Spiking Neural Networks for Handwritten Digit Recognition – Supervised Learning and Network Optimization

Shruti R. Kulkarni[a], Bipin Rajendran[a,*]

[a] *Department of Electrical and Computer Engineering, New Jersey Institute of Technology, NJ, 07102 USA*

Abstract

We demonstrate supervised learning in Spiking Neural Networks (SNNs) for the problem of handwritten digit recognition using the spike triggered Normalized Approximate Descent (NormAD) algorithm. Our network that employs neurons operating at sparse biological spike rates below 300 Hz achieves a classification accuracy of 98.17% on the MNIST test database with four times fewer parameters compared to the state-of-the-art. We present several insights from extensive numerical experiments regarding optimization of learning parameters and network configuration to improve its accuracy. We also describe a number of strategies to optimize the SNN for implementation in memory and energy constrained hardware, including approximations in computing the neuronal dynamics and reduced precision in storing the synaptic weights. Experiments reveal that even with 3-bit synaptic weights, the classification accuracy of the designed SNN does not degrade beyond 1% as compared to the floating-point baseline. Further, the proposed SNN, which is trained based on the precise spike timing information outperforms an equivalent non-spiking artificial neural network (ANN) trained using back propagation, especially at low bit precision. Thus, our study shows the potential for realizing efficient neuromorphic systems that use spike based information encoding and learning for real-world applications.

*Corresponding author
Email address: bipin@njit.edu (Bipin Rajendran)

Keywords: Neural networks, spiking neurons, supervised learning, pattern recognition, approximate computing, neuromorphic computing

1. Introduction

The superior computational efficiency of biological systems has inspired the quest to reverse engineer the brain in order to develop intelligent computing platforms that can learn to execute a wide variety of data analytics and inference tasks [1]. Artificial neural networks (ANNs), inspired by the network architecture of the brain, have emerged as the state-of-the-art for various machine learning applications. In particular, inspired by the Nobel prize winning work of Hubel and Weisel on elucidating the mechanisms of information representation in the visual cortex [2], multi-layer convolutional neural networks have shown impressive performance for a wide variety of applications such as image recognition, natural language processing, speech recognition and video analytics [3, 4, 5, 6, 7, 8, 9, 10, 11].

Nevertheless, the neurons in ANNs implement a memoryless nonlinear transformation of the input synaptic signals to create real-valued output signals. This is vastly different from the behavior of neurons in the brain, which encode information in the timing of binary signals, called action potentials or spikes based on the timing of incoming spike signals from upstream nodes. The third generation of artificial neural networks, also called spiking neural networks (SNNs), have been introduced to mimic this key aspect of information processing in the brain [12]. There is growing evidence that SNNs have significant computational advantages as a result of their higher information representational capacity due to the incorporation of the temporal dimension [13, 14, 15, 16]. Furthermore, SNNs issue spikes sparsely - the observed spike rate in biological networks is in the range of 0.1 to 300 Hz - and they operate in an event-driven manner [17, 18, 19, 20]. Therefore, highly energy efficient neuromorphic systems can be realized in hardware based on SNNs, as is evidenced by recent demonstrations [21, 22, 23, 24, 25].

Earlier efforts to build learning algorithms for SNNs were inspired by recent discoveries from neuroscience that shed light on the synaptic (neuronal interconnections) mechanisms of adaptation based on the difference in the times of issue of pre- and post-synaptic spikes. The most prominent among them is the Remote Supervised Method (ReSuMe) [26], that adjusts the synaptic weights based on the precise timing differences of the input and output neurons, inspired by the spike timing dependent plasticity (STDP) rule. Other spike based learning algorithms that have been proposed include the SpikeProp algorithm (though it was restricted to single spike learning) [27], SPAN and PSD, which converted spikes to smoothened analog signals and defined a continuous time cost function for training [28, 29]. Another important spike based supervised learning rule was the Chronotron rule which used piece-wise gradient descent and was demonstrated to be efficient in identifying different classes of random spike trains [30]. Recently, the reward modulated STDP or R-STDP learning has shown superior performance on several benchmark problems compared to STDP SNNs and even traditional CNNs, even though training was limited to a single layer in the network [31]. A variant of ReSuMe algorithm, called the Delay Learning (DL)-ReSuMe, in addition to the synaptic weights, made use of the transmission delays of synapses interconnecting the neurons as parameters to train the network [32]. This algorithm has been shown to be superior in terms of accuracy and speed of convergence compared to the basic ReSuMe algorithm. The accurate synaptic efficiency adjustment method is another spike-error triggered supervised learning rule based on STDP, which optimizes a cost function defined in terms of membrane potential differences [33]. This method has been used to demonstrate excellent performance in several UCI datasets with few training parameters. The Synaptic Kernel Inverse Method (SKIM) [34], evaluates the weights analytically rather than learning them iteratively and has been applied to the problem of speech based digit recognition in a small network with 50 neurons. Based on the SKIM method, the convex optimized synaptic efficiencies (CONE) algorithm was developed [35] and was used for the problem of gait detection. The generalization capability of this algorithm and the

noise tolerance of a variation of the algorithm called CONE-R has also been demonstrated.

Our work focuses on applying a precise spike based supervised learning algorithm to the MNIST (Modified National Institute of Standards and Technology database) handwritten digit classification problem and optimizing the network in terms of the number of learning parameters for implementation in energy and memory constrained hardware.

In addition to the above mentioned learning methods, unsupervised learning algorithms for SNNs have also been explored, based on the biological spike timing dependent plasticity (STDP) rule [36, 37, 38, 39, 40, 41, 42]. While these networks use multi-layered convolution architectures with more than one million parameters and have achieved over 98% accuracy on the MNIST dataset [38, 39], we demonstrate similar accuracy with 13$\times$ fewer parameters.

There are also several efforts directed towards developing architectures with adaptive and evolving network structures [43, 44, 45, 46, 47]. SpikeTemp and SpikeComp are algorithms where neurons are progressively added in the classifier layer as the training algorithm approaches the optimal point [45, 46]. The recently developed evolving architecture called NeuCube, directly inspired by the brain [43], incorporates weight adjustments based on supervised and unsupervised rules and additionally, adds new network neurons as per training requirements.

Besides the above-mentioned approaches for designing learning algorithms for SNNs that operate directly in the spike domain, several authors have proposed to convert ANNs trained with the well-established backpropagation algorithm to SNNs so that the latter can be used as inference engines [48, 49, 50, 51, 52, 53]. ANN-to-SNN conversion imposes that the firing rate of a spiking neuron in the SNN be proportional to the activation output of a non-spiking neuron in the ANN. Various techniques such as approximating the response of a spiking neuron with a smooth differentiable ReLU-like function, weight normalization, noise addition, lateral inhibition or spiking rate based pooling masks, which is similar to max pooling operation, have been employed to this end.

Using these approaches, state-of-the-art inference accuracies have been demonstrated in spike domain equivalent of deep learning networks such as VGG-16 and Inception-V13 for ImageNet classification problem, and close to 2× reduction in the number of operations needed compared to CNNs for smaller problems such as MNIST and CIFAR-10 [53]. Recently, a more biologically plausible algorithm called the Feedback Alignment (FA) has been proposed, which unlike the standard backpropagation uses two different sets of weights in the feedforward and feedback paths [54]. This method has also been demonstrated in SNNs, using approximate differentiable functions of leaky integrate and fire (LIF) spiking neurons to train them in an online manner. However, the FA rule has lower performance compared to the standard backpropagation rule [52].

Towards the goal of demonstrating a learning SNN capable of high accuracy and efficiency, we use the recently proposed Normalized Approximate Descent (NormAD) algorithm to train the output layer weights of a three-layered network with fixed convolutional kernel weights in the hidden layer. This spike-triggered weight update rule frames the learning task as a supervised optimization problem aimed at tuning the membrane potential to create spikes at desired time instants. Compared to other deterministic learning algorithms in the spike domain such as ReSuMe, at least 10× faster convergence characteristics have been demonstrated using this algorithm for generating arbitrarily desired spike streams [55].

Prior SNN based demonstration of handwritten digit recognition using spiking versions of backpropagation of errors has achieved 98.7% based on a fully connected $4-$layer network and 99.31% with convolutional spiking networks, but also with more than 4× higher number of trainable synapses compared to our network [56]. The training algorithm employed in that work has a cost function that is continuous in time defined in terms of the low pass filtered spike trains (both input and output). Compared to the state-of-the-art networks which have shown over 99% accuracy, our SNN trained with NormAD shows an accuracy of 98.17% on the test set of the MNIST database, with 4× fewer synaptic learning parameters [3, 4, 11, 56]. Furthermore, if the network architecture and number

of synaptic parameters are kept the same, we show that the accuracy and performance of the NormAD trained SNN is slightly better than that of an equivalent ANN trained using backpropagation.

This paper is organized as follows. We introduce the basic units of SNNs in Section 2. Section 3 describes the architecture of our network, the spike encoding at the input and output of the network, and the training algorithm used for weight updates. Section 4 describes several hyper-parameter tuning experiments and the results achieved on the MNIST database. Section 5 discusses the optimization of the network for implementation in energy and memory constrained hardware platforms by approximating the neuronal dynamics and using low-precision bits for storing the synaptic weights. Finally, section 6 summarizes the key conclusions of our work.

2. Spiking Neural Networks

SNNs are the third generation of neural networks employing neuron models that are inspired by the biological mechanisms of neuronal signaling. While the mechanism of spike process in biological neurons depends on complex interactions of ion-channels on the cell membrane, a computationally simpler leaky integrate and fire (LIF) model is typically used for simulation of spiking neural networks [57]. This model represents the potential of a neuron as the voltage across a capacitor connected in parallel with a leaky conductance path, and is charged by incoming input currents. Accordingly, the membrane potential $V(t)$ evolves according to the differential equation:

$$C\frac{dV(t)}{dt} = -g_L(V(t) - E_L) + I_{syn}(t). \tag{1}$$

When $V(t) \geq V_T$, a spike is issued and transmitted to the downstream synapses; the membrane potential is reset to its resting value E_L after the spike. We use $E_L = -70\,\text{mV}$ and $V_T = 20\,\text{mV}$ in our simulations. $C = 300\,\text{pF}$ and $g_L = 30\,\text{nS}$ model the membrane's capacitance and leak conductance, respectively. Biological neurons enter a refractory period immediately after a spike is issued

during which another spike cannot be issued. We implement this by holding the membrane potential at $V(t) = E_L$ for a short period $t_{ref} = 3\,\text{ms}$ after the issue of a spike. We also limit the membrane potential to the range $[E_L, V_T]$ through clipping.

The spikes arriving at a synapse having a strength (weight) w, will generate a post-synaptic current ($I_{syn}(t)$) in its downstream neuron, given by the expressions:

$$c(t) = \sum_i \delta(t - t^i) * \left(e^{-t/\tau_1} - e^{-t/\tau_2}\right) \quad (2)$$

$$I_{syn}(t) = w \times c(t). \quad (3)$$

where t^i denotes the time of issue of the i^{th} incoming spike and $*$ is the convolution operator. The variables $\tau_1 = 5\,\text{ms}$ and $\tau_2 = 1.25\,\text{ms}$ model the shape of the synaptic current kernel $c(t)$ and denote its falling and rising time constants, respectively. Note that the time of issue of spikes of a LIF neuron depends on the incoming spike times and synaptic strength in a strong nonlinear fashion, due to the weighted summation, integration and reset.

3. Network Architecture

As illustrated in Fig. 1, we designed a simple 3-layer SNN for classification of handwritten digits from the MNIST database. Since MNIST images are 28×28 pixels, our network's input layer has 784 neurons and the output layer has 10 neurons, each corresponding to a particular digit. The input layer neurons connect to 8112 hidden layer neurons through twelve *a priori* fixed 3×3 sized convolutional kernels. The synapses connecting this hidden layer to the output layer are trained using the NormAD algorithm.

3.1. Input encoding

Biological sensory neurons employ complex transformations such as rate coding, time-of-spike coding, population coding and phase coding to encode real-world information in the spike domain [58]. Time-encoding machines that

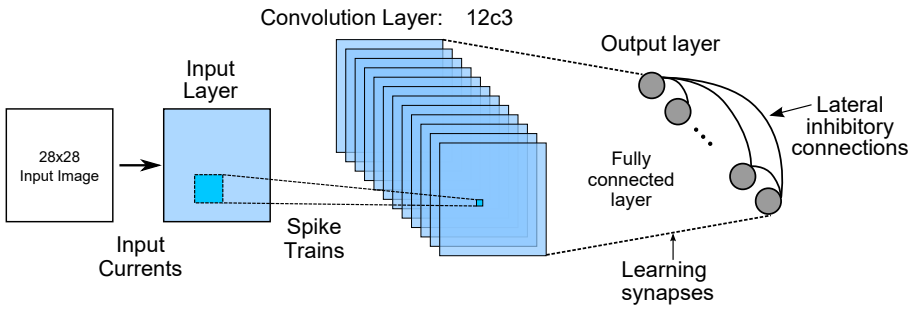

Figure 1: The proposed spiking neural network architecture for handwritten digit classification. The spike trains from the input layer with 28×28 neurons are spatially convolved with twelve filters (or convolution kernels) of size 3×3, resulting in the twelve feature maps of size 26×26. The synapses connecting the 8112 convolution layer neurons and the 10 output layer neurons are tuned during training. There is a fixed winner-take-all (WTA) lateral inhibition between the neurons in the output layer.

convert band-limited input signals to the spike domain such that their perfect reconstruction is possible have been proposed in [59]. There are also some recent works that use Gaussian receptive fields or Poisson encoding to directly translate real-valued inputs to spike times [60, 40]. As we are dealing with static images, we translate each gray-scale pixel value, in the range $[0, 255]$, to currents that can be applied as inputs to the spiking neurons. Accordingly, each pixel value k is converted into a constant input current for the LIF neuron as:

$$i(k) = I_0 + (k \times I_p)\,. \tag{4}$$

where $I_p = 101.2\,\text{pA}$ is a scaling factor and $I_0 = 2700\,\text{pA}$ is the maximum constant amplitude current that does not generate a spike in the LIF neuron in equation 1. As a result, a LIF neuron in the input layer issues spikes that are uniformly spaced in time, with a frequency that is sub-linearly proportional to the magnitude of its input current [61].

3.2. Convolutional feature extraction

The convolution layer of our network uses *a priori* determined fixed weights for the different feature maps and serves to detect the key features of the image.

The filter kernels are continuous curves as shown in Fig. 2(left), and incorporate both excitatory and inhibitory connections. Our kernels are only 3×3 pixels and were inspired by biological studies that suggest that the first few layers of the visual cortex consist of small-sized visual receptive fields [2].

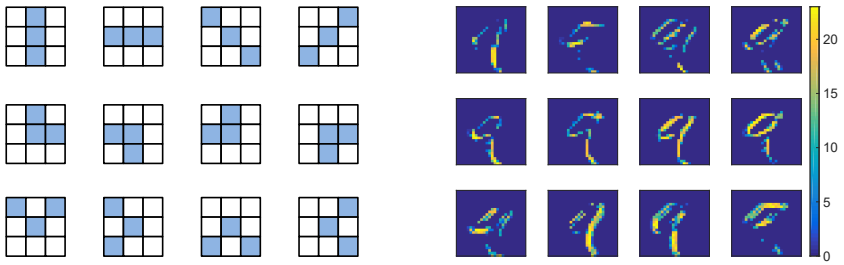

Figure 2: (left) Convolution filters used in our SNN are of size 3×3 pixels. The blue pixels are the excitatory weights, while white pixels are inhibitory values. The magnitude of the excitatory weight is 1.6 times that of the inhibitory weight. (right) The twelve spike count feature maps corresponding to these filters obtained when an exemplary image of digit '9' was presented to the network. The color intensities in the 2D map depict the number of spikes generated by the neurons of the hidden layer when the input was presented for $T = 100\,\text{ms}$.

The filter kernels are spatially convolved with 28×28 spike trains arriving from the input layer neurons, over a simulation period T, with a stride of 1, resulting in feature maps of size 26×26. The weight kernels have an overall net higher inhibition than excitation, as it helped to better suppress the spikes from unwanted edges of the input digit image in the corresponding feature map. Fixed weights based on Gabor filters have been used before as the first layer in a deep convolution neural network, and have shown an improvement in the accuracy for the MNIST dataset compared to the original LeNet-5 network [3, 62]. We use relatively simpler edge detection filters in the hidden layer of our network.

The spikes from the input layer neurons pass through these synaptic weight kernels to generate currents to the hidden layer neurons. The magnitude of the current entering the hidden layer neurons is scaled such that on an average their output spike rate is limited to 10 Hz. Fig. 2(right) shows the 2D feature maps depicting the number of spikes generated by the neurons in the hidden layer

when an exemplary image of digit 9 from the MNIST data-set is presented to the network for $T = 100\,\text{ms}$. The different kernels are able to effectively encode the edges and features of the input image in spike domain.

3.3. Learning layer

The synaptic weights connecting the hidden layer to the output classifier layer are trained using the NormAD algorithm [55]. The weights are initialized to zero at the beginning of training. Weights of all the 8112×10 synapses in this fully-connected layer of the network are updated at the end of presentation of each image, which lasts for the interval T, as:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \Delta\mathbf{w}. \tag{5}$$

The weight update, $\Delta\mathbf{w}$ is calculated only when there is a discrepancy between the spike times in the desired ($S^d(t)$) and observed ($S^o(t)$) spike trains, $e(t) = S^d(t) - S^o(t)$. As described in [55], this is achieved by defining a cost function in terms of the error between the desired ($V_{des}(t)$) and observed ($V(\mathbf{w}, t)$) neuron membrane potentials as:

$$J(\mathbf{w}) = \frac{1}{2}\int_0^T |e(t)|(V_{des}(t) - V(\mathbf{w}, t))^2 dt \tag{6}$$

Using gradient descent on the instantaneous cost $J(\mathbf{w}, t)$ obtained by restricting the limits of integral in equation 6 to an infinitesimally small interval around time t, the instantaneous weight update term can be written as:

$$\Delta\mathbf{w}(t) = \eta(t)\nabla_{\mathbf{w}} J(\mathbf{w}, t) \tag{7}$$

with

$$\nabla_{\mathbf{w}} J(\mathbf{w}, t) = |e(t)|(V_{des}(t) - V(\mathbf{w}, t))\nabla_{\mathbf{w}} V(\mathbf{w}, t) \tag{8}$$

$\eta(t)$ is a time dependent proportionality constant in equation 7. By *normalizing* and *approximating* the dependence of membrane potential on the weights, it is possible to obtain a closed form relationship for the weight update as:

$$\Delta\mathbf{w} = r\int_0^T e(t)\frac{\hat{\mathbf{d}}(t)}{\|\hat{\mathbf{d}}(t)\|}dt \tag{9}$$

where,

$$\hat{\mathbf{d}}(t) = \mathbf{c}(t) * \hat{h}(t), \text{with } \hat{h}(t) = \exp(-t/\tau_L)u(t). \tag{10}$$

Here, $c(t)$ is the synaptic kernel as described in equation 2 and $u(t)$ is the Heaviside step function. The constant $\tau_L = 1\,\text{ms}$ represents the approximation for the neuronal time constant, during training phase. Normalization helps in eliminating the dependency on $V_{des}(t)$, which is an unknown term. The weight update depends only on the output spike error $e(t)$ and the incoming spike trains, captured in $\hat{d}(t)$. The constant r, having the dimensions of synaptic conductance, is a function of the number of input neurons, and is set to 200 pS for our network with 8112 incoming synapses per output neuron.

In our network, the desired signal $S^d(t)$ for the label neuron is a uniform spike train with a frequency of 285 Hz, corresponding to a spike every 3.5 ms, which is slightly higher than the LIF refractory period of 3 ms. There are no spikes in the $S^d(t)$ for all the other neurons.

3.4. Lateral inhibition at the output layer

In addition to the feed-forward inputs from the convolution layer neurons, each output layer neuron also receives lateral inhibitory inputs from the remaining 9 output neurons, implement winner-take-all (WTA) dynamics, similar to [56]. When a neuron spikes, its outgoing WTA synapses inject a negative current to other neurons, thereby suppressing their spikes, as illustrated in Fig. 3.

3.5. Training methodology

During training, each image is presented to the network for a duration T and all the output layer weights are updated after every image, similar to a stochastic gradient descent (SGD) rule. We divide the MNIST training set into two parts: $50,000$ for training and remaining $10,000$ for validation. In each training epoch, all the $50,000$ images are presented once to the network. All the neurons' membrane potentials are initialized to their resting value of $E_L = -70\,\text{mV}$ and the synaptic current variables are cleared at the beginning of each simulation.

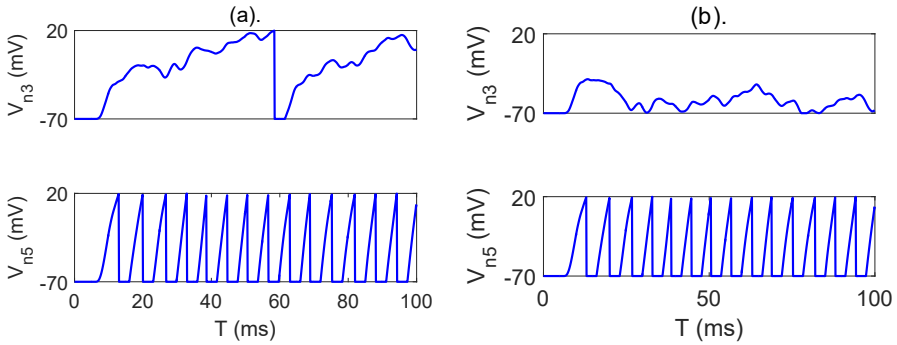

Figure 3: Membrane potential of two output layer neurons '3' and '5', when an input image of digit '5' was presented to the network. (left) Membrane potential without lateral inhibition and (right) with lateral inhibition. It can be seen that lateral inhibition has suppressed the incorrect neuron '3' from issuing a spike.

The dynamics of the SNN is evaluated by numerical integration with a time-step of $\Delta t = 0.1\,\text{ms}$ which is 10 times smaller than the learning time constant, $\tau_L = 1\,\text{ms}$ used in the NormAD algorithm (see section 3.3). The validation set is used to tune the hyper-parameters of the network such as the variation in the learning rate, optimal number of convolution kernels and the presentation duration as discussed in the following subsections. The network accuracy was determined on the MNIST test set consisting of $10,000$ images. The details of the GPU implementation of the algorithm are available in the supplementary material.

4. Results

We now discuss the results of various experiments that we conducted in our study to optimize the performance of our network. We start with the baseline experiments that were conducted to analyze network performance, and then discuss the sensitivity of the network to signal encoding parameters such as image presentation duration, learning rate schedules and the network size.

4.1. Accuracy metrics in spike domain

We primarily used two metrics to measure the accuracy of our network – the first based on the spike count and the second based on the correlation C, of the observed spike trains with respect to a reference spike train. In the count metric, the network's output is decided based on the neuron having the highest spike count. The spike correlation measure [63] between the output spike train $S_i^o(t)$ for each neuron i in the output layer and a reference spike train $S^r(t)$ is defined as:

$$C_i = \frac{\langle L[S_i^o(t)], L[S^r(t)] \rangle}{\|L[S_i^o(t)]\| \, \|L[S^r(t)]\|} \tag{11}$$

where

$$L[S(t)] = S(t) * \exp(-t/\tau)u(t). \tag{12}$$

Here $\langle \mathbf{x}, \mathbf{y} \rangle$ represents the dot product of vectors $\mathbf{x}$ and $\mathbf{y}$. The training signal with a frequency $f_{out} = 285\,\text{Hz}$ is also used as the reference signal during inference. The neuron with the highest value of C is declared the winner of the classification.

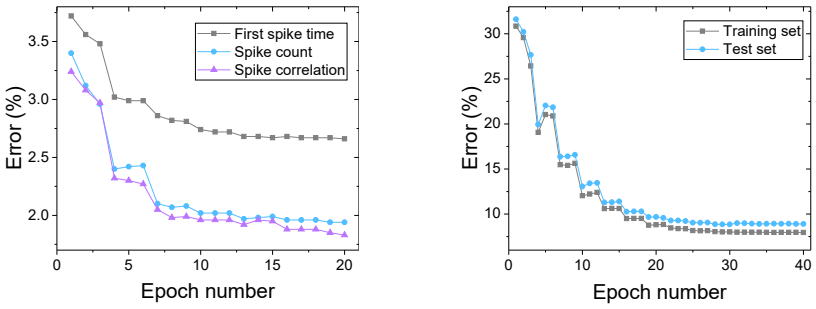

Figure 4: (left) The 3-layer SNN error on the MNIST test data-set based on the count, correlation and first-spike-time metrics. It can be seen that the network classification error in terms of first neuron to spike (in gray) during the presentation interval T, is worse by almost 1% compared to either count (blue) or the correlation metric (magenta). (right) For a 2-layer SNN without the hidden layer, the error saturates to about 8%, even at 40 epochs of training, illustrating the importance of the hidden layer.

The SNN is trained on the MNIST training set for 20 epochs. It can be seen from Fig. 4 (left) that precise timing of spikes measured using the correlation

metric gives a slightly higher accuracy for classification, though the spike count metric is a simpler metric to evaluate. The classification accuracy of the network is reported using the correlation metric for the succeeding sections in this paper, with explicit mention of the count metric whenever it is used. We also considered the classification accuracy based on the output neuron that spiked first during the input presentation. However, the accuracy based on this metric at the end of 20 epochs was only about 97.34%. While there is a significant drop in accuracy compared to the correlation and spike count metrics, the prediction can be made within 20 ms of image presentation in 99% of input samples using the first-to-spike metric. This trade-off between latency and accuracy may be especially attractive for low-power approximate computing applications.

We also note the crucial role the convolutional hidden layer plays in improving the network accuracy – in a 2-layer network with the 784 input neurons connected directly to the 10 output layers, the network's error saturates around 8% (Fig. 4(right)).

4.2. Learning rate schedule optimization

As discussed in [55], the optimal learning rate for the NormAD algorithm depends on the number of input neurons, N_{inp} and scales according to a $N_{inp}^{-1/2}$ rule. We studied several protocols (learning rate schedules) to decrease the learning rate during training (Table 1), which resulted in lowering the network error by nearly 0.5% (Fig. 5).

Epoch dependent learning rate schedules have shown accuracy improvement in previous works for ANN training [3, 64, 56]; in our study, we experimented with these and several other schedules, shown in the Table 1. We use schedule 5 which gave the best validation error after convergence, for the rest of experiments in the paper.

4.3. Network parameter optimization

We also optimized the design parameters of the network such as the number of the convolution kernels used in the hidden layer and the time period T used

Table 1: Learning rate schedules

Scheme	Learning rate (pS)
Schedule 1	$r_0 = 200$, constant over all epochs, n
Schedule 2	$(1/n)$ decrease: $r(n) = \frac{r_0}{(1+k\times n)}$
Schedule 3	Exponential decrease: $r(n) = r_0 \exp(-k \times n)$
Schedule 4	Step decrease by half every 5 epochs
Schedule 5	Step decrease by half every 3 epochs

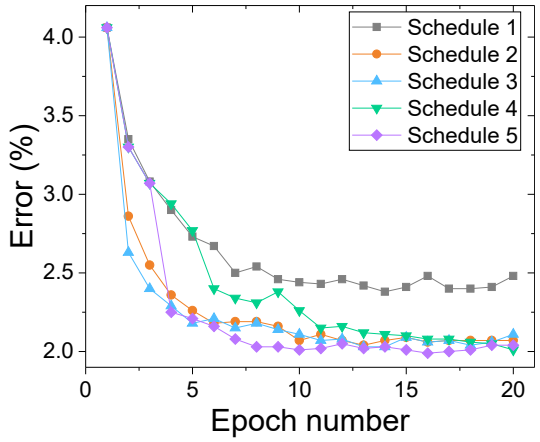

Figure 5: Network error on the validation set for five different rate schedules listed in Table 1.

for presenting each input image to the network. Increasing T results in longer integration time to learn the features of each image, as more spikes (or error points) are produced, resulting in a larger magnitude for the weight update. However, from the perspective of improving the throughput for network performance and preventing over-fitting, smaller values of T are more desirable. Fig. 6 shows the network performance as a function of the number of convolution kernels and the presentation duration T for the images. The network accuracy is optimized with 12 kernels and a presentation duration of $T = 100\,\text{ms}$. We used a constant inhibitory WTA synaptic strength of 1 nS for all connections in the output layer.

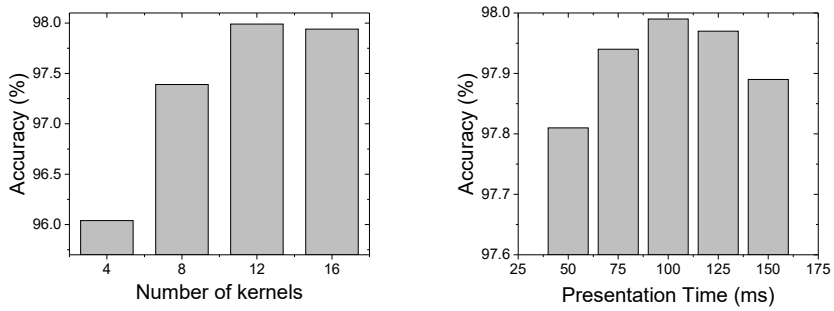

Figure 6: (left) Classification accuracy on the MNIST test set as a function of the number of convolutional kernels; (right) the presentation duration, T. The network accuracy is optimized with 12 kernels and a presentation duration of $T = 100\,\text{ms}$.

4.4. MNIST accuracy results

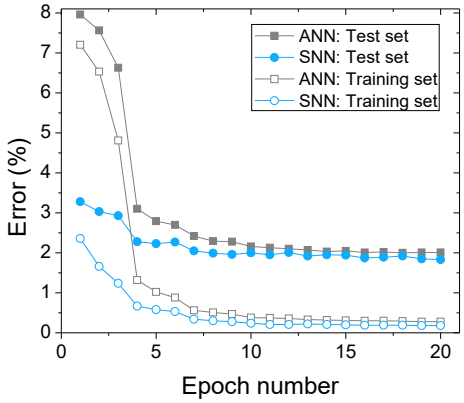

Figure 7: Comparison of the MNIST error for the 3-layer SNN and an equivalent ANN with the same network structure during 20 epochs of training. The SNN performance (0.18% error for training set and 1.83% error for test set at convergence) is slightly better than that of the ANN (0.28% error for training set and 2.0% for test set at convergence).

Having optimized the network hyper-parameters, we trained our SNN with the complete MNIST dataset ($60,000$ images) for 20 epochs. The SNN achieved an accuracy of 99.82% on the MNIST training set and 98.17% on the test set.

We also trained an equivalent ANN with the same architecture, i.e., the

same number of neurons and connectivity patterns (but without the lateral WTA connection) as the SNN in Fig. 1. We used the rectified linear unit (ReLU) as the activation function of the neurons in this network. The weights of the fully-connected layer were adjusted by the standard gradient descent rule by back-propagating the network error. After fine-tuning the learning rate schedule, this ANN achieved an accuracy of 98.0% on the MNIST test set, which is close to the best case accuracy of around 98.50% reported on an equivalently sized three-layered ANN [65]. The performance for training and test sets for the SNN and ANN networks for 20 epochs of training is shown in Fig. 7. This comparison shows that SNNs trained using the NormAD algorithm can obtain performance similar to equivalent ANNs in benchmark classification problems.

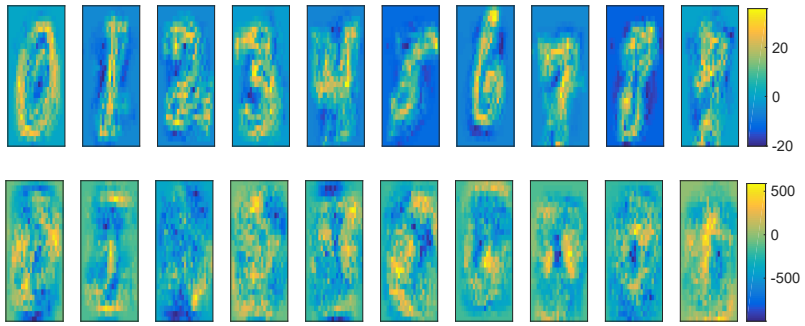

Figure 8: The average of the trained weights (in pS) from the 12 kernels in the hidden layer to the 10 neurons in the output layer is the effective internal representation of the digits learned by the network. (Top) The average weights in the output layer of the SNN after 100 images presented once for training (when the test set accuracy was only 65.8%) and; (Bottom) average weights after training (i.e., with 98.17% accuracy).

Fig. 8 shows the average of the trained weights of the synapses from the 12 feature maps to each of the 10 output neurons of SNN. When the network is trained on the first 100 images, the weight maps closely resemble the images of the training set digits, though the test set accuracy using these weights was only about 65.8%. When the network is trained with all the $60,000$ images in the training set, the test set accuracy rises to 98.17%, thanks to a more complex representation of the images that are captured by the synaptic weights in the

network.

Table 2: MNIST classification accuracy comparison - our network architecture achieves over 98% accuracy with atleast four times fewer parameters than the state-of-the-art networks.

Network and learning algorithm (BP stands for back-propagation)	Number of learning synapses	Test set Accuracy
ANN (LeNet-5) [3]	$331,984$	99.05%
GCNN (LeNet-5 + Gabor filters) [62]	$331,984$	99.32%
MCDNN (Multi-column Deep NN) [4]	$1,574,600$	99.77%
DNN with DropConnect [66]	$2,508,470$	99.79%
SNN, with STDP [40]	$5,017,600$	95.0%
Deep SNN with STDP [38]	$5,875,456$	98.40%
Fully connected SNN, with BP [56]	$328,984$	98.77%
Convolution SNN with BP[56]	$581,520$	99.31%
Spiking ConvNet [49]	$1,422,848$	99.11%
SNN, with NormAD (this work)	$81,120$	98.17%
ANN, with BP (this work)	$81,120$	98.0%

To benchmark the classification performance of our network, we compare the accuracy and number of learning synapses in other state-of-the-art approaches for MNIST handwritten digit classification (Table 2). We note that while the accuracy of our approach is about 1.6% worse than the best in class approach, our network achieves this accuracy with four to twenty times fewer number of trainable synaptic weights.

Table 3 presents the confusion matrix for the SNN based classification of the MNIST test data-sets into 10 classes. It can be seen that for all the digits, the true positive rate is 97% and above, demonstrating the high selectivity of the classifier layer, even though this is not easily discernible from the weight maps (Fig. 8). Only five images failed to elicit any spikes in the output neurons.

Table 3: Confusion matrix for the SNN's predicted output shows high selectivity of the NormAD trained classifier layer for each digit.

Actual Predicted	0	1	2	3	4	5	6	7	8	9
0	973	0	3	0	2	2	9	1	4	4
1	0	1126	1	0	0	0	2	4	0	4
2	2	3	1015	4	1	1	0	9	1	1
3	0	2	0	996	0	7	1	1	6	4
4	0	1	2	0	964	0	1	1	5	7
5	0	1	0	6	0	876	3	0	1	3
6	2	1	1	0	5	3	940	0	1	0
7	1	1	6	2	0	1	0	1005	3	7
8	1	0	1	1	1	2	1	3	947	3
9	0	0	2	1	9	0	0	3	6	975
No spike	1	0	1	0	0	0	1	1	0	1
Total	980	1135	1032	1010	982	892	958	1028	974	1009

5. Network optimization

We now discuss the network optimization studies to translate the software design for energy and memory constrained hardware platforms.

5.1. Low precision weight encoding

The ability of a network to maintain its accuracy even when the precision for storing the network parameters is limited, is crucial for efficient hardware implementations. It has been observed that accuracy degrades significantly when low-precision weights are used for network emulation. For instance, a 5% drop in accuracy (with the MNIST data-set) was observed even with 5-bits of fixed-point precision for the synaptic weights in [67].

We test the ability of our SNN and ANN for inference as a function of the precision of trained weights. We train the weights of both these networks in double-precision and then measure the inference accuracy by quantizing these weights, similar to the approach taken in [68] for designing a scalable hardware solution. The histograms of the weights of our SNN and ANN after training with NormAD and gradient descent, respectively, are observed to be log-normally distributed. Our quantization studies showed that dividing the range of weights

into linear bins, rather than log-linear bins gives lesser degradation in performance. Fig. 9 shows the drop in accuracy for our networks as the number of levels for representing the trained weights are reduced. It can be seen that even at 3-bit quantization, the degradation in SNN accuracy is within 1.0% for $T = 100\,\text{ms}$ compared to the floating point baseline. Further, across all quantization values, the degradation in accuracy of the ANN is slightly worse than that of the spiking network. It is also worth pointing out that compared to previous reports such as [67], where the input spike rate was as high as 1500 Hz, the firing rate in our SNN is in the range of 10 to 300 Hz, which is closer to the observed biological spike rates. These results hence demonstrate the robustness of the SNN architecture and its suitability for memory constrained hardware platforms.

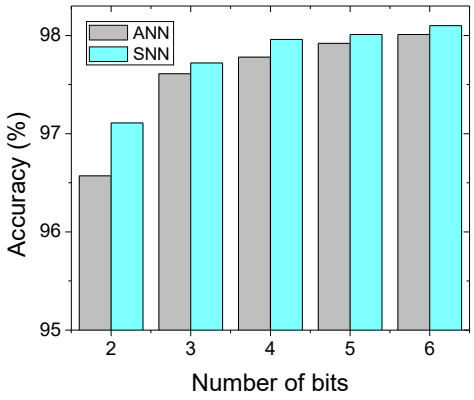

Figure 9: Test accuracy as a function of the precision of the trained weights in the SNN and ANN. Even at 2-bit precision, the SNN accuracy is only about 1% worse than the floating point baseline. Further, the SNN accuracy is better than the corresponding ANN especially at low bit-precision.

5.2. Approximating neuronal dynamics

We also study the SNN's performance when the dynamics of the neurons is evaluated with lower precision. As mentioned in the section 3.5, the time step

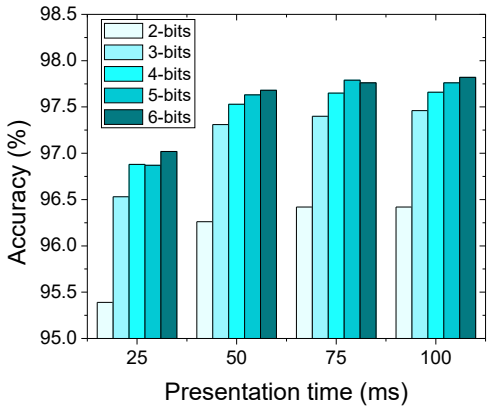

Figure 10: MNIST test accuracy (count metric) as a function of bit-precision of weights and the presentation time T, when the neuronal dynamics is approximated with a larger integration time step of 1 ms. Even at 3-bits of precision and with $T = 50$ ms, the drop in accuracy is within 1% of the baseline.

for numerical integration was chosen to be 0.1 ms for learning. Even though there will be some error in the precise time of spike issue, a larger time step can be used when the network is used for inference.

With $\Delta t = 1$ ms, the neuronal response can be calculated $10\times$ faster; Fig. 10 shows the test accuracy as a function of bit-precision and presentation times for the 3-layer SNN. Here, we used the count metric to determine the test accuracy to simplify the computation further. At a bit-precision of 3-bits, the digit identification can be completed in just 50 ms or with 50 points of neuronal integration with an accuracy of 97.31%. Hence, close to base-line accuracies can be maintained in approximate network evaluation that permits higher throughput for classification.

6. Conclusion and Future Work

We presented a highly compact and efficient 3-layer spiking neural network for identifying handwritten digits, that achieved an accuracy of 98.17% on the MNIST data set using the NormAD learning algorithm. All information in the

network is encoded and processed in the spike domain at sparse biological spike rates. Our studies show that using the precise time of spike issue for classification gives slightly better accuracy compared to the simpler rate coding method. We have also presented two techniques to co-optimize the network for hardware implementation, by reducing the bit-precision of weights and approximating the neuronal dynamics with higher integration time-step size.

The best convolution networks in both spiking and non-spiking versions that have achieved over 99% accuracy on the MNIST database use at least over $300,000$ adjustable synapses. The NormAD-trained SNN, on the other hand, has $4\times$ fewer learning parameters, making it amenable for implementation on custom neuromorphic hardware with on-chip learning. Our studies also show that as low as 3-bits of weight precision is sufficient to maintain close to baseline accuracies in the SNN when used for inference. Compared to an equivalent ANN with similar network architecture, the spike based training approach also shows better accuracy, especially at lower precision for synaptic weight storage.

The NormAD weight update rule as used in this study can be applied only for tuning the strength of synapses connected to the output layer of a network. However, the methodology used to derive this rule can be extended to adjust the weights of networks with hidden layers in a spike-triggered manner, based on the chain rule of derivatives. Such weight update rules could be then used to pre-train autoencoders which could be stacked and trained to develop deep spiking networks, following the approaches used in deep learning today [5]. Quantifying the performance of such deep spiking networks and determining their accuracy-efficiency trade-offs for large benchmark classification problems is identified as a topic for future exploration.

Acknowledgment

The authors acknowledge valuable discussions with Prof. Osvaldo Simeone at NJIT and Navin Anwani and Bharath Sastha from IIT Bombay. We also gratefully acknowledge the comments of the anonymous reviewers which helped

improve the paper.

This research was supported in part by the CAMPUSENSE project grant from CISCO Systems Inc; the Semiconductor Research Corporation; and the National Science Foundation grant 1710009. Resources of the High-Performance Computing facility at NJIT was used in this work.

References

[1] National academy of engineering - Reverse-engineer the brain, available at `http://bit.ly/1PmsLiX` (2009).

[2] D. H. Hubel, T. N. Wiesel, Receptive fields and functional architecture of monkey striate cortex, The Journal of physiology 195 (1) (1968) 215–243.

[3] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324. `doi:10.1109/5.726791`.

[4] D. Ciregan, U. Meier, J. Schmidhuber, Multi-column deep neural networks for image classification, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 3642–3649. `doi:10.1109/CVPR.2012.6248110`.

[5] G. E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural computation 18 (7) (2006) 1527–1554.

[6] A. Krizhevsky, I. Sutskever, G. E. Hinton, ImageNet Classification with deep convolutional neural networks, in: F. Pereira, C. J. C. Burges, L. Bottou, K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems 25, Curran Associates, Inc., 2012, pp. 1097–1105.

[7] Y. Goldberg, A primer on neural network models for natural language processing., J. Artif. Intell. Res.(JAIR) 57 (2016) 345–420.

[8] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, B. Kingsbury, Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups, IEEE Signal Processing Magazine 29 (6) (2012) 82–97. doi:10.1109/MSP.2012.2205597.

[9] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, L. Fei-Fei, Large-scale Video Classification with Convolutional Neural Networks, in: 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1725–1732. doi:10.1109/CVPR.2014.223.

[10] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning.
URL http://www.deeplearningbook.org

[11] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, Y. Bengio, Maxout networks, in: S. Dasgupta, D. McAllester (Eds.), Proceedings of the 30th International Conference on Machine Learning, Vol. 28 of Proceedings of Machine Learning Research, PMLR, Atlanta, Georgia, USA, 2013, pp. 1319–1327.
URL http://proceedings.mlr.press/v28/goodfellow13.html

[12] W. Maass, Networks of spiking neurons: The third generation of neural network models, Neural networks 10 (9) (1997) 1659–1671, = http://www.sciencedirect.com/science/article/pii/S0893608097000117. doi:http://dx.doi.org/10.1016/S0893-6080(97)00011-7.

[13] R. Gutig, H. Sompolinsky, The tempotron: a neuron that learns spike timing-based decisions, Nat Neurosci 9 (3) (2006) 420–428. doi:10.1038/nn1643.

[14] J. M. Brader, W. Senn, S. Fusi, Learning Real-World Stimuli in a Neural Network with Spike-Driven Synaptic Dynamics, Neural Computation 19 (11) (2007) 2881–2912. doi:10.1162/neco.2007.19.11.2881.

[15] J. J. Hopfield, C. D. Brody, Learning rules and network repair in spike-timing-based computation networks, Proceedings of the National Academy of Sciences 101 (1) (2004) 337–342. doi:10.1073/pnas.2536316100.

[16] P. Crotty, W. B. Levy, Energy-efficient interspike interval codes, Neurocomputing 65 (2005) 371 – 378, Computational Neuroscience: Trends in Research 2005. doi:http://dx.doi.org/10.1016/j.neucom.2004.10.031.

[17] F. Gabbiani, W. Metzner, Encoding and processing of sensory information in neuronal spike trains, Journal of Experimental Biology 202 (10) (1999) 1267–1279. arXiv:http://jeb.biologists.org/content/202/10/1267.full.pdf.

[18] S. Shoham, D. H. O'Connor, R. Segev, How silent is the brain: is there a "dark matter" problem in neuroscience?, Journal of Comparative Physiology A 192 (8) (2006) 777–784. doi:10.1007/s00359-006-0117-6.

[19] A. Roxin, N. Brunel, D. Hansel, G. Mongillo, C. van Vreeswijk, On the distribution of firing rates in networks of cortical neurons, Journal of Neuroscience 31 (45) (2011) 16217–16226. arXiv:http://www.jneurosci.org/content/31/45/16217.full.pdf, doi:10.1523/JNEUROSCI.1677-11.2011.
URL http://www.jneurosci.org/content/31/45/16217

[20] B. Wang, W. Ke, J. Guang, G. Chen, L. Yin, S. Deng, Q. He, Y. Liu, T. He, R. Zheng, Y. Jiang, X. Zhang, T. Li, G. Luan, H. D. Lu, M. Zhang, X. Zhang, Y. Shu, Firing frequency maxima of fast-spiking neurons in human, monkey, and mouse neocortex, Frontiers in Cellular Neuroscience 10 (2016) 239, 27803650[pmid]. doi:10.3389/fncel.2016.00239.

[21] 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, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, D. S. Modha, A million spiking-neuron integrated circuit

with a scalable communication network and interface, Science 345 (6197) (2014) 668–673. doi:10.1126/science.1254642.

[22] J. Gehlhaar, Neuromorphic Processing: A New Frontier in Scaling Computer Architecture, in: Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS '14, ACM, New York, NY, USA, 2014, pp. 317–318. doi:10.1145/2541940.2564710.

[23] N. Qiao, H. Mostafa, F. Corradi, M. Osswald, F. Stefanini, D. Sumislawska, G. Indiveri, A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses, Frontiers in Neuroscience 9 (2015) 141. doi:10.3389/fnins.2015.00141.

[24] S. B. Furber, F. Galluppi, S. Temple, L. A. Plana, The SpiNNaker project, Proceedings of the IEEE 102 (5) (2014) 652–665. doi:10.1109/JPROC.2014.2304638.

[25] B. V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J. M. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. A. Merolla, K. Boahen, Neurogrid: A Mixed-Analog-Digital Multichip System for Large-scale Neural Simulations, Proceedings of the IEEE 102 (5) (2014) 699–716. doi:10.1109/JPROC.2014.2313565.

[26] F. Ponulak, A. Kasinski, Supervised Learning in Spiking Neural Networks with ReSuMe: Sequence Learning, Classification, and Spike Shifting, Neural Computation 22 (2) (2010) 467–510. doi:10.1162/neco.2009.11-08-901.

[27] S. M. Bohte, J. N. Kok, H. La Poutre, Error-backpropagation in temporally encoded networks of spiking neurons, Neurocomputing 48 (1) (2002) 17–37. doi:http://dx.doi.org/10.1016/S0925-2312(01)00658-0.

[28] A. Mohemmed, S. Schliebs, S. Matsuda, N. Kasabov, SPAN: Spike Pattern Association Neuron for Learning Spatio-Temporal Spike Patterns, In-

ternational Journal of Neural Systems 22 (04), pMID: 22830962. doi:10.1142/S0129065712500128.

[29] Q. Yu, H. Tang, K. C. Tan, H. Li, Precise-Spike-Driven Synaptic Plasticity: Learning Hetero-Association of Spatiotemporal Spike Patterns, PLOS ONE 8 (11) (2013) 1–16. doi:10.1371/journal.pone.0078318.

[30] R. V. Florian, The chronotron: A Neuron That Learns to Fire Temporally Precise Spike Patterns, PloS ONE 7 (8) (2012) e40233. doi:https://doi.org/10.1371/journal.pone.0040233.

[31] M. Mozafari, S. R. Kheradpisheh, T. Masquelier, A. Nowzari-Dalini, M. Ganjtabesh, First-spike based visual categorization using reward-modulated stdp, arxiv preprint arXiv:1705.09132 [q-bio.NC].

[32] A. Taherkhani, A. Belatreche, Y. Li, L. P. Maguire, DL-ReSuMe: A Delay Learning-Based Remote Supervised Method for Spiking Neurons, IEEE Transactions on Neural Networks and Learning Systems 26 (12) (2015) 3137–3149. doi:10.1109/TNNLS.2015.2404938.

[33] X. Xie, H. Qu, Z. Yi, J. Kurths, Efficient Training of Supervised Spiking Neural Network via Accurate Synaptic-Efficiency Adjustment Method, IEEE Transactions on Neural Networks and Learning Systems 28 (6) (2017) 1411–1424. doi:10.1109/TNNLS.2016.2541339.

[34] J. Tapson, G. Cohen, S. Afshar, K. Stiefel, Y. Buskila, R. Wang, T. J. Hamilton, A. van Schaik, Synthesis of neural networks for spatio-temporal spike pattern recognition and processing, arXiv preprint arXiv:1304.7118.

[35] W. W. Lee, S. L. Kukreja, N. V. Thakor, CONE: Convex-Optimized-Synaptic Efficacies for Temporally Precise Spike Mapping, IEEE Transactions on Neural Networks and Learning Systems 28 (4) (2017) 849–861. doi:10.1109/TNNLS.2015.2509479.

[36] T. Masquelier, S. J. Thorpe, Unsupervised Learning of Visual Features through Spike Timing Dependent Plasticity, PLOS computational biology 3 (2) (2007) 1–11. doi:10.1371/journal.pcbi.0030031.

[37] P. Panda, K. Roy, Unsupervised regenerative learning of hierarchical features in spiking deep networks for object recognition, in: 2016 International Joint Conference on Neural Networks, 2016, pp. 299–306. doi:10.1109/IJCNN.2016.7727212.

[38] S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, T. Masquelier, STDP-based spiking deep convolutional neural networks for object recognition, Neural Networksdoi:https://doi.org/10.1016/j.neunet.2017.12.005.

[39] A. Tavanaei, A. S. Maida, Multi-layer unsupervised learning in a spiking convolutional neural network, in: 2017 International Joint Conference on Neural Networks (IJCNN), 2017, pp. 2023–2030. doi:10.1109/IJCNN.2017.7966099.

[40] P. U. Diehl, M. Cook, Unsupervised learning of digit recognition using spike-timing-dependent plasticity, Frontiers in computational neuroscience 9 (2015) 99. doi:10.3389/fncom.2015.00099.

[41] S. Roy, A. Basu, An Online Unsupervised Structural Plasticity Algorithm for Spiking Neural Networks, IEEE Transactions on Neural Networks and Learning Systems 28 (4) (2017) 900–910. doi:10.1109/TNNLS.2016.2582517.

[42] J. M. Allred, K. Roy, Unsupervised incremental STDP learning using forced firing of dormant or idle neurons, in: 2016 International Joint Conference on Neural Networks (IJCNN), 2016. doi:10.1109/IJCNN.2016.7727509.

[43] N. K. Kasabov, NeuCube: A spiking neural network architecture for mapping, learning and understanding of spatio-temporal brain data, Neural

Networks 52 (2014) 62–76. doi:http://dx.doi.org/10.1016/j.neunet.2014.01.006.

[44] N. Kasabov, N. M. Scott, E. Tu, S. Marks, N. Sengupta, E. Capecci, M. Othman, M. G. Doborjeh, N. Murli, R. Hartono, J. I. Espinosa-Ramos, L. Zhou, F. B. Alvi, et al., Evolving spatio-temporal data machines based on the NeuCube neuromorphic framework: design methodology and selected applications, Neural Networks 78 (2016) 1–14. doi:http://dx.doi.org/10.1016/j.neunet.2015.09.011.

[45] J. Wang, A. Belatreche, L. P. Maguire, T. M. McGinnity, SpikeTemp: An Enhanced Rank-Order-Based Learning Approach for Spiking Neural Networks with Adaptive Structure, IEEE Transactions on Neural Networks and Learning Systems 28 (2017) 30–43. doi:10.1109/TNNLS.2015.2501322.

[46] J. Wang, A. Belatreche, L. P. Maguire, T. M. McGinnity, SpikeComp: An Evolving Spiking Neural Network with Adaptive Compact Structure for Pattern Classification, Springer International Publishing, Cham, 2015, pp. 259–267. doi:10.1007/978-3-319-26535-3_30.

[47] T. Takuya, T. Haruhiko, K. Hiroharu, T. Shinji, A training algorithm for spike sequence in spiking neural networks – a discussion on growing network for stable training performance, in: 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery, 2016, pp. 1773–1777. doi:10.1109/FSKD.2016.7603446.

[48] Y. Cao, Y. Chen, D. Khosla, Spiking deep convolutional neural networks for energy-efficient object recognition, International Journal of Computer Vision 113 (1) (2015) 54–66. doi:10.1007/s11263-014-0788-3.

[49] P. U. Diehl, D. Neil, J. Binas, M. Cook, S. C. Liu, M. Pfeiffer, Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing, in: 2015 International Joint Conference on Neural Networks (IJCNN), 2015, pp. 1–8. doi:10.1109/IJCNN.2015.7280696.

[50] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, Theory and tools for the conversion of analog to spiking convolutional neural networks, arXiv preprint arXiv:1612.04052.

[51] E. Hunsberger, C. Eliasmith, Training spiking deep networks for neuromorphic hardware, arXiv preprint arXiv:1611.05141.

[52] Hunsberger, Eric, Spiking deep neural networks: Engineered and biological approaches to object recognition, Ph.D. thesis (2018).
URL http://hdl.handle.net/10012/12819

[53] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, S.-C. Liu, Conversion of continuous-valued deep networks to efficient event-driven networks for image classification, Frontiers in Neuroscience 11 (2017) 682. doi:10.3389/fnins.2017.00682.
URL https://www.frontiersin.org/article/10.3389/fnins.2017.00682

[54] T. P. Lillicrap, D. Cownden, D. B. Tweed, C. J. Akerman, Random synaptic feedback weights support error backpropagation for deep learning, Nature Communicationsdoi:http://dx.doi.org/10.1038/ncomms13276.

[55] N. Anwani, B. Rajendran, NormAD - Normalized Approximate Descent based supervised learning rule for spiking neurons, in: International Joint Conference on Neural Networks, 2015, pp. 1–8. doi:10.1109/IJCNN.2015.7280618.

[56] J. H. Lee, T. Delbruck, M. Pfeiffer, Training Deep Spiking Neural Networks Using Backpropagation, Frontiers in Neuroscience 10 (2016) 508. doi:10.3389/fnins.2016.00508.

[57] L. F. Abbott, Lapicque's introduction of the integrate-and-fire model neuron (1907), Brain Research Bulletin 50 (1999) 303–304. doi:http://dx.doi.org/10.1016/S0361-9230(99)00161-6.

[58] S. Panzeri, N. Brunel, N. K. Logothetis, C. Kayser, Sensory neural codes using multiplexed temporal scales, Trends in Neurosciences 33 (3) (2010) 111 – 120. doi:http://dx.doi.org/10.1016/j.tins.2009.12.001.

[59] A. A. Lazar, E. K. Simonyi, L. T. Tóth, Time encoding of bandlimited signals, an overview, in: Proceedings of Conference on Telecommunication Systems, Modeling and Analysis, Citeseer, 2005.

[60] J. Wang, A. Belatreche, L. Maguire, M. McGinnity, Online versus offline learning for spiking neural networks: A review and new strategies, in: 2010 IEEE 9th International Conference on Cyberntic Intelligent Systems, 2010, pp. 1–6. doi:10.1109/UKRICIS.2010.5898113.

[61] S. Q. Khan, A. Ghani, M. Khurram, Population coding for neuromorphic hardware, Neurocomputing 239 (2017) 153 – 164.

[62] A. Calderón, S. Roa, J. Victorino, Handwritten digit recognition using convolutional neural networks and gabor filters, Proc. Int. Congr. Comput. Intell, 2003.

[63] S. Schreiber, J.-M. Fellous, D. Whitmer, P. Tiesinga, T. J. Sejnowski, A new correlation-based measure of spike timing reliability, Neurocomputing 52 (2003) 925–931, computational Neuroscience: Trends in Research 2003. doi:http://dx.doi.org/10.1016/S0925-2312(02)00838-X.

[64] T. Gokmen, Y. Vlasov, Acceleration of Deep Neural Network Training with Resistive Cross-Point Devices: Design Considerations, Frontiers in Neuroscience 10 (2016) 333. doi:10.3389/fnins.2016.00333.

[65] J. Tapson, P. De Chazal, A. van Schaik, Explicit computation of input weights in extreme learning machines, Springer International Publishing, Cham, 2015, pp. 41–49. doi:10.1007/978-3-319-14063-6_4.

[66] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, R. Fergus, Regularization of Neural Networks using DropConnect, in: Proceedings of the 30th International

Conference on Machine Learning, Vol. 28 of Proceedings of Machine Learning Research, PMLR, Atlanta, Georgia, USA, 2013, pp. 1058–1066.
URL http://proceedings.mlr.press/v28/wan13.html

[67] E. Stromatias, D. Neil, M. Pfeiffer, F. Galluppi, S. B. Furber, S.-C. Liu, Robustness of spiking Deep Belief Networks to noise and reduced bit precision of neuro-inspired hardware platforms, Frontiers in Neuroscience 9 (2015) 222. doi:10.3389/fnins.2015.00222.

[68] S. R. Kulkarni, B. Rajendran, Scalable digital CMOS Architecture for Spike based Supervised Learning, Springer International Publishing, Cham, 2015, pp. 149–158. doi:10.1007/978-3-319-23983-5_15.