UC Irvine
UC Irvine Previously Published Works

Title

Event-Driven Random Back-Propagation: Enabling Neuromorphic Deep Learning Machines.

Permalink

https://escholarship.org/uc/item/335376xb

Authors

Neftci, Emre O
Augustine, Charles
Paul, Somnath
et al.

Publication Date

2017

DOI

10.3389/fnins.2017.00324

Neuromorphic Deep Learning Machines

Emre Neftci[1], Charles Augustine[3], Somnath Paul[3], and Georgios Detorakis[1]

[1]Department of Cognitive Sciences, UC Irvine, Irvine, CA, USA,
[3]Circuit Research Lab, Intel Corporation, Hilsboro, OR, USA,

January 24, 2017

Abstract

An ongoing challenge in neuromorphic computing is to devise general and computationally efficient models of inference and learning which are compatible with the spatial and temporal constraints of the brain. One increasingly popular and successful approach is to take inspiration from inference and learning algorithms used in deep neural networks. However, the workhorse of deep learning, the gradient descent Back Propagation (BP) rule, often relies on the immediate availability of network-wide information stored with high-precision memory, and precise operations that are difficult to realize in neuromorphic hardware. Remarkably, recent work showed that exact backpropagated weights are not essential for learning deep representations. Random BP replaces feedback weights with random ones and encourages the network to adjust its feed-forward weights to learn pseudo-inverses of the (random) feedback weights. Building on these results, we demonstrate an event-driven random BP (eRBP) rule that uses an error-modulated synaptic plasticity for learning deep representations in neuromorphic computing hardware. The rule requires only one addition and two comparisons for each synaptic weight using a two-compartment leaky Integrate & Fire (I&F) neuron, making it very suitable for implementation in digital or mixed-signal neuromorphic hardware. Our results show that using eRBP, deep representations are rapidly learned, achieving nearly identical classification accuracies compared to artificial neural network simulations on GPUs, while being robust to neural and synaptic state quantizations during learning.

1 Introduction

Biological neurons and synapses can provide the blueprint for inference and learning machines that are potentially thousandfold more energy efficient than mainstream computers. However, the breadth of application and scale of present-day neuromorphic hardware remains limited, mainly due to a lack of general and efficient inference and learning algorithms compliant with the spatial and temporal constraints of the brain.

Machine learning and deep learning are well poised for solving a broad set of applications using neuromorphic hardware, thanks to their general-purpose, modular, and fault-tolerant nature [Esser et al., 2016; Lee et al., 2016; Neftci et al., 2016]. One outstanding question is whether the learning phase in deep neural networks can be efficiently carried out in neuromorphic hardware. Performing learning on-the-fly in less controlled environments where no prior, representative dataset exists confers more fine-grained context awareness to behaving cognitive agents. However, deep learning usually relies on the immediate availability of network-wide information stored with high-precision memory. In digital computers, the access to this information funnels through the von Neumann bottleneck, which dictates the fundamental limits of the computing substrate. Distributing computations along multiple cores (such as in GPUs) is a popular solution to mitigate this problem, but even there the scalability of BP is often limited by its memory-intensive operations [Zhu et al., 2016].

The implementation of gradient BP on a neural substrate is even more challenging [Baldi et al., 2016; Grossberg, 1987; Lee et al., 2016] because it requires 1) using synaptic weights that are identical with

forward passes (symmetric weights requirements, also known as the weight transport problem), 2) carrying out the operations involved in BP including multiplications with derivatives and activation functions, 3) propagating error signals with high precision, 4) alternating between forward and backward passes, 5) changing the sign of synaptic weights, and 6) availability of targets (labels). The essence of these challenges is that it requires precise linear and non-linear computations, and more importantly because gradient BP requires information that is not local to the computational building blocks in a neural substrate, meaning that special communcation channels must be provisioned. Whether a given operation is local or not depends on the physical substrate that carries out the computations. For example, while symmetric weights in neural networks are compatible with von Neumann architectures (and even desirable since weights in both directions are shared), the same is not true in a distributed system such as the brain: elementary computing units do not have bidirectional connections with the same weight in each direction. Since neuromorphic implementations generally assume dynamics closely related to the those in the brain, requirements (1-4) above also hinder efficient implementations of BP in neuromorphic hardware.

Although previous work [Lee et al., 2016; Lillicrap et al., 2016; O'Connor and Welling, 2016] overcomes some of the fundamental difficulties of gradient BP listed above in spiking networks, here we tackle all of the key difficulties using event-driven random BP (eRBP), a learning rule for deep spiking neural networks achieving classification accuracies that are similar to those obtained in artificial neural networks, potentially running on a fraction of the energy budget with dedicated neuromorphic hardware.

eRBP builds on the recent advances in approximate forms of the gradient BP rule [Baldi et al., 2016; Lee et al., 2014; Liao et al., 2015; Lillicrap et al., 2016] for training spiking neurons of the type used in neuromorphic hardware to perform supervised learning. These approximations solve the non-locality problem by replacing BP weights with random ones, leading to remarkably little loss in classification performance on benchmark tasks [Baldi et al., 2016; Lillicrap et al., 2016] (requirement 1 above). Although a general theoretical understanding of random BP (RBP) is still lacking, extended simulations and analyses of linear networks show that, during learning, the network adjusts its feed-forward weights to learn an approximation of the pseudo-inverse of the (random) feedback weights, which is equally good in communicating gradients. eRBP is an asynchronous (event-driven) adaptation of random BP that can be tightly embedded with the dynamics of dual compartment I&F neurons that costs one addition and two comparisons per synaptic weight update. Extended experimentations show that the spiking nature of neuromorphic hardware and the lack of general linear and non-linear computations at the neuron does not prevent accurate learning on classification tasks (requirement 2, 3), and operates continuously and asynchronously without alternation of forward or backward passes (requirement 4). Additional experimental evidence shows that eRBP is robust to fixed width representations with limited neural and synaptic state precision, making it suitable for dedicated digital hardware. The success of eRBP lays out the foundations of neuromorphic deep learning machines, and paves the way for learning with streaming spike-event data in neuromorphic platforms at artificial neural network proficiencies. We demonstrate this in simulation of a custom digital neuromorphic processor using fixed point, discrete-time dynamics.

2 Results

2.1 Event-driven Random Backpropagation

The central contribution of this article is event-driven RBP (eRBP), a presynaptic spike-driven plasticity rule modulated by top-down errors and gated by the state of the postsynaptic neuron. The idea behind this additional modulation factor is motivated by supervised gradient descent learning in artificial neural networks and biologically plausible models of three-factor plasticity rules [Urbanczik and Senn, 2014], which was argued to subserve supervised, unsupervised and reinforcement learning, an idea that was also reported in [Lillicrap et al., 2016]. For a given neuron, the eRBP learning dynamics for synapse k can be summarized as follows:

$$\Delta w_k = T[t]\,\Theta(I[t])\mathrm{S}_{\mathrm{k}}^{\mathrm{pre}}[t] \tag{1}$$
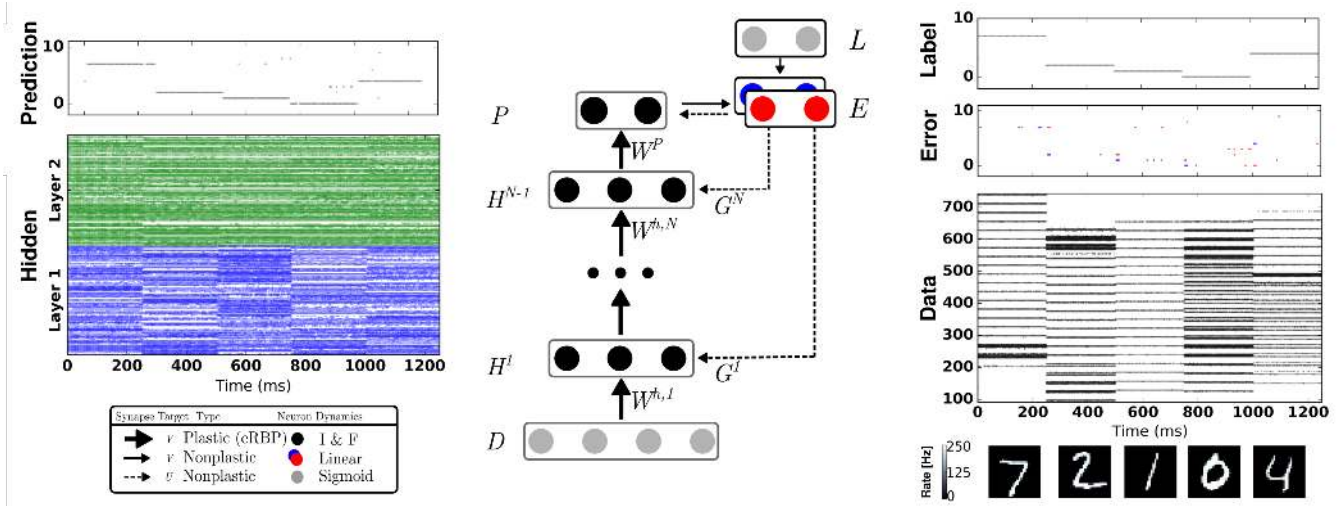
Figure 1: *Network Architecture for Event-driven Random Backpropagation (eRBP) and example spiking activity after training a 784-200-200-10 network for 60 epochs.* The network consists in feed-forward layers $(H^1, ..., H^N)$ for prediction and feedback layers for supervised training with labels (targets) L. Full arrows indicate synaptic connections, thick full arrows indicate plastic synapses, and dashed arrows indicate synaptic plasticity modulation. In this example, digits 7,2,1,0,4 were presented in sequence to the network. The digit pixel values are transformed to spike trains (layer D) using a Spike Response Model (Eq. (20)). Neurons in the network indicated by black circles were implemented as two-compartment leaky I&F neurons (Eq. (17) and Eq. (19)). The error is the difference between labels (L) and predictions (P), as is implemented using a pair of neurons coding for positive error (blue) and negative error (red), following Eq. (16). Each hidden neuron receives inputs from a random combination of the pair of error neurons to implement random BP. Output neurons receives inputs from the pair of error neurons in a one-to-one fashion. At the moment of data sample (digit) transitions, bursts of activity (about 3 spikes) in the error neurons occur. To prevent the perturbation of the weights during these transitions, no weight updates were undertaken immediately after changing data sample.

where $S_k^{pre}[t]$ represents the spike train of presynaptic neuron k ($S_k^{pre}[t] = 1$ if pre-synaptic neuron k spiked at time step t), and Θ is the derivative of the spiking neuron's activation function evaluated at the total synaptic input $I[t]$. The essence of eRBP is contained in the factor T. For the final output (prediction) layer, T is equal to the classification error e of the considered neuron, similarly to the delta rule. For hidden layers, the T is equal to the error projected randomly to the hidden neurons, *i.e.* $T = \sum_j g_j e_j$. where the g_j are random weights that are fixed during the learning.

We found that a boxcar function in place of Θ provides very good results, while being more amenable to hardware implementation compared to the alternative of computing the exact derivative of the activation function. This choice is motivated by the fact that the activation function of leaky I&F neurons with absolute refractory period can be approximated by a linear threshold unit with saturation whose derivative is exactly the boxcar function. Using a boxcar function with boundaries b_{min} and b_{max}, the eRBP synaptic weight update consists of additions and comparisons only, and can be captured using the following operations for each neuron:

> **function** ERBP
>   **for** $k \in \{$presynaptic spike indices $\mathbf{S}^{pre}\}$ **do**
>   **if** $b_{min} < I < b_{max}$ **then** $w_k \leftarrow w_k + T$,
>   **end if**
>   **end for**
> **end function**

where $\mathbf{S}_{pre}$ is the list of presynaptic neuron indices that have spiked, T is the linear combination of the error vector. In the spiking network, T is equal to the voltage of a auxiliary compartment that integrates spikes from the error neurons. Provided the second compartment dynamics, no multiplications are necessary for an eRBP update. Furthermore, the second compartment dynamics can be made multiplication free (see Methods) given custom bit shift operations and can be disabled after learning. Although eRBP is presented using a discrete-time notation, it is straightforward to generalize it in a continuous-time framework. This rule is reminiscent of membrane voltage-based rules, where spike-driven plasticity is induced only when membrane voltage is inside an eligibility window [Brader et al., 2007; Chicca et al., 2013].

The realization of eRBP on neuromorphic hardware requires an auxiliary learning variable for integrating and storing top-down error signals during learning, which can be substantiated as a dendritic compartment. Provided this variable, each synaptic weight update incurs only two comparison operations and one addition. Additions and comparisons can be implemented very naturally in neuromorphic VLSI circuits [Liu et al., 2002], and costs in the order of tens of femtojoules in digital circuits (45 nm processes [Horowitz, 2014]). In a large enough network, the cost of the second compartment dynamics is small compared to the cost of the synaptic update, since for N neurons, there are N^2 synapses. As a concrete example we use leaky, two compartment, current-based Integrate-and-Fire neurons with additive and multiplicative noise and linear synapses (See Methods). The gating term Θ, implemented as two comparisons, operates on the total synaptic input, rather than membrane or calcium as used in other work. This choice is guided by the gradient descent rule, which dictates that the derivative should be evaluated on the total input (Methods). The linearity of the synaptic dynamics allows to use a single dynamical variable for all synapses, such that the value of this dynamical variable is exactly equal to the total synaptic input I, and thus readily available at the neuron and the synapses.

The eRBP rule combined with its ability to learn deep representations with near equal accuracies as described below can enable neuromorphic deep learning machines on a wide variety of tasks. In the following, we focus on a design that is tailored for digital neuromorphic design, namely that some non-plastic synaptic weights can be exactly matched. Its implementation in a mixed signal design prone to fabrication mismatch and other non-idealities is the subject of ongoing work.

Network	Classification Error			
	eRBP	peRBP	RBP (GPU)	BP (GPU)
784-100-10	**3.94%**	**3.02%**	2.74%	2.19%
784-200-10	**3.13%**	**2.51%**	2.15%	1.81%
784-500-10	**2.96%**	**2.35%**	2.08%	1.8%
784-200-200-10	**3.04%**	**2.25%**	2.42%	1.91%

Table 1: Classification error on the permutation invariant MNIST task (test set). Bold indicates online trained spiking network

2.2 Spiking Networks Equipped with eRBP Learn with High Accuracy

We demonstrate eRBP in networks consisting of one and two hidden layers trained on permutation invariant MNIST (Tab. 1) , although eRBP can in theory generalize to other datasets, tasks and network architectures as well. Rather than optimizing for absolute classification performance, we compare to equivalent artificial (non-spiking) neural networks trained with RBP and standard BP, with free parameters fine-tuned to achieve the highest accuracy on the considered classification tasks (Tab. 1). On most network configurations eRBP achieved performances equivalent to those achieved with RBP in artificial networks. When equipped with probabilistic connections (peRBP) that randomly blank-out presynaptic spikes, the network performed better overall. This is because, as learning progresses, a significant portion of the neurons tend to fire near their maximum rate and synchronize their spiking activity across layers as a result of large synaptic weights (and thus presynaptic inputs). Synchronized spike activity is not well captured by our rate model, which is assumed by the eRBP (see Methods). Additive noise has relatively small effect when the magnitude of the presynaptic input is large. However, multiplicative blank-out noise improves learning by introducing irregularity in the presynaptic spike-trains even when presynaptic neurons fire regularly. Our previous work [Neftci et al., 2016] suggested that the probabilistic connections implement DropConnect regularization [Wan et al., 2013]. In contrast with [Wan et al., 2013], the probabilistic connections remain enabled both during learning and inference because the network dynamics depend strongly on this stochasticity. Interestingly, this type of "always-on" stochasticity also was argued to approximate Bayesian inference with Gaussian processes [Gal and Ghahramani, 2015].

Overall, the learned classification accuracy is comparable with those obtained with offline training of spiking neural networks (*e.g.* GPUs) using standard BP.

Transitions between two data samples of different class (digit) are marked by bursts of activity in the error neurons (Fig. 1). To overcome this problem, weight updates were disabled the first 50 ms after the new digit onset. In future work involving practical applications on autonomous systems, it will be beneficial to interleave learning and inference stages without explicitly controlling the learning rate. One way to achieve this is to introduce a negative bias in the error neurons by means of a constant negative input and an equal positive bias in the label neurons such that the error neuron can be only be active when an input label is provided[1]. The same solution can overcome the perturbations caused by bursts of error activity during digit transitions (see red and blue spikes in Fig. 1).

The presence of these bursts of error activity suggest that eRBP could learn spatiotemporal sequences as well. However, learning useful latent representations of the sequences requires solving a temporal credit assignment problem at the hidden layer – a problem that is commonly solved with gradient BP-through-time in artificial neural networks [Rumelhart et al., 1988] – which could be tackled using synaptic eligibility dynamics based on ideas of reinforcement learning [Sutton and Barto, 1998].

---

[1] Such logical "and" operation on top of a graded signal was previously used for conditional signal propagation in neuromorphic VLSI spiking neural networks [Neftci et al., 2013].
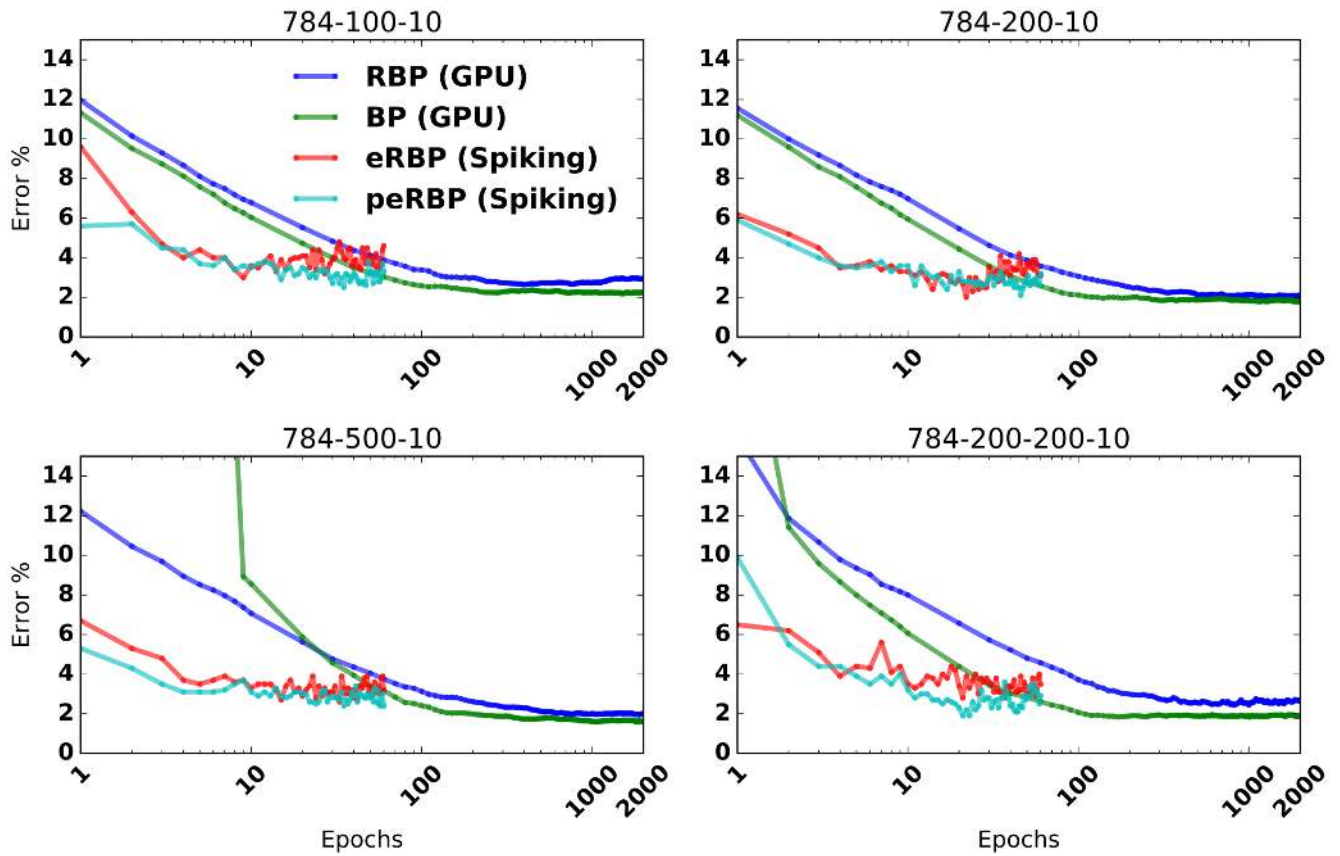
Figure 2: MNIST Classification error on fully connected artificial neural networks (BP and RBP) and on spiking neural networks (eRBP and peRBP).
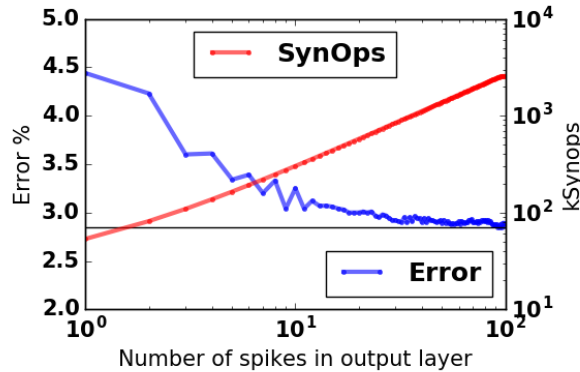
Figure 3: Classification error in the 784-200-10 network as a function of the number of spikes in the prediction layer, and total number of synaptic operations incurred up to each output spike. Horizontal line is 2.85%. To obtain this data, the network was first stimulated with random patterns, and the spikes in the output layer were counted after $\tau_{syn} = 4$ ms.

2.3 Classification with Single Spikes is Highly Accurate and Efficient

The response of the 784-200-10 network after stimulus onset is about one synaptic time constant. Using the first spike after $2\tau_s = 8$ms from the stimulus onset for classification leads to about 5% error (Fig. 3), and improves steadily as the number output layer spikes increase.

In this example, classification using the first spike incurred about 100 k synaptic operations (averaged over 10000 test samples), most of which occur between the data and the hidden layer (784 neurons and 200 neurons respectively). In existing dedicated neuromorphic hardware [Merolla et al., 2014; Park et al., 2014; Qiao et al., 2015], the energetic cost of a synaptic operation is about 20pJ. On such hardware, single spike classification in eRBP trained networks can potentially result in about $2\mu J$ energy per classification. This figure is comparable to the state-of-the-art in digital neuromorphic hardware ($\cong 2\,\mu J$ at this accuracy [Esser et al., 2015]) and current GPU technology ($> $ mJ). We note that no sparsity criterion was enforced in this network. We expect that sparsity implemented explicitly using weight regularization or implicitly using Dropout or DropConnect techniques [Baldi and Sadowski, 2013] can further reduce this energy, by virtue of the lower activity in the hidden layer.

The low latency response with high accuracy may seem at odds with the inherent firing rate code underlying the network computations (See Methods). However, a code based on the time of the first-spike is consistent with a firing rate code, since a neuron with a high firing rate is expected to fire first [Gerstner and Kistler, 2002]. In addition, the onset of the stimulus provokes a burst of synchronized activity, which further favors the rapid onset of the prediction response. These results suggest that despite the underlying firing rate code, eRBP can take advantage of the spiking dynamics, with classification accuracies comparable to spiking networks trained exclusively for single-spike classification [Mostafa, 2016].

2.4 Spiking Networks Equipped with eRBP Learn Rapidly and Efficiently

The spiking neural network requires fewer iterations of the dataset to reach the peak classification performance compared to the artificial neural network trained with batch gradient descent (Fig. 2). In minibatch learning, weight updates are averaged across the minibatch. Batch or Minibatch learning improves learning speed in conventional hardware thanks to vectorization libraries or efficient parallelization with GPUs' SIMD architecture, and lead to smoother convergence. However, this approach result in n_{batch} times fewer weight updates per epoch compared to online gradient descent. In contrast, the spiking neuron network is updated after each sample presentation, accounting in large part for the faster convergence of learning. Other spiking networks trained online using stochastic gradient descent achieved comparable speedup [Lee

et al., 2016; O'Connor and Welling, 2016]. These results are not entirely surprising since seminal work in stochastic gradient descent established that, with suitable conditions on the learning rate, the solution to a learning problem obtained with stochastic gradient descent is asymptotically as good as the solution obtained with batch gradient descent [Le Cun and Bottou, 2004] for a given number of samples. Furthermore, for equal computational resources, online gradient descent can process more data samples [Le Cun and Bottou, 2004], while requiring less memory for implementation. Thus, for an equal number of compute operations per unit time, online gradient descent converges faster than batch learning. Standard artificial neural networks can be trained using $n_{batch} = 1$, but learning becomes much slower on standard platforms because the operations cannot be vectorized across data samples. (The converse is not true, however: $n_{batch} > 1$ in spiking networks is non-local because it requires storing synaptic weight gradients.) It is fortunate that synaptic plasticity is an inherently "online" in the machine learning sense, given that potential applications of neuromorphic hardware often involve real-time streaming data.

2.5 eRBP can Learn with low Precision, Fixed Point Representations

The effectiveness of stochastic gradient descent degrades when the precision of the synaptic weights using a fixed point representation is smaller than 16 bits [Courbariaux et al., 2014]. This is because quantization determines the smallest learning rate and bounds the range of the synaptic weights, thereby preventing averaging the variability across dataset iterations. The tight integration of memory with computing circuits as pursued in neuromorphic chip design is challenging due to space constraints and memory leakage. For this reason, full precision (or even 16 bit) computer simulations of spiking networks may be unrepresentative of performance that can be attained in dedicated neuromorphic designed due to quantization of neural states and parameters, and synaptic weights.

Extended simulations suggest that the random BP performances at 10 bits precision is indistinguishable from unquantized weights [Baldi et al., 2016], but whether this is the case for online learning was not yet tested. Here, we hypothesize that 8 bit synaptic weight is a good trade-off between the ability to learn with high accuracy and the cost of implementation in hardware. To demonstrate robustness to such constraints, we simulate quantized versions of the eRBP network using low precision fixed point representations (8 bits per synaptic weight and 16 bits for neural states). Consistent with existing findings, our simulations of eRBP in a quantized 784-100-10 network show that eRBP still performs reasonably well under these conditions (Fig. 4). While many weights aggregate at the boundaries, a majority of them remain away from the boundaries. Although the learned accuracies using quantized simulations fall slightly short of the full precision ones, we emphasize that no specific rounding mechanisms [Muller and Indiveri, 2016] was used to obtain these results and are expected to tighten this gap.

3 Discussion

The gradient descent BP rule is a powerful algorithm that is ubiquitous in deep learning, but when implemented in a von Neumann or neural architecture, it relies on the immediate availability of network-wide information stored with high-precision memory. More specifically, [Baldi et al., 2016] and [Lee et al., 2016] list several reasons why the following requirements of gradient BP make them biologically implausible. The essence of these difficulties is that gradient BP is non-local in space and in time when implemented on a neural substrate, and requires precise linear and non-linear computations. The feedback alignment work demonstrated that symmetric weights were not necessary for communicating error signals across layers. Here we demonstrated a learning rule inspired by Building on feedback alignment [Lillicrap et al., 2016], and membrane voltage-gated plasticity rules, and three-factor synaptic plasticity rules proposed in the computational neuroscience literature. With an adequate network architecture, we find that the spike-based computations and the lack of general linear and non-linear computations and alternating forward and backward steps does not prevent accurate learning. Although previous work overcome some of the non-locality problems of gradient BP [Lee et al., 2016; Lillicrap et al., 2016; O'Connor and Welling, 2016],
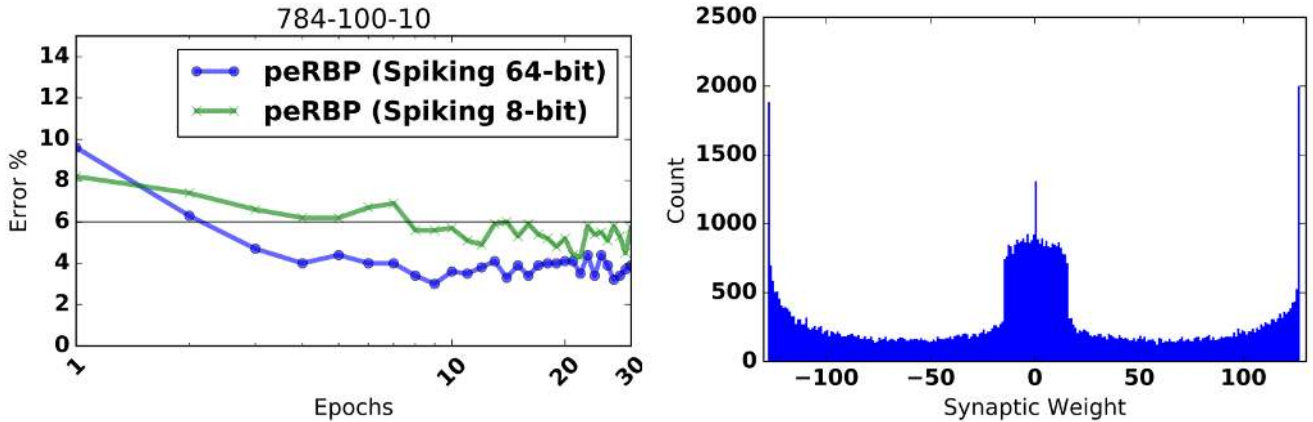
Figure 4: (**Left**) MNIST Classification error using a fully connected 784-100-10 network with quantized synaptic weights and neural states (8 bits and 16 bits respectively) (**Right**). Histogram of synaptic weights of the quantized network after training.

eRBP overcomes all of the key difficulties using a simple rule that incurs one addition and two comparisons per synaptic weight update.

Taken together, our results suggest that general-purpose deep learning using streaming spike-event data in neuromorphic platforms at artificial neural network proficiencies is realizable. To emphasize this, we have implemented eRBP using a software simulations using fixed point, discrete-time dynamics, called the neural and synaptic array transceiver. This simulator is compatible with a digital neuromorphic hardware currently in development, the full details of which will be published elsewhere.

Our experiments target digital implementations of spiking neural networks with embedded plasticity. However, membrane-voltage based learning rules implemented in mixed-signal neuromorphic hardware [Huayaney et al., 2016; Qiao et al., 2015] are compatible with eRBP provided that synaptic weight updates can be modulated by an external signal on a neuron-to-neuron basis. Following this route, and combined with the recent advances in neuromorphic engineering and emerging nanotechnologies, eRBP can become key to ultra low-power processing in space and power constrained platforms.

3.1 Why Neuromorphic Learning Machines?

Spiking neural networks, especially those based on the I&F neuron types severely restrict the possible computations during learning and inference. With the wide availability of graphical processing units and future dedicated machine learning accelerators, the neuromorphic spike-based approach more machine learning tasks is often heavily criticized as being misguided. While this is true for some cases and on metrics based on absolute accuracy at some standardized benchmark task, the premise of neuromorphic engineering, *i.e.* that electronic and biological share similar constraints on communication, power and reliability [Mead, 1990], extend to the algorithmic domain. That is, accommodating machine learning algorithms within the constraints ultra-low power hardware for adaptive behavior (*i.e.* embedded learning) is likely to result in solutions for communication, computations and reliability that are in strong resemblance with how the brain solves similar problems. In addition, the neuromorphic approach offers a few advantages over straight artificial neural networks: 1) Asynchronous, event-based communication most often used in neuromorphic hardware considerably reduce the communication between distributed processes, 2) Spiking networks naturally exploit "rate" codes and "spike" codes where single spikes are meaningful, leading to fast and thus power-efficient and gradual responses (Fig. 3, see also [O'Connor and Welling, 2016]).

One prominent example is the Binarized Neural Network (BNN) [Courbariaux and Bengio, 2016]. The BNN is trained such that weights and activities are -1 or 1, which considerably reduces the energetic footprint of inference, because multiplications are not necessary and the memory requirements for inference

are much smaller. The discrete, quantized dynamics used in this work and developed independently from the BNN shares many similarities, such as binary activations (spikes), low-precision variables, and straight-through gradient estimators. Our neurally inspired approach innovates several new features for binarized networks: network activity is sparse and data-driven (asynchronous), random variables for stochasticity are generated only when neurons spike, errors are backpropagated only for misclassified examples, and learning is ongoing leading to accurate early single spike classification. Many other examples that led to the unprecedented success in machine learning were discovered independently of equivalent neural mechanisms, such as normalization techniques for improving deep learning [Ioffe and Szegedy, 2015; Ren et al., 2016], attention, short-term memory for learning complex tasks [Graves et al., 2014], and memory consolidation through fast replays for reinforcement learning [Kumaran et al., 2016; Mnih et al., 2015]. The convergence between the two approaches (neuromorphic *vs.* artificial) will not only improve the design of neuromorphic learning machines, but can also widen the breadth of knowledge transfer between computational neuroscience and deep learning.

3.2   Relation to Prior Work in Random Backpropagation

Our learning rule builds on the feedback alignment learning rule proposed in [Lillicrap et al., 2016], showing that random feedback can deliver useful teaching signals by aligning the feed-forward weights with the feedback weights. The authors also demonstrated a spiking neural network implementing feedback alignment, demonstrating that feedback alignment is able to implicitly adapt to random feedback when the forward and backward pathways both operate continuously. However, their learning rule is not event-based as in eRBP, but operates on a continuous-time fashion that is not directly compatible with spike-driven plasticity, and a direct neuromorphic implementation thereof would be inadequate due to the high bandwidth communication required between neurons. Furthermore, their model is a spike response model that does not emulate the physical dynamics of spiking neurons such as I&F neurons. Another difference between eRBP and the network presented in [Lillicrap et al., 2016] is that eRBP contains only one error-coding layer, whereas feedback alignment contains one error-coding layer per hidden layer. Such direct feedback alignment was recently proposed in [Nø kland, 2016] and [Baldi et al., 2016], and a theoretical analysis there showed that gradient computed in this fashion point is within 90 degrees of the backpropagated gradient. [Baldi et al., 2016] studied feedback alignment in the framework of local learning and the learning channel, and derived several other flavors of random BP such as adaptive, sparse, skipped and indirect RBP, along with their combinations. In related work, [Lee et al., 2014] showed how feedback weights can be learned to improve the classification accuracy by training the feedback weights to learn the inverse of the feedforward mapping.

3.3   Relation to Prior Work in Spiking Deep Neural Networks

Several approaches successfully realized the mapping of pre-trained artificial neural networks onto spiking neural networks using a firing rate code [Cao et al., 2014; Das et al., 2015; Diehl et al., 2015; Esser et al., 2016; Hunsberger and Eliasmith, 2015; Marti et al., 2015; Neftci et al., 2014b; O'Connor et al., 2013; O'Connor and Welling, 2016] Such mapping techniques have the advantage that they can leverage the capabilities of existing machine learning frameworks such as Caffe [Jia et al., 2014] or Theano [Goodfellow et al., 2013] for brain-inspired computers. More recently, [Mostafa, 2016] used a temporal coding scheme where information is encoded in spike times instead of spike rates and the dynamics are cast in a differentiable form. As a result, the network can be trained using standard gradient descent to achieve very accurate, sparse and power-efficient classification. Although eRBP achieves comparable results, their approach naturally leads to sparse activity in the hidden layer which can be more advantageous in large and deep networks.

An intermediate approach is to learn online with standard BP using spike-based quantization of network states [O'Connor and Welling, 2016] and the instantaneous firing rate of the neurons [Lee et al., 2016]. [O'Connor and Welling, 2016] eschews neural dynamics and instead operates directly on event-based (spiking) quantizations of vectors. Using this representation, common neural network operations including online gradient BP are mapped on to basic addition, comparison and indexing operations applied to streams

of signed spikes. As in eRBP, their learning rule achieves better results when weight updates are made in an event-based fashion, as this allows the network to update its parameters many times during the processing of a single data sample. [Lee et al., 2016] propose a method for training spiking neural networks via a formulation of the instantaneous firing rate of the neuron obtained by low-pass filtering the spikes. There, quantities that can be related to the postsynaptic potential (rather than mean rates) are used to compute the derivative of the activity of the neuron, which can provide a useful gradient for backpropagation. [Esser et al., 2016] use multiple spiking convolutional networks trained offline to achieve near state-of-the-art classification in standard benchmark tasks. Their approach maps onto the all-digital spiking neural network architecture using trinary weights. For the above approaches, the eRBP learning rule presented here can be used as a drop-in replacement and can reduce the computational footprint of learning by simplifying the backpropagated chain path and by operating directly with locally available variables *i.e.* membrane potentials and spikes.

3.4 Relation to Prior Work in Spike-Driven Plasticity Rules

STDP has been shown to be very powerful in a number of different models and tasks [Neftci et al., 2014a; Nessler et al., 2013; Thorpe et al., 2001]. Although the implementation of acausal updates (triggered by presynaptic firing) is typically straightforward in cases where presynaptic lookup tables are used, the implementation of causal updates (triggered by postsynaptic firing) can be challenging due to the requirement of storing a reverse look-up table. Several approximations of STDP exist to solve this problem [Galluppi et al., 2014; Pedroni et al., 2016], but require dedicated circuits.

Thus, there is considerable benefit in hardware implementations of synaptic plasticity rules that forego the causal updates. Such rules, which we referred to as spike-driven plasticity, can be consistent with STDP [Brader et al., 2007; Clopath et al., 2010; Qiao et al., 2015; Sheik et al., 2016a], especially when using dynamical variables that are representative of the pre- and postsynaptic firing rates (such as calcium or average membrane voltage).

A common feature among spike-driven learning rules is a modulation or gating with a variable that reflects the average firing rate of the neuron for example through calcium concentration [Graupner and Brunel, 2012; Huayaney et al., 2016] or the membrane potential [Clopath et al., 2010; Sheik et al., 2016a] or both [Brader et al., 2007]. [Sheik et al., 2016a] recently proposed a membrane-gated rule inspired by calcium and voltage-based rules with homeostasis for learning unsupervised spike pattern detection. Their rule statistically emulates pairwise STDP using presynaptic spike timing only and using additions and multiplications. Except for homeostasis, eRBP follows similar dynamics but potentiation and depression magnitudes are dynamic and determined by external modulation, and comparisons are made on total synaptic currents to avoid the effect of the voltage reset after firing.

The two compartment neuron model used in this work is motivated from conductance-based dynamics in [Urbanczik and Senn, 2014] and previous neuromorphic realizations of two compartment mixed signal spiking neurons [Park et al., 2014]. Although the spiking network used in this work is current-based rather than conductance-based, eRBP shares strong similarities to the three-factor learning rule employed in [Urbanczik and Senn, 2014]. The latter is composed of three factors: an approximation of the prediction error, the derivative of the membrane potential with respect to the synaptic weight, and a positive weighting function that stabilizes learning in certain scenarios. The first factor corresponds to the error modulation, while the second and third factors roughly correspond to the presynaptic activity and the derivative of the activation function. The differences between eRBP and [Urbanczik and Senn, 2014] (besides from the *random* BP which was considered in [Lillicrap et al., 2016]) stems mainly from two facts: 1) the firing rate description used here for simplicity and for easier comparisons between artificial neural networks and spiking neural networks and 2) eRBP is fully event-based in the sense that weights are updated only when the presynaptic neurons spike, in order to make memory and compute operations more efficient in hardware.

4 Conclusions and Future Directions

This article demonstrates a local learning rule for deep, feed-forward neural networks achieving classification accuracies on par with those obtained using equivalent machine learning algorithms. The learning rule combines two features: 1) algorithmic simplicity: one addition and two comparisions per synaptic update provided one auxiliary state per neuron and 2) Locality: all the information for the weight update is available at each neuron and the synapse. The combination of these two features enables learning dynamics for deep learning in neuromorphic hardware.

Existing literature suggests that that random BP also works for unsupervised learning [Baldi et al., 2016; Lee et al., 2014] in deeper and convolutional networks. It can be reasonably expected that the deep learning community will uncover many variants of random BP, including in recurrent neural networks for sequence learning and memory augmented neural networks. In tandem with these developments, we envision that such RBP techniques will enable the embedded learning of pattern recognition, attention, working memory and action selection mechanisms which promise transformative hardware architectures for embedded computing.

This work has focused on unstructured, feed-forward neural networks and a single benchmark task across multiple implementations for ease of comparison. Limitations in deep learning algorithms are often invisible on "small" datasets like MNIST [Liao et al., 2015]. Random BP was demonstrated to be effective in a variety of tasks and network structures [Baldi et al., 2016; Liao et al., 2015], including convolutional neural networks. Although random BP was reported to work well in this case [Liao et al., 2015], the parameter sharing in convnets is inherently non-local. Despite this non-locality, neuromorphic implementation of convnets are still possible in neuromorphic [Qiao et al., 2015] if presynaptic connectivity tables are stored rather than postsynaptic tables.

5 Methods

5.1 Derivation of Event-driven Random Backpropagation

In artificial neural networks, the mean-squared cost function for one data sample in a single layer neural network is:

$$L = \frac{1}{2} \sum_i e_i^2,$$
$$e_i = (y_i - l_i),$$

(2)

where e_i is the error of prediction neuron i, $y_i = \phi(\sum_j w_{ij} x_j)$ is the activity of the prediction neuron i with activation function ϕ, $\mathbf{x}$ is the data sample and l_i is the label associated to the data sample. The task of learning is to minimize this cost over the entire dataset. The gradient descent rule in artificial neural networks is often used to this end by modifying the network parameters $\mathbf{w}$ in the direction opposite to the gradient:

$$w_{ij}[t+1] = w_{ij}[t] - \eta \frac{\partial}{\partial w_{ij}} L,$$
$$\text{where } \frac{\partial}{\partial w_{ij}} L = \phi'(\sum_j w_{ij} x_j) e_i x_j.$$

(3)

where η is a small learning rate. In deep networks, *i.e.* networks containing one or more hidden layers, the weights of the hidden layer neurons are modified by backpropagating the errors from the prediction layer

using the chain rule:

$$\frac{\partial}{\partial w_{ij}^l} L = \delta_{ij}^l y_j^{l-1},$$

$$\text{where } \delta_{ij}^l = \phi'\left(\sum_j w_{ij}^{l-1} y_j^{l-1}\right) \sum_k \delta_{ik}^{l+1} w_{ik}^{l+1}. \tag{4}$$

where the δ for the topmost layer is e_i, as in Eq. (3) and y at the bottommost layer is the data x. This update rule is the well-known gradient back propagation algorithm ubiquitously used in deep learning [Rumelhart et al., 1988]. Learning is typically carried out in forward passes (evaluation of the neural network activities) and backward passes (evaluation of the δs). The computation of the δ requires knowledge of the forward weights, thus gradient BP relies on the immediate availability of a symmetric transpose of the network for computing the backpropagated errors δ_{ij}^l. Often the access to this information funnels through the von Neumann bottleneck, which dictates the fundamental limits of the computing substrate.

In the random BP rule considered here, the BP term δ is replaced with:

$$\delta_{RBP}^l = \phi'\left(\sum_j w_{ij}^{l-1} y_j^{l-1}\right) \sum_k e_k g_{ik}^l \tag{5}$$

where g_{ik}^l are fixed random numbers. This backpropagated term does not depend on the previous layer $l+1$, and thus does not have a recursive structure as in standard BP (Eq. (4)) or feedback alignment [Lillicrap et al., 2016]. This form was previously referred to as direct feedback alignment [Nø kland, 2016] or skipped RBP [Baldi et al., 2016] and was shown to perform equally well on a broad spectrum of tasks. A detailed justification of random BP is out of the scope of this article, and interested readers are referred to [Baldi et al., 2016; Lillicrap et al., 2016; Nø kland, 2016].

In the context of models of biological spiking neurons, RBP is appealing because it circumvents the problem of calculating the backpropagated errors and does not require bidirectional synapses or symmetric weights. RBP works remarkably very well in a wide variety of classification and regression problems, using supervised and unsupervised learning in feed-forward networks, with a very small penalty in accuracy.

The above BP rules are commonly used in artificial neural networks, where neuron outputs are represented as single scalar variables. To derive an equivalent spike-based rule, we start by matching this scalar value is the neuron's instantaneous firing rate. The cost function and its derivative for one data sample is then:

$$L_{sp} = \frac{1}{2} \sum_i (\nu_i^p(t) - \nu_i^l(t))^2$$

$$\frac{\partial}{\partial w_{ij}} L_{sp} = \sum_i e_i(t) \frac{\partial}{\partial w_{ij}} \nu_i^p(t) \tag{6}$$

where $e_i(t)$ is the error of prediction unit i and ν^p, ν^l are the firing rates of prediction and label neurons, respectively.

Random BP (Eq. (5)) is straightforward to implement in artificial neural network simulations. However, spiking neurons and synapses, especially with the dynamics that can be afforded in low-power neuromorphic implementations typically do not have arbitrary mathematical operations at their disposal. For example, evaluating the derivative ϕ can be difficult depending on the form of ϕ and multiplications between the multiple factors involved in RBP can become very costly given that they must be performed at every synapse for every presynaptic event.

In the following, we derive an event-driven version of RBP that uses only two comparisons and one addition for each presynaptic spike to perform the weight update. The derivation proceeds as follows: 1) Derive the firing rate ν, $i.e$ the equivalent of ϕ in the spiking neural network, 2) Compute its derivative $\frac{\partial}{\partial w_{ij}} \nu_i(t)$, 3) Introduce modulation with a random linear combination of the classification error to the hidden neurons, 4) Devise a plasticity rule that increments the weight with the product of the latter two factors times the presynaptic activity.

Activation Function of Spiking Neurons with Background Poisson Noise and its Derivative

The dynamics of spiking neural circuits driven by Poisson spike trains is often studied in the diffusion approximation [Brunel, 2000; Brunel and Hakim, 1999; Deco et al., 2008; Fusi and Mattia, 1999; Renart et al., 2003; Tuckwell, 2005; Wang, 1999]. In this approximation, the firing rates of individual neurons are replaced by a common time-dependent population activity variable with the same mean and two-point correlation function as the original variables, corresponding here to a Gaussian process. The approximation is true when the following assumptions are verified: 1) the charge delivered by each spike to the postsynaptic neuron is small compared to the charge necessary to generate an action potential, 2) the number of afferent inputs to each neuron is large, 3) the spike times are uncorrelated. In the diffusion approximation, only the first two moments of the synaptic current are retained. The currents to the neuron, $I(t)$, can then be decomposed as:

$$I(t) = \mu + \sigma \eta(t), \tag{7}$$

where $\mu = \langle I(t) \rangle = \sum_j w_j \nu_j$ and $\sigma^2 = w_{bg}^2 \nu_{bg}$, where ν_{bg} is the firing rate of the background activity, and $\eta(t)$ is the white noise process. We restrict neuron dynamics to the case of synaptic time constants that are much larger than the membrane time constant, *i.e.* $\tau_m \ll \tau_{syn}$, such that we can neglect the fluctuations caused by synaptic activity from other neurons in the network *i.e.* σ is constant. Although the above dynamics are not true in general, in a neuromorphic approach, the parameters can be chosen accordingly during configuration or at design.

In this case, the neuron's membrane potential dynamics is an Ornstein-Uhlenbeck (OU) process [Gardiner, 2012]. The stationary distribution of the freely evolving membrane potential (no firing threshold) is a Gaussian distribution:

$$V_{nt} \sim N(\frac{\mu}{g_L}, \frac{\sigma^2}{2g_L^2 \tau_m}). \tag{8}$$

where g_L is the leak conductance and τ_m is the membrane time constant. Although this distribution is generally not representative of the membrane potential of the I&F neuron due to the firing threshold [Gerstner and Kistler, 2002], the considered case $\tau_m \ll \tau_{syn}$ yields approximately a truncated Gaussian distribution, where neurons with $V_{nt} > 0$ fire at their maximum rate of $\frac{1}{\tau_{refr}}$. This approximation is less exact for very large μ due to the resetting, but the resulting form highlights the essence of eRBP while maintaining mathematical tractability. Furthermore, using a first-passage time approach, [Petrovici et al., 2013] computed corrections that account for small synaptic time constants and the effect of the firing threshold on this distribution.

The firing rate of neuron i is approximately equal to the inverse of the refractory period, $\nu_i = \tau_{refr}^{-1}$ with probability $P(V_{nt,i}(t+1) \geq 0|\mathbf{s}(t))$ and zero otherwise. The probability is equal to one minus the cumulative distribution function of $V_{nt,i}$:

$$P(V_{nt,i}(t+1) \geq 0|\mathbf{s}(t)) = \frac{1}{2} \left(1 + \mathrm{erf}\left(\frac{\mu_i(t)}{\sigma_{OU} \sqrt{2}} \right) \right),$$

where "erf" stands for the error function. The firing rate of neuron i becomes:

$$\nu_i = \frac{1}{\tau_{refr}} \frac{1}{2} \left(1 + \mathrm{erf}\left(\frac{\sqrt{\tau_m}}{\sigma} \sum_j w_{ij} \nu_j \right) \right). \tag{9}$$

For gradient descent, we require the derivative of the neuron's activation function with respect to the weight w. By definition of the cumulative distribution, this is the Gaussian function in Eq. (8) times the presynaptic activity:

$$\frac{\partial}{\partial w_{ij}} \nu_i \propto \frac{1}{\sigma_{OU} \sqrt{2\pi}} \exp(-\frac{\mu_i(t)^2}{2\sigma_{OU}^2}) \nu_j(t). \tag{10}$$

14

As in previous work [Neftci et al., 2014a], we replace $\nu_j(t)$ in the above equations with the presynaptic spike train $s_j(t)$ to obtain an asynchronous, *event-driven* update, where the derivative is evaluated only when the presynaptic neuron spikes. This approach is justified by the fact that the learning rate is typically small, such that the event-driven updates are averaged at the synaptic weight variable [Gerstner and Kistler, 2002]. Thus the derivative becomes:

$$\frac{\partial}{\partial w_{ij}}\nu_i \propto \begin{cases} \exp(-\frac{\mu_i(t)^2}{2\sigma_{OU}^2}) \text{ if } s_j(t) = 1 \\ 0 \text{ otherwise} \end{cases} . \tag{11}$$

In the considered spiking neuron dynamics, the Gaussian function is not directly available. Although, a sampling scheme based on the membrane potential to approximate the derivative is possible, here we follow a simpler solution: Backed by extensive simulations, and inspired by previously proposed learning rules based on membrane potential gated learning rules [Brader et al., 2007; Clopath et al., 2010; Sheik et al., 2016a], we find that replacing the Gaussian function with a boxcar function Θ operating on the total synaptic input, $I(t)$, with boundaries b_{min} and b_{max} yields results that are as good as using the exact derivative. With appropriate boundaries, $\Theta(I(t))$ can be interpreted as a piecewise constant approximation of the Gaussian function[2] since $I(t)$ is proportional to its argument $\sum_j w_{ij}\nu_j$, and has the advantage that an explicit multiplication with the modulation is unnecessary in the random BP rule (explained below).

$$\frac{\partial}{\partial w_{ij}}\nu_i \propto \begin{cases} 1 \text{ if } s_j(t) = 1 \text{ and } b_{min} < I_i(t) < b_{max} \\ 0 \text{ otherwise} \end{cases} \tag{12}$$

The resulting derivative function is similar in spirit to straight-through estimators used in machine learning [Courbariaux and Bengio, 2016].

Derivation of Event-Driven Random Backpropagation

For simplicity, the error $e_i(t)$ is computed using a pair of spiking neurons with a rectified linear activation function. One neuron computes the positive values of $e_i(t)$, while the other neuron computes the negative values of $e_i(t)$ such that:

$$\begin{aligned} \nu_i^{E+}(t) &\propto \nu_i^p(t) - \nu_i^l(t), \\ \nu_i^{E-}(t) &\propto -\nu_i^p(t) + \nu_i^l(t). \end{aligned} \tag{13}$$

Each pair of error neurons synapse with a leaky dendritic compartment U of the hidden and prediction neurons using equal synaptic weights with opposite sign, generating a dendritic potential proportional to $(\nu_i^{E+}(t) - \nu_i^{E-}(t)) \cong e_i$. Several other schemes for communicating the errors are possible. For example an earlier version of eRBP used on a positively biased error neuron per class (rather than a positive negative pair as above) such that the neuron operated (mostly) in the linear regime. This solution led to similar results but was computationally more expensive due to error neurons being strongly active even when the classification was correct. Population codes of heterogeneous neurons as in [Eliasmith and Anderson, 2004; Salinas and Abbott, 1994] may provide even more flexible dynamics for learning. The weight update for the last layer becomes:

$$\Delta w_{ij} \propto \begin{cases} -e_i(t) \text{ if } s_j^h(t) = 1 \text{ and } b_{min} < I_i < b_{max} \\ 0 \text{ otherwise} \end{cases} . \tag{14}$$

The weight update for the hidden layers is similar, except that a random linear combination of the error is used instead of e_i:

$$\Delta w_{ij}^C \propto \begin{cases} -\sum_k g_{ik} e_k^E(t) \text{ if } s_j^C(t) = 1 \text{ and } b_{min} < I_i < b_{max} \\ 0 \text{ otherwise} \end{cases} . \tag{15}$$

---

[2]or equivalently, for the purpose of the derivative evaluation, the activation function is approximated as a rectified linear with hard saturation at τ_{refr}^{-1}, also called "hard tanh" in the machine learning community.

where $C = \{d, h\}$. All weight initializations are scaled with the number of rows and the number of columns as $g_{ik} \sim U(\sqrt{\frac{6}{N_E + N_H}})$, where N_E is the number of error neurons and N_H is the number of hidden neurons. In the following, we detail the spiking neuron dynamics that can efficiently implement eRBP.

5.2 Spiking Neural Network and Plasticity Dynamics

The network used for eRBP consists of one or two feed-forward layers (Fig. 1) with N_d "data" neurons, N_h hidden neurons and N_p prediction neurons. The top layer, labeled P, is the prediction. The feedback from the error population is fed back directly to the hidden layers' neurons. The network is composed of three types of neurons:

1) **Error-coding neurons** are non-leaky I&F neurons following the linear dynamics:

$$C\frac{\mathrm{d}}{\mathrm{d}t}V_i^{E+} = w^{L+}(s_i^P(t) - s_i^L(t))$$
$$\text{if } V^{E+} > V_T^E \text{ then } V^{E+} \leftarrow V^{E+} - V_T^E, \tag{16}$$

where $s_i^P(t)$ and $s_i^L(t)$ are spike trains from prediction neurons and labels (teaching signal). To prevent negative runaway dynamics, a rigid boundary at zero is imposed. In addition, the membrane potential is lower bounded to V_T^E. Each error neuron has one counterpart neuron with weights of opposite sign, i.e. $w^{L-} = -w^{L+}$ to encode the negative errors. The firing rate of the error-coding neurons is proportional to a linear rectification of the inputs. For simplicity, the label spike train is regular with firing rate equal to τ_{refr}^{-1}. When the prediction neurons classify correctly, $(s_i^P(t) - s_i^L(t)) \cong 0$, such that the error neurons remain silent.

2) **Hidden neurons** follow current-based leaky I&F dynamics:

$$C\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} V_i^h \\ U_i^h \end{pmatrix} = -\begin{pmatrix} g_V V_i^h \\ g_U U_i^h \end{pmatrix} + \begin{pmatrix} I_i^h + \sigma_w \eta_i^h(t) \\ \sum_{k=1}^{N_L} g_{ik}^{E+} s_k^{E+}(t) - g_{ik}^{E-} s_k^{E-}(t) \end{pmatrix}$$
$$\tau_{syn}\frac{\mathrm{d}}{\mathrm{d}t}I_i^h = -I_i^h + \sum_{k=1}^{N_d} w_{ik}^d s_k^d(t)\xi(t) + \sum_{j=1}^{N_h} w_{ij}^h s_j^h(t)\xi(t) \tag{17}$$
$$\text{if } V_i(t) > V_T \text{ then } V_i^h \leftarrow 0 \text{ during refractory period } \tau_{refr}.$$

where $s_k^d(t)$ and $s_j^h(t)$ are the spike trains of the data neurons and the hidden neurons, respectively, I^h are current-based synapse dynamics, $\sigma_w s_i^{bg}(t)$ a Poisson process of rate 1kHz and amplitude σ_w, and ξ is a stochastic Bernouilli process with probability p (indices i, j are omitted for clarity). The Poisson process simulates background Poisson activity and contributes additively to the membrane potential, whereas the Bernouilli process contributes multiplicatively by randomly "blanking-out" the proportion $(1 - p)$ of the input spikes. In this work, we consider feed-forward networks, i.e the weight matrix w^h is restricted to be upper diagonal. Each neuron is equipped with a separate "dendritic" compartment U_i^h following similar subthreshold dynamics as the membrane potential and where $s^E(t)$ is the spike train of the error-coding neurons and g_{ij}^E is a fixed random matrix. The dendritic compartment is not directly coupled to the "somatic" membrane potential V_i^h, but indirectly through the learning dynamics. For every hidden neuron i, $\sum_j w_{ij}^E = 0$, ensuring that the spontaneous firing rate of the error-coding neurons does not bias the learning. The synaptic weight dynamics follow a dendrite-modulated and gated rule:

$$\frac{\mathrm{d}}{\mathrm{d}t}w_{ij}^h = \eta U_i^h \Theta(I_i^h)s_j^h(t). \tag{18}$$

where Θ is a boxcar function with boundaries b_{min} and b_{max}.

3) **Prediction neurons**, synapses and synaptic weight updates follow the same dynamics as the hidden neurons except for the dendritic compartment, and one-to-one connection with pairs of error-neurons
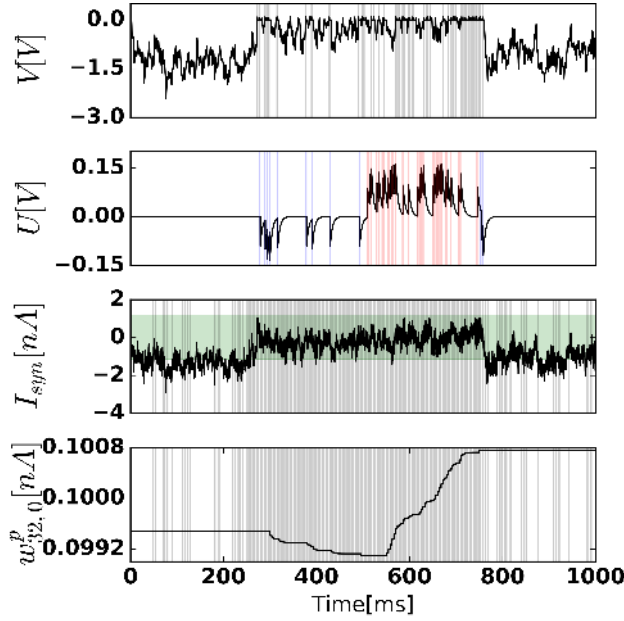
Figure 5: *Neural states and synaptic weight of the prediction neuron after 500 training examples.* (top) Somatic membrane potential dynamics of prediction neuron 0, where output spikes are superimposed as grey vertical bars. (Middle-top) Dendritic membrane potential, where blue and red bars indicate negative error neuron 0 spikes and positive error neuron 0 spikes, respectively. In the time range (500,750), the digit 0 is presented to the network. (Middle-bottom) Total synaptic current of prediction neuron 0, where superimposed vertical bars are presynaptic spikes of hidden neuron 32. The green shaded area (b_{min}, b_{max}) corresponds to the plasticity-enabled region, *i.e.* the approximate derivative function $\Theta(I^p(t))$. (Bottom) Synaptic weight between hidden neuron 32 and prediction neuron 0.

associated to the same class:

$$
C\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} V_i^P \\ U_i^P \end{pmatrix} = -\begin{pmatrix} g_V V_i^P \\ g_U U_i^P \end{pmatrix} + \begin{pmatrix} I_i^P + \sigma_w \eta_i^P(t) \\ w^E s_i^{E+}(t) - w^E s_i^{E-}(t) \end{pmatrix}. \tag{19}
$$

The spike trains at the data layer were generated using a stochastic neuron with instantaneous firing rate (exponential hazard function [Gerstner and Kistler, 2002] with absolute refractory period):

$$
\nu^d(d, t - t') = \begin{cases} 0 & \text{if } t - t' < \tau_{refr} \\ \tau_{refr}^{-1} \exp(\beta d + \gamma) & t - t' \ge \tau_{refr} \end{cases}, \tag{20}
$$

where d is the intensity of the pixel (scaled from 0 to 1), and t' is the time of the last spike. Although neurons with I&F neuron dynamics similar to the prediction and hidden neurons could be employed here, we assumed that data will be provided by external sensors in the form of spike trains that do not necessarily follow I&F dynamics. Fig. 5 illustrates the neural dynamics in a prediction neuron, in a network trained with 500 training samples (1/100 of an epoch).

5.2.1 Stochastic, blank-out Synapses and peRBP

In practice, we find that neurons tend to strongly synchronize in late stages of the training. The analysis provided above does not accurately describe synchronized dynamics, since one of the assumptions for the diffusion approximation is that spike times are uncorrelated. Multiplicative stochasticity was previously shown to be beneficial for regularization and decorrelation of spike trains, while being easy to implement in neuromorphic hardware [Neftci et al., 2016]. Following the ideas of synaptic sampling [Neftci et al.,

2016], we find that replacing the background Poisson noise with multiplicative, blank-out noise [Vogelstein et al., 2002] at the plastic synapses slightly improves the results and mitigates the energetic footprint of the stochasticity [Sheik et al., 2016b].

5.3 Quantized, Discrete-time dynamics

In order to demonstrate the effectiveness of eRBP in dedicated digital hardware with realistic constraints on precision and limited computations, we created a simulation of a digital neuromorphic learning core. The hardware simulator is a bit-accurate, fixed point emulation of a spiking neural network implemented on a digital hardware platform for ultra-efficient and flexible learning dynamics that is currently under development. Full details of the model and its architecture will be discussed elsewhere. Here, we describe the subset of the model dynamics necessary for peRBP. The dynamics of the neuron i implemented at the hidden layer is the following:

$$V_i^h[t+1] = V_i^h[t] - m(a_V \diamond V_i^h[t], V_i^h[t]) + a_{IV} \diamond I_i + b_V$$
$$U_i^h[t+1] = U_i^h[t] - m(a_U \diamond U_i^h[t], U_i^h[t]) + \sum_j g_U \diamond w_{ij}^E[t] s_j^E[t]$$
$$I_i^h[t+1] = I_i^h[t] - m(a_{syn} \diamond I_i^h[t], I_i^h[t]) + \sum_j g_I \diamond w_{ij}^h[t] \xi[t] s_j^h[t] + \sum_j g_I \diamond w_{ij}^d[t] \xi[t] s_j^d[t] \qquad (21)$$
$$\text{if } V_i^h[t+1] \geq V_T, s_i^h[t+1] \leftarrow 1 \text{ and } V_i^h[t+1] \leftarrow V_{reset}$$
$$\text{if } V_i^h[t+1] < V_T, s_i^h[t+1] \leftarrow 0$$

where the fourth and fifth line account for thresholds, resets, and spiking outputs s_i. $s_j^d[t]$, $s_j^h[t]$, $s_j^h[t]$, and $s_j^E[t] \in \{0, 1\}$ are the spiking output of the input (data), hidden, and error-coding neurons i at time t, respectively. Indices from $\xi[t]$ were dropped for clarity, and every instance of $\xi[t]$ in the equations above refers to an independent and identically distributed Bernouilli draw with probability p. The terms g_U and g_I are weight gain factors used to adjust the range of the synaptic weights.

Parameters a, including a_V a_U a_{syn}, and a_{IV} are integers implementing the coupling between and within the states V, U and I. The $\diamond$ operator is a custom bit shift that performs multiplication by powers of two and that can be implemented using only bitwise operations:

function $a \diamond x$
    if $a \geq 0$ **then return** $x \ll a$,
    else if $a < 0$ **then return** $sign(x)(|x| \gg -a)$,
    end if
end function

The reason for using $\diamond$ rather than left and right bit shifting is because integers stored using a two's complement representation have the property that right shifting by a of values such that $x > -2^{a'}, \forall a' < a$ is -1, whereas 0 is expected in the case of a multiplication by 2^{-a}. The $\diamond$ operator corrects this problem by modifying the bit shift operation such that $-2^{a'} \diamond a = 0, \forall a' < a$. Furthermore, such multiplications by powers of 2 have the advantage that fewer bits are required to *store* parameters on a logarithmic scale, which is a natural parametrization for such linear difference equations. A similar operation was used in the BNN [Courbariaux and Bengio, 2016] for an approximate power-of-two operation, although in our simulations, the first argument a is considered constant.

The function $m(\cdot, \cdot)$ defined as

function $m(x, y)$
    if $y \neq 0$ and $x = 0$ **then return** $sign(-x)$,
    else return x,
    end if
end function

ensures that all states leak towards zero in the absence of external drive.

Weight dynamics are implemented as:

$$w_{ij}^h[t+1] = \text{Clip}(w_{ij}^h[t] + \eta \diamond U_i^h[t]\Theta(I_i^h[t])s_j[t], -128, 128), \tag{22}$$

where Clip clips the weights higher than 128 and lower than -128. Dynamics for the prediction neurons were the same except that they reflected the connectivity of the output layer. Positive error neurons followed the following dynamics:

$$
\begin{aligned}
&V_i^E[t+1] = V_i^E[t] - m(a_V \diamond V_i^E[t], V_i^E) + (g_E \diamond w^L)(s_i^P(t) - s_i^L(t)) \\
&\text{if } V_i^E[t+1] \geq V_T^E, s_i^E[t+1] \leftarrow 1 \text{ and } V_i^E[t+1] \leftarrow V_i^E[t+1] - V_T^E \\
&\text{if } V_i^E[t+1] < V_T^E, s_i^E[t+1] \leftarrow 0,
\end{aligned}
\tag{23}
$$

As for continuous dynamics, negative error neurons follow the exact same dynamics with w^L of opposite sign and error neuron membrane voltages are lower bounded to zero.

For data neurons, input spike trains were generated as Poisson spike trains with rate γd, where d is the pixel intensity. For label neurons, input spikes were regular, *i.e.* spikes were spaced regularly with interspike interval τ_{refr}^{-1}

In the simulations used for eRBP, all states V, U and I and parameters were stored in 16 bit fixed point precision (ranging from -32768 to 32767), except for synaptic weights which were stored with 8 bit precision (ranging from -128 to 128) and coupling parameters were stored with 5 bits precision (from 0 to 32).

5.4   Experimental Setup and Software Simulations

We trained fully connected feed-forward networks MNIST hand-written digits, separated in three groups, training, validation, and testing (50000, 10000, 10000 samples respectively). During a training epoch, each of the training digits were presented in sequence during 250ms. We tested eRBP using two configurations: one with additive noise ($\sigma_w > 0$, $p = 1$, labeled eRBP), and one with multiplicative noise implemented as blank-out noise on the connections (blank-out probability $p < 1$ and $\sigma_w = 0$, labeled peRBP). To prevent the network from learning (spurious) transitions between digits, the synaptic weights did not update in the first 50ms window of each digit presentation.

We tested eRBP training on two different implementations: 1) Spiking neural network based on the Auryn simulator [Zenke and Gerstner, 2014], and 2) Hardware compatible simulator with quantized neural states and weights. Results are compared against GPU implementations of RBP and standard BP performed in Theano [Bergstra et al., 2010] using an equivalent, non-spiking neural network. Besides layered connectivity, all networks were unstructured (*e.g.* no convolutions or poolings).

References

Pierre Baldi and Peter J Sadowski. Understanding dropout. In *Advances in Neural Information Processing Systems*, pages 2814–2822, 2013.

N_d	Number of data neurons	all networks	784
N_h	Number of hidden neurons		Variable
N_l	Number of label neurons	all networks	10
N_{E+}	Number of positive error neurons	all networks	10
N_{E-}	Number of negative error neurons	all networks	10
N_p	Number of prediction neurons	all networks	10
σ	Poisson noise weight	eRBP	$50 \cdot 10^{-3}\,\text{nA}$
		peRBP	$0 \cdot 10^{-3}\,\text{nA}$
p	Blank-out probability	eRBP	1.0
		peRBP	.65
τ_{refr}	Refractory period	Prediction and hidden neurons	$3.9\,\text{ms}$
		Data neurons	$4.0\,\text{ms}$
τ_{syn}	Synaptic Time Constant	all synapses	$4\,\text{ms}$
g_V	Leak conductance state V	Prediction and hidden neurons	$1\,\text{nS}$
g_U	Leak conductance state U	Prediction and hidden neurons	$5\,\text{nS}$
C	Membrane capacitance	all neurons	$1\,\text{pF}$
V_T	Firing threshold	Prediction and Hidden neurons	$100\,\text{mV}$
V_T^E		Error neurons	$100\,\text{mV}$
N_{train}	Number of training samples used	all figures	50000
N_{test}	Number of training samples used	Tab. 1 eRBP, peRBP	10000
		Tab. 2 eRBP, peRBP	1000
		Tab. 2 RBP, BP	10000
T_{train}	Training sample duration	all models	$100\,\text{mV}$
T_{test}	Testing sample duration	Tab. 1,Fig. 3	$500\,\text{ms}$
		Tab. 2	$250\,\text{ms}$
w^h, w^d, w^p, g	Initial weight matrix	RBP, BP	$U(\sqrt{\frac{6}{\#rows+\#cols}})$
		eRBP	$U(\sqrt{\frac{6}{\#rows+\#cols}})\text{nA}$
		peRBP	$U(\sqrt{\frac{7}{\#rows+\#cols}})\text{nA}$
w^E		eRBP, peRBP	$90 \cdot 10^{-3}\text{nA}$
w^{L+}		eRBP, peRBP	$90 \cdot 10^{-3}\text{nA}$
w^{L-}		eRBP, peRBP	$-90 \cdot 10^{-3}\text{nA}$
b_{min}		eRBP, peRBP	$-1.15\,\text{nA}$
b_{max}		eRBP, peRBP	$1.15\,\text{nA}$
β	Data neuron input scale	eRBP, peRBP	.5
γ	Data neuron input threshold	eRBP, peRBP	$-.215$
T_{sim}	Simulation time per epoch	all models	$100\,\text{s}$
ϵ	Learning Rate	eRBP	$6 \cdot 10^{-4}$
		peRBP	$10 \cdot 10^{-4}$
		RBP, BP 1 hidden layer	.4
		RBP, BP 2 hidden layers	.5
n_{batch}	Batch size	RBP, BP	100

Table 2: Parameters used for the continuous-time spiking neural network simulation implementing eRBP.

$a_V, a_U, a_{syn}, a_{IV}$	State Couplings	Hidden and prediction neurons	$-3, -7, -6, 4$
g_V, g_U, g_I, g_E	Synaptic weight gain factors		$-3, -7, -6, 4$
V_T	Firing threshold	Prediction and hidden neurons	32797
V_T^E	Firing threshold	Error neurons	1025
b_V	Bias	Prediction and hidden neurons	1000
V_{reset}	Reset Voltage	Prediction and hidden neurons	32796
p	Blank-out probability	all plastic synapses	.6
η	Learning rate	all neurons	-10
τ_{refr}	Refractory period	Prediction and hidden neurons	39
		Data neurons	40
T_{sim}	Simulation time per epoch	all models	$90 \cdot 10^6$
T_{train}	Training sample duration	all models	1500
T_{test}	Testing sample duration	all models	3000
b_{min}	eRBP lower bound	all plastic synapses	-2560
b_{max}	eRBP upper bound	all plastic synapses	2560
β	Data neuron input scale	data neurons	25
n_{batch}	Batch size	All GPU (Theano) simulations	100

Table 3: Parameters used for the discrete-time spiking neural network simulation implementing eRBP.

Pierre Baldi, Peter Sadowski, and Zhiqin Lu. Learning in the machine: Random backpropagation and the learning channel. *arXiv preprint arXiv:1612.02734*, 2016.

J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, volume 4, 2010.

J. Brader, W. Senn, and S. Fusi. Learning real world stimuli in a neural network with spike-driven synaptic dynamics. *Neural Computation*, 19:2881–2912, 2007.

N. Brunel. Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons. *Journal of computational neuroscience*, 8(3):183–208, 2000.

N. Brunel and V. Hakim. Fast global oscillations in networks of integrate-and-fire neurons with low firing rates. *Neural Computation*, 11(7):1621–1671, 1999. ISSN 0899-7667.

Yongqiang Cao, Yang Chen, and Deepak Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, pages 1–13, 2014.

E. Chicca, F. Stefanini, and G. Indiveri. Neuromorphic electronic circuits for building autonomous cognitive systems. *Proceedings of IEEE*, 2013.

C. Clopath, L. Büsing, E. Vasilaki, and W. Gerstner. Connectivity reflects coding: a model of voltage-based STDP with homeostasis. *Nature Neuroscience*, 13(3):344–352, 2010.

Matthieu Courbariaux and Yoshua Bengio. Binarynet: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Low precision arithmetic for deep learning. *arXiv preprint arXiv:1412.7024*, 2014.

Srinjoy Das, Bruno Umbria Pedroni, Paul Merolla, John Arthur, Andrew S Cassidy, Bryan L Jackson, Dharmendra Modha, Gert Cauwenberghs, and Ken Kreutz-Delgado. Gibbs sampling with low-power

spiking digital neurons. In *Circuits and Systems (ISCAS), 2015 IEEE International Symposium on*, pages 2704–2707. IEEE, 2015.

G. Deco, V.K. Jirsa, P.A. Robinson, M. Breakspear, and K. Friston. The dynamic brain: From spiking neurons to neural masses and cortical fields. *PLoS Comput Biol*, 4(8):e1000092, 2008.

Peter U Diehl, Daniel Neil, Jonathan Binas, Matthew Cook, Shih-Chii Liu, and Michael Pfeiffer. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *International Joint Conference on Neural Networks (IJCNN),*, pages 1–8. IEEE, 2015.

C. Eliasmith and C.H. Anderson. *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. MIT Press, 2004.

S.K. Esser, R. Appuswamy, P. Merolla, J.V. Arthur, and D.S. Modha. Backpropagation for energy-efficient neuromorphic computing. In *Advances in Neural Information Processing Systems*, pages 1117–1125, 2015.

Steven K Esser, Paul A Merolla, John V Arthur, Andrew S Cassidy, Rathinakumar Appuswamy, Alexander Andreopoulos, David J Berg, Jeffrey L McKinstry, Timothy Melano, Davis R Barch, et al. Convolutional networks for fast, energy-efficient neuromorphic computing. *PNAS*, 113:11441–11446, 2016. doi: doi: 10.1073/pnas.1604850113.

S. Fusi and M. Mattia. Collective behavior of networks with linear (VLSI) integrate and fire neurons. *Neural Computation*, 11:633–52, 1999.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *arXiv preprint arXiv:1506.02142*, 2015.

Francesco Galluppi, Xavier Lagorce, Evangelos Stromatias, Michael Pfeiffer, Luis A Plana, Steve B Furber, and Ryad Benjamin Benosman. A framework for plasticity implementation on the spinnaker neural architecture. *Frontiers in Neuroscience*, 8:429, 2014.

Crispin W Gardiner. Handbook of stochastic methods. 2012.

W. Gerstner and W. Kistler. *Spiking Neuron Models. Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002.

Ian J. Goodfellow, David Warde-Farley, Pascal Lamblin, Vincent Dumoulin, Mehdi Mirza, Razvan Pascanu, James Bergstra, Frédéric Bastien, and Yoshua Bengio. Pylearn2: a machine learning research library. *arXiv preprint arXiv:1308.4214*, 2013. URL http://arxiv.org/abs/1308.4214.

M. Graupner and N. Brunel. Calcium-based plasticity model explains sensitivity of synaptic changes to spike pattern, rate, and dendritic location. *Proceedings of the National Academy of Sciences*, 2012. doi: 10.1073/pnas.1109359109. URL http://www.pnas.org/content/early/2012/02/21/1109359109.abstract.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

S. Grossberg. Competitive learning: From interactive activation to adaptive resonance. *Cognitive science*, 11(1):23–63, 1987.

Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 10–14. IEEE, 2014.

F. L. Maldonado Huayaney, S. Nease, and E. Chicca. Learning in silicon beyond stdp: A neuromorphic implementation of multi-factor synaptic plasticity with calcium-based dynamics. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 63(12):2189–2199, Dec 2016. ISSN 1549-8328. doi: 10.1109/ TCSI.2016.2616169.

Eric Hunsberger and Chris Eliasmith. Spiking deep networks with lif neurons. *arXiv preprint arXiv:1510.08829*, 2015.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

Dharshan Kumaran, Demis Hassabis, and James L McClelland. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in Cognitive Sciences*, 20(7):512–534, 2016.

Leon Bottou Yann Le Cun and Leon Bottou. Large scale online learning. *Advances in neural information processing systems*, 16:217, 2004.

Dong-Hyun Lee, Saizheng Zhang, Antoine Biard, and Yoshua Bengio. Target propagation. *arXiv preprint arXiv:1412.7525*, 2014.

Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. Training deep spiking neural networks using back-propagation. *Frontiers in Neuroscience*, 10, 2016.

Qianli Liao, Joel Z Leibo, and Tomaso Poggio. How important is weight symmetry in backpropagation? *arXiv preprint arXiv:1510.05067*, 2015.

Timothy P Lillicrap, Daniel Cownden, Douglas B Tweed, and Colin J Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, 7, 2016.

S.-C. Liu, M. Boegerhausen, and S. Pascal. Circuit model of short-term synaptic dynamics. In *Advances in Neural Information Processing Systems*, volume 15, Cambridge, MA, USA, December 2002. MIT Press.

Daniel Marti, Mattia Rigotti, Mingoo Seok, and Stefano Fusi. Energy-efficient neuromorphic classifiers. *arXiv preprint arXiv:1507.00235*, 2015.

C. Mead. Neuromorphic electronic systems. *Proceedings of the IEEE*, 78(10):1629–36, 1990.

Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, Andrew S Cassidy, Jun Sawada, Filipp Akopyan, Bryan L Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Hesham Mostafa. Supervised learning based on temporal coding in spiking neural networks. *arXiv preprint arXiv:1606.08165*, 2016.

Lorenz K Muller and Giacomo Indiveri. Neural sampling by irregular gating inhibition of spiking neurons and attractor networks. *arXiv preprint arXiv:1605.06925*, 2016.

E. Neftci, S. Das, B. Pedroni, K. Kreutz-Delgado, and G Cauwenberghs. Restricted boltzmann machines and continuous-time contrastive divergence in spiking neuromorphic systems, May 2013.

E. Neftci, S. Das, B. Pedroni, K. Kreutz-Delgado, and G. Cauwenberghs. Event-driven contrastive divergence for spiking neuromorphic systems. *Frontiers in Neuroscience*, 7(272), Jan. 2014a. ISSN 1662-453X. doi: 10.3389/fnins.2013.00272. URL `http://www.frontiersin.org/neuromorphic_engineering/10.3389/fnins.2013.00272/abstract`.

E. Neftci, S. Das, B. Pedroni, K. Kreutz-Delgado, and G. Cauwenberghs. Event-driven contrastive divergence for spiking neuromorphic systems. *Frontiers in Neuroscience*, 7(272), Jan. 2014b. ISSN 1662-453X. doi: 10.3389/fnins.2013.00272. URL `http://www.frontiersin.org/neuromorphic_engineering/10.3389/fnins.2013.00272/abstract`.

Emre Neftci, Charles Augustine, Somnath Paul, and Georgios Detorakis. Event-driven random back-propagation: Enabling neuromorphic deep learning machines. *arXiv preprint arXiv:1612.05596*, 2016.

B. Nessler, E. Pfeiffer, and W. Maass. Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity. *PLoS Computational Biology*, 2013.

Arild Nø kland. Direct feedback alignment provides learning in deep neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1037–1045. Curran Associates, Inc., 2016. URL `http://papers.nips.cc/paper/6441-direct-feedback-alignment-provides-learning-in-deep-neural-networks.pdf`.

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*, 7(178), 2013. ISSN 1662-453X. doi: 10.3389/fnins.2013.00178. URL `http://www.frontiersin.org/neuromorphic_engineering/10.3389/fnins.2013.00178/abstract`.

Peter O'Connor and Max Welling. Deep spiking networks. *arXiv preprint arXiv:1602.08323*, 2016.

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, Oct. 2014. URL `https://www.dropbox.com/s/mwex6dnytnjhew7/Park_etal14.pdf?dl=0`.

Bruno U Pedroni, Sadique Sheik, Siddharth Joshi, Georgios Detorakis, Somnath Paul, Charles Augustine, Emre Neftci, and Gert Cauwenberghs. Forward table-based presynaptic event-triggered spike-timing-dependent plasticity. Oct 2016. URL `IEEEBiomedicalCircuitsandSystemsConference(BioCAS),https://arxiv.org/abs/1607.03070`.

Mihai A Petrovici, Johannes Bill, Ilja Bytschok, Johannes Schemmel, and Karlheinz Meier. Stochastic inference with deterministic spiking neurons. *arXiv preprint arXiv:1311.3211*, Nov 2013.

Ning Qiao, Hesham Mostafa, Federico Corradi, Marc Osswald, Fabio Stefanini, Dora Sumislawska, and Giacomo Indiveri. A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses. *Frontiers in neuroscience*, 9, 2015.

Mengye Ren, Renjie Liao, Raquel Urtasun, Fabian H Sinz, and Richard S Zemel. Normalizing the normalizers: Comparing and extending network normalization schemes. *arXiv preprint arXiv:1611.04520*, 2016.

A. Renart, N. Brunel, and X. Wang. *Computational Neuroscience: A Comprehensive Approach*, chapter Mean field theory of irregularly spiking neuronal populations and working memory in recurrent cortical networks, pages 431–490. Chapman and Hall, Boca Raton, 2003.

David E Rumelhart, James L McClelland, PDP Research Group, et al. *Parallel distributed processing*, volume 1. IEEE, 1988.

Emilio Salinas and LF Abbott. Vector reconstruction from firing rates. *Journal of computational neuroscience*, 1(1-2):89–107, 1994.

S. Sheik, S. Paul, C. Augustine, C. Kothapalli, and G. Cauwenberghs. Membrane-dependent neuromorphic learning rule for unsupervised spike pattern detection. In *BioMedical Circuits and Systems, (BioCAS), 2016*. IEEE, Oct 2016a.

S. Sheik, S. Paul, C. Augustine, C. Kothapalli, G. Cauwenberghs, and E. Neftci. Synaptic sampling in hardware spiking neural networks. In *International Symposium on Circuits and Systems, (ISCAS), 2016*. IEEE, May 2016b. URL http://ieeexplore.ieee.org/abstract/document/7538991/.

R.S. Sutton and A.G. Barto. *Reinforcement learning: An introduction*, volume 28. Cambridge Univ Press, 1998.

Simon Thorpe, Arnaud Delorme, Rufin Van Rullen, et al. Spike-based strategies for rapid processing. *Neural networks*, 14(6-7):715–725, 2001.

Henry C Tuckwell. *Introduction to theoretical neurobiology: volume 2, nonlinear and stochastic theories*, volume 8. Cambridge University Press, 2005.

Robert Urbanczik and Walter Senn. Learning by the dendritic prediction of somatic spiking. *Neuron*, 81 (3):521–528, 2014.

R Jacob Vogelstein, Francesco Tenore, Ralf Philipp, Miriam S Adlerstein, David H Goldberg, and Gert Cauwenberghs. Spike timing-dependent plasticity in the address domain. In *Advances in Neural Information Processing Systems*, pages 1147–1154, 2002.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann L Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1058–1066, 2013.

X. Wang. Synaptic basis of cortical persistent activity: the importance of NMDA receptors to working memory. *J. Neurosci.*, 19:9587–9603, November 1999.

F. Zenke and W. Gerstner. Limits to high-speed simulations of spiking neural networks using general-purpose computers. *Frontiers in neuroinformatics*, 8, 2014.

Xian Zhu, Mihir Awatramani, Diane Rover, and Joseph Zambreno. Onac: Optimal number of active cores detector for energy efficient gpu computing. In *Computer Design (ICCD), 2016 IEEE 34th International Conference on*, pages 512–519. IEEE, 2016.