2-2022

A Deep Unsupervised Feature Learning Spiking Neural Network with Binarized Classification Layers for the EMNIST Classification

Ruthvik Vaila
Boise State University

John Chiasson
Boise State University

Vishal Saxena
University of Delaware

 https://doi.org/10.1109/TETCI.2020.3035164

A Deep Unsupervised Feature Learning Spiking Neural Network with Binarized Classification Layers for the EMNIST Classification

Ruthvik Vaila, *Student Member, IEEE,* John Chiasson, *Fellow, IEEE,* and Vishal Saxena, *Senior Member, IEEE*

***Abstract*—End user AI is trained on large server farms with data collected from the users. With ever increasing demand for IoT devices, there is a need for deep learning approaches that can be implemented at the edge in an energy efficient manner. In this work we approach this using spiking neural networks. The unsupervised learning technique of spike timing dependent plasticity (STDP) and binary activations are used to extract features from spiking input data. Gradient descent (backpropagation) is used only on the output layer to perform training for classification. The accuracies obtained for the balanced EMNIST data set compare favorably with other approaches. The effect of stochastic gradient descent (SGD) approximation on learning capabilities of our network are also explored.**

***Index Terms*—STDP, Spiking Networks, Surrogate Gradients, EMNIST, Binary Activations, Reduced Multiplications.**

I. Introduction

THE deep learning community has shifted its attention towards energy efficiency. Proposals for energy-efficient artificial neural network primarily fall into two categories: hardware and algorithm based approaches. Hardware based approaches [1] [2] [3] tend to propose new/hybrid computing architectures or tend to trade off precision with classification accuracy whereas algorithm based approaches tend to simplify the network synapses and activations to avoid expensive multiplication operations. Algorithmic approaches like Binary Neural Networks (BNNs) with binary weights and activations have been proposed and were shown to achieve the state-of-the-art classification accuracy with MNIST [4] and CIFAR-10 [5] datasets. Another approach towards energy-efficient machine learning is bio-inspired and this type of networks are called Spiking Neural Networks (SNNs). In this work our focus is on bio-inspired algorithmic approaches. Biological neurons communicate with each other by transmitting spikes which are $70mV$ voltage pulses while artificial neural networks (ANNs) communicate with each other using floating point real-valued activations. There are two popular theories pertaining to how the information is encoded in the spiking input image: rate coding and latency coding. Rate coding stipulates that the information transfer from the input image to the next (hidden) layer is embedded in the rate of spikes coming out of the input neurons. In this work, latency coding is used and it refers to the information in the image being encoded in the relative spike times [6] [7]. According to latency coding, earlier spikes (in time) carry more information than later (in time) spikes [6]. The synapses (weights) between spiking neurons are modified according to spike timing dependent plasticity (STDP), where the synapse is strengthened if an input neuron aides the output neuron in spiking (spikes before the output neuron spikes) while the synapse is weakened if an input neuron does not aide the output neurons in spiking (spikes after the output neuron spikes) [8]. As STDP is an unsupervised learning rule, SNNs can be trained layer by layer. The synapses (weights) in ANNs are modified using gradient descent (backpropagation) to reduce the loss which is defined by an appropriate cost function on the last (output) layer [9]. More specifically, gradient descent is used to update the weights of the network to an acceptable local minimum of the cost function on the output layer [10]. In ANNs input data is fed forward through the network and then the gradient of the cost function is computed layer by layer by going backwards to update the weights in each layer. That is, the weights cannot be updated immediately as one feed forwards the input data giving rise to the update locking problem [11]. This makes backpropagation a global update rule unlike STDP which is a local update rule [12]. Further, backpropagation uses the same weights for the forward as well as the backward steps, which is referred to as the weight transport problem [13] [14] [15]. Random backpropagation (feedback alignment) was shown to mitigate this problem [16]. A neuromorphic variant of the feedback alignment (random backpropagation) was proposed in [17] and was shown to achieve an accuracy of approximately 98% on the MNIST dataset. Panda et al. [18] reported a reduction in energy consumption by a factor of 25 for CIFAR-10 and reduction by a factor of 2 for the IMAGENET dataset [19] by combining existing techniques in deep learning with rate encoded spiking networks. Other works such as [20] [21] [22] [23] approximate backpropagation with rate coding and have achieved approximately 98% accuracy on the MNIST dataset. Apart from training spiking networks directly either with supervised or unsupervised methods, alternative methods that convert an existing ANNs to SNNs using transfer learning were introduced in [24]. Masquelier et al. have shown that the neurons in coincidence detection mode have less than $10\ ms$ to produce a response and only their earliest spike(s) are processed for rapid object classification. In this case the time to process the inputs is so short that it is highly unlikely to use feedback connections to impact decision making. It was

R. Vaila is a PhD candidate at Department of Electrical and Computer Engineering, Boise State University, Boise, ID, 83706 USA e-mail: ruthvikvaila@u.boisestate.edu

J. Chiasson is with Boise State University, johnchiasson@boisestate.edu.

V. Saxena is with University of Delaware, vsaxena@udel.edu.

also shown that neuronal recordings in the Inferior Temporal Cortex over small time durations ($\approx 12\ ms$) represent object categories in a linearly separable manner via a Support Vector Machine (SVM) [25]. The authors in [26] [27] [28] [29] [30] proposed various algorithms to learn the exact spike times for SNNs with latency (temporal) coding using gradient descent (backpropagation) on an output cost function. In those approaches, activations in ANNs are replaced with spike times and the loss is obtained by calculating the time difference between the desired output spike times and the actual output neuron spike times. In latency (temporal) coding minimizing a spike time is conceptually similar to maximizing the activation of a target neuron. In the mammalian brain, individual neurons only have $\approx 10\ ms$ to produce a response [25]. This rapid object categorization motivated us to decouple the supervised training of output classification layers from the unsupervised training of the low-level feature extraction layer. The current literature on spiking networks reports a lower accuracy on classification tasks [31]. However, standard ANNs employing SGD are energy inefficient as they require high precision computations. Energy-efficiency (low power consumption) and state-of-the-art classification accuracy are both important goals for an object classification system.

A primary contribution of this work is to combine spiking feature extraction (based on the latency encoded first spike) with approximate SGD and binary activations to achieve near state-of-the-art classification accuracy on EMNIST and MNIST datasets. We also demonstrate that the unsupervised feature extraction showed robustness to input noise in terms of the final classification accuracy. Such noisy inputs arise in several practical scenarios such as the silicon retina sensors and readout circuits [32]. The proposed SNN architecture reduces the number of multiplication operations between 3-4 orders of magnitude and thus facilitates low-power realization on neuromorphic architectures employing either digital asynchronous or analog in-memory computing [33] [34] [35]. Finally we also introduce the software tool SPYKEFLOW, developed by the authors for latency encoded SNNs.

II. NETWORK DESCRIPTION

Our network is shown in Figure 1. The extraction layers of this network are similar to that of [36] [37].

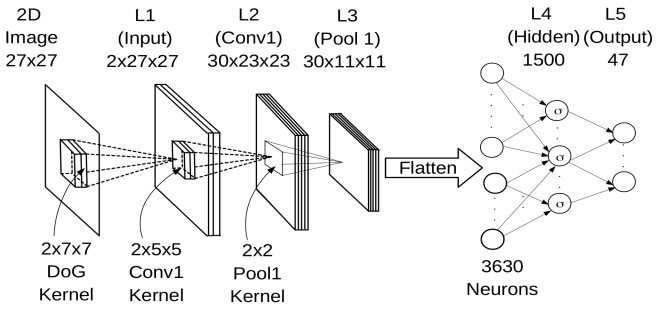

Fig. 1. Proposed SNN architecture. Here, layers $L1 - L3$ are the feature extraction layers and layer $L3 - L5$ are the feature classification layers.

A. Input Encoding

Following [36] [37], K_{σ_1,σ_2} is a Difference of Gaussian (DoG) filter with $\sigma_1 = 1, \sigma_2 = 2$ for the ON-center and $\sigma_1 = 2, \sigma_2 = 1$ for the OFF-center, given by

$$K_{\sigma_1,\sigma_2}(i,j) = \begin{cases} \frac{1}{2\pi\sigma_1^2} e^{-\frac{i^2+j^2}{2\sigma_1^2}} - \frac{1}{2\pi\sigma_2^2} e^{-\frac{i^2+j^2}{2\sigma_2^2}} & \text{for } -3 \leq i,j \leq 3 \\ 0, \text{otherwise} \end{cases} \quad (1)$$

Plots of ON and OFF center filters are shown in Figure 2. The input image is convolved with these ON and OFF centered filters, resulting in two "images" which are then converted to their corresponding ON and OFF spiking image.

$$\Gamma_{\sigma_1,\sigma_2}(u,v) = \sum_{j=-3}^{j=3} \sum_{i=-3}^{i=3} \mathbf{I}_{in}(u+i, v+j) K_{\sigma_1,\sigma_2}(i,j) \quad \text{for } 0 \leq u \leq 26, 0 \leq v \leq 26. \quad (2)$$

At each location (u,v) of the output image $\Gamma_{\sigma_1,\sigma_2}(u,v)$, a unit spike $s_{(u,v)}$ is produced if and only if $\Gamma_{\sigma_1,\sigma_2}(u,v)$ exceeds a threshold i.e.,

$$\Gamma_{\sigma_1,\sigma_2}(u,v) > \gamma_{DoG} \quad (3)$$

where $\gamma_{DoG} = 50$ was arbitrarily chosen [38]. The spike times are encoded relative to the start time depending on magnitude of the membrane potentials and the relation is given by

$$\tau_{(u,v)} = \frac{1}{\Gamma_{\sigma_1,\sigma_2}(u,v)} \text{ in milliseconds.}$$

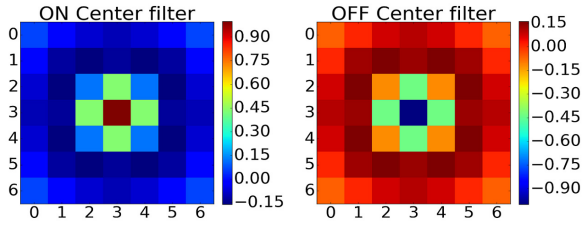

Fig. 2. ON and OFF center DoG filters. Color code indicates the filter values.

The spike signal $s_{(u,v)}(t)$, is latency (temporally) encoded [6] by delaying it by an amount inversely proportional to $\Gamma_{\sigma_1,\sigma_2}(u,v)$ as shown in Figure 3. That is, the greater the value of $\Gamma_{\sigma_1,\sigma_2}(u,v)$, the sooner the neurons spikes and vice versa. Equivalently, the value of $\Gamma_{\sigma_1,\sigma_2}(u,v)$ is encoded in the value $\tau_{(u,v)}$. Note that a neuron at location (u,v) can generate at most one spike. Silicon retina sensors such as eDVS [32], ATIS [39] directly provide spiking images. Such images have been used in the SNN literature [40] [41].

B. Convolution Layers and STDP

We denote a spike at time t emanating from the (u,v) neuron of a spiking image by $S_{L1}(t,k,u,v)$, where $k = 0$ (ON center) or $k = 1$ (OFF center) and $(0,0) \leq (u,v) \leq (27,27)$.

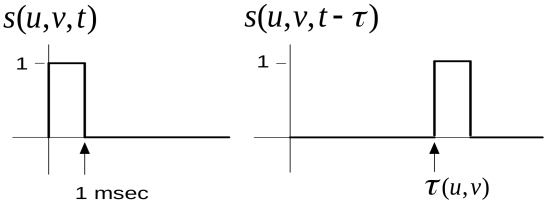

Fig. 3. Spike signal. Left: Spike signal from the input image with no time delay. Right: Spike signal from the input image with a delay of τ milliseconds.

Layer $L2$ (Conv1) consists of 30 (feature) maps with each map having its own convolution kernel given by

$$W_{C1}(w,k,i,j) \in \mathbb{R}^{30\times 2\times 5\times 5} \quad \text{for} \quad w = 0,1,2,...,29$$

The instantaneous "membrane potential" of the (u,v) neuron of a feature map w ($w = 0,1,2,...,29$) in layer $L2$ (Conv1) at time t is given by

$$V_{L2}(t,w,u,v) = \sum_{\tau=0}^{t}\left(\sum_{k=0}^{1}\sum_{i=0}^{4}\sum_{i=0}^{4} S_{L1}(\tau,k,u+i,v+j)W_{C1}(w,k,i,j)\right) \quad (4)$$
$$\text{for } 0 \le (u,v) \le 22$$

If at time t the membrane potential of a neuron in a feature map w at location (u,v) crosses a set threshold value

$$V_{L2}(t,w,u,v) > \gamma_{L2} = 15$$

then the neuron at (w,u,v) produces a spike at time t.

At any time t, *all* of the potentials $V_{L2}(t,w,u,v)$ for $(0,0) \le (u,v) \le (22,22)$ and $w = 0,1,2,...,29$ are computed in parallel. Neurons in different locations within a map and in different maps may have spiked. In particular, at the location (u,v), there can be multiple spikes (up to 30) produced by 30 different neuron belonging to 30 different maps. The idea here is to have different maps learn different features so that all the important features in the input image can be captured. To enforce this condition, *lateral inhibition* and *STDP competition* are used [36].

1) Lateral Inhibition: To explain lateral inhibition, suppose at the location (u,v) there were potentials $V_{L2}(t,w,u,v)$ in different maps (w goes from 0 to 29) at time t that exceeded the threshold γ_{L2}. Then the neuron in a map with the highest potential $V_{L2}(t,w,u,v)$ at (u,v) inhibits the neurons in all other maps at the location (u,v) from spiking for the current image (even if the potentials in other maps exceeded the threshold). Figure 4 (left) shows the accumulated spikes (from an MNIST image of "5") for 12 time steps from all 30 maps of Layer $L2$ at each location (u,v) *without* lateral inhibition. For example, at location (19,14) in Figure 4 (left) the color code is yellow indicating in excess of 20 spikes, i.e., more than 20 of the maps produced a spike at that location.

Figure 4 (center) shows the accumulation of spikes from all 30 maps for 12 time steps, but now *with* lateral inhibition imposed. Note that at each location there is at most one spike indicated by the color code. Also, as explained next, only a few of these spikes will actually result in the update of any of the 30 kernels (weights) in layer L2.

2) STDP Competition: After lateral inhibition we consider each of the maps in layer L2 that had one or more neurons with their membrane potential V exceeding γ. Let these maps be $w_{k1}, w_{k2}, ..., w_{km}$ where[1] $0 \le k_1 < k_2 < \cdots < k_m \le 29$. Then in each map w_{ki}, we locate the neuron in that map that has the maximum membrane potential value. Let

$$(u_{k1},v_{k1}),(u_{k2},v_{k2}),...,(u_{km},v_{km}) \quad (5)$$

be the location of these maximum potential neurons in each map. Then neuron (u_{ki},v_{ki}) inhibits all other neurons in its map w_{ki} from spiking for the remainder of the time steps of the current spiking image. Further, these m neurons can inhibit each other depending on their relative location as we explain next. Suppose neuron (u_{ki},v_{ki}) of map w_{ki} has the highest potential of the m neurons in Equation (5). Then, in an 11×11 area centered around (u_{ki},v_{ki}), this neuron inhibits all the neurons of all the other maps in the same 11×11 area. Next, suppose neuron (u_{kj},v_{kj}) of map w_{kj} has the second highest potential of the remaining $m-1$ neurons. If the location (u_{kj},v_{kj}) of this neuron was within the 11×11 area centered on neuron (u_{ki},v_{ki}) of map w_{ki}, then it is inhibited. Otherwise, this neuron at (u_{kj},v_{kj}) inhibits all the neurons of all the other maps in a 11×11 area centered on it. This process is continued for the remaining $m-2$ neurons. In summary, there can be no more than one neuron that spikes in the same 11×11 area across all the maps.[2] The right side of Figure 4 shows the spike accumulation for the final winner neuron for 12 time steps across 30 maps after both lateral inhibition and STDP competition have been imposed. It also shows that there is at most one winner neuron from all the maps in any 11×11 area. For this particular input image (the digit 5), these five winning neurons are from maps 14, 16, 19, 21, and 23 at locations (19, 4), (3,10), (17, 15), (9,12) and (3,19), respectively and will result in weight updates for these 5 map kernels. Lateral inhibition STDP competition resulted in an average of only *5.8 spikes per image* from the $30\times 22\times 22$ neurons in $L2$ during training with EMNIST dataset. Figure 5 shows the evolution of the randomly initialized weights for all 30 maps after training is performed with 6000 images.

3) Spike Feature Vectors : After (unsupervised) training of the weights (synapses) in the $L2$ layer, these weights are fixed. Spike feature vectors are created by passing spiking input images through layer $L2$ (Conv1) *with lateral inhibition enforced* and *without STDP competition* as there is no training involved. The spikes coming out of the $L2$ layer are then pooled in the $L3$ layer *without lateral inhibition*. The pooling is done on an area of 2×2 neurons in L2 with a stride of 2. Specifically, in each 2×2 area of L2 which contains 4 neurons, the spike of the neuron with the maximum membrane potential V_{L2}, assuming it exceeds the threshold γ_{L2}, is then the spike of the corresponding neuron of the $L3$ (pooling) layer (i.e., thresholding on maxpooling). For the EMNIST dataset each input image results in a spike tensor of shape $\tau\times 30\times 11\times 11$. We set τ to be 12 and these tensors were summed across their

[1]The other maps did not have any neurons whose membrane potential crossed the threshold and therefore could not spike.

[2]The use of the number 11 for the 11×11 inhibition area of neurons was suggested by Dr. Kheradpisheh [38].

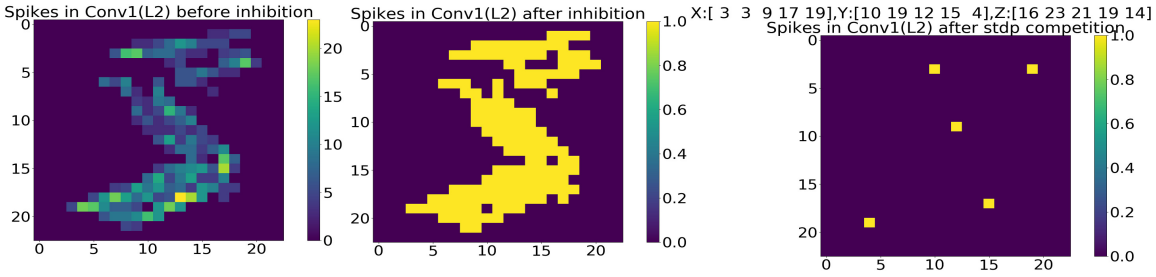

Fig. 4. Left: EMNIST digit "5" input. Accumulation of spikes from all 30 maps and 12 time steps in L2 layer *without* lateral inhibition. Center: Accumulation of spikes from all 30 maps and all 12 time steps in L2 *with* lateral inhibition. Right: Accumulation of spikes across all maps and 12 time steps with both lateral inhibition and STDP competition imposed for a single image. X, Y denote the location of neuron in a map and Z denotes the map number. Note that these five winner neuron spikes suppress all the other neurons that crossed the threshold.

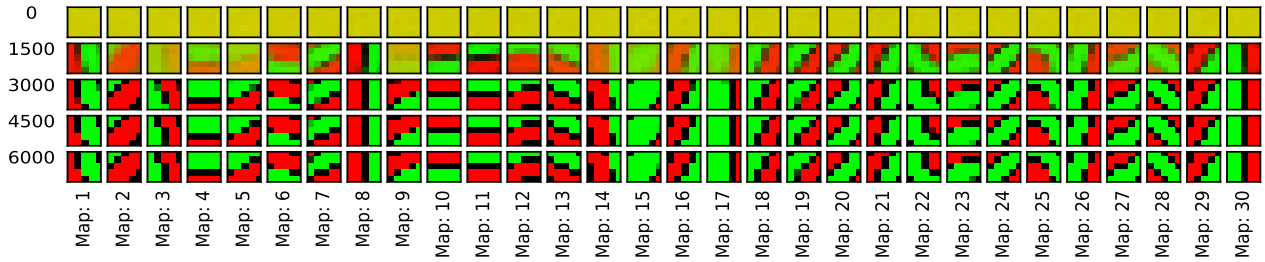

Fig. 5. Evolution of learned feature maps in the first convolution layer. Red and Green indicate ON and OFF center synapses respectively.

first axis (i.e, along the time axis). The resulting tensors in $R^{30\times11\times11}$ were flattened.[3]

Once a neuron in layer $L3$ spikes, it is not allowed to spike again during the rest of the time steps for the current image. This results in the spike feature vectors being *binary* valued (i.e., vectors of zeros and ones). In our experiments an average of 125 spikes/image emerge out of L3 from the $30\times11\times11 = 3630$ neurons and an average of 300 spikes/image emerge out of L4 from the $30\times23\times23 = 15870$ neurons for the EMNIST dataset. As the activations of L4 are *binary* (non differentiable), in order to perform backpropagation from layer L5 back to layer L3, a *surrogate* gradient is used (see Section IV).

4) Weight Initialization : The weights of the $L2$ layer are initialized from the normal distribution $\mathcal{N}(0.8, 0.04)$. The weights of layers $L4$ & $L5$ layers are initialized from the normal distribution $\mathcal{N}(0, 0.01)$, but truncated to restrict them between ±0.02. A softmax activation is used for the classification layer $L5$ with its inputs converted to integers using the floor function. A look-up table containing predefined values of the exponential function e^x can be used to calculate softmax activation in a hardware implementation. The activation functions employed in layer $L4$ (denoted by σ in Figure 1) are discussed below (see Section IV).

C. Spike Timing Dependent Plasticity (STDP)

Spike timing dependent plasticity defines how a synapse (weight) between an input (pre-synaptic) neuron and an output (post-synaptic) neuron is modulated (updated). In its simplest

[3]If more convolution layers are desired, spike tensors collected in L3 layer can be used for unsupervised training of any subsequent convolutional layers.

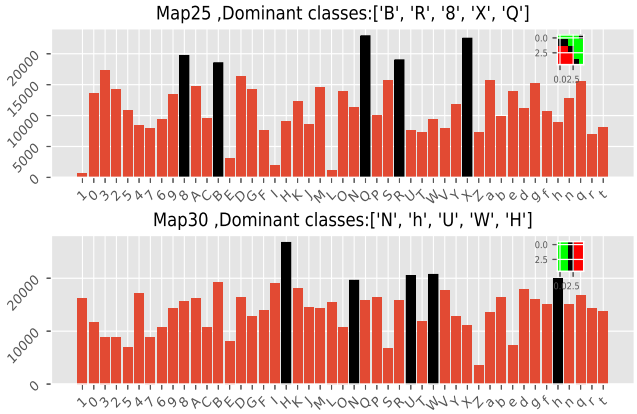

Fig. 6. Spikes per map per label in L3 (Pool 1). Highlighted (in black) are the classes that resulted in most number of spikes in a particular feature map. Feature learned by the corresponding map is shown in the inset.

form [36], STDP strengthens the synapse (weight) between an input and output neuron if the input neuron aids the output neuron in overcoming the membrane threshold (i.e., it spikes); otherwise the synapses are weakened. With t_{out} and t_{in} denoting the spike time of the output (post-synaptic) and the input (pre-synaptic) neuron, respectively, the STDP learning rule used here is given by

$$\Delta w_i = \begin{cases} -a^- w_i(1-w_i), & \text{if } t_{out} - t_{in} < 0 \\ +a^+ w_i(1-w_i), & \text{if } t_{out} - t_{in} \geq 0 \end{cases} \qquad (6)$$
$$w_i \leftarrow w_i + \Delta w_i.$$

Learning in spiking networks refers to the change Δw_i in the (synaptic) weight. The learning rate parameters a^+ and a^- are

initialized with relatively lower values i.e., $(0.004, 0.003)$ [36] [37] and are typically increased as the learning progresses. In our experiments we doubled the learning rate for every 1500 input images. As there are neither labels nor a cost function involved in the process of STDP, it is an *unsupervised* learning algorithm. That is, the weights can be updated during the feed forward step in SNNs. In contrast, ANNs update their weights during the error feed back. So, STDP does not suffer from the update locking restriction [11]. Synapses in feature extraction section of the network in Figure 1 were updated at the end of every time step.

III. Backpropagation in the L3-L5 Layers

Stochastic gradient descent (SGD) via backpropagation is the primary choice for state-of-the-art classification, regression, and generative learning [42]. A cost function is assigned to the last layer of the network and the synapses are updated to minimize the cost. In our network, backpropagation is used only in the classification layers (L3-L4-L5) of the network with a single hidden layer L4. Let $\delta^l, a^l(=\sigma(z^l)), b^l, W^l, z^l(=w^l z^{l-1}+b^l)$ denote the error vector, the activation vector, the bias vector, the weights and the net input to the activation function for the l^{th} layer, respectively [43]. σ is the activation function. With C denoting the output cost, the backpropagation equations are:

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \tag{7}$$

where δ^L denotes the error vector on the last layer and the error vector for the hidden layers is given by

$$\delta^l = ((W^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \tag{8}$$

Updates to biases and weights of layer l are calculated with

$$\frac{\partial C}{\partial b^l} = \delta^l \tag{9}$$

$$\frac{\partial C}{\partial W^l} = \delta^l a^{(l-1)T} \tag{10}$$

C denotes the cost in the final layer. We used a softmax activation with a cross entropy cost function for the last layer so that equation (7) becomes

$$\delta^L = -(y - a^L), \tag{11}$$

where a^L and y are softmax activation of the output layer and the one hot label vector, respectively. For the remainder of this article, we refer to gradients obtained using Equations (7)-(11) as *true gradients* with $\sigma(z)$ a Rectified Linear Unit (ReLU) activation function.

IV. Binary Activations and Surrogate Gradients

In order to significantly reduce the number of high precision multiplications, the activations of the L4 layer are restricted to be binary. That is, if the net input to a neuron is greater than zero the output is one, otherwise the output is zero. Consequentially this activation function is not differentiable (i.e., the gradient doesn't exist). Here, we provide two different possible functions that we have used to replace the true gradient, i.e., to be its surrogate [44].

A. Surrogate Gradient 1

The activation function of a neuron in layer $L4$ is defined as

$$a^l = \sigma(z^l) \triangleq \begin{cases} 0, & z < 0 \\ z, & 0 \leq z < \tau \leq 1 \\ \tau, & z \geq \tau. \end{cases} \tag{12}$$

Figure 7 plots this ReLU activation function that saturates at τ ($\tau \leq 1$).

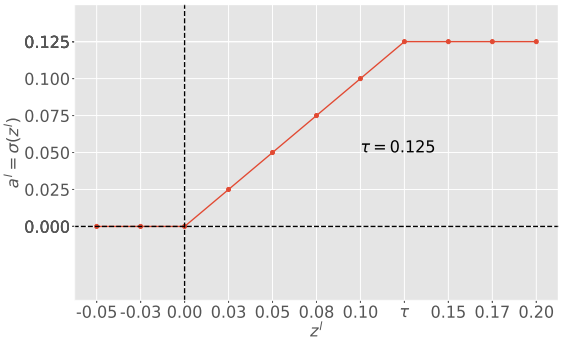

Fig. 7. Activation function $a^l = \sigma(z^l)$ for neurons in layer $L4$.

Since we require the activation to be binary its definition is modified to be ($\lceil \cdot \rceil$ denotes the ceiling function)

$$a^l = \lceil \sigma(z^l) \rceil \triangleq \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases} \tag{13}$$

As this binarized activation (13) is non-differentiable we define its surrogate gradient as

$$\sigma'(z^l) \triangleq \begin{cases} 1, & 0 \leq z < \tau \leq 1 \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

which is the derivative of the function in Equation 12.

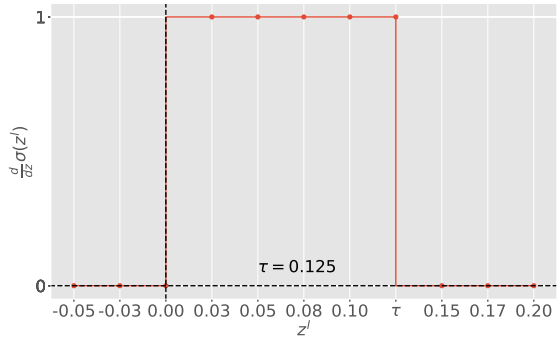

Fig. 8. Surrogate gradient of activation function defined in equation (12).

Simulations were performed by setting τ to $0.25, 0.125, 0.05$ and we found that 0.125 maximizes the validation accuracy. Since error backpropagation is not feasible with equation (13), we take derivative of $\sigma(z)$ using equation (14). In SNN convention, we denote an activation value of 1 as spike and an activation value of 0 as no spike.

B. Surrogate Gradient 2

We also considered a second activation given by

$$a^l = \sigma(z^l) \triangleq \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases} \tag{15}$$

and define its surrogate gradient as

$$\sigma'(z^l) \triangleq \begin{cases} 1, & z \geq 0 \\ 0, & z < 0 \end{cases} \tag{16}$$

Note that $\sigma'(z) = \sigma(z)$ and is binary so that $a^l = \sigma'(z^l)$ in the hidden layer. Equation (8) then becomes

$$\delta^l = ((W^{l+1})^T \delta^{l+1}) \odot a^l \tag{17}$$

where a^l determines if a neuron spikes in the l^{th} layer. Hence a^l determines if a neuron in the l^{th} layer is to receive error information from the $l+1$ layer. Substituting Equation (17) in Equation (10) results in

$$\frac{\partial C}{\partial W^l} = \left((w^{l+1})^T \delta^{l+1} \odot a^l\right) a^{(l-1)T} \tag{18}$$

We see that a neuron in $l-1$ layer gets to update its synapse with a neuron in l^{th} layer if both neurons have spiked, i.e., for $\partial C/\partial W^l_{pq}$ to be a non-zero both a^l_p and a^{l-1}_q have to be non-zero.

V. MNIST

Our interest here is the EMNIST dataset. However, as the MNIST handwritten digits dataset is a popular benchmark, we briefly present our results using this dataset [4]. The MNIST digits were passed through the network in Figure 1 and encoded into spike vectors as (described in Section II-B3). Note that the extracted features are *binary* valued. Table I shows that surrogate gradient 1 yields a test accuracy 0.74% higher or 74 more correct classifications compared to surrogate gradient 2 with $10,000$ test images. Figure 9 shows the classification accuracy per class using the surrogate gradient 1. For results reported in Table I, dropout mechanism (50%) was used in hidden layer for regularization, number of neurons in layer L4 were set to 900, mini-batch size was set to 5 and η for the actual and true gradients was set to 0.0125 and 0.01, respectively. These results were obtained by averaging over five experiments with the classification layers of the network in Figure 1 trained for 30 epochs. For classification accuracies reported using the true gradient a quadratic cost function with a ReLU activation function for layers L4, L5 was used. Whereas for accuracies reported using the surrogate gradients a cross-entropy cost function with softmax approximation (see Section II-B4) for layer L5 and binary activation function for layers L3, L4 was employed.

TABLE I
MNIST RESULTS. TRUE GRADIENT REFERS TO EQUATIONS (7)-(11).

Gradient Type	Mean Test Acc.	Max. Test Acc.
True Gradient	98.58%	98.66%
Surrogate Gradient 1	98.49%	98.54%
Surrogate Gradient 2	97.75%	97.77%

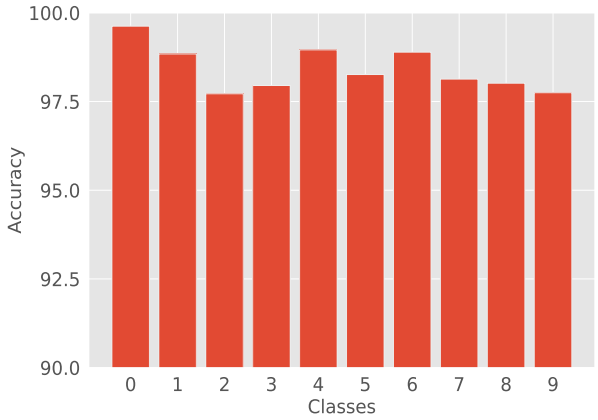

Fig. 9. Classification accuracy per class with surrogate gradient 1.

VI. EMNIST

EMNIST dataset has 47 classes containing handwritten upper & lower case letters of the English alphabet in addition to the digits. This dataset is divided into $102,648$ training images, $10,151$ validation images, and $18,800$ test images [45]. Mini-batch size was set to 5 and a dropout of 50% was used in the hidden layer (L4). The number of neurons in layer L4 was 1500. Number of epochs was set to 35 and the all experiments were averaged over 5 trials.

A. Why Use First Spike Based Feature Extraction?

In this section, *binary* valued features vectors (i.e., vector with 0s and 1s) were collected in the layer $L3$ as described in Section II-B3. Classification was performed using an ANN with binary activation for the hidden layer $L4$ neurons and an approximated softmax output explained in Section II-B4. The synapses of L2 layer (Conv1) were fixed with random weights and the binary spike features collected in layer L3 were classified using surrogate gradient 1 resulting in 80.43% maximum test accuracy. Similarly, binary spike features collected from layer L3 with unsupervised trained weights in layer L2 (Conv2) were classified using surrogate gradient 1 resulting in a maximum test accuracy of 85.6%, ≈ 972 more correct classifications when compared to random weights in L2. Results averaged over five trials are provided in Table II. Figures 11 and 12 show the confusion matrices for the network with random synapses in L2 and STDP trained synapses in L2 respectively. When the layer L2 was trained with STDP, Figure 12 shows that there is frequent misclassification between the classes {f} and {F}, the classes {0} and {O}, the classes {q} and {9}, the classes {1}, {I} and {L}, the classes{S} and {5}, and the classes {2} and {Z}. Misclassifications for this case are explainable in the sense that one might expect humans to make such errors. For example, in the lower left corner of Figure 10, the network predicted a lower case "f", while the label was an upper case "F". In contrast, when layer L2 was not trained, Figure 11 shows that the network frequently misclassified the classes {H} and {0}, the classes {E} and {1}, the classes {A} and {1}, the classes {Z} and {7}, and

the classes {h} and {L}. One would not expect humans to make such mistakes.

TABLE II
EMNIST ACCURACY WITH RANDOM AND TRAINED L2 LAYER.

Gradient Type	Mean Test Acc.	Max Test Acc.	L2 Synapses
Surrogate Gradient 1	80.21%	80.43%	Random
Surrogate Gradient 1	85.35%	85.60%	STDP trained

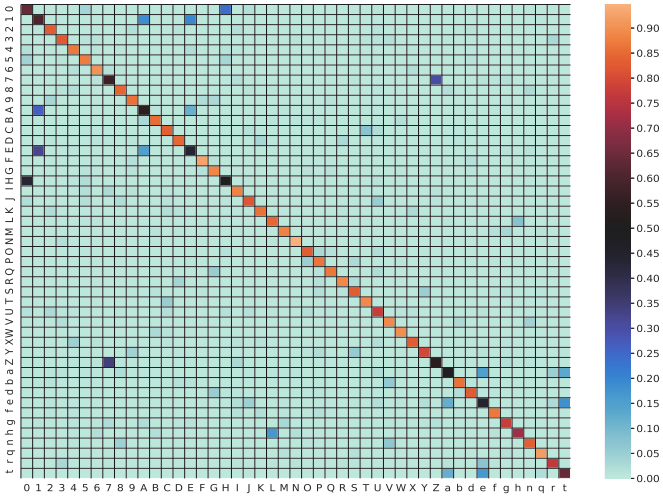

Fig. 11. Confusion matrix of predictions with the EMNIST dataset when random weights were used in layer L2.

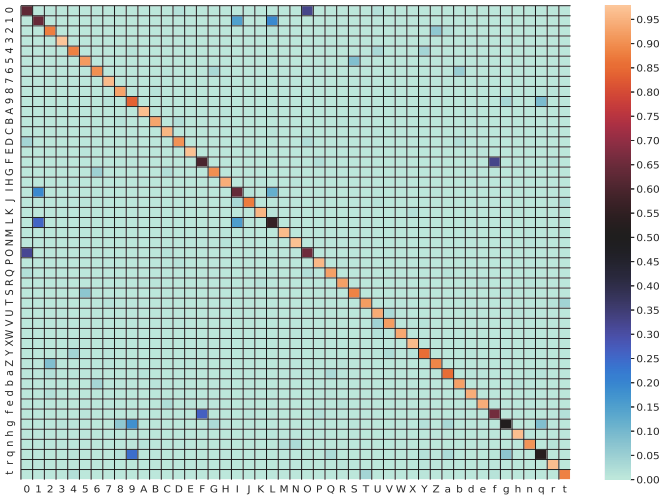

Fig. 12. Confusion matrix of predictions with the EMNIST dataset when the synapses in layer L2 were learned in an unsupervised fashion using STDP.

We performed experiments to study the classification accuracy in the presence of noise in the spiking input images (layer L1). To explain, suppose a particular image resulted in 100 spikes in L1. Then by introduction of 10% noise, we imply that 5 of the randomly chosen neurons that spiked were set to zero, while 5 randomly chosen non-spiking neurons were forced to spike. Figure 13 shows the result of this input noise on the final classification accuracy. As shown in Figure 13, the network can withstand $\approx 40\%$ this input noise before the classification accuracy decreases to that of the case where the L2 layer synapses were randomly set.

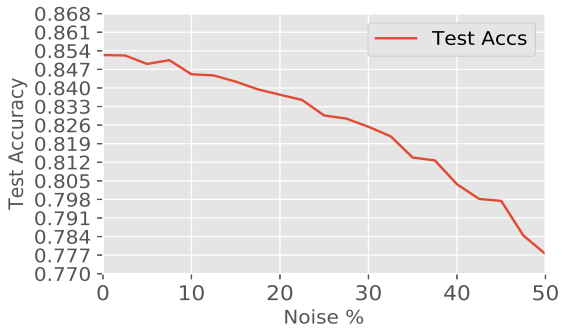

Fig. 13. Effect of input noise on the final classification accuracy.

B. Effect of Gradient Approximation on Classification

Table III shows that the true gradients results in higher classification accuracy and surrogate gradient 1 outperforms surrogate gradient 2 by 1.0% (i.e., 188 more correct classifications with 18800 test images).

C. Conditioning on Upper Case, Lower Case, and Digits

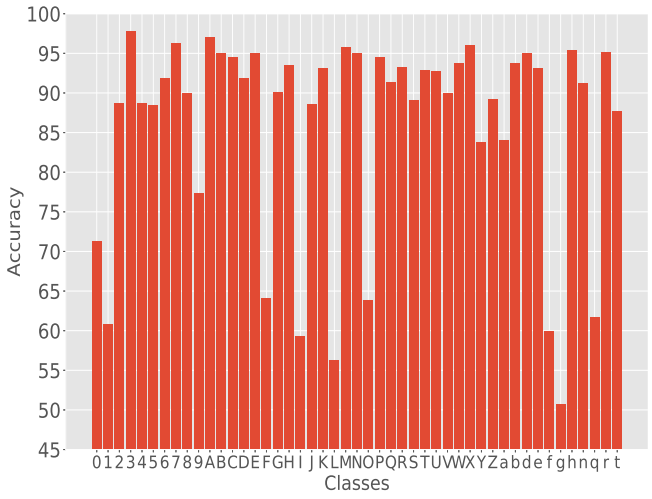

Fig. 14. Classification accuracy per class with surrogate gradient 1.

Figure 14 shows the accuracy per class when surrogate gradient 1 is used for classification. With handwritten data, even a human classifier may not be able to tell the difference between, for example, the upper case letter "O" and the digit "0". To study this we also ran the classifier conditioned on (given that) the image under test was an upper case letter, a lower case letter or a digit. No retraining was performed for this section. Table III shows dramatic increase in accuracy under this conditioning. The accuracy per class using this conditioning is given in Figure 15. It is seen that the classes I, L, g, q have the least recognition rate, but still well above their accuracies previously seen in Figure 14 where conditioning was not used. In more detail we found that about 13% of the letters "q" were misclassified as the letter "g", about 4% of

Fig. 10. Frequently misclassified classes in the EMNIST dataset. *P* and *L* denote predicted class and actual label, respectively.

TABLE III
EMNIST RESULTS. TRUE GRADIENT REFERS TO EQUATIONS (7)-(11).

Gradient Type	Mean Test Acc.	Max. Test Acc.	Conditioned Max. Test Acc.	η	Activation
True Gradient	85.47%	85.7 %	94.49 %	0.05	ReLU
Surrogate Gradient 1	85.35 %	85.60 %	94.1 %	0.02	Binary
Surrogate Gradient 2	84.24 %	84.47 %	93.72 %	0.02	Binary

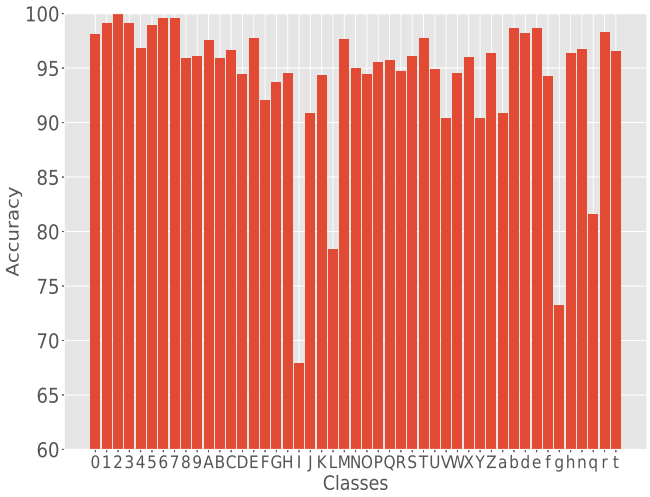

Fig. 15. Classification accuracy per class of EMNIST dataset with surrogate gradient 1 after the proposed conditioning.

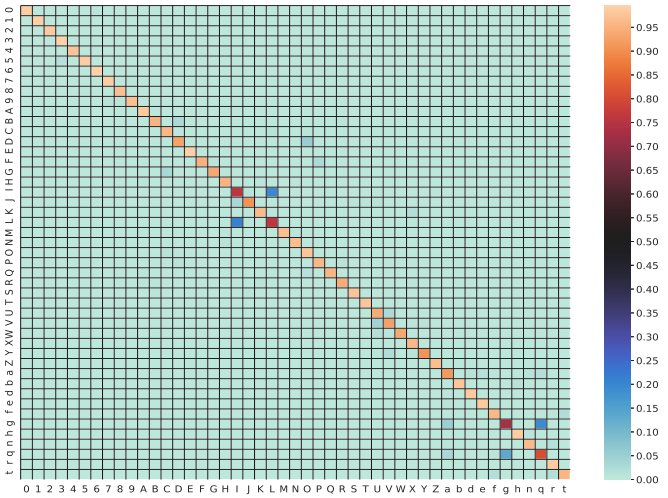

Fig. 16. Confusion matrix of predictions with EMNIST dataset when the inputs are conditioned on Upper Case, Lower Case and Digits.

letters "q" were misclassified as the letter "a", while about 83% of letters "q" were correctly classified. About 20% of letters "g" were misclassified as the letter "q" while about 73% of letters "g" were correctly classified. Similarly, we found that about 27% of letters of upper case "I" (eye) were misclassified as the upper case letter "L" while 68% of upper case "I" were correctly classified. As a final observation about 20% of upper case letters "L" were misclassified as an upper case "I" (eye) while about 78% of upper case letters "L" were correctly classified. Figure 16 shows the confusion matrix for the conditioned case.

D. Computational Advantage of Binary Activations

In the feedforward paths L1 through L4 the matrix-vector multiplication operations can altogether be avoided in a hardware implementation as all these layers have binary activations. For example, executing the multiplication of a set of (floating point) weights times a set of spikes (binary activations) is simply

$$\begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \\ & \vdots & \\ w_{n1} & w_{n2} & w_{n3} \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} w_{12} \\ w_{22} \\ w_{32} \\ \vdots \\ w_{n2} \end{pmatrix} + \begin{pmatrix} w_{13} \\ w_{23} \\ w_{33} \\ \vdots \\ w_{n3} \end{pmatrix}. \tag{19}$$

That is, multiplication is replaced by addition. This technique avoids the need for dedicated multiplier hardware and allows the feasibility of in-memory computing [34] [35]

Another advantage is found in backpropagation computations. Specifically, as the surrogate gradient $\sigma'(z^l)$ is binary, the error vector δ^l for the hidden layer can be obtained without having to compute majority of the row-column multiplications in

$$\delta^l = ((W^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l).$$

For example

$$\left(\underbrace{\begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix}}_{(w^{l+1})^T} \times \underbrace{\begin{pmatrix} 1 \\ 2.5 \\ 3.1 \end{pmatrix}}_{\delta^{l+1}} \right) \odot \underbrace{\begin{pmatrix} 0 \\ 1 \end{pmatrix}}_{\sigma'(z^l)} = \begin{pmatrix} 0 \\ w_{21} + 2.5w_{22} + 3.1w_{23} \end{pmatrix}. \tag{20}$$

That is, in equation (20) the row-column multiplications of the first row are avoided as the result will zero out due to the element-wise (Hadamard product) vector multiplication. All the weight updates, $\partial C/\partial W^l$ can be obtained without explicitly calculating vector outer product $\delta^l a^{(l-1)T}$ as the activations of $L3$ and $L4$ layers are binarized. For example

$$\underbrace{\begin{pmatrix} a \\ b \\ c \end{pmatrix}}_{\delta^l} \times \underbrace{\begin{pmatrix} 0 & 1 & 0 \end{pmatrix}}_{a^{(l-1)T}} = \begin{pmatrix} 0 & a & 0 \\ 0 & b & 0 \\ 0 & c & 0 \end{pmatrix}. \tag{21}$$

That is, the matrix on the right side of Equation (21) is found by simply transcribing δ^l into its columns as specified by $a^{(l-1)T}$.

E. Number of High-Precision Multiplications

The majority of computations in a DNN are high-precision multiplications of the weights with the activations during both the forward inference as well as the backpropagation of the error. Energy consumption of the network is hardware architecture dependent, but in order to provide an estimate for the energy savings in our SNN, we compare the number of high precision multiplications between a DNN and our SNN [46]. It requires $m \times n \times p$ high precision multiplications in order to multiply an $m \times n$ matrix by an $n \times p$ matrix in a fully connected network. Convolution (in valid mode) of an $I \times I$ image with an $F \times F$ filter requires $(I - F + 1) \times (I - F + 1) \times F \times F$ multiplications. As we employ temporally encoded spikes with binary activations used in the classification layer, the forward path can be implemented without any multiplications (See Equation (19) and Equation (21)). Further, orders of magnitude less number of multiplications are required for backpropagation as we explain next. Table IV below compares the number of high-precision multiplications required for a DNN with our SNN-based approach. In a neuromorphic system the input spikes are typically provided by a silicon retina (eDVS [32]). Thus we assume that the images are available in spike form. We begin by estimating the number of multiplications in the L4 layer for our SNN. Figure 17 shows the average number of neurons in the L4 layer (1500 total neurons) for each epoch that have a non-zero activation. The number of multiplications required to calculate the error in layer L4 according to Equation (20) is as follows: In the earliest epochs, the number of multiplications during the training is approximately 1.45×10^9 which is computed as

$$20500 \text{ m-batches} \times 5 \frac{\text{images}}{\text{m-batch}} \times 47 \text{ classes} \times 300 \text{ non-zero activations}. \tag{22}$$

In the latter epochs, the number of non-zero activations decreases to 100 making the number of multiplications approximately $20500 \times 5 \times 47 \times 100 = 4.5 \times 10^8$. Summing over the 35 epochs results in approximately 1.87×10^{10} multiplications.

To compute the number of multiplications in the L2 layer of our SNN, note that during the unsupervised training of L2 (Conv1), lateral inhibition and STDP competition results in sparse neuronal activity in that there are only 5.8 weight updates (winner spikes) per spiking input image (see Section II-B2). L2 was trained (unsupervised) on 6000 spiking input images . The number of multiplications required is approximately 5.22×10^7 computed as

$$5.8 \text{ avg updates} \times 2 \times 5 \times 5 \times 30 \text{ L2 synapses} \times 6000 \text{ images}. \tag{23}$$

Due to the binary activations used in L4, layer L5 of our network can be implemented in a custom hardware without any multiplications. Based on this quantitative analysis our approach makes a suitable candidate for low power implementation as it uses $\approx 3 - 4$ orders of magnitude less multiplications compared to a standard DNN.

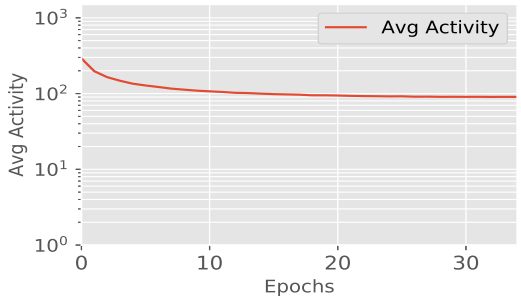

Fig. 17. Number of neurons with non-zero activations in layer L4 as the training in classification sections of the network in Figure 1 progresses.

VII. SpykeFlow

The authors did consider the PyNN simulator with Neuron [47] [48]. However these tools are designed for neuroscientists where the neuron models are much more complex than needed in our case. The software tool Nengo [49], developed for bio-inspired machine learning, uses a more complex neuronal model than required here. Motivated by the simple spiking models in Kheradpisheh et al.'s work [36], we developed customized software tools. Following [36] our package supports instantaneous (non leaky integrate and fire) neurons, latency encoding, and inhibition mechanisms to be able to simply extract meaningful features from the input images. The feature extraction in our SNNs is performed in an unsupervised manner using STDP, which requires monitoring the weight updates (synapse changes) in the spiking network. Our software provides the capability to monitor spike activity, weight evolution (updates), feature extraction (spikes per map per label), synapse convergence. This software tool was used in our other work [40] [41]. Similar to our work, Mozafari et al released the software tool SpykeTorch in [50], which is based on the PyTorch deep learning tool. Our software is named SpykeFlow[4] and primarily uses NumPy to do the calculations of lateral inhibition, STDP updates, neuron spike accumulation, etc. However, we also use TensorFlow for computationally intensive calculations such as convolution

[4]https://github.com/ruthvik92/SpykeFlow

TABLE IV
COMPARISON OF MULTIPLICATIONS FOR A DNN AND THE SNN IN FIGURE 1

Architecture	L2		L4		L5		Total
	Forward	Updates	Forward	Updates	Forward	Updates	
DNN	1.42×10^{12}	1.42×10^{12}	1.954×10^{13}	1.96×10^{13}	2.53×10^{11}	2.53×10^{11}	$\approx 4.25 \times 10^{13}$
Proposed SNN	0	5.22×10^{7}	0	1.87×10^{10}	0	0	$\approx 1.87 \times 10^{10}$

and pooling. Therefore, the users will have the ability to use a GPU, if one is available. Detailed instructions to use the software are provided in [51].

VIII. CONCLUSIONS

A comparison of our work with recent publications that employ the EMNIST dataset is provided in Table V. Rate encoded spiking networks require hundreds of time-steps of simulation for a single input image resulting in very high spike counts. Whereas, latency encoded inputs to an SNN equipped with first spike based feature extraction results in very few spikes thus requiring fewer synapse updates implying lower power consumption. In this work, neurons were essentially used as coincidence detectors with latency encoded input spikes and first spike based feature extraction to transform the inputs to spike feature vectors that contain robust object category information as observed in biology [25]. This spike features were then classified using the proposed backpropagation with surrogate gradients to demonstrate up to 85.60% accuracy with the EMNIST dataset. This was achieved by employing backpropagation only in the classification layers of the network as the classification layers are decoupled from the feature extraction layers. The accuracy achieved here is quite comparable to the 85.57% accuracy reported in [53] which used rate encoded (Poisson) input spikes in a network with one hidden layer comprised of 800 neurons and with backpropagation performed in all the layers. Furthermore, [53] uses complex Leaky integrate-and-fire (LIF) neurons as opposed to our simple instantaneous summation neurons that act as coincidence detectors. Using a conventional deep convolution network, Shawon et al [52] report an accuracy of 90.59% on the balanced EMNIST (refer to survey paper [54]). The deep network in [52] consisted of 6 convolution layers, a hidden layer with 64 neurons followed by a classification layer. Though our accuracy is lower than DNNs, we have proposed an energy-efficient solution using bio-inspired unsupervised techniques. This energy efficiency can be realized by implementing the proposed architecture using a Neuromorphic ASIC or FPGA.

TABLE V
COMPARISON OF EMNIST CLASSIFICATION RESULTS.

Learning method	Neuron model	Input Encoding	Max. Test Acc.
Supervised DNN [52]	ReLU	-	90.59 %
Supervised SNN [53]	LIF	Rate	85.57 %
Unsupervised SNN + Supervised DNN (This work)	Summation (Instantaneous) + Binary	Latency	85.60 %

As shown in Table III we also demonstrated an accuracy of 94.49% when the classifier was given the information that an input image was either a letter (upper or lower case) or a digit. As discussed in the paper, this conditioning was considered due to the indistinguishability of few samples between some of the classes (e.g., $\{0\}$ and $\{O\}$ in Figure 10).

The computational advantages (i.e., high precision multiplications reduced by 3-4 orders of magnitude) of using binary activations with respect to a custom hardware implementation [55] [34] [35] [56] of SNNs were also discussed. We also demonstrated the robustness of the classification accuracy when noisy spikes are present in the input spiking image. All the simulations were performed using a custom software tool SPYKEFLOW [51] developed by the authors.

IX. ACKNOWLEDGMENTS

Ruthvik Vaila is thankful to the Electrical and Computer Engineering Department, Boise State University, Idaho, USA for providing him a graduate assistantship (GA) to carry out this work.

REFERENCES

[1] J. Liu, H. Zhao, M. A. Ogleari, D. Li, and J. Zhao, "Processing-in-memory for energy-efficient neural network training: A heterogeneous approach," in *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2018, pp. 655–668.

[2] X. Jiao, V. Akhlaghi, Y. Jiang, and R. K. Gupta, "Energy-efficient neural networks using approximate computation reuse," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2018, pp. 1223–1228.

[3] E. H. Lee, D. Miyashita, E. Chai, B. Murmann, and S. S. Wong, "Lognet: Energy-efficient neural networks using logarithmic computation," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 5900–5904.

[4] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *ATT Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, vol. 2, 2010.

[5] A. Krizhevsky, V. Nair, and G. Hinton, "The CIFAR-10 dataset," May 2012. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html

[6] A. Delorme, L. Perrinet, and S. J. Thorpe, "Networks of integrate-and-fire neurons using Rank Order Coding B: Spike timing dependent plasticity and emergence of orientation selectivity," *Neurocomputing*, vol. 38-40, pp. 539 – 545, 2001, computational Neuroscience: Trends in Research 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231201004039

[7] M. Kiselev, "Rate coding vs. temporal coding - is optimum between?" in *2016 International Joint Conference on Neural Networks (IJCNN)*, July 2016, pp. 1355–1359.

[8] T. Masquelier, R. Guyonneau, and S. J. Thorpe, "Spike Timing Dependent Plasticity Finds the Start of Repeating Patterns in Continuous Spike Trains," *PLOS ONE*, vol. 3, no. 1, pp. 1–9, 01 2008. [Online]. Available: https://doi.org/10.1371/journal.pone.0001377

[9] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization Methods for Large-Scale Machine Learning," 2016.

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

[11] C. Frenkel, M. Lefebvre, and D. Bol, "Learning without feedback: Direct random target projection as a feedback-alignment algorithm with layerwise feedforward training," 2019.

[12] J. C. R. Whittington and R. Bogacz, "Theories of Error Back-Propagation in the Brain," *Trends in Cognitive Sciences*, vol. 23, no. 3, pp. 235–250, Jan. 2020. [Online]. Available: https://doi.org/10.1016/j.tics.2018.12.005

[13] S. Grossberg, "Competitive learning: From interactive activation to adaptive resonance," *Cognitive Science*, vol. 11, no. 1, pp. 23 – 63, 1987. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0364021387800253

[14] Q. Liao, J. Z. Leibo, and T. Poggio, "How Important is Weight Symmetry in Backpropagation?" *arXiv e-prints*, p. arXiv:1510.05067, Oct 2015.

[15] A. Rocke, *The weight transport problem*. paulispace, Jun 2017. [Online]. Available: https://paulispace.com/deep/learning/2017/06/30/weight-transport.html

[16] T. Lillicrap, D. Cownden, D. Tweed, and C. J. Akerman, "Random synaptic feedback weights support error backpropagation for deep learning," *Nature Communications*, vol. 7, p. 13276, 11 2016.

[17] E. O. Neftci, C. Augustine, S. Paul, and G. Detorakis, "Event-Driven Random Back-Propagation: Enabling Neuromorphic Deep Learning Machines," *Frontiers in Neuroscience*, vol. 11, p. 324, 2017. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2017.00324

[18] C. Lee, P. Panda, G. Srinivasan, and K. Roy, "Training Deep Spiking Convolutional Neural Networks With STDP-Based Unsupervised Pre-training Followed by Supervised Fine-Tuning," *Frontiers in Neuroscience*, vol. 12, p. 435, 08 2018.

[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," 2009.

[20] N. Anwani and B. Rajendran, "NormAD - Normalized Approximate Descent based supervised learning rule for spiking neurons," in *2015 International Joint Conference on Neural Networks (IJCNN)*, July 2015, pp. 1–8.

[21] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training Deep Spiking Neural Networks Using Backpropagation," *Frontiers in Neuroscience*, vol. 10, p. 508, 2016. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2016.00508

[22] A. Tavanaei, Z. Kirby, and A. Maida, "Training Spiking ConvNets by STDP and Gradient Descent," *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 07 2018.

[23] J. C. Thiele, O. Bichler, A. Dupret, S. Solinas, and G. Indiveri, "A Spiking Network for Inference of Relations Trained with Neuromorphic Backpropagation," p. arXiv:1903.04341, Mar 2019.

[24] B. Rueckauer, I.-A. Lungu, Y. Hu, and M. Pfeiffer, "Theory and Tools for the Conversion of Analog to Spiking Convolutional Neural Networks," *arXiv e-prints*, p. arXiv:1612.04052, Dec 2016.

[25] T. Masquelier, "Spike-based computing and learning in brains, machines, and visual systems in particular (hdr report)," Ph.D. dissertation, Université Toulouse III - Paul Sabatier, 10 2017.

[26] I. M. Comsa, K. Potempa, L. Versari, T. Fischbacher, A. Gesmundo, and J. Alakuijala, "Temporal coding in spiking neural networks with alpha synaptic function," *arXiv e-prints*, p. arXiv:1907.13223, Jul 2019.

[27] B. Gardner and A. Grüning, "Supervised Learning in Spiking Neural Networks for Precise Temporal Encoding," *PLOS ONE*, vol. 11, no. 8, pp. 1–28, 08 2016. [Online]. Available: https://doi.org/10.1371/journal.pone.0161335

[28] S. R. Kheradpisheh and T. Masquelier, "S4NN: temporal backpropagation for spiking neural networks with one spike per neuron," p. arXiv:1910.09495, 2019.

[29] H. Mostafa, "Supervised Learning Based on Temporal Coding in Spiking Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 7, pp. 3227–3235, July 2018.

[30] Y. Sakemi, K. Morino, T. Morie, and K. Aihara, "A Supervised Learning Algorithm for Multilayer Spiking Neural Networks Based on Temporal Coding Toward Energy-Efficient VLSI Processor Design," p. arXiv:2001.05348, 2020.

[31] P. Panda, A. Aketi, and K. Roy, "Towards Scalable, Efficient and Accurate Deep Spiking Neural Networks with Backward Residual Connections, Stochastic Softmax and Hybridization," 2019.

[32] J. Conradt, R. Berner, M. Cook, and T. Delbruck, "An embedded AER dynamic vision sensor for low-latency pole balancing," in *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, Sep. 2009, pp. 780–785.

[33] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The SpiNNaker Project," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652–665, May 2014.

[34] X. Wu, V. Saxena, and K. Zhu, "Homogeneous Spiking Neuromorphic System for Real-World Pattern Recognition," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 5, no. 2, pp. 254–266, June 2015.

[35] X. Wu, V. Saxena, K. Zhu, and S. Balagopal, "A CMOS Spiking Neuron for Brain-Inspired Neural Networks With Resistive Synapses and In Situ Learning," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 62, no. 11, pp. 1088–1092, Nov 2015.

[36] S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, and T. Masquelier, "STDP-based spiking deep convolutional neural networks for object recognition," *Neural Networks*, vol. 99, pp. 56 – 67, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608017302903

[37] S. R. Kheradpisheh, M. Ganjtabesh, and T. Masquelier, "Bio-inspired unsupervised learning of visual features leads to robust invariant object recognition," *Neurocomputing*, vol. 205, pp. 382 – 392, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231216302880

[38] S. R. Kheradpisheh, private communication.

[39] C. Posch, D. Matolin, R. Wohlgenannt, M. Hofstätter, P. Schön, M. Litzenberger, D. Bauer, and H. Garn, "Live demonstration: Asynchronous time-based image sensor (ATIS) camera with full-custom AE processor," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, May 2010, pp. 1392–1392.

[40] R. Vaila, J. Chiasson, and V. Saxena, "Deep Convolutional Spiking Neural Networks for Image Classification," *arXiv e-prints*, p. arXiv:1903.12272, Mar 2019.

[41] R. Vaila, J. Chiasson, and V. Saxena, "Feature Extraction Using Spiking Convolutional Neural Networks," in *Proceedings of the International Conference on Neuromorphic Systems*, ser. ICONS '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3354265.3354279

[42] R. Vaila, D. Lloyd, and K. Tetz, "Regression with Deep Learning for Sensor Performance Optimization," p. arXiv:2002.11044, 2020.

[43] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, Jan 2015. [Online]. Available: http://neuralnetworksanddeeplearning.com/

[44] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1," 2016.

[45] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "EMNIST: an extension of MNIST to handwritten letters," *arXiv e-prints*, p. arXiv:1702.05373, Feb. 2017.

[46] B. D. Rouhani, A. Mirhoseini, and F. Koushanfar, "Delight: Adding energy dimension to deep neural networks," in *Proceedings of the 2016 International Symposium on Low Power Electronics and Design*, ser. ISLPED '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 112–117. [Online]. Available: https://doi.org/10.1145/2934583.2934599

[47] R. Vaila, J. Chiasson, and V. Saxena, "Spiking CNNs with PYNN and NEURON," in *NICE Workshop Series*. Portland, Oregon, USA: Intel, Feb. 2019. [Online]. Available: https://www.researchgate.net/publication/335635588_Spiking_CNNs_with_PYNN_and_NEURON

[48] M. Hines and T. Carnevale, *NEURON Simulation Environment*. New York, NY: Springer New York, 2013, pp. 1–8, In Encyclopedia of Computational Neuroscience, Jaeger, Dieter and Jung, Ranu, Editors. [Online]. Available: https://doi.org/10.1007/978-1-4614-7320-6_795-1

[49] T. Bekolay, J. Bergstra, E. Hunsberger, T. DeWolf, T. Stewart, D. Rasmussen, X. Choo, A. Voelker, and C. Eliasmith, "Nengo: a Python tool for building large-scale functional brain models," *Frontiers in Neuroinformatics*, vol. 7, p. 48, 2014. [Online]. Available: https://www.frontiersin.org/article/10.3389/fninf.2013.00048

[50] M. Mozafari, M. Ganjtabesh, A. Nowzari-Dalini, and T. Masquelier, "SpykeTorch: Efficient Simulation of Convolutional Spiking Neural Networks With at Most One Spike per Neuron," *Frontiers in Neuroscience*, vol. 13, p. 625, 2019. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2019.00625

[51] R. Vaila, J. Chiasson, and V. Saxena, "A Deep Unsupervised Feature Learning Spiking Neural Network with Binarized Classification Layers for EMNIST Classification," p. arXiv:2002.11843, 2020.

[52] A. Shawon, M. Jamil-Ur Rahman, F. Mahmud, and M. M. Arefin Zaman, "Bangla handwritten digit recognition using deep cnn for large and unbiased dataset," in *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, Sep. 2018, pp. 1–6.

[53] Y. Jin, P. Li, and W. Zhang, "Hybrid Macro/Micro Level Backpropagation for Training Deep Spiking Neural Networks," *arXiv-eprints*, 05 2018.

[54] A. Baldominos, Y. Saez, and P. Isasi, "A Survey of Handwritten Character Recognition with MNIST and EMNIST," *Applied Sciences*, vol. 9, no. 15, 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/15/3169

[55] V. Saxena, X. Wu, I. Srivastava, and K. Zhu, "Towards Neuromorphic Learning Machines Using Emerging Memory Devices with Brain-Like Energy Efficiency," *Journal of Low Power Electronics and Applications*, vol. 8, no. 4, 2018. [Online]. Available: http://www.mdpi.com/2079-9268/8/4/34

[56] X. Wu and V. Saxena, "Dendritic-Inspired Processing Enables Bio-Plausible STDP in Compound Binary Synapses," *IEEE Transactions on Nanotechnology*, vol. PP, 01 2018.