Unsupervised Learning of Digit Recognition Using Spike-Timing-Dependent Plasticity

Peter U. Diehl, *Student Member, IEEE,* Matthew Cook, *Member, IEEE,*

***Abstract*—The recent development of power-efficient neuromorphic hardware offers great opportunities for applications where power consumption is a main concern, ranging from mobile platforms to server farms. However, it remains a challenging task to design spiking neural networks (SNN) to do pattern recognition on such hardware. We present a SNN for digit recognition which relies on mechanisms commonly used on neuromorphic hardware, i.e. exponential synapses with spike-timing-dependent plasticity, lateral inhibition, and an adaptive threshold. Unlike most other approaches, we do not present any class labels to the network; the network uses unsupervised learning. The performance of our network scales well with the number of neurons used. Intuitively, the used algorithm is comparable to k-means and competitive learning algorithms such as vector quantization and self-organizing maps, each neuron learns a representation of a part of the input space, similar to a centroid in k-means. Our architecture achieves 95% accuracy on the MNIST benchmark, which outperforms other unsupervised learning methods for SNNs. The fact that we used no domain-specific knowledge points toward a more general applicability of the network design.**

***Index Terms*—Spiking Neural Network, STDP, Classification, Digit Recognition.**

I. Introduction

THE mammalian neocortex offers an unmatched pattern recognition performance given a power consumption of only 10-20 watts [1]. Since energy consumption is a major cost factor for companies with lots of data [2], there is a strong motivation to decrease power consumption of chips. Current implementations of spiking neural networks (SNN) on neuromorphic hardware [3, 4, 5, 6] use only a few nJ or even pJ for transmitting a spike [7, 8, 9] (for some setups as little energy as 0.02 pJ per spike [10]) and consume only few pW of power per synapse [11]; some of those neuromorphic systems also offer on-chip learning mechanisms [4, 12, 13]. However, so far it remains a challenging task to achieve good performance on classical machine learning benchmarks like MNIST [14] using SNNs. There are two main approaches of designing (and training) SNNs for machine learning applications (1) learning the weights of the SNN using different variants of spike-timing dependent plasticity (STDP), which we will call "spike-based learning" [15, 16, 17, 18, 19], and (2) training a rate-based neural network with backpropagation [20] (or other rate-based learning algorithms) and using those weights for an SNN, which we will call "rate-based learning" [8, 21, 22, 23].

Here we present an SNN that is trained in an unsupervised fashion using STDP, i.e. the weights of the network learn the structure of the input examples without using labels. No preprocessing of the MNIST dataset is used (besides the necessary conversion of the intensity images to spike-trains). The performance of this approach scales well with the number of neurons in the network, and achieves a performance of 95% using 6400 learning neurons.

In the next section we explain the architecture including the neuron and synapse models, and the training and evaluation process. Section III contains the simulation results and in section IV we compare our results to those of other architectures as well as giving an intuition into how our network works.

Peter U. Diehl and Matthew Cook are with the Institute of Neuroinformatics, ETH Zurich and University Zurich e-mail: {diehlp, cook}@ini.ethz.ch.

II. Methods

To simulate our SNNs we used Python and the BRIAN simulator [24][1]. Here we describe the dynamics of a single neuron and a single synapse, then the network architecture and the used mechanisms, and finally we explain the MNIST training and classification procedure.

A. Neuron and Synapse Model

To model neuron dynamics, we chose the leaky integrate-and-fire model. The membrane voltage V is described by

$$\tau \frac{dV}{dt} = (E_{rest} - V) + g_e(E_{exc} - V) + g_i(E_{inh} - V) \quad (1)$$

where E_{rest} is the resting membrane potential, E_{exc} and E_{inh} the equilibrium potentials of excitatory and inhibitory synapses, and g_e and g_i the conductances of excitatory and inhibitory synapses, respectively. As observed in biology, we use a time constant τ, which is longer for excitatory neurons than for inhibitory neurons. When the neuron's membrane potential crosses its membrane threshold v_{thres}, the neuron fires and its membrane potential is reset to v_{reset}. Within the next few milliseconds after the reset, the neuron is in its refractory period and cannot spike again.

Synapses are modelled by conductance changes, i.e. synapses increase their conductance instantaneously when a presynaptic spike arrives at a synapse, otherwise the conductance is decaying exponentially. If the presynaptic neuron is excitatory, the dynamics of the conductance g_e are

$$\tau_{g_e} \frac{dg_e}{dt} = -g_e \quad (2)$$

where τ_{g_e} is the time constant of an excitatory postsynaptic potential. Similarly, if the presynaptic neuron is inhibitory, a

[1]The code (including used parameters) is available under http://www.ini.uzh.ch/admin/extras/doc_get.php?id=52538 (only the Brian package has to be installed additionally to the standard python packages to run the code).

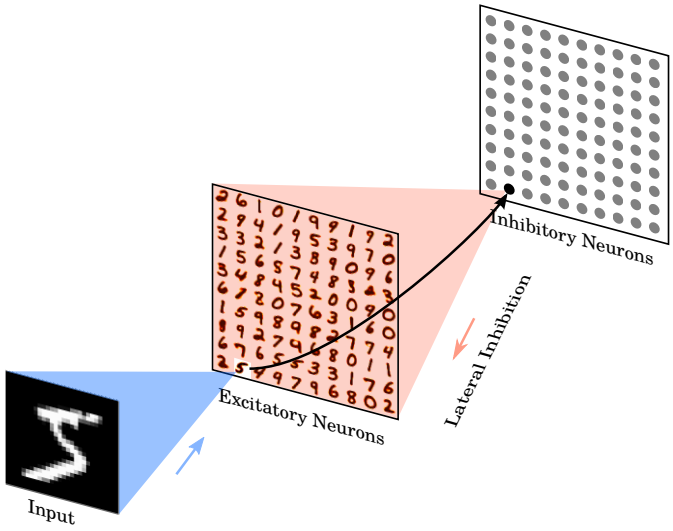

Fig. 1. Network architecture. The intensity values of the 28 x 28 pixel MNIST image are converted to Poisson-spike with firing rates proportional to the intensity of the corresponding pixel. Those Poisson-spike trains are fed as input to excitatory neurons in an all-to-all fashion. The blue shaded area shows the input connections to one specific excitatory example neuron. Excitatory neurons are connected to inhibitory neurons via one-to-one connections, as shown for the example neuron. Each inhibitory neuron is connected to all excitatory neurons, except for the one it receives a connection from. The red shaded area denotes all connections from one inhibitory neuron to the excitatory neurons. Class labels are not presented to the network, so the learning is unsupervised. Excitatory neurons are assigned to classes after training, based on their highest average response to a digit class over the training set. No additional parameters are used to predict the class, specifically no linear classifier or similar methods are on top of the SNN.

conductance g_i is updated using the same equation but with the time constant of the inhibitory postsynaptic potential τ_{g_i}.

We use biologically plausible ranges for almost all of the parameters in our simulations, including time constants of membranes, synapses and learning windows [25]; the exception is the time constant of the membrane voltage of excitatory neurons. By increasing the time constant of the excitatory neuron membrane potential to 100 ms (from the 10 to 20 ms that are typically observed for biological neurons) we were able to increase the classification accuracy.

B. Network Architecture

The network consists of two layers, see **figure 1**. The first layer is the input layer, containing 28×28 neurons (one neuron per image pixel), and the second layer is the processing layer, containing a variable number of excitatory neurons and as many inhibitory neurons. Each input is a Poisson spike-train, which is fed to the excitatory neurons of the second layer. The rates of each neuron are proportional to the intensity of the corresponding pixel in the example image, see subsection input coding.

The excitatory neurons of the second layer are connected in a one-to-one fashion to inhibitory neurons, i.e. each spike in an excitatory neuron will trigger a spike in its corresponding inhibitory neuron. Each of the inhibitory neurons is connected to all excitatory ones, except for the one from which it receives a connection. This connectivity provides lateral inhibition and leads to competition among excitatory neurons.

C. Learning

All synapses from input neurons to excitatory neurons are learned using STDP. To improve simulation speed, the weight dynamics are computed using synaptic traces [26]. This means that, besides the synaptic weight, each synapse keeps track of another value, namely the presynaptic trace x_{pre}, which models the recent presynaptic spike history. x_{pre} decays exponentially

$$\tau_{x_{pre}} \frac{dx_{pre}}{dt} = -x_{pre} \tag{3}$$

where $\tau_{x_{pre}}$ is the time constant of the decay. Every time a presynaptic spike arrives at the synapse, the trace is increased by 1. When a postsynaptic spike arrives at the synapse the weight change Δw is calculated based on the presynaptic trace

$$\Delta w = \eta (x_{pre} - x_{tar})(w_{max} - w)^{\mu} \tag{4}$$

where η is the learning-rate, w_{max} is the maximum weight, and μ determines the dependence of the update on the previous weight. x_{tar} is the target value of the presynaptic trace at the moment of a postsynaptic spike. The higher the target value, the lower the synaptic weight will be. The target value ensures that presynaptic neurons that rarely lead to firing of the postsynaptic neuron will become more and more disconnected and is especially useful if the presynaptic neuron is only rarely active. A similar effect can be achieved by adding some noise to the input and adding a weight decrease mechanism to the learning rule (like in classical STDP [27]) to disconnect irrelevant inputs. However, in our simulations it comes at the cost of an increased simulation time. The learning rule is similar to the one used in [18] but here we use an exponential learning window instead of a rectangular one, which is more biologically plausible [28] and more similar to the learning mechanisms available on some neuromorphic systems [4, 12, 13].

D. Homoeostasis

The inhomogeneity of the input leads to different firing rates of the excitatory neurons, and lateral inhibition further increases this difference. However, it is desirable that all neurons have approximately equal firing rates to prevent single neurons from dominating the response pattern and to ensure that the receptive fields of the neurons differentiate. To achieve this, we employ an adaptive membrane threshold similar to the mechanism used in [18]. Specifically, each excitatory neuron's membrane threshold is not only determined by v_{thresh} but by the sum $v_{thresh}+\theta$, where θ is increased every time the neuron fires and is exponentially decaying according to

$$\tau_{\theta} \frac{\theta}{dt} = -\theta \tag{5}$$

with τ_{θ} being the time constant of the decay. Therefore, the more a neuron fires, the higher will be its membrane threshold and in turn the neuron requires more input to spike in the near future. Using this mechanism, the firing rate of the neurons is limited because the conductance-based synapse model limits the maximum membrane potential to the excitatory reversal potential E_{exc}, i.e. once the neuron membrane threshold is

close to E_{exc} (or higher) it will fire less often (or even stop firing completely) until θ decreases sufficiently.

E. Input encoding

The input to the network is based on the MNIST dataset which contains 60,000 training examples and 10,000 test examples of 28×28 pixel images of the digits 0 to 9 [14]. The input is presented to the network for 350 ms in the form of Poisson-distributed spike trains, with firing rates proportional to the intensity of the pixels of the MNIST images. Specifically, the maximum pixel intensity of 255 is divided by 4, resulting in input firing rates between 0 and 63.75 Hz. Additionally, if the excitatory neurons in the second layer fire less than five spikes within 350 ms, the maximum input firing rate is increased by 32 Hz and the example presented again for 350 ms. This process is repeated until at least five spikes have been fired during the entire time the particular example was presented.

F. Training and Classification

To train the network, we present the entire MNIST training set (60,000 examples) ten times to the network. Before presenting a new image, there is a 150 ms phase without any input to allow all variables of all neurons decay to their resting values (except for the adaptive threshold). After training is done, we set the learning rate to zero, fix each neuron's spiking threshold, and assign a class to each neuron, based on its highest response to the ten classes of digits over one presentation of the training set. This is the only step where labels are used, i.e. for the training of the synaptic weights we do not use labels.

The response of the class-assigned neurons is then used to measure the classification accuracy of the network on the MNIST test set (10,000 examples). The predicted digit is determined by averaging the responses of each neuron per class and then choosing the class with the highest average firing rate.

III. Results

We trained and tested a network with 100 excitatory neurons by presenting 40,000 examples of the MNIST training set. The resulting rearranged input to excitatory neuron weights are shown in **Figure 2 A**. For each neuron, the 784-dimensional input vector is rearranged into a 28×28 matrix to visualize that the neurons learn prototypical inputs.

Additionally to the 100 neuron network, we trained and tested three other networks with 400, 1600 and 6400 excitatory neurons by presenting 3, 7 and 15 times the entire MNIST training set; the four networks achieved an average classification accuracy of 82.9%, 87.0%, 91.9% and 95.0%, respectively. The accuracies are averaged over ten presentations of the 10,000 examples of the MNIST test set, see **figure 2 B**. Since the intensity images of the MNIST test set are converted into Poisson-distributed spike trains, the accuracy can differ for different spike timings. However, the standard deviation of the performance over ten presentations of the entire test set (using the same trained network) is small ($\approx 0.1\%$), as shown in the error bars in **figure 2 B**.

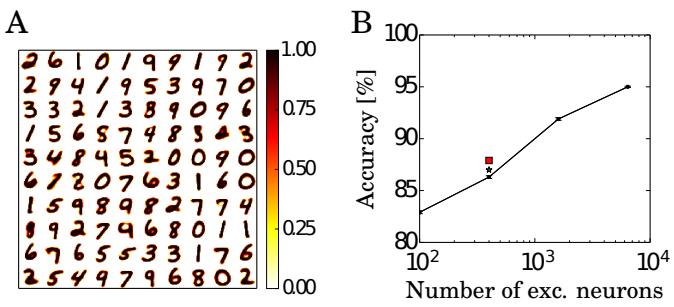

Fig. 2. Training results. A) Rearranged weights (from 784 to 28×28) of the connections from input to excitatory neurons of for a network with 100 excitatory neurons in a 10 by 10 grid. B) Performance as a function of the number of excitatory neurons. Each dot shows the performance for a certain network size as an average over ten presentations of the entire MNIST test set, during which no learning occurs. Error bars denote the standard deviation between ten presentations of the test set. Performances of two additional learning rules are denoted by the red square (exponential weight dependence STDP) and the green star (pre-and-post STDP).

For all simulations, the same neuron, synapse, and STDP parameters were used (except for the parameters of the adaptive threshold and the inhibition strength). The time constant of the adaptive threshold is chosen such that approximately 5 spikes are fired per presented example.

The chosen STDP rule has the advantage that only one addition is needed for each spike received by an excitatory neuron (which is necessary for updating the presynaptic trace). Since the firing rate of the postsynaptic neurons is quite low, a more complex STDP update for postsynaptic firing doesn't require many computational resources.

In order to compare the robustness of the chosen architecture to the exact form of the learning rule, we tested two other STDP learning rules. The first one uses an exponential weight dependence as in [18] to compute the weight change

$$\Delta w = \eta_{post}(x_{pre} - x_{tar}) \exp(-\beta w) \tag{6}$$

where β determines the strength of the weight dependence.

The second rule uses not only a presynaptic trace but also a postsynaptic trace, which works in the same way as the presynaptic trace but its increase is triggered by a postsynaptic spike. Additionally, for this learning rule weight changes occur for pre- and postsynaptic spikes. The weight change Δw for a presynaptic spike is

$$\Delta w = -\eta_{pre} x_{post} w^{\mu} \tag{7}$$

where η_{pre} is the learning-rate for a presynaptic spike and μ determines the weight dependence. The weight change for a postsynaptic spike is

$$\Delta w = \eta_{post}(x_{pre} - x_{tar})(w_{max} - w)^{\mu} \tag{8}$$

where η_{post} is the learning rate, w_{max} is the maximum weight, and x_{tar} is the target average value of the presynaptic trace at the moment of a postsynaptic spike. The second learning rule is computationally more expensive to simulate using software simulations since for every presynaptic event the weight

TABLE I
CLASSIFICATION ACCURACY OF SPIKING NEURAL NETWORKS ON MNIST TEST SET

Architecture	Preprocessing	Training-type	(Un-)supervised	Learning-rule	# Neurons/ # Plastic synapses	Performance
Dendritic neurons [21]	Thresholding	Rate-based	Supervised	Morphology learning	794/50,100	90.3%
Spiking RBM [8]	None	Rate-based	Supervised	Contrastive divergence, linear classifier	740/123,904	89.0%
Spiking RBM [22]	Enhanced training set to 120,000 examples	Rate-based	Supervised	Contrastive divergence	1,794/647,000	**94.1%**
Spiking RBM [23]	As in [22], implemented on FPGA	Rate-based	Supervised	Contrastive divergence	1,794/647,000	92.0%
Spiking RBM [29]	Thresholding	Rate-based	Supervised	Contrastive divergence	1,324/412,000	92.6%
Spiking RBM [29]	Thresholding	Spike-based	Supervised	Contrastive divergence	1,324/412,000	91.9%
Spiking convolutional neural network [19]	Scaling, orientation detection, thresholding	Spike-based	Supervised	Tempotron-rule	?/?[2]	91.3%
Two layer feedward network [16]	Edge-detection	Spike-based	Supervised	STDP with calcium variable	934/117,600	**96.5%**[3]
Multi-layer hierarchical network [15]	Orientation-detection	Spike-based	Supervised	STDP with calcium variable	71,026/ 133,000,000	91.6%
Two layer network [18]	None	Spike-based	Unsupervised	Rectangular STDP	1184/313,600	93.5%
Two layer network (this paper)	None	Spike-based	Unsupervised	Exponential STDP	7184/5,017,600	**95.0%**

change has to be calculated for every single postsynaptic neuron. Therefore we train a network with only 400 excitatory neurons; the exponential weight dependence network achieves 87.9% accuracy (red square in **figure 2 B**) and the pre-and-post STDP network achieves 87.0% accuracy (green star).

IV. DISCUSSION

1) Comparison: The presented network achieves the best classification performance of SNNs with unsupervised learning on the MNIST data set. One common way the weights are trained is using a rate-based algorithm and then transfer those trained weights into a SNN (referred to as "Rate-based" training in table **table I**). The standard training procedure used for such rate-based training is based on Restricted Boltzman Machines (RBM). The other approach is to train the weights using spike-based training procedures, typically relying on STDP. A comparison of spiking neural networks used for MNIST classification is shown in **table I**. Note that [29] and [21] tested their networks only on 1000 and 5000 digits, respectively. Also, in [16] it was not possible to determine whether the examples from the training set were also part of the test set. This question is important because in machine learning the goal is to be able to generalize from seen examples and not to memorize previously seen ones (although this is still interesting for other tasks). A likely reason for the good performance of our network is its strong generalization ability, which is indicated by the fact that the training error is essentially the same as the testing error (less than 1% between both of them). This hints that the network is still underfitting the data and that a further increase of the network size should also increase its classification performance.

A network architecture style similar to ours is presented in [18, 30, 31], using the learning rule presented in [32]. The main differences between their network and ours is that we use more biologically plausible mechanisms, most of which are closer to many neuromorphic hardware implementations. Those differences include that we use an exponential conductance-based synapse instead of a current pulse, that we use an exponential shape of the STDP time-window instead of a rectangular one, and that in our network inhibition is applied using an inhibitory exponential conductance instead of clamping the postsynaptic membrane voltage to a certain value for some inhibition time t_{inh}. Especially the latter modification makes learning more difficult, i.e. using 400 excitatory neurons for each one of the networks, the one in [18] outperforms the one presented here by about 5%. Nevertheless, since the performance of our network scales well with the number of neurons it is possible to achieve an even higher accuracy. The likely reason for the increased difficulty in learning is that in [18] learning works best when t_{inh} is equal to the refractory period of the neuron, such that after one neuron fires, all neurons have the same chance of firing after the refractory period. It is not easily possible to achieve the same effect using inhibitory exponential conductances, since it would be necessary to simultaneously fine tune the time constant of the inhibitory conductance, the refractory period, and the strength of the connection from inhibitory to excitatory neurons. Even if such a fine tuning is achieved, neurons that are not in their refractory period can still integrate incoming excitatory potentials and thus increase their chance of firing. Since many neuromorphic systems exhibit exponential dynamics for postsynaptic potentials and for learning that are similar to the ones in our network, it should possible with only minor changes to port our network to existing neuromorphic systems.

2) Inhibition: In the current implementation we used as many inhibitory neurons as excitatory neurons, such that every spike of an excitatory neuron (indirectly) leads to an inhibition of all other excitatory neurons. This can be changed by substituting the big pool of inhibitory neurons by a smaller one and using a one-to-many connectivity from excitatory to inhibitory neurons. This would result in a network where a spike of an excitatory neuron leads to inhomogeneous inhibitory inputs to other excitatory neurons and thus might favour the activation

[2]Number of neurons or plastic synapses cannot be inferred from the paper.
[3]From the paper it is not clear whether the training set was separate from the test set.

of some neurons over others. Nonetheless, in a big network those effects should be averaged out and the performance of the network should stay approximately the same.

3) Spike-Based Learning: Given that power consumption is most likely going to be one of the main reasons to use spike-based machine learning architectures, it maybe preferable to use spike-based learning instead of rate-based learning since the learning procedure itself has a high power consumption. Specifically, spike-based learning is important when the learning procedure takes up a significant part of time the network will be used. Another application where spike-based learning is needed is for systems which have to adapt dynamically to their environment, i.e. when it's not enough to train the system once and run it with the pre-trained weights. Possible examples include speech recognition systems that are pre-trained but adaptable on the user's accent, or vision processors that have to be tuned to the specific vision sensor. Such an adaptive vision processing system is especially interesting in conjunction with a spiking vision sensor like the ATIS or the DVS [33, 34, 35] as it provides an end-to-end low-power spike-based vision system. If our network were implemented on a low-power neuromorphic chip [3, 4, 5, 6], it could be run on a very low power budget; for example, using IBM's TrueNorth chip [6] which consumes about 72mW for 1 million neurons, the network would consume less than 1mW.

4) Competitive Learning: Intuitively, the function of the network is similar to competitive learning procedures [36] like self-organizing maps [37] or neural-gas [38], which share aspects with k-means. The main idea is that each neuron learns and represents one prototypical input. Every time an input is presented, the network determines the prototypes that are most similar to the input. Those winner prototypes are then used to predict the class of the input and their weights are adapted such that they become more similar to the current input. In our network this means that every time a neuron spikes, because an example is similar enough to its receptive field, it will make its receptive field more similar to the example. The lateral inhibition prevents the prototypes from becoming too similar to each other (which means that they spread out in the input space), since only a few different neurons will be able to respond to each example and in turn only a few neurons can adapt their receptive fields towards it. Homoeostasis can be thought of as a tool to keep an approximately constant number of examples within range of the prototype. This analogy to k-means-like learning algorithms is especially interesting since recently such approaches have been shown to be very successful in complex machine learning tasks [39].

The main advantages of our system are its scalability, making use of the large number of neurons in neuromorphic systems, and its flexibility in spike-based unsupervised learning rules, allowing training of the network without labels and using only few labels to assign neurons to classes while achieving state-of-the-art performance.

ACKNOWLEDGMENTS

We would like to thank Damien Querlioz and Olivier Bichler for helpful comments and Miura Hawkins for careful proofreading. This work was supported by the SNF Grant 200021-143337 "Adaptive Relational Networks".

REFERENCES

[1] F. Javed, Q. He, L. E. Davidson, J. C. Thornton, J. Albu, L. Boxt, N. Krasnow, M. Elia, P. Kang, S. Heshka *et al.*, "Brain and high metabolic rate organ mass: contributions to resting energy expenditure beyond fat-free mass," *The American journal of clinical nutrition*, vol. 91, no. 4, pp. 907–912, 2010.

[2] L. A. Barroso, "The price of performance," *Queue*, vol. 3, no. 7, pp. 48–53, 2005.

[3] B. V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. Merolla, and K. Boahen, "Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 699–716, 2014.

[4] G. Indiveri, E. Chicca, and R. Douglas, "A vlsi array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity," *Neural Networks, IEEE Transactions on*, vol. 17, no. 1, pp. 211–221, 2006.

[5] M. Khan, D. Lester, L. Plana, A. Rast, X. Jin, E. Painkras, and S. Furber, "Spinnaker: mapping neural networks onto a massively-parallel chip multiprocessor," in *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*. IEEE, 2008, pp. 2849–2856.

[6] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[7] C. Mayr, J. Partzsch, M. Noack, S. Hänzsche, S. Scholze, S. Höppner, G. Ellguth, and R. Schüffny, "A biological-realtime neuromorphic system in 28 nm cmos using low-leakage switched capacitor circuits," in *Transactions in Biomedical Circuits and Systems (TBCAS)*. IEEE, 2014.

[8] P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar, and D. S. Modha, "A digital neurosynaptic core using embedded crossbar memory with 45pj per spike in 45nm," in *Custom Integrated Circuits Conference (CICC), 2011 IEEE*. IEEE, 2011, pp. 1–4.

[9] J. Park, S. Ha, T. Yu, E. Neftci, and G. Cauwenberghs, "A 65k-neuron 73-mevents/s 22-pj/event asynchronous micro-pipelined integrate-and-fire array transceiver," in *Biomedical Circuits and Systems Conference (BioCAS)*. IEEE, 2014.

[10] M. R. Azghadi, N. Iannella, S. Al-Sarawi, and D. Abbott, "Tunable low energy, compact and high performance neuromorphic circuit for spike-based synaptic plasticity," *PloS one*, vol. 9, no. 2, p. e88326, 2014.

[11] M. Rahimi Azghadi, N. Iannella, S. F. Al-Sarawi, G. Indiveri, and D. Abbott, "Spike-based synaptic plasticity in silicon: Design, implementation, application, and chal-

lenges," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 717–737, 2014.

[12] P. U. Diehl and M. Cook, "Efficient implementation of stdp rules on spinnaker neuromorphic hardware," in *Neural Networks (IJCNN), 2014 International Joint Conference on*. IEEE, 2014, pp. 4288–4295.

[13] F. Galluppi, X. Lagorce, E. Stromatias, M. Pfeiffer, L. A. Plana, S. B. Furber, and R. B. Benosman, "A framework for plasticity implementation on the spinnaker neural architecture," *Frontiers in Neuroscience*, vol. 429, no. 8, 2014.

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[15] M. Beyeler, N. D. Dutt, and J. L. Krichmar, "Categorization and decision-making in a neurobiologically plausible spiking network using a stdp-like learning rule," *Neural Networks*, vol. 48, pp. 109–124, 2013.

[16] J. M. Brader, W. Senn, and S. Fusi, "Learning real-world stimuli in a neural network with spike-driven synaptic dynamics," *Neural computation*, vol. 19, no. 11, pp. 2881–2912, 2007.

[17] S. Habenschuss, J. Bill, and B. Nessler, "Homeostatic plasticity in bayesian spiking networks as expectation maximization with posterior constraints," in *Advances in Neural Information Processing Systems*, 2012, pp. 773–781.

[18] D. Querlioz, O. Bichler, P. Dollfus, and C. Gamrat, "Immunity to device variations in a spiking neural network with memristive nanodevices," *Nanotechnology, IEEE Transactions on*, vol. 12, no. 3, pp. 288–295, 2013.

[19] B. Zhao, R. Ding, S. Chen, B. Linares-Barranco, and H. Tang, "Feedforward categorization on aer motion events using cortex-like features in a spiking neural network," 2014.

[20] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," DTIC Document, Tech. Rep., 1985.

[21] S. Hussain, S.-C. Liu, and A. Basu, "Improved margin multi-class classification using dendritic neurons with morphological learning," in *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on*. IEEE, 2014, pp. 2640–2643.

[22] P. O'Connor, D. Neil, S.-C. Liu, T. Delbruck, and M. Pfeiffer, "Real-time classification and sensor fusion with a spiking deep belief network," *Frontiers in Neuroscience*, vol. 7, 2013.

[23] D. Neil and S.-C. Liu, "Minitaur, an event-driven fpga-based spiking network accelerator," 2014.

[24] D. Goodman and R. Brette, "Brian: A simulator for spiking neural networks in python," *Frontiers in neuroinformatics*, vol. 2, p. 5, 2008.

[25] F. Jug, "On competition and learning in cortical structures," Ph.D. dissertation, Diss., Eidgenössische Technische Hochschule ETH Zürich, Nr. 20194, 2012, 2012.

[26] A. Morrison, A. Aertsen, and M. Diesmann, "Spike-timing-dependent plasticity in balanced random networks," *Neural Computation*, vol. 19, no. 6, pp. 1437–1467, 2007.

[27] G. Bi and M. Poo, "Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type," *The Journal of Neuroscience*, vol. 18, no. 24, pp. 10 464–10 472, 1998.

[28] L. Abbott and S. Song, "Temporally asymmetric hebbian learning, spike timing and neuronal response variability," *Advances in neural information processing systems*, vol. 11, pp. 69–75, 1999.

[29] E. Neftci, S. Das, B. Pedroni, K. Kreutz-Delgado, and G. Cauwenberghs, "Event-driven contrastive divergence for spiking neuromorphic systems," *Frontiers in Neuroscience*, vol. 7, no. 272, 2013.

[30] O. Bichler, D. Querlioz, S. J. Thorpe, J.-P. Bourgoin, and C. Gamrat, "Extraction of temporally correlated features from dynamic vision sensors with spike-timing-dependent plasticity," *Neural Networks*, vol. 32, pp. 339–348, 2012.

[31] D. Querlioz, O. Bichler, and C. Gamrat, "Simulation of a memristor-based spiking neural network immune to device variations," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*. IEEE, 2011, pp. 1775–1781.

[32] D. Querlioz, P. Dollfus, O. Bichler, and C. Gamrat, "Learning with memristive devices: How should we model their behavior?" in *Nanoscale Architectures (NANOARCH), 2011 IEEE/ACM International Symposium on*. IEEE, 2011, pp. 150–156.

[33] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128$\times$ 128 120 db 15 μs latency asynchronous temporal contrast vision sensor," *Solid-State Circuits, IEEE Journal of*, vol. 43, no. 2, pp. 566–576, 2008.

[34] C. Posch, D. Matolin, and R. Wohlgenannt, "High-dr frame-free pwm imaging with asynchronous aer intensity encoding and focal-plane temporal redundancy suppression," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*. IEEE, 2010, pp. 2430–2433.

[35] J. A. Leñero-Bardallo, T. Serrano-Gotarredona, and B. Linares-Barranco, "A signed spatial contrast event spike retina chip," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*. IEEE, 2010, pp. 2438–2441.

[36] J. L. McClelland, D. E. Rumelhart, P. R. Group *et al.*, "Parallel distributed processing," *Explorations in the microstructure of cognition*, vol. 2, 1986.

[37] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[38] B. Fritzke *et al.*, "A growing neural gas network learns topologies," *Advances in neural information processing systems*, vol. 7, pp. 625–632, 1995.

[39] A. Coates and A. Y. Ng, "Learning feature representations with k-means," in *Neural Networks: Tricks of the Trade*. Springer, 2012, pp. 561–580.