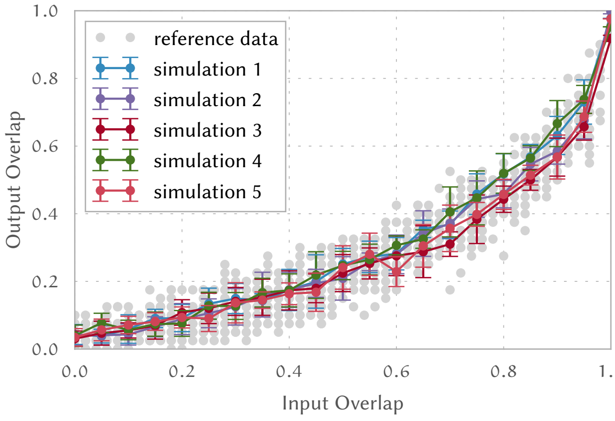

Fig. 7. Output overlap as a dependency of the input vector's overlap score. In each of the five simulation runs, the stimulus' was gradually changed starting from a random vector. As required for a spatial pooler, two similar input stimuli get mapped to similar output patterns, while disjunct input vectors result in low overlap scores. The simulations fully reproduce data from an existing software implementation which is also shown in this figure.

The temporal memory was verified in a first sequence prediction experiment. A reference software implementation was trained with three disjunct sequences of length three.

Consecutive sequences were separated by a random input pattern. The trained network's lateral connectivity was dumped and loaded in a simulation. When presented with the same stimulus, the LIF-based implementation was able to correctly predict sequences, as shown in Figure 8.

3.2 Learning Algorithms

Implementing online learning mechanisms in neuromorphic hardware is a challenge, especially for accelerated systems. Although the HMF features implementations of nearest-neighbor Spike-Timing Dependent Plasticity (STDP) and Short Term Plasticity (STP) [Friedmann, 2013a, Billaudelle, 2014b], more complex update algorithms are hard to implement. Numenta's networks rely on structural plasticity rules which go beyond these mechanisms.

The spatial pooler's stimulus changes significantly for learned input patterns. Verification of its functionality under these conditions is important. In order to follow the HTM specification as closely possible, a supervised update rule was implemented in an outer loop: for each time step, a matrix containing the connections' permanence values is updated according to the activity patterns of the previous time step. This allows us to implement the concepts of structural plasticity presented in the original whitepaper. For the target platform, the learning algorithms could be implemented on the Plasticity Processing Unit (PPU) which is planned for the next version of the HICANN chip [Friedmann, 2013b]. Simulation results of the implementation described above are shown in Figure 9.

Experiments to replace the HTM structural plasticity rules by a classic nearest-neighbor STDP model did not yield the desired results. The HTM learning rules require negative modifications to inactive synapses in segments that contribute to cell activity. In contrast, STDP does not affect inactive synapses.

3.3 Map and Route

Applying abstract network models to the hardware platform requires algorithms for placing the neuron populations and routing the synaptic connections. In a best-case scenario, this processing step results in an isomorphic projection of the network graph to the hardware topology. For networks with extreme connectivity requirements, however, synaptic losses must be expected.

Mapping the simulated networks does not represent a challenge for the routing algorithms. The temporal memory can be projected to a single wafer without synaptic loss. The same still applies with assumed lateral all-to-all connectivity resulting in approximately 2 million synapses. The latter network corresponds to a network with a potential pool of 100 % which would allow the exploration of learning algorithms even without creating and pruning hardware synapses.

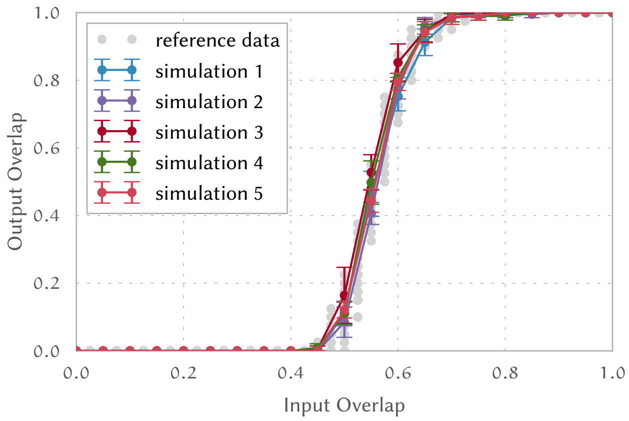

Fig. 9. Dependency of output and input overlap for a trained spatial pooler. Results of five independent simulation runs are shown as well as reference data from a custom software implementation.

On the hardware platform, a tradeoff between the number of afferent connections per cell and the number of neurons must be taken into consideration: while it is possible to connect the dendritic membrane circuits such that a single neuron can listen on roughly 14×10^3 synases, such a network could only consist of approximately 3×10^3 neurons per wafer. With just 226 synapses, just under 200×10^3 neurons can be allocated per wafer.

Scaling up the proof-of-concept models to a size useful for production purposes, however, challenges the hardware topology as well as the projection algorithms.

A minimal useful HTM network spans 1024 columns with 8 cells each. In such a scenario the neurons would receive lateral input on 32 dendritic segments. Allowing approximately 1×10^3 afferent connections per dendritic segment, this network could be realized on approximately 1×10^6 dendritic membrane circuits, or six wafers. The existing system set up for the BSS would suffice for this scenario. Even larger networks could be brought to the HBP's platform.

4 CONCLUSION AND OUTLOOK

Implementing machine intelligence algorithms as spiking neural networks and porting them to a neuromorphic hardware platform presents high demands in terms of precision and scalability.

We have shown in this paper that HTMs can be successfully modeled in dynamic simulations. The basic functionality of spatial pooler and temporal memory networks could be reproduced based on AdEx neurons. In theory, the proof of concept networks can be easily transferred to the HMF, since the high-level software interfaces are designed to be interchangable. Of course, emulating the models on the actual hardware platform will bring up a new set of challenges.

Adapting the HTM's learning rules to the native plasticity features available on the HMF has turned out to be nontrivial. The learning rules could not be replicated with the current implementation of classic STDP. As a freely programmable microprocessor directly embedded into the neuromorphic core, the PPU provides the ability to extend the system's plasticity mechanisms in order to implement the HTM rules. Further investigation is required to map out a complete implementation of the HTM update rules on the PPU.

Analog neuromorphic hardware is susceptible to transistor mismatches due to e.g. dopand fluctuations in the production process [Mihai A. Petrovici, 2014]. A careful calibration of the individual neurons is required to compensate for these variations. Due to the complexity of the problem and the high number of interdependent variables, a perfect calibration is hard to accomplish. Therefore, network models are required to be tolerant regarding certain spatial, and trial-to-trial variations on the computing substrate. Carrying out additional Monte Carlo simulations with slightly randomized parameters is important to investigate the robustness of the presented networks.

Finally, a multicompartmental neuron model is planned for a later version of the neurmorphic platform. Making use of this extended feature set will significantly increase the level of biophysical detail. This will account for the detailed dendritic model used in HTMs and help to stay closer to the whitepaper as well as the reference implementation.

Besides paving the road towards a highly accelerated execution of HTM models, the HMF offers a high level of detail in its neuron implementation. With the multicompartmental extension and a flexible plasticity framework, we anticipate the platform will prove valuable as a tool for further low-level research on HTM theories.

ACKNOWLEDGEMENTS

Special thanks to Jeff Hawkins, Prof. Dr. Karlheinz Meier, Paxon Frady, and the Numenta Team.

REFERENCES

Subutai Ahmad and Jeff Hawkins. Properties of Sparse Distributed Representations and their Application to Hierarchical Temporal Memory. March 2015. URL http://arxiv.org/abs/1503.07469.

Sebastian Billaudelle. PyHMF – eine PyNN-kompatible Schnittstelle für das HMF-System, 2014a.

Sebastian Billaudelle. Characterisation and calibration of short term plasticity on a neuromorphic hardware chip. Bachelor's thesis, Universität Heidelberg, 2014b.

Romain Brette and Wulfram Gerstner. Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *Journal of neurophysiology*, 94(5): 3637–3642, 2005.

Andrew P Davison, Daniel Brüderle, Jochen Eppler, Jens Kremkow, Eilif Muller, Dejan Pecevski, Laurent Perrinet, and Pierre Yger. Pynn: a common interface for neuronal network simulators. *Frontiers in neuroinformatics*, 2, 2008.

Ruth Erlanson and Yaser S Abu-Mostafa. Analog neural networks as decoders. In *Advances in neural information processing systems*, pages 585–588, 1991.

Andrew C Felch and Richard H Granger. The hypergeometric connectivity hypothesis: Divergent performance of brain circuits with different synaptic connectivity distributions. *Brain research*, 1202:3–13, 2008.

Simon Friedmann. *A new approach to learning in neuromorphic hardware*. PhD thesis, Heidelberg, Univ., Diss., 2013, 2013a.

Simon Friedmann. *A new approach to learning in neuromorphic hardware*. PhD thesis, Heidelberg, Univ., Diss., 2013, 2013b.

SB Furber, F Galluppi, S Temple, and LA Plana. The spinnaker project. *0018-9219*, (99):1–14, 2014.

Marc-Oliver Gewaltig and Markus Diesmann. Nest (neural simulation tool). *Scholarpedia*, 2(4):1430, 2007.

Jeff Hawkins, Subutai Ahmad, and Donna Dubinsky. Cortical Learning Algorithm and Hierarchical Temporal Memory, 2011. URL http://numenta.org/resources/HTM_CorticalLearningAlgorithms.pdf.

HBP SP9 partners. *Neuromorphic Platform Specification*. Human Brain Project, March 2014.

A.L. Hodgkin and A.F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology*, 117:500–544, 1952.

Renaud Jolivet, Felix Schürmann, Thomas K Berger, Richard Naud, Wulfram Gerstner, and Arnd Roth. The quantitative single-neuron modeling competition. *Biological cybernetics*, 99(4-5):417–426, 2008.

Wolfgang Maass. Neural computation with winner-take-all as the only nonlinear operation. Citeseer, 2000.

Paul Müller Oliver Breitwieser Mikael Lundqvist Lyle Muller Matthias Ehrlich Alain Destexhe Anders Lansner RenÃľ Schüffny Johannes Schemmel Karlheinz Meier Mihai A. Petrovici, Bernhard Vogginger. Characterization and compensation of network-level anomalies in mixed-signal neuromorphic modeling platforms. *PLOS ONE*, October 2014. doi: dx.doi.org/10.1371/journal.pone.0108590. URL http://www.plosone.org/article/info%3Adoi%2F10.1371%2Fjournal.pone.0108590.

Sebastian Millner. *Development of a Multi-Compartment Neuron Model Emulation*. PhD thesis, Heidelberg, Univ., Diss., 2013, 2012.

Numenta, Inc. Numenta Platform for Intelligent Computing (NuPIC). URL http://numenta.org/.

PavloV. Tymoshchuk. A continuous-time model of analogue k-winners-take-all neural circuit. In Chrisina Jayne, Shigang Yue, and Lazaros Iliadis, editors, *Engineering Applications of Neural Networks*, volume 311 of *Communications in Computer and Information Science*, pages 94–103. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-32908-1.

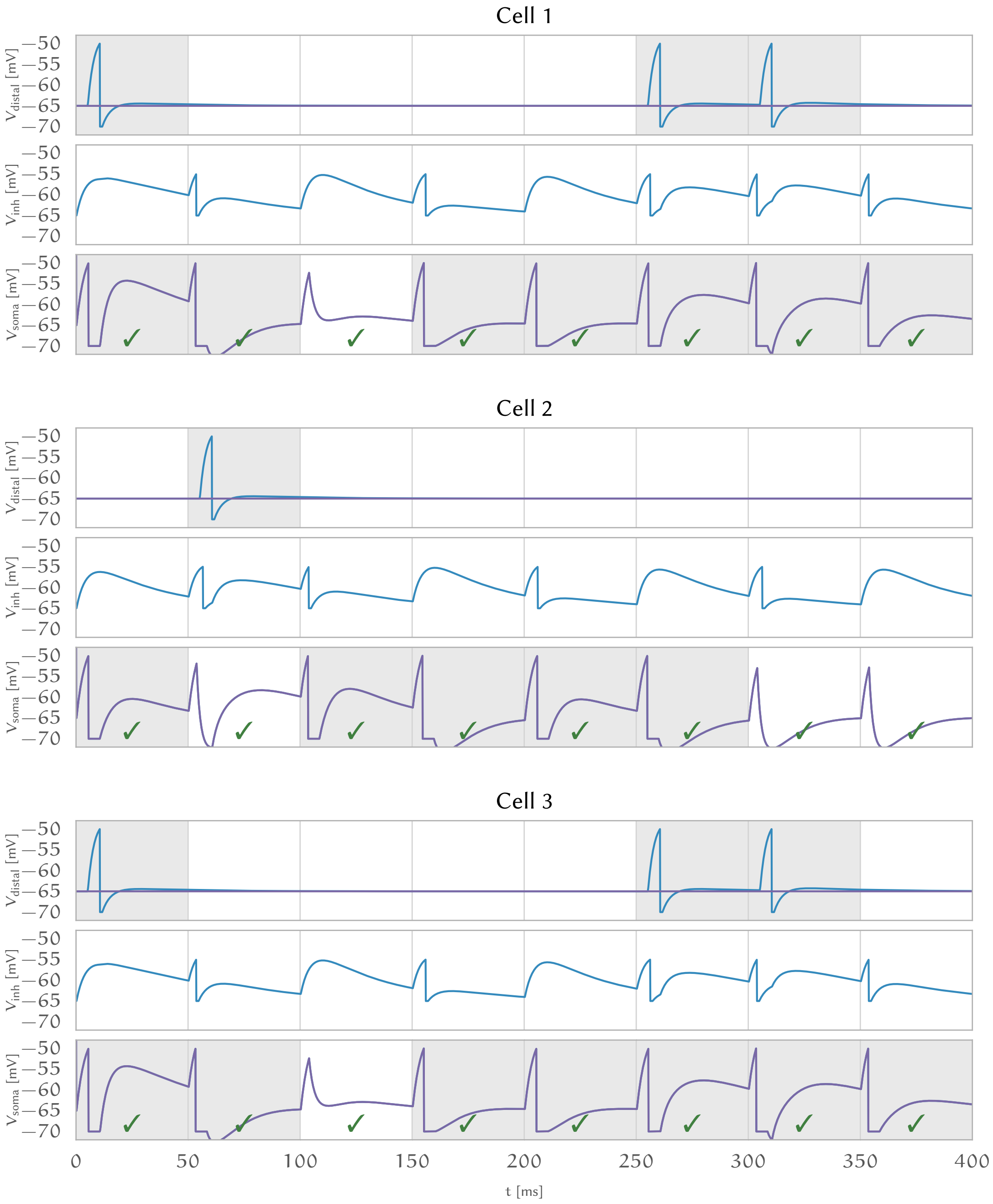

Fig. 4. Neuron traces for a temporal memory column containing three HTM cells. Each of these cells is represented by a somatic compartment, an inhibitory helper cells and two dendritic segments. The column is activated by proximal input in every time step and receives random distal stimulus predicting none, one or more cells per step. As indicated by the automatic classification algorithm, the column exhibits a correct response pattern to these predictions.

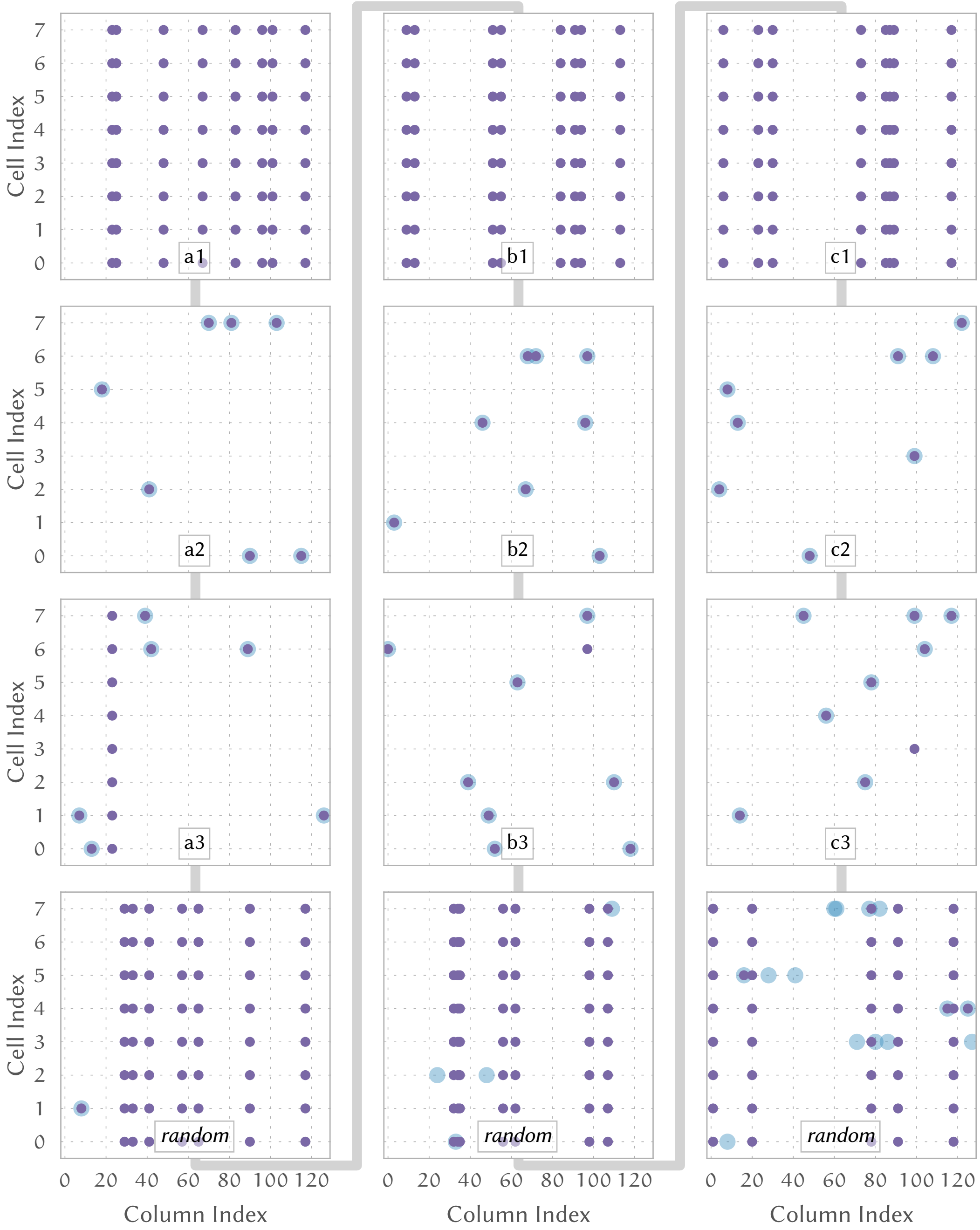

Fig. 8. A LIF neuron based temporal memory implementation correctly predicting different patterns. Predicted cells are marked blue, active cells in purple. The network spans 128 columns with each of their eight HTM cells collecting distal stimuli via two dendritic segments. Connectivity for the distal inputs was configured externally. The model was presented three disjunct sequences of size three. The individual patterns were separated by a random input Sparse Distributed Representation (SDR).