Stable Spike-Timing Dependent Plasticity Rule for Multilayer Unsupervised and Supervised Learning

Amar Shrestha, Khadeer Ahmed, Yanzhi Wang, Qinru Qiu
Department of Electrical Engineering and Computer Science, Syracuse University, NY 13244, USA
Email {amshrest, khahmed, ywang393, qiqiu} @syr.ed

***Abstract*—Spike-Timing Dependent Plasticity (STDP), the canonical learning rule for spiking neural networks (SNN), is gaining tremendous interest because of its simplicity, efficiency and biological plausibility. However, to date, multilayer feed-forward networks of spiking neurons are either only partially trained using STDP or pre-trained using traditional deep neural networks which are converted to deep spiking neural networks or a two-layer network where STDP learnt features are manually labelled. In this work, we present a low-cost, simplified, yet stable STDP rule for layer-wise unsupervised and supervised training of a multilayer feed-forward SNN. We propose to approximate Bayesian neuron using Stochastic Integrate and Fire (SIF) neuron model and introduce a supervised learning approach using teacher neurons to train the classification layer with one neuron per class. A SNN is trained for classification of handwritten digits with multiple layers of spiking neurons, including both the feature extraction and classification layer, using the proposed STDP rule. Our method achieves comparable to better accuracy on MNIST dataset than manually labelled two layer networks for the same sized hidden layer. We also analyze the parameter space to provide rationales for parameter fine-tuning and provide additional methods to improve noise resilience and input intensity variations. We further propose a Quantized 2-Power Shift (Q2PS) STDP rule, which reduces the implementation cost of digital hardware while achieves comparable performance.**

Keywords—spiking neural network; STDP; digit recognition; unsupervised learning; supervised learning; quantized STDP

I. INTRODUCTION

The brain's ability to perform complex tasks such as pattern recognition, classification and inference with merely 10-20 watts [1] [2] is far superior to any state-of-the-art computer system. This arises from the fact that action potential (spike) of neurons (processing units) in the brain are used to process information in biological neural networks. Here, the communication of information is asynchronous and event driven, thus utilizing energy only when required. This mechanism is most closely modeled with Spiking Neural Networks (SNNs), the third generation of neural networks [3]. In today's world of Internet of Things, large amounts of data need to be processed and a major concern is the energy consumption. Mimicking the brains massive computation capability and energy efficiency is gaining tremendous interest and novel spiking neural network (SNN) implementations are actively being researched upon to achieve this goal.

Developments in this domain of neural networks have been pursued along two different branches. With the first approach, biologically plausible SNNs are developed to model biochemical principles from neuroscience aspects of the brain. These models solve differential equations which are computationally expensive and thus, are not functionally efficient for large scale systems [4] [5]. The second class of works are functional models at abstract level, which are far from biological brain in terms of computational principles. They are built from deep networks trained through backpropagation, and then converted into spike domain through various methods [6] [7]. In this work, we use a spike-based learning approach utilizing the primary learning method in biological neurons called Spike Timing Dependent Plasticity (STDP). This approach is more biologically plausible and with simplifications, garners to large-scale implementations as well.

STDP is a learning rule that potentiates or depresses a synapse depending on the relative timing between single pre and post-synaptic spikes [8]: long-term potentiation (LTP) occurs if the pre-synaptic spike arrives before the post-synaptic spike and long-term depression (LTD) occurs otherwise. It is local as it always pertains to a pair of pre- and post-synaptic neurons. And although it is correlation-based and causality is important for the synaptic plasticity, it differs from the Hebbian learning rule as STDP also requires temporal precedence [8]. Purely rate-based Hebbian rules on its own leads to runaway processes of potentiation causing instability. That requires additional constraints and mechanisms [9] [10] for containment. Whereas many theoretical studies concerned with non-Hebbian rules of plasticity look for desirable properties, such as a trend towards inherent stability in weight distribution, neural activity and competition among correlated inputs. STDP rules which are independent of the current synaptic weight (additive STDP) [11] induces competition but need mechanisms to prevent weights from either disappearing or exploding. Weight-dependent STDP rules (multiplicative STDP) [12] are inherently stable producing unimodal weight distribution but induce weak competition thus requiring additional mechanisms to induce competition [13]. These additional mechanisms increase complexity in their implementation. This is undesirable when it comes to large-scale or neuromorphic hardware implementations. In this work, we introduce a variation of a weight-dependent STDP rule which is inherently stable and yet simple and lends well to computationally efficient and inexpensive implementations.

STDP is known to be selective to patterns. When used in a network with lateral inhibition producing a Winner Take All (WTA) effect, STDP allows for learning discriminative features without supervision [14]. These features could either be used as intermediate features [15] or could be labelled [14] for classification. In addition to improving the STDP rule for unsupervised learning, in this work we also apply it for supervised learning by introducing teacher neurons. The unsupervised feature learning and unsupervised classifier learning are stacked forming a multilayer SNN trained with STDP.

This work is partially supported by the National Science Foundation under Grants CCF-1337300.

The main contributions of this work are summarized as following.

1. We present a simplified approximation of conventional Bayesian neuron and an improved STDP rule with extended LTD window, exponential weight dependence and different learning rates during LTD and LTP. Experimental results show that the modified STDP rule provides stable and competitive learning.
2. A layer-wise approach is used to combine unsupervised and supervised STDP learning and train a multilayer SNN on MNIST dataset. It achieves comparable or better accuracy for handwritten digit recognition than other existing STDP approaches with the similar size of trainable parameters and manually labelled features for classification. The parameter space is analyzed to further fine-tune the network.
3. We introduce an approximation of our STDP rule named Quantized 2-Power Shift (Q2PS) rule which is hardware implementation friendly. This approximation produces comparable results on the handwritten recognition to the original STDP rule.

The remainder of the paper is organized as the following. In Section II we discuss the related work, in Section III we introduce the proposed STDP rule and its stability, and the Quantized 2-Power Shift STDP rule. Section IV presents the network architecture and layer-wised learning that are used for MNIST classification. The results and the parameter space analysis are presented in Section V and finally, is the conclusions.

II. Related works

There are many variations of STDP learning rule. Additive STDP [11] induces competition among inputs but require hard weight constraints. Multiplicative STDP rules [12] are stable and unimodal with no or weak competition between inputs. This again requires complementary mechanisms such as rate normalization, weight scaling [16], homeostasis [14] and activity-dependent scaling [13]. These mechanisms add complexities either onto the neuron model or the synapse model. In this work, we introduce a simplified STDP rule which is stable but also develops good competition among correlated inputs utilizing an elongated symmetric STDP window.

The STDP rules are also used in various ways for different applications. [15] use STDP to learn intermediate features in a single layer in a deep SNN for recognizing faces. [14] train a layer of neurons on handwritten digits without supervision using STDP, label them afterwards and then utilize the now labelled neurons firing rates to classify the test digits. [17] also, use a similar approach. In the work, along with learning the features without supervision, we combine it with a classification layer that is trained with teacher neurons on top of the trained features. Here we train the combination layer-wise. [18] also adopts a "teacher signal" to train classifier neurons for their evolving SNN using a STDP rule similar to additive STDP with fixed minimum and maximum weights where the weights saturate at whereas [19] requires multiple classifier neurons per class for a SNN using event-driven Contrastive Divergence.

Some implementations of STDP rule is complex with several parameters involved. [14] use presynaptic and postsynaptic traces, weight constraints and additional factors for weight dependence, whereas [17] use normalized weights using the weight constraints. Because of the complexity in the STDP rules and complementary mechanisms in the neuron models, implementing them in hardware becomes expensive. Thus, in this work, we reduce the STP rule to the bare bones to simplify it and then further propose Quantized 2-power shift rule which is neuromorphic hardware implementation friendly.

III. Simplified Bayesian Neuron and STDP rules

In this section, we describe the neuron and the synapse model and discuss the simplifications that allows for an efficient implementation. We also discuss the network architecture for the MNIST digit pattern recognition and the methods used for its implementation.

A. Neuron Model

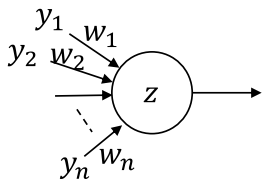


Fig. 1. Generic neuron model

We utilize the generic Bayesian neuron model proposed in [20] as the starting point and simplify the overall computation model as mentioned in [21]. For this model, we propose a stable and simplified STDP rule for efficient online learning. Bayesian model has two computational stages as compared to the regular integrate and fire neuron. The first being the exponential function and the other being the Poisson firing. In the generic neuron model as shown in Fig. 1, the membrane potential $u(t)$ of neuron Z is computed as

$$u(t) = w_0 + \sum_{i=1}^{n} w_i \cdot y_i(t) \qquad (1)$$

where w_i is the weight of the synapse connecting Z to its i^{th} presynaptic neuron y_i, $y_i(t)$ is 1 if y_i issues a spike at time t, and w_0 models the intrinsic excitability of the neuron Z. The stochastic firing model for Z, in which the firing probability depends exponentially on the membrane potential, is expressed as

$$prob(Z\ fires\ at\ time\ t) \propto \exp(u(t)) \qquad (2)$$

In (1), small variations of $u(t)$ resulting from the synaptic weight changes will have an exponential impact on the firing probability, which is not desirable. A range mapping function is proposed as detailed in [22] to mitigate this effect. But this introduces additional complexities.

Since the probability of the Bayesian neuron's output firing rate has an exponential dependence on the membrane potential, the computation must be limited to a small region of the exponential curve. This keeps the neuron computation within its dynamic range, else it will saturate quickly or the firing probability builds up extremely slowly. This small region of exponential curve can be safely approximated with a linear equation. However, the accumulation of weighted spikes approximates a linear function therefore a good dynamic range can be achieved.

For a Bayesian neuron output firing pattern resembles a Poisson process. To model this, we randomly vary the threshold after every spike generation. By limiting the range of threshold change to a small interval which satisfies the exponential distribution with unit rate, we achieve a firing pattern which is similar to Poisson spiking behavior as described in the model. With these simplifications and approximations, we can replace the Bayesian neuron model with an Integrate and Fire neuron with stochastic threshold model. The general behavior of neuron is still similar to the Bayesian neuron model even with these simplifications. Now the membrane potential $u(t)$ of neuron Z is computed as

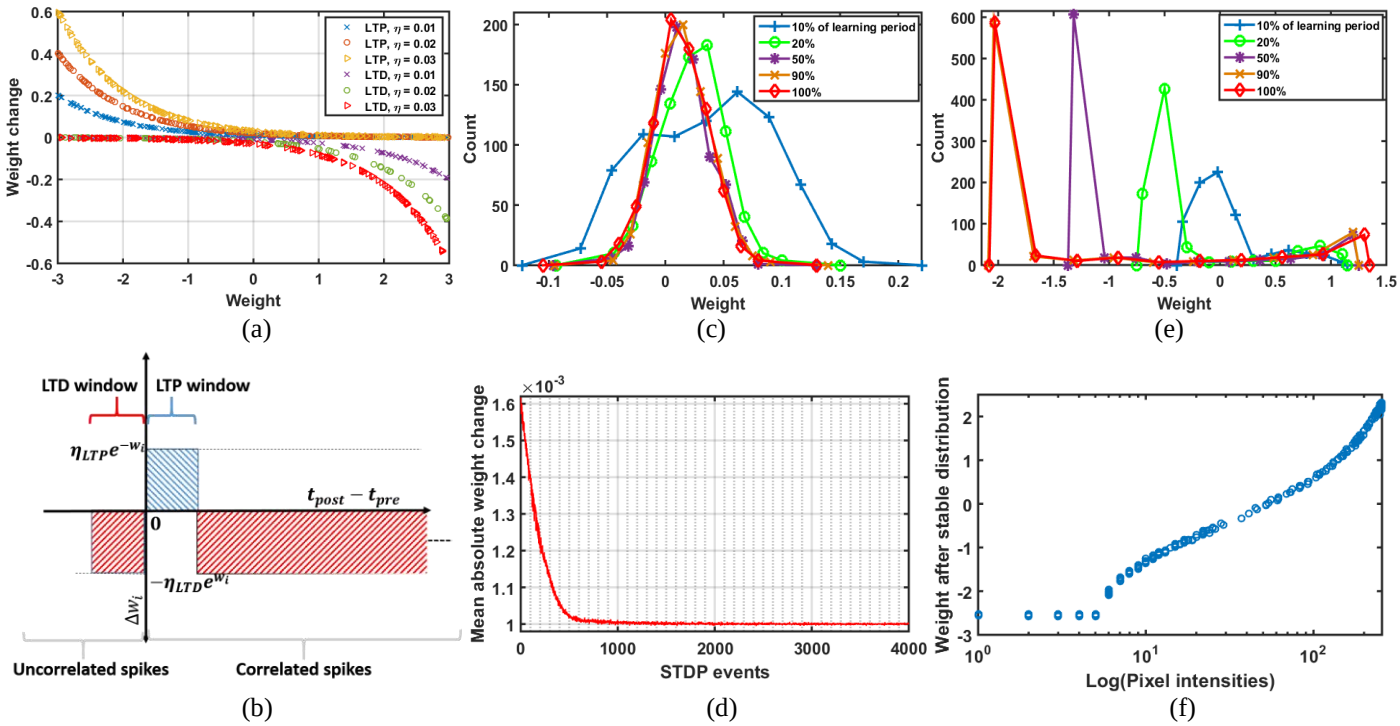


Fig. 2.(a)) Current weight vs Weight change for learning rates (b) STDP windows (c) Stable unimodal distribution over period of learning (20, 60 and 100% of final distribution) (d) Convergence of mean absolute weight change (e) Bimodal distribution over period of learning (f) Learnt weights for pixel intensities

$$u(t) = w_0 + u(t-1) + \sum_{i=1}^{n} w_i \cdot y_i(t) \quad (3)$$

The neuron Z spikes when the membrane potential crosses the threshold and is set to 0 (reset potential).

B. *Stable STDP Rule*

STDP forms the basis for learning in our synapse model . We use a multiplicative STDP rule where the amount of weight increase scales inversely with present weight size. Thus, learning is inherently stable and robust producing a unimodal weight distribution. But it lacks synaptic competition which is an attractive feature which enables learning discriminative features in the input. In sharp contrast, in additive STDP the weight change is independent of the current weight producing a bimodal distribution of weights and strong competition. But without any hard weight constraints, the learning is fragile and unstable.

We model our multiplicative STDP rule such that weight change of a synapse has an exponential dependence on its current weight as shown in Fig. 2(a). Thus, in the text we refer to this rule as the Exp rule. Update for the weight w_i of i^{th} synapse of the neuron Z from (3) is calculated as below.

$$\text{If } t_{post} - t_{pre} < \tau_{LTP}$$
$$\text{then, } \Delta w_i = \eta_{LTP} e^{-w_i}, w_i = w_i + \Delta w_i \quad (4)$$

$$\text{If } t_{post} - t_{pre} > \tau_{LTP} \text{ or } t_{pre} - t_{post} < \tau_{LTD}$$
$$\text{then, } \Delta w_i = \eta_{LTD} e^{w_i}, w_i = w_i - \Delta w_i \quad (5)$$

Where t_{post} and t_{pre} are the time steps at which the pre and post-synaptic neuron spikes, τ_{LTP} and τ_{LTD} are the LTP and LTD window and η_{LTP} and η_{LTD} are the LTP and LTD learning rates respectively. The intrinsic excitability w_0 of neuron Z from (3) is potentiated when neuron Z fires and depressed when it doesn't fire with the same exponential weight dependency as for the synaptic weights. These updates are linearly added to their current counterpart.

Plasticity is implemented with LTP and LTD windows as shown in Fig. 2 (b): when a postsynaptic spike occurs after a presynaptic spike and is within the LTP window, the synapse is potentiated according to (4). We assume all the STDP events to be independent such that only the first postsynaptic spike causes potentiation on that synapse even when the subsequent postsynaptic spikes are within the LTP window. If the postsynaptic spike falls outside the LTP window or there is no presynaptic spike, then the synapse is depressed as per (5). And similarly, only the first presynaptic spike within the LTD window after a given spike depresses the synapse; subsequent presynaptic spikes do not depress the synapse further before another postsynaptic spike occurs. Whereas a presynaptic spike outside the LTD window neither potentiate nor depress the synapse.

Ubiquity and fidelity of STDP as a general learning rule [23] [24] has regularly been in question despite it being biologically realistic. Phenomenological STDP rules that have simple biological precedence such as rate-based and spike-timing based models, are inherently unstable [13]. For proper utilization, they demand complementary mechanisms. Biologically-inspired mechanisms and ad-hoc mechanisms such as weight constraints, weight normalization [16], firing rate normalization and homeostasis [14] are usually used. These add computational complexity when simplicity is necessary especially for an efficient and large-scale implementation.

The above mentioned complementary mechanisms are not required for our STDP rule presented in (4) and (5) as it is inherent stable. The stability of a STDP rule can be shown with three main properties [25] [26]: (a) shape of the weight distribution is stable over time such that even if the synaptic weights change, the distribution follow similar pattern, (b) unimodal distribution such that all the weights are not concentrated at the boundaries and (c) limited weights without hard weight constraints such that no synaptic weights explode. We check our STDP rule for these properties through simulations to verify its stability empirically.

Fig. 2 (c) shows the distribution of synaptic weight during different learning stages when the input to the network with one SIF neuron and 28x28 input neurons is random and uncorrelated, and the synapse is updated through our STDP rule. Pertaining to the properties of a stable STDP rule. Fig. 2 (c) shows that our method resonates those properties very closely. The distribution achieved is unimodal, and its shape is stable over the period of learning. And the weights are naturally constrained between soft limits imposed by the formulation of the rule itself and not with any complimentary mechanism. Fig. 2 (a) shows that the weight updates are exponentially proportional to the current weights during both LTP and LTD; If the current weight is highly positive, then it is penalized with a lower weight update for LTP case (larger weight update for LTD case). On the contrary if the current weight is highly negative, then larger weight update is produced for LTP case (lower weight update for LTD case). Because of this weight dependence, strong synapses experience a net depression, whereas weak synapses experience a net potentiation whose magnitudes are controlled separately through separate learning rates for LTP and LTD in our STDP rule. This net depression and potentiation confines the synaptic weights and stabilizes the weight distribution. As the mean absolute weight change converges asymptotically, distribution reaches an equilibrium and becomes stationary. We can see that in Fig. 2 (c) where the change in weight distribution decreases and becomes stationary as the mean absolute weight change shown in Fig. 2 (d) converges and remains in that equilibrium.

Usually in weight-dependent STDP models, there is a lack of competition [13]. Competition between inputs allows for a specific set of input synapses to drive a neuron into firing. This is important for discriminative applications. In STDP models in which potentiation and depression are independent of the synaptic weight, there is strong competition [12] [27]. In these models of constrained plasticity, the potentiation mainly occurs if the input has caused the spike. When one input starts driving the postsynaptic spikes and its weight increases, the other inputs will become less correlated with the postsynaptic spikes, and these inputs will effectively be depressed. This pushes the distribution of the weights towards the applied hard constraints forming a bimodal distribution.

To replicate such competitive behavior amongst the inputs and still maintain a stable distribution, [13] utilizes Activity-dependent scaling of the synaptic weights. Activity-dependent scaling is a homeostatic mechanism in which the neuron reacts to changes in the postsynaptic activity, scales all synapses to keep the activity of the neuron within bounds. The scaling is multiplicative. If one synapse is potentiated, the postsynaptic activity rises, and the activity-dependent scaling kicks in to reduce all the synaptic weights. The shape and stability of the weight distribution are not affected by the scaling.

In our STDP model, behavior similar to activity-dependent scaling is induced through the elongated symmetric STDP window used as shown in Fig. 2 (b). LTD not only happens when the presynaptic spike falls in the LTD window but also when the postsynaptic spike falls outside LTP window or there is no presynaptic spike at all. So, when a certain synapse is potentiated due to strong correlation, synapses with weak correlation are depressed along with synapses with no correlation. This induces good competition and learns discriminative separation even between correlated inputs. When trained on a MNIST image, this induces good competition producing a distribution similar to that of the input or a bimodal distribution with sparse strong synapses and dense weak or silent synapses [28] as seen in Fig. 2 (e). And Fig. 2 (f) shows that higher pixel intensities induce higher weights whereas lower pixel intensities induce lower to negative weights in corresponding synapses driving weights to clusters in opposite ends reiterating the sharpness of the discriminative ability.

One important feature of the Bayesian neuron model is that it allows for neural sampling [20] such that the weight of the synapse is the log conditional probability of pre-synaptic neuron firing given the post-synaptic neuron has fired with a log constant ($\log P(y = 1 \mid Z = 1) + \log c$) and each spike is a sample of the posterior distribution. This allows for Bayesian inference. Using an exponential dependence of weight update on the current weight allows to retain the log conditional property as shown empirically by the correlation graph in Fig. 3. As the theoretical results remain true [20] for any $c > 1$, we choose $c = 30$. Under this condition the correlation coefficient of the synaptic weight and the log conditional probability ($\log P(y = 1 \mid Z = 1) + \log 30$) is 0.9885. This also accounts for learnt weights being positive and negative.

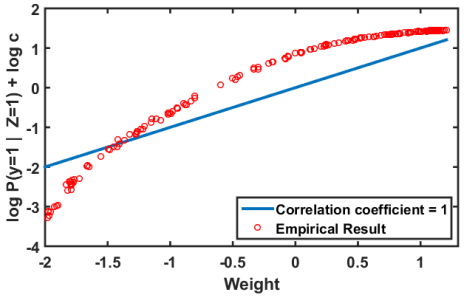


Fig. 3. Correlation graph between synaptic weight and log conditional probability with constant c=30

C. *Quantized 2-Power Shift Rule*

In terms of computation, the proposed STDP rule requires an exponential and a multiplication operation for both LTP and LTD for each synapse. From the perspective of efficient digital hardware implementation these are expensive operations in terms of circuit area and computation time as explained in [29].Thus, substituting these with simplified operations is highly desirable. In the next, we introduce a Quantized 2-power shift rule (Q2PS), which approximates our STDP rule in (4) by removing both multiplication and exponential. The approximation is summarized in (6) and (7).

If $t_{post} - t_{pre} < \tau_{LTP}$

$$\begin{aligned} \Delta w_i &= \eta_{LTP} 2^{-w_i} \\ &= 2^{\eta'_{LTP} - w_i} \end{aligned} \tag{6}$$

where $\eta'_{LTP} = \log_2 \eta_{LTP}$.

Similarly, If $t_{post} - t_{pre} > \tau_{LTP}$ or $t_{pre} - t_{post} < \tau_{LTD}$

$$\begin{aligned} \Delta w_i &= \eta_{LTD} 2^{w_i} \\ &= 2^{\eta'_{LTD} + w_i} \end{aligned} \tag{7}$$

where $\eta'_{LTD} = \log_2 \eta_{LTD}$.

We denote $Q = \eta'_{LTP} - w_i$ for LTP and $Q = \eta'_{LTD} + w_i$ for LTD, also let $\bar{Q}$ be the quantization of Q through priority encoding. This encoding converts the binary representation of Q into a new binary representation with the index of the most significant active input bit as the highest priority. For example, if $Q = 12$ then its binary representation is 1100. The priority encoding of this is 1000 hence $\bar{Q} = 8$. After the quantization, the change of the synaptic weight is calculated by shifting the value 1 by $\bar{Q}$, either left of right, based on the sign of that result. In other words, for both cases

$$\Delta w_i = \begin{cases} 1 \ll |\bar{Q}|, & if\ \bar{Q} > 0 \\ 1 \gg |\bar{Q}|, & if\ \bar{Q} < 0 \end{cases} \tag{8}$$

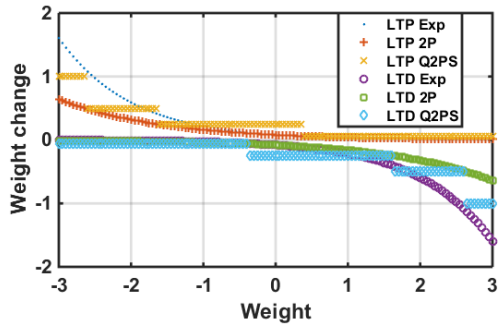


Fig. 4. Comparison of the introduced STDP rules

where $\ll$ and $\gg$ represent binary shift left and shift right operations respectively. This approximation allows implementation of the STDP rule presented in (4) and (5) on digital hardware by using a priority encoder, negligibly small lookup to determine $|\bar{Q}|$ from the encoded value, barrel shifter and an adder circuit. Please note that, based on (6) and (7), Δw_i should be calculated as 2^Q, which can be obtained by shifting value 1 by Q. We refer to this as 2P approximation. However, because the value of $\bar{Q}$ has much coarser resolution than Q, the implementation of Q2PS approximation is much simpler than the 2P approximation. We refer to the original STDP rule in (4) and (5) as Exp rule. Fig. 4 compares the Δw_i calculated using the *Exp*, *2P* and *Q2PS* rules, with a learning rate of 0.08 for all the cases. As we can see, the Q2PS rule provides multi-level quantization, which enables similar quality of trained weights even with approximations when compared to Exp rule. This way we can perform online learning in a quick and efficient manner from the hardware perspective with a smaller footprint in terms of circuit area and energy consumption.

IV. Experiments

We build a spiking neural network to classify MNIST [30] handwritten digits and perform analysis of our proposed STDP learning rules. Both supervised learning and unsupervised learning for training the entire network including the classifier layer is discussed. We use a SNN simulator SpNSim [22] to simulate all the experiments presented in the paper. SpNSim is a multithreaded and scalable simulation platform built using C++. It is flexible and vectorized for efficient evaluation and capable of handling large-scale networks of mixed neuron models.

A. Network Architecture

We create a 3-layer network as shown in Fig. 5. The input layer contains 28x28 neurons (one per pixel), the second layer has variable number of stochastic integrate and fire (SIF) neurons for different trials (the hidden layer), and the third layer is the classification layer with 10 SIF neurons (one per class). The input is fed to the input layer which encodes the pixel intensities with 0 Hz – 300 Hz firing rates and the classification result is obtained from the classification layer. Input layer is fully connected to the hidden layer, and hidden layer to the classification layer and all these synapses are plastic.

Along with the SIF neurons, hidden and the classification layer neurons there are supporting ReLU neurons [22] for lateral inhibition. At both layers, the ReLU neurons (inhibitory) form WTA (Winner Take All) circuits with a connectivity as mentioned in [22]; one-to-one connection from SIF neurons to the ReLU neurons and the ReLU neurons are connected to all the SIF neurons except for the one from which it receives a connection. Hard or soft WTA behavior can be achieved based on the degree of inhibition delivered. *Hard WTA* happens when the inhibition is strong such that it brings down the firing rate of the non-preferred SIF neurons to zero, resulting in only one neuron with highest excitation being active. On the other hand, if plural voting action is required within the set, the degree of inhibition is tuned to be moderate. This makes SIF neurons fire with different stable rates which is, *soft WTA* behavior where firing rate is proportional to their relative excitation levels.

The hidden layer learns features of the MNIST images and more than one neurons could learn feature concerning a certain class, thus requiring multiple neurons in the layer to be firing. Thus, the hidden layer is set to a soft WTA inhibition level. Whereas in the classification layer, a neuron is associated with a class in a one-to-one basis thus requiring only one neuron to fire in a period. Hence the classification layer is set to a hard WTA inhibition level. And these synapses are not plastic.

The stochasticity of the SIF neuron is important during learning to allow the neurons in both layers learn unique features. But having similar stochasticity during recall becomes counterproductive as inappropriate inputs could excite a certain feature in the hidden layer, and similarly incorrect feature could excite an incorrect neuron in the classification layer producing an incorrect classification. Thus, during the recall phase, we disable learning and fix each neuron's spiking threshold to make them deterministic.

B. Learning

Using our STDP rule, we perform both unsupervised and supervised learning. STDP is known to have the effect of concentrating high synaptic weights on afferents that systematically fire early. It makes neurons naturally selective to patterns that are reliably present in the input. These patterns are learnt without supervision and can be used for categorization and discrimination. In this way, the hidden layer learns patterns from the input layer as shown in Fig. 5.

For classification purposes, supervision is necessary to label the neurons that have learnt discriminative capabilities thus categorization is in order during recall. In [14], the feature learning neurons are manually labeled after training and their collective firing rate are used to classify digits. This methodology creates issues in cases when the learnt feature is not of a distinguishable class or it is common to several classes e.g. slanted "1" is a sub feature of "7", "3" has feature elements common to "5" and "8" and so on. Thus, labeling these features to one class or the other could lead to mislabeling of undistinguished features or underutilization of a common feature. To avoid such situations, we add a classification layer which learns intermediate features from all the available learnt features in the hidden layer without having to manually label them. The classification layer has one neuron per class to perform categorization such that the neuron with the highest firing rate is the predicted class. And to train the classification layer, we introduce a supervised learning approach using our STDP rule on top of the already trained hidden layer.

During supervised training in ANN, the neurons in the classification layer are driven by the class labels and the error is propagated down the network layers. Here we add a neuron per class which fires at a specific rate based on the class being observed. We call this Teacher neuron. These neurons are connected one-to-one to the classification layer as shown in Fig. 5. They excite one neuron per class to fire at a specific rate when the label is presented. The teacher neurons to classification layer neuron synapses are not learnt. However, it excites the connected classification neuron and creates differentiable time dependent relations between the classification layer and the hidden layer. Based on our STDP rule, active input synapses from the classification layer to the hidden layer are potentiated and inactive ones are depressed, thus propagating the error down through the incoming synapses

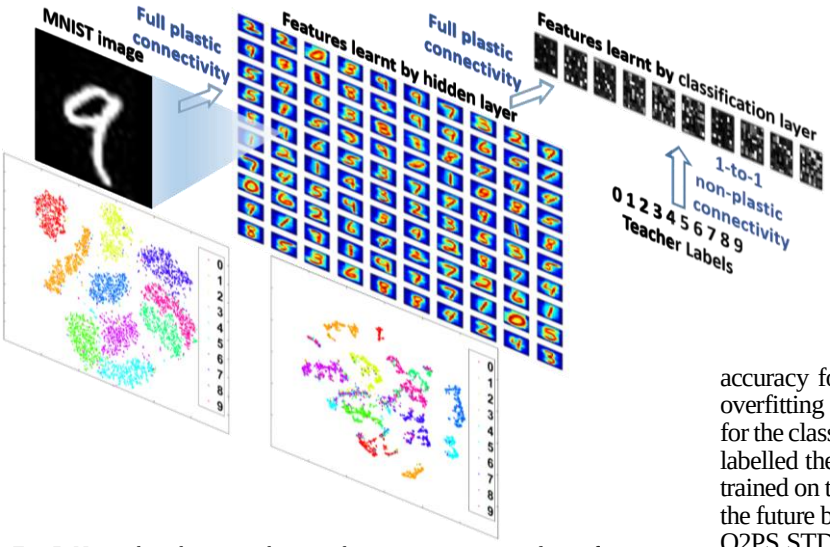


Fig. 5. Network architecture showing the connectivity, input, learnt features by hidden and classification layer, the labels and t-SNE visualizations

V. Results and Discussions

In this section, we present the results on the handwritten digits classification. We also analyze the performance for different sets of parameters to present rationale on how to tune the network, and what could be adapted in case of noise and variable intensities in input images. Finally, we discuss the quality of the features learnt.

A. Handwritten Digits Classification

We trained a 3-layer network as presented in section IV with 10 classification neurons and with 3 variations of hidden layers consisting of 100, 400 and 1600 neurons. The training is done layer-wise; the input to hidden layer synapses are trained without supervision by presenting the complete training set once, and then the hidden to classification layer synapses are trained with supervision on top of the trained hidden layer by presenting the complete training set again. Each input neuron is connected to one pixel in the image and fires at a rate (maximum rate is 300Hz) that is proportional to the pixel intensity. No preprocessing is done on the images and we present each training image for 200 time steps during training. The learnt features and hidden to classification layer synaptic weights are shown visualized in Fig. 5. The same parameters are used throughout all these experiments.

Fig. 5 also shows the t-SNE [31] visualizations for the input layer and the hidden layer which provides a qualitative analysis. This technique performs t-distributed stochastic neighbor embedding which maps high-dimensional data that lie on several different, but related, low-dimensional manifolds to lower dimensions by capturing local structure of the high-dimensional data. In our case 784 dimensions of input layer and 100 dimensions of feature layer are mapped to 2 dimensions respectively. These visualizations are made using the firing rates of all the neurons in that layer. It is clear from the figure that the proposed STDP rule can form tight clustering of input space when mapping it into the feature space. This generates features that can be more easily classified by the classification layer.

The three networks; 100, 400 and 1600 hidden neurons, achieved an average classification accuracy of 85, 87.4 and 89.7% for our STDP rule, respectively, as shown in Fig. 6. We achieve better classification accuracy for networks with 100 and 400 hidden neurons compared to [14] for their respective network sizes, whereas for 1600 hidden neurons our results are slightly lower. This lower

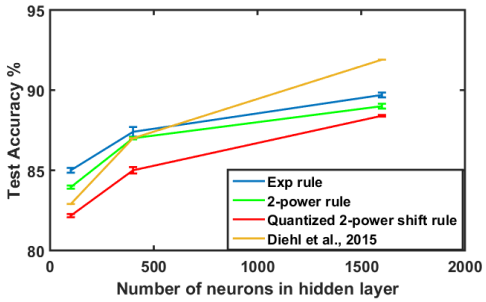


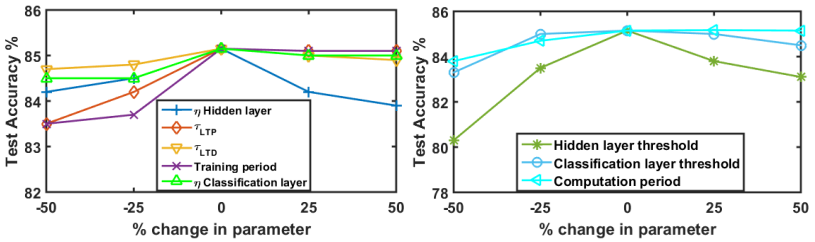
Fig. 6. Test accuracies for different sized networks

accuracy for high number of hidden neurons can be attributed to overfitting due to large number of trainable parameters per neuron for the classification layer. [14] don't face this issue as they manually labelled the features for classification instead of classification layer trained on top of those learnt features. We could resolve this issue in the future by using dropout while training. The accuracies of 2P and Q2PS STDP rules for these networks are also shown in Fig. 6. As can be seen from the results, even with approximations and quantization, the Q2PS rule achieves accuracies reasonably close to the Exp STDP rule.

B. Sensitivity Analysis

Even with a simplified STDP rule and SIF neuron model, several parameters involved during learning and recall. As the networks are stochastic and no mathematical measures are present to pinpoint these parameters, it is important to empirically fine-tune them. This step is analogous to the process of selecting the hyper-parameters such as learning rate, momentums, etc., for a conventional artificial neural network. The empirical exploration of the parameter space was done for individual parameters within a sensible range through multi-model simulations on SpNSim to achieve optimal parameters used for all training and testing. This approach is similar to grid search. To show these parameter's impact on the quality of learning, we present each parameter's sensitivity relative to the optimal parameters in separate sensitivity graphs for training and testing parameters in Fig. 7 using the 3-layer MNIST classification SNN with 100 hidden neurons as an example. The rationales on how to configure those parameters and their specific sensitivity are also discussed.

1) Training parameters: STDP windows τ_{LTP} and τ_{LTD}, learning rates η_{LTP} and η_{LTD} and the training period for each input image are the important parameters during the training. Learning rates control the step size of weight updates. Larger step size leads to faster learning but also faster forgetting of the learnt features . Hence can result in overfitting of features. Given a training period, if the learning rates of LTP and LTD are equal and small (~10^{-3}), it learns and retains features without overfitting. In Fig. 7 (a), while increasing the learning rates, the performance increases, peaks at a certain rate and then start decreasing because of overfitted features. The quality of the learnt features on that peak learning rate value is discussed in Section V.E. Having higher LTP window allows more opportunities for potentiation. As our STDP rule is setup for competition among inputs, lower LTP window leads to high net depression. Thus we set the LTP window (50 time steps) higher than the LTD window (10 time steps). The LTP window determines how strongly correlated the input must be to be potentiated whereas the LTD window determines how strongly uncorrelated the input must be to be depressed. As discriminative features are present in correlated inputs, the impact of LTP window is more which is visible in Fig. 7 (a). The training period per image is set such that weight updates with small step sizes have enough time to capture features



(a) (b)

Fig. 7. Sensitivity of recognition accuracy for % change in (a) training and (b) testing parameters from the optimal parameters. Only one parameter is varied at a time from base values for 100 hidden neuron network

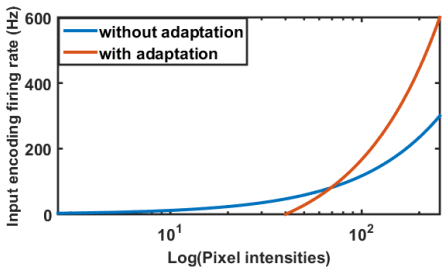


Fig. 8. Firing rate curve with and without adaptation for different pixel intensities

from an input. In Fig. 7 (a), we also see that at training period of 200 time steps, the performance saturates.

2) Testing parameters: Hidden and classification layer neuron thresholds and the computation period for each image are parameters during the testing. Decreasing and increasing threshold increases and decreases the firing rates respectively. Lower thresholds make the neurons extremely sensitive thus may allow unnecessary features/classifier neuron to also fire whereas very high thresholds might mean less sensitivity such that even a favorable feature/classifier neuron may not be able to gather enough excitation to fire in the testing period. As the hidden layer is exposed to the input, high sensitivity of a neuron in that layer is exploited by similarity between different inputs. Thus in Fig. 7 (b) we see a sudden drop in the performance with decrease in hidden layer threshold whereas only a small drop for decrease in classification layer threshold. Lower computation period decreases the latency from input to a classification result but leads to less time to resolve the result. Higher computation period increases the latency but allows for enough time to resolve the results as the inputs are rate coded. So, we use a computation period where the accuracy starts saturating as seen in Fig. 7 (b).

C. Classifying Images with White Noise

Background clutter is a common form of noise which reduces the effective contrast in a real-world image inputs. This reduces the discriminative properties between images. To test the resilience of our network at different levels of background clutter, we create artificial MNIST test sets with Additive White Gaussian Noise (AWGN) of different SNRs (15, 20 and 25). The results are shown in TABLE I.

TABLE I. PERFORMANCE WITH AND WITHOUT ADAPTION IN PRESENCE OF AWGN NOISE

AWGN noise SNR	*Without adaptation*	*With adaptation*
15	19.1%	34.5%
20	40.9%	83.9%
25	78.6%	85%

With the current temporal sparseness (maximum rate 300 Hz) for encoding the input, the AWGN noise has a significant impact on the accuracy. In biology, the LGN neuron in the early visual pathway employs an adaptive strategy [32] which decreases the temporal sparseness (increasing the firing rate) and shifts stimulus–response curves toward higher stimulus intensities when the effective contrast is reduced due to white noise. This reduces the spike-timing jitters induced by lower effective contrast by increasing the level of discrimination (in terms of firing rate) between similar stimuli and ignores low intensity stimuli in the background. We apply a similar adaptation strategy to deal with the AWGN noise. As shown in Fig. 8, we shift the intensity-firing rate (stimulus-response) curve towards higher intensities and increase the range of the firing rate to 600 Hz. This leads to a dramatic improvement in accuracy in the presence of AWGN noise. But this comes with the decrease in sparsity and reduction in sensitivity to lower intensity discriminative properties.

D. Classifying Images with Variable Intensity

Resistance to variation in the image is important. The most common variation is the intensity of the image. In real life, lighting conditions result in different intensity images. Thus, we create artificial MNIST test sets with 25% intensity, 50% intensity and mixed intensity to test our network on. The results are shown in TABLE II.

This leads into the discussion of when normalization becomes useful in our network. Normalization mechanisms allow for intrinsic modulation of the firing rates of the neurons in a layer; increase if the rate is low, and decrease if the rate is high. Thus, keeping all the neurons approximately within a range of firing rates. With low intensity images, there can be a propensity of a lack of excitation for the neurons in the hidden layer to fire and thus for the classification layer. This negatively impacts the classification layer's ability to accurately categorize the image's class.

Normalization mechanisms such as homeostasis [14] and weight normalization or scaling [6] [33] add complexities onto the neuron model and the learning rule. With our aim of simplicity, we utilize the normalized winner-take-all (NWTA) mechanism as described in [21]. It adds three additional neurons in the hidden layer; *upper limiter* (*UL*), *lower limiter* (*LL*), and *exciter* (*Ex*), which checks if the upper limit of the desired firing rate range is reached and increases lateral inhibition, checks if the lower limit of the desired firing rate is reached and provides additional excitation, respectively. We incorporate NWTA in the hidden layer as low intensity image directly impacts its firing rates. The results with and without NWTA mechanism is shown in TABLE II. for a network with 100 neurons in the hidden layer averaged over three experiments.

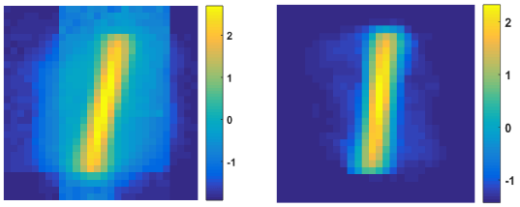
TABLE II. PERFORMANCE WITH AND WITHOUT NWTA FOR DIFFERENT INTENSITY MNIST IMAGES

Intensity level	*Without NWTA*	*With NWTA*
25%	62.9%	79.7%
50%	81.5%	84.1%
100%	85%	84.9%
Mixed	82.7%	84.4%

From TABLE II, we can see that normalization results in improvement for lower intensity images whereas for normal intensity images normalization is redundant as the network was trained using normal intensity images.

E. Quality of the learnt features

During unsupervised learning, neurons in the hidden layer get associated with a class without supervision and learn common patterns of that class. These features, along with being discriminative, should also encapsulate variations of the associated class as well.



(a) (b)

Fig. 9. Example of (a) non-overfit feature and (b) overfit feature

For example, a color map of two features of class "1" is shown in Fig. 9. The feature of class "1" in Fig. 9 (b) is overfit to a particular variation of "1" in the MNIST training set. It will not contribute to or will negatively impact the classification when a different variation of "1" is presented, because it can lead to a lack of excitation for the neurons in the hidden layer to fire. Whereas in Fig. 9 (a) the feature has a strict core of "1" but a fuzziness around it such that the feature contributes to the classification of different variation of "1". That fuzziness could be considered as the remnant of the other 1's that the neuron has been exposed to. When we keep the learning rates low and equal for LTP and LTD (10^{-3}), and increase the length of the LTP window (50 time steps) as compared to the LTD window (10 time steps), it provides the neuron more opportunities to learn other variations of "1" and only allows it to forget them slowly. Hence, leaving the fuzzy remnant.

VI. Conclusion

In this paper, we presented a simplified neuron model and introduced a stable and competition inducing STDP learning rule. With simplicity in terms of computation and hardware implementation being a motivating factor we also introduced Q2PS rule which can be implemented with minimal digital hardware resources. We also combined unsupervised and supervised learning with STDP using a layer-wise training approach, applied the network for handwritten digit classification with competitive results and provided analysis for parameter tuning.

References

[1] F. Javed, Q. He, L. E. Davidson, J. C. Thornton, J. Albu, L. Boxt, N. Krasnow, M. Elia, P. Kang, S. Heshka and others, "Brain and high metabolic rate organ mass: contributions to resting energy expenditure beyond fat-free mass," *The American journal of clinical nutrition*, vol. 91, pp. 907-912, 2010.

[2] R. Ananthanarayanan, S. K. Esser, H. D. Simon and D. S. Modha, "The cat is out of the bag: cortical simulations with 109 neurons, 1013 synapses," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, 2009.

[3] W. Maass, "Networks of spiking neurons: the third generation of neural network models," *Neural networks*, vol. 10, pp. 1659-1671, 1997.

[4] H. Z. Shouval, M. F. Bear and L. N. Cooper, "A unified model of NMDA receptor-dependent bidirectional synaptic plasticity," *Proceedings of the National Academy of Sciences*, vol. 99, pp. 10831-10836, 2002.

[5] G. C. Castellani, E. M. Quinlan, F. Bersani, L. N. Cooper and H. Z. Shouval, "A model of bidirectional synaptic plasticity: from signaling network to channel conductance," *Learning \& Memory*, vol. 12, pp. 423-432, 2005.

[6] P. U. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu and M. Pfeiffer, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *2015 International Joint Conference on Neural Networks (IJCNN)*, 2015.

[7] P. O'Connor, D. Neil, S.-C. Liu, T. Delbruck and M. Pfeiffer, "Real-time classification and sensor fusion with a spiking deep belief network," *Neuromorphic Engineering Systems and Applications*, p. 61, 2015.

[8] W. Gerstner and W. M. Kistler, Spiking neuron models: Single neurons, populations, plasticity, Cambridge university press, 2002.

[9] R. Kempter, W. Gerstner and V. a. L. J. Hemmen, "Hebbian learning and spiking neurons," *Physical Review E*, vol. 59, p. 4498, 1999.

[10] S. Song, K. D. Miller and L. F. Abbott, "Competitive Hebbian learning through spike-timing-dependent synaptic plasticity," *Nature neuroscience*, vol. 3, pp. 919-926, 2000.

[11] S. Sengupta, K. S. Gurumoorthy and A. Banerjee, "Sensitivity Analysis for additive STDP rule," *arXiv preprint arXiv:1503.07490*, 2015.

[12] M. Gilson and T. Fukai, "Stability versus neuronal specialization for STDP: long-tail weight distributions solve the dilemma," *PloS one*, vol. 6, p. e25339, 2011.

[13] V. a. M. C. W. Rossum, G. Q. Bi and G. G. Turrigiano, "Stable Hebbian learning from spike timing-dependent plasticity," *The Journal of Neuroscience*, vol. 20, pp. 8812-8821, 2000.

[14] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers in computational neuroscience*, vol. 9, 2015.

[15] T. Masquelier and S. J. Thorpe, "Unsupervised learning of visual features through spike timing dependent plasticity," *PLoS Comput Biol*, vol. 3, p. e31, 2007.

[16] N. Levy, D. Horn, I. Meilijson and E. Ruppin, "Distributed synchrony in a cell assembly of spiking neurons," *Neural networks*, vol. 14, pp. 815-824, 2001.

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

[18] N. Kasabov, K. Dhoble, N. Nuntalid and G. Indiveri, "Dynamic evolving spiking neural networks for on-line spatio-and spectro-temporal pattern recognition," *Neural Networks*, vol. 41, pp. 188-201, 2013.

[19] E. Neftci, S. Das, B. Pedroni, K. Kreutz-Delgado and G. Cauwenberghs, "Event-driven contrastive divergence for spiking neuromorphic systems," 2013.

[20] B. Nessler, M. Pfeiffer, L. Buesing and W. Maass, "Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity," *PLoS Comput Biol*, vol. 9, p. e1003037, 2013.

[21] K. Ahmed, A. Shrestha, Q. Qiu and Q. Wu, "Probabilistic inference using stochastic spiking neural networks on a neurosynaptic processor," in *Neural Networks (IJCNN), 2016 International Joint Conference on*, 2016.

[22] K. Ahmed, A. Shrestha and Q. Qiu, "Simulation of bayesian learning and inference on distributed stochastic spiking neural networks," in *Neural Networks (IJCNN), 2016 International Joint Conference on*, 2016.

[23] L. F. Abbott and S. B. Nelson, "Synaptic plasticity: taming the beast," *Nature neuroscience*, vol. 3, pp. 1178-1183, 2000.

[24] J. Lisman and N. Spruston, "Questions about STDP as a general model of synaptic plasticity," *Spike-timing dependent plasticity*, vol. 26, p. 53, 2010.

[25] F. S. Matias, P. V. Carelli, C. R. Mirasso and M. Copelli, "Self-organized near-zero-lag synchronization induced by spike-timing dependent plasticity in cortical populations," *PloS one*, vol. 10, p. e0140504, 2015.

[26] K. S. Burbank and G. Kreiman, "Depression-biased reverse plasticity rule is required for stable learning at top-down connections," *PLOS Comput Biol*, vol. 8, p. e1002393, 2012.

[27] J. T. A. Kepecs, "Why neuronal dynamics should control synaptic learning rules," in *Advances in Neural Information Processing Systems 14: Proceedings of the 2001 Neural Information Processing Systems (NIPS) Conference*, 2002.

[28] J.-n. Teramae and T. Fukai, "Computational implications of lognormally distributed synaptic weights," *Proceedings of the IEEE*, vol. 102, pp. 500-512, 2014.

[29] K. Ahmed, A. Shrestha, Y. Wang and Q. Qiu, "System Design for In-Hardware STDP Learning and Spiking Based Probablistic Inference," in *VLSI (ISVLSI), 2016 IEEE Computer Society Annual Symposium on*, 2016.

[30] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278-2324, 1998.

[31] L. v. d. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579-2605, 2008.

[32] N. A. Lesica, J. Jin, C. Weng, C.-I. Yeh, D. A. Butts, G. B. Stanley and J.-M. Alonso, "Adaptation to stimulus contrast and correlations during natural visual stimulation," *Neuron*, vol. 55, pp. 479-491, 2007.

[33] S. Afshar, L. George, C. S. Thakur, J. Tapson, A. van Schaik, P. de Chazal and T. J. Hamilton, "Turn Down That Noise: Synaptic Encoding of Afferent SNR in a Single Spiking Neuron," *IEEE transactions on biomedical circuits and systems*, vol. 9, pp. 188-196, 2015.