The Impact of Encoding-Decoding Schemes and Weight Normalization in Spiking Neural Networks

Zhengzhong Liang[a], David Schwartz[a], Gregory Ditzler[a], O. Ozan Koyluoglu[b]

[a]*Dept. of Elec. & Comp. Engineering, The University of Arizona, Tucson, AZ, USA.*
[b]*EECS, University of California, Berkeley, CA, USA.*

Abstract

Spike-timing Dependent Plasticity (STDP) is a learning mechanism that can capture causal relationships between events. STDP is considered a foundational element of memory and learning in biological neural networks. Previous research efforts endeavored to understand the functionality of STDP's learning window in spiking neural networks (SNNs). In this study, we investigate the interaction among different encoding/decoding schemes, STDP learning windows and normalization rules for the SNN classifier, trained and tested on MNIST, NIST and ETH80-Contour datasets. The results show that when no normalization rules are applied, classical STDP typically achieves the best performance. Additionally, first-spike decoding classifier requires much less decoding time than a spike count decoding classifier. Thirdly, when no normalization rule is applied, the classifier accuracy decreases as the encoding duration increases from 10 ms to 34 ms using count decoding scheme. Finally, normalization of output weights is shown to improve the performance of a first-spike decoding classifier.

Keywords: Spiking Neural Network, Spike-Timing Dependent Plasticity, Learning Window, Encoding, Decoding, Normalization.

Email addresses: `zhengzhongliang@email.arizona.edu` (Zhengzhong Liang), `dmschwar@email.arizona.edu` (David Schwartz), `ditzler@email.arizona.edu` (Gregory Ditzler), `ozan.koyluoglu@berkeley.edu` (O. Ozan Koyluoglu)

1. Introduction

Spiking Neural Networks (SNN) are the third generation of artificial neural networks that aim to emulate the biological activity of neurons while providing a parsimonious compromise in the natural trade-off between realism and computational complexity [1]. Different from traditional Artificial Neural Networks (ANN), SNN represent data in sequences of spikes [2], the impulses of a neuron's membrane potential. Signals can be encoded in several forms, including temporal sequences of spikes, the rate of emission of spikes, or other forms [2].

Spike-Timing Dependent Plasticity (STDP) is a biological learning mechanism observed in multiple species' neural systems, and it is believed that STDP can capture the causal relationship between events that are encoded by spikes [3]. The idea behind STDP is that the connection between two neurons is strengthened or weakened depending on the relative spike times of two neurons. If the pre-synaptic spike arrives before the post-synaptic spike, the connection is strengthened, a process known as long-term potentiation (LTP). However, if the post-synaptic spike arrives first then long-term depression (LTD) is induced: the connection is weakened. STDP dwells in a vast swathe of neural systems, including the hippocampus, cerebral cortex, cerebellum-like structures and retinotectal projection [4]. Recently, it was shown that STDP may exist in absence of LTD [5].

Earlier works focus on using a variety of particular implementations of STDP to train a SNN. In [6], the authors applied classical STDP to a SNN, for which images are converted to Poisson spikes and fed into the network. An accuracy of 95% on MNIST is achieved using one proposed configuration. However, this encoding scheme requires a relatively long time to encode input samples [6]. [7] proposed a variant of STDP which enables the network to learn input patterns encoded by precise times of spikes. Additionally, input signals that are encoded by arrival of the first spike can also be learned via STDP: In [8], a SNN emulating the human visual system is constructed in order to demonstrate fast feature detection. This work shows that first-spike encoding and decoding schemes coupled with unsupervised STDP enable the network to quickly detect visual features after training is complete. Later in [9], this network is augmented with a supervised STDP regulated by reward [10]. The modified network is then used for image classification. In addition to these comparisons of variants of STDP, multiple forms of STDP have been observed in biological experiments [11]. It is shown that the locations and types of synapses can largely influence the STDP learning windows. Further, normalization mechanisms have been proposed to account for global properties of synapse change [12].

In this work, we experiment with an SNN classifier to emulate the structure and coding scheme of the human visual system, and consider multiple STDP variants (including rewarded STDP) and normalization rules. In section 2, we formalize the framework for training a SNN classifier, including neuronal and synaptic dynamics, classifier configuration and performance evaluation. In Section 3, we present results obtained by training and testing with MNIST, NIST, and ETH80-Contour datasets. Several conclusions are drawn from the experimental results. Firstly, although a count-decoding scheme achieves a greater classification accuracy, it consumes more decoding time than first-spike decoding. Secondly, when no normalization rules are applied, the accuracy of the classifier under count-decoding scheme decreases if the encoding duration is too long (relative to the length of STDP learning window). This shows the importance of normalization in the context of SNN. Finally, choosing an appropriate normalization rule is shown to improve the classification performance of a first-spike decoding network.

2. Theoretical Framework

2.1. Spiking Neurons

We construct a spiking neural classifier from Leaky Integrate and Fire (LIF) units. This neural model coarsely mimics real neurons while maintaining a reasonable trade-off with computational complexity [13]. We prescribe the dynamics of this model as governed by (1)

$$\frac{\partial v}{\partial t} = \frac{(V_{\text{rest}} - v + E)}{\tau_m} \tag{1}$$

$$v = V_{\text{reset}}, \text{ if } v \geq V_t, \tag{2}$$

where v is membrane potential, V_{rest} is the resting potential of neuron, E is the post-synaptic potential evoked by a pre-synaptic spike (i.e., E is the increase in membrane potential produced by an input spike), τ_m is the membrane potential time constant, and V_t is the neuron's spiking threshold. The neuron emits a spike when v exceeds V_t and its membrane potential resets to V_{reset}. A neuron's membrane potential settles to V_{rest} at equilibrium (e.g., when it receives no pre-synaptic spikes).

There are several properties of membrane potential dynamics that emerge from (1). First, observe that a neuron's spiking is driven by its membrane potential. This spike can be stimulated by increasing E, an effect induced by the reception of input spikes. Additionally, note that choice of the parameter, τ_m, determines a neuron's excitability. If τ_m is large, then the neuron tends to be reluctant to vary its membrane potential. Conversely, when τ_m is small, even very small perturbations

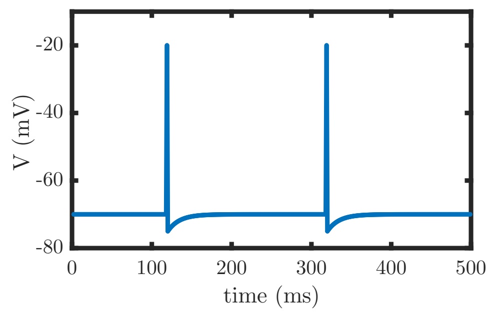


Figure 1: Depicted above is a trace of membrane potential (shown as a function of time) of a typical spiking neuron engaging in two spiking events

in E can produce spikes. Figure 1 demonstrates a spiking behavior emerging from the prescribed neuronal dynamics.

2.2. Synapse Dynamics and Plasticity

Synapse dynamics determine how pre-synaptic activity affects future post-synaptic spiking by adjusting connection strength in response to coactivity. We model synaptic dynamics with the Spike Response Model (SRM) [14]. A major assumption is that a pre-synaptic spike causes an exponentially decreasing post-synaptic voltage. STDP serves as the main rationale of synapse plasticity in our model. In addition to the STDP with the classical learning window, we also propose three new learning windows. Three normalization rules are also applied to our model, an augmentation that improves global stability of our network. Finally, inspired by the reinforcement learning found in the brain, we couple STDP and the normalization rules with a reward signal [10].

2.2.1. Synapse Dynamics

Equation 4 describes our dynamical model of synaptic transmission (i.e., the effect on post-synaptic potential induced by incoming spikes).

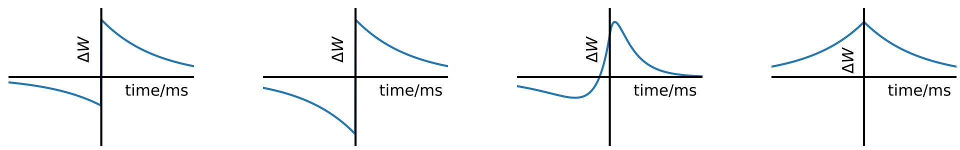
$$E_j = E_j + \alpha \sum_{i=1}^{I} w_{i,j} s_i, \quad \text{if a pre-synaptic spike is received} \tag{3}$$

$$\frac{\partial E_j}{\partial t} = -\frac{E_j}{\tau_n}, \qquad \text{otherwise} \tag{4}$$

where i and j are the indices for the pre- and post-synaptic neurons, respectively. E_j is the increase in post-synaptic potential evoked by the spike in question, I is the number of pre-synaptic neurons, $w_{i,j}$ is the strength of the connection from neuron i to neuron j, and s_i is an indicator function taking the value 1 when the pre-synaptic neuron spikes. α, a constant in our model, is included to incorporate the effect of synaptic resistance/conductance. In the absence of pre-synaptic spikes, E decays exponentially.

2.2.2. STDP Learning Windows

The learning window was described as asymmetrical when STDP was observed for the first time [4, 15]. Furthermore, recent work has reported that the learning window of STDP may show a symmetrical property [15]. The particular shapes of real learning windows are quite diverse [15, 4, 16, 5]. Experimental results have shown that learning window is not only affected by the relative arrival times of pre-synaptic spike and post-synaptic spikes, but also by inter-spike interval (ISI), spike pair pattern and the synapse type [17, 18, 19]. In this work, we consider the following four STDP learning windows.



(a) Classical STDP (b) STDP Variant I (c) STDP Variant II (d) STDP Variant III

Figure 2: Learning Windows of STDP Variants

Classical STDP.

$$\Delta w = \begin{cases} \mathrm{A_{pre}} \cdot \exp\left(-\frac{t_{\mathrm{post}} - t_{\mathrm{pre}}}{\tau_{s1}}\right), & t_{\mathrm{post}} > t_{\mathrm{pre}} \\ \mathrm{A_{post}} \cdot \exp\left(-\frac{t_{\mathrm{pre}} - t_{\mathrm{post}}}{\tau_{s2}}\right), & t_{\mathrm{post}} < t_{\mathrm{pre}} \end{cases} \tag{5}$$

Equation (5) shows a classical STDP weight update rule, demonstrated in [15] to govern variation of synaptic weights as functions of relative spike times, where t_{pre} is the time of the most recent pre-synaptic spike, t_{post} is the time of the most recent post-synaptic spike, and $\mathrm{A_{pre}}$ and $\mathrm{A_{post}}$ determine the corresponding learning rates. $\mathrm{A_{pre}} > 0$ and $\mathrm{A_{post}} < 0$ so that $w_{i,j}$ strengthens (and $w_{j,i}$ weakens) when neuron j spikes after neuron i. Notably, the change in synaptic strength is maximized when the time between pre- and post-synaptic spikes is minimized.

Figure 2(a) graphically depicts this change in synaptic efficacy as a function of the time between the relevant pre-synaptic and post-synaptic spikes. We choose A_{pre} to be $0.0096 \cdot w_{\mathrm{max}}$ and A_{post} to be $-0.0053 \cdot w_{\mathrm{max}}$ where w_{max} is the maximum weight of each synapse, so that the ratio of $A_{\mathrm{pre}} : A_{\mathrm{post}}$ is the same as the ratio reported in [15]. τ_{s1} and τ_{s2} are chosen to be 16.8 ms and 33.7 ms, the values reported in [15].

STDP Variant I. Shortly after the discovery of classical STDP, it was discovered that the STDP learning window might appear with a symmetric depression window [4]. In the symmetric case, $\tau_{\text{pre}} = \tau_{\text{post}}$, which produces potentiation of $w_{i,j}$, equal in magnitude to depression experienced by $w_{j,i}$, assuming both connections exist. Here, we assume the STDP Variant I has the same potentiation window as the classical STDP, while maintaining a symmetric depression window. Equation (6) shows the mathematical expression of STDP Variant I, and Figure 2(b) shows a graphical representation of it. In (6), A is chosen to be $0.0096 \cdot w_{\max}$, just as in the classical STDP introduced above. τ_s is chosen to be 16.8 ms.

$$\Delta w = \begin{cases} \mathrm{A} \cdot \exp\left(-\frac{t_{\text{post}} - t_{\text{pre}}}{\tau_s}\right), & t_{\text{post}} > t_{\text{pre}} \\ -\mathrm{A} \cdot \exp\left(-\frac{t_{\text{pre}} - t_{\text{post}}}{\tau_s}\right), & t_{\text{post}} < t_{\text{pre}} \end{cases} \tag{6}$$

STDP Variant II. Figure 2(c) shows a second variant of STDP. This learning window, introduced in [16], aims to fit the STDP data collected in experiments. The potentiation regime extends to $t_{post} - t_{pre} < 0$, a minor departure from classical STDP [20].

$$\Delta w = \begin{cases} E_N \cdot (A_p e^{-\Delta t/\tau_p} - A_d e^{\eta \Delta t/\tau_p}), \text{ if } \Delta t > 0 \\ E_N \cdot (A_p e^{-\eta \Delta t/\tau_d} - A_d e^{\Delta t/\tau_d}), \text{ if } \Delta t < 0 \end{cases} \tag{7}$$

where A_p and A_d are given by:

$$A_p = \gamma [1/\tau_p + \eta/\tau_d]^{-1} \tag{8}$$

$$A_d = \gamma [\eta/\tau_p + 1/\tau_d]^{-1} \tag{9}$$

Equation (7) shows a mathematical description of STDP Variant II. A special property of this learning window is that the integration over $-\infty$ to ∞ is 0. In our experiment, the normalization coefficient E_N is chosen to make the magnitude of STDP Variant II the same as that of Classical STDP.

STDP Variant III. A prototype of this learning window is found in [5], a symmetrical variant of STDP with no LTD. Our third instantiation of STDP uses symmetric depression and potentiation. The time constant for the learning window described in [5] is too large, so we select a compromise larger than that of classical STDP: τ_s is chosen to be 33.7 ms, which is equal to the time constant of the depression window of classical STDP. It is about twice the value of the time constant for the potentiation window of classical STDP. The learning window is formulated as in (10) and illustrated in Figure 2(d).

$$\Delta w = Ae^{-|\Delta t|/\tau_s} \tag{10}$$

2.2.3. Weight Normalization

Indubitably, STDP is a powerful tool. However, this plasticity is a point-to-point mechanism: dynamics of a synapse are completely determined by the activity of the two neurons attached it. However, we can avoid this restriction and allow plasticity to act with a wider scope. For example, we can allow plasticity to be a function of the activity of larger set of neurons. A simple form of this network-wide operation is normalization of synaptic weights, which we show allows the network to achieve globally desirable properties. In this work, we propose three normalization rules.

Input Normalization on Sum of the Weights. Input normalization of synapse strength is discussed in the context of rate-based neural models [12]. However, little work has been done concerning the impact of normalization in spike-time based models. Here, we proposed a spike-time based normalization rule: If the sum of weights input to a post-synaptic neuron exceeds an imposed maximum, all of the synapses to that post-synaptic neuron will be weakened so that their sum remains less than or equal to this bound.

$$\text{If } \sum_i w_{i,j} > w_{\text{inCons}}, \text{ then } w_{i,j} \leftarrow w_{i,j} \frac{w_{\text{inCons}}}{\sum_i w_{i,j}}. \tag{11}$$

Input Normalization on Sum of the Squared Weights. Similar to input normalization rule introduced above, the squared sum of weights projected to a neuron is regulated by a chosen maximum, w_{inCons}. Here, the weights are normalized as follows.

$$\text{If } \sum_i w_{i,j}^2 > w_{\text{inCons}}, \text{ then } w_{i,j} \leftarrow w_{i,j} \sqrt{\frac{w_{\text{inCons}}}{\sum_i w_{i,j}^2}}. \tag{12}$$

Output Normalization on Sum of the Weights.

$$\Delta w_{i,j} = \mathrm{A}_{\text{pre}} \cdot \exp\left(\frac{t_i - t_j}{\tau_s}\right) \tag{13}$$

$$\Delta w_{i,k} = -|w_{i,k}| \cdot \frac{\Delta w_{i,j}}{w_{\text{cons}} - w_{i,j}} \tag{14}$$

Under output normalization defined by (13), when $w_{i,j}$ is strengthened, the connections from neuron i to other neurons $k \neq j$ are weakened. We implement this competitivity via normalization of output synapses to the summed strengths of outputs of each input neuron at each synaptic update operation. To accomplish this, we impose a constraint, w_{cons}, which bounds from above the strength of connections departing a neuron. Traditionally (e.g. as discussed in [12]) synaptic competition is considered as implemented by normalization of weights. This competitive spike time based learning differs from examples discussed in [12, 21] in that their approaches implement competition as normalization of synaptic strengths to the summed strengths of common inputs (i.e., they consider competition among synapses projecting to a common post-synaptic neuron) while we consider competition among synapses originating from a common pre-synaptic neuron. Our approach follows from an intuitionistic argument: A neuron projecting synapses are burdened by physics with a strict upper bound on the energy it may expend on communicating a spike to its post-synaptic neighbors. Additionally, physics limits a neuron's neurotransmitter[1] budget. It follows that if the neuron is driven to invest more energy in a particular channel, it must divest of others.

Our competitive learning rule has three important features. Firstly, it imposes an upper bound on the sum of efficacies of synapses departing a neuron. Secondly, this learning rule allows this sum to increase slowly and more stably. Finally, the learning rule ramps up competitivity (i.e., increases the impact of this normalization) as strength approaches a hypothetical maximum, w_{cons}.

We first analyze the situation in which all synapses have a non-negative strength. In this case, we can remove the absolute value symbol from $w_{i,k}$ in (13). Then, we obtain (15), where we assume that neuron i emits a spike shortly before j. In response, synapse $w_{i,j}$ is strengthened, and all other synapses $w_{i,k}$ are weakened.

[1]Neurotransmitters are molecules released at a synapse, and communicated to dendrites of post-synaptic neurons via diffusion across a gap [22].

We know that

$$\sum_{k=1,k\neq j}^{K} \Delta w_{i,k} = \sum_{k=1,k\neq j}^{K} \left(-\frac{w_{i,k}\Delta w_{i,j}}{w_{\text{cons}} - w_{i,j}} \right), \tag{15}$$

where K is the number of synapses projected by the neuron in question. We divide the analysis into two cases. First, if the sum of outgoing synaptic efficacies, $\Psi = w_{i,j} + \sum_{k=1,k\neq j}^{K} w_{i,k}$ hits w_{cons}, then we have

$$w_{\text{cons}} = w_{i,j} + \sum_{k=1,k\neq j}^{K} w_{i,k}. \tag{16}$$

Combining (15) and (16) we have:

$$\sum_{k=1,k\neq j}^{K} \Delta w_{i,k} = \sum_{k=1,k\neq j}^{K} \left(-\frac{w_{i,k}\Delta w_{i,j}}{\sum_{k=1,k\neq j}^{K} w_{i,k}} \right) = -\Delta w_{i,j} \tag{17}$$

Equation (17) shows that when Ψ reaches w_{cons}, the increase in $w_{i,j}$ is equal to the sum of decreases in $w_{i,k}$ over $k \neq j$, due to competition among synapses. This should drive the network towards equilibrium and prevents epileptic destabilization that results from run-away potentiation.

If Ψ remains much smaller than w_{cons}, then

$$w_{\text{cons}} = B + w_{i,j} + \sum_{k=1,k\neq j}^{K} w_{i,k}, \tag{18}$$

where B defines competitivity equal to the difference between w_{cons} and the quantity of synaptic efficacy already invested after the potentiation induced by the most recent pair of spikes. When $\frac{1}{B}$ is small, ample synaptic efficacy is still available for synapses to be strengthened. Thus when one synapse is strengthened, other synapses will only be weakened mildly (a low-competitivity situation). On the other hand, if $\frac{1}{B}$ is large, there is little efficacy available for the synapse in question to be strengthened, thus there is greater competitivity. Combining (18) and (15), we have

$$\sum_{k=1,k\neq j}^{K} \Delta w_{i,k} = \sum_{k=1,k\neq j}^{K} \left(-\frac{w_{i,k}\Delta w_{i,j}}{B + \sum\limits_{k=1,k\neq j}^{K} w_{i,k}} \right) = -\Delta w_{i,j} \left(\frac{\sum\limits_{k=1,k\neq j}^{K} w_{i,k}}{B + \sum\limits_{k=1,k\neq j}^{K} w_{i,k}} \right) \tag{19}$$

As before, when B is relatively large, the decrease, $\Delta w_{i,k}$ is small. Conversely, when Ψ is sufficiently close to w_{cons}, B is nearly zero. Equation 19 also shows that in this case, total synaptic depression (i.e. depression summed over all outgoing synapses, $w_{i,k}$ where $k \neq i$) is equal to potentiation, $w_{i,j}$. If a negative weight is allowed then we place an absolute value operation on $w_{i,k}$, to obtain (13). Thus $w_{i,k}$ always decreases when $w_{i,j}$ increases.

2.2.4. Reward STDP

The weight change of reward STDP is not only affected by the relative spiking time of pre- and post-synaptic neurons but also modulated by the reward signal. The reward is given by the output of classifier and the target (the label of each image). If a classifier's output matches the target, then the reward given to network is a normal application of the STDP learning rule. However, if a classifier's output differs from the target, then a punishment is supplied. In our model, we apply this rule whenever a post-synaptic spike occurs. A more detailed treatment of rewarded

STDP is discussed in section 2.3.

2.2.5. STDP Trace

The modification of a single synapse caused by STDP is expressed using (20), where pre means all pre-synaptic spikes, post means all post-synaptic spikes and K is the STDP learning window. (20) shows that total modification caused by STDP is the summation of modifications over all combinations of pre-synaptic spikes and post-synaptic spikes.

$$\Delta W = \sum_{t_{\text{pre}}} \sum_{t_{\text{post}}} K(t_{\text{pre}} - t_{\text{post}}) \tag{20}$$

In practice, recording and processing of all previous spikes is computationally expensive. Therefore, we use an alternative representation of this relationship. Since, in our classifier, each pre-synaptic neuron is allowed to emit only one spike during the simulation of each image, then (20) can be simplified as (21).

$$\Delta W = \sum_{t_{\text{post}}} K(t_{\text{pre}} - t_{\text{post}}) \tag{21}$$

When t_{post} is larger than t_{pre}, this equation does not need reformulation because there is only one pre-synaptic spike. However, when every t_{post} is less than t_{pre}, (21) implies that all post-synaptic spikes that arrive before the pre-synaptic spike will contribute to depression of the synapse. If we assume the depression window of STDP is an exponentially-decaying function and that $t_{\text{post}} > t_{\text{pre}}$ then (21) can be reformulated as (22).

$$\Delta W = \sum_{t_{\text{post}}} A \cdot \exp\left(-\frac{t_{\text{pre}} - t_{\text{post}}}{\tau}\right) \tag{22}$$

$$= A_j \cdot \exp\left(-\frac{t_{\text{pre}} - t_j}{\tau}\right) \tag{23}$$

where t_{pre} is the time of pre-synaptic spike, t_j is the time of the j^{th} post-synaptic spike and A is the amplitude of the STDP depression learning window. Consider j as the latest post-synaptic spike, and $j-1$ as the post-synaptic spike arriving earlier than j. A_j is the "trace" of STDP, defined with the following recursive formula:

$$A_j = \begin{cases} A, & \text{if } j = 1 \\ A_{j-1} \cdot \exp\left(-\frac{t_j - t_{j-1}}{\tau}\right) + A, & \text{if } j > 1. \end{cases} \tag{24}$$

The expression in (24) accounts for all of the previous post-synaptic spikes, which allows us to compute A_j in an efficient way because we only need to store the value of A_{j-1} to make the next calculation. We exploit this in our implementations of classical STDP, STDP Variant I and STDP Variant III, because it is assumed that the depression window of STDP decays exponentially with increases in inter-spike interval [23]. However, we omit this assumption for STDP Variant II by ignoring the impact of previous post-synaptic spikes except the latest one.

2.3. Spiking Classifiers

We implemented a network of leaky integrate and fire (LIF) neurons with plastic synapses with the dynamics described in Section 2.1 to construct a 4-layer feedforward SNN. The image preprocessing protocols and network properties are adapted from [9]. As compared to [9], where the network has an additional hidden

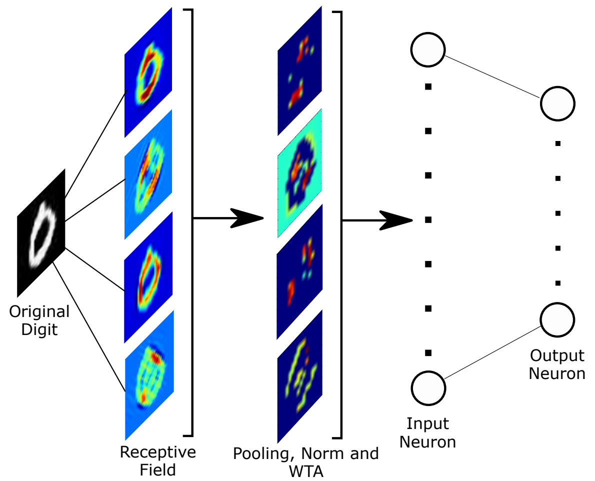


Figure 3: Structure of our SNN Classifier

layer between the input neurons and output neurons, we considered input neurons directly mapping to output neurons in our network. The structure of our proposed network is illustrated in Figure 3.

We now describe preprocessing of the data. The images are fed into a receptive field layer of four orientations, Gabor filters of 0, 45, 90, and 135 degrees. After this operation, the original image, of size 28×28 pixels, is projected to a higher dimensional space of 32×32 pixels by four orientations. The output of this first layer processing is then sent through a layer of max-pooling and winner-takes-all circuitry (WTA). The max-pooling and WTA operations impose a dimensionality reduction on the input in order to remove noise and speed up the simulation. After max-pooling, the image is 16×16 pixels by four orientations; then the image is converted to a sequence of spikes. Each pixel is converted to a single spike of a corresponding neuron in the input layer. That is, during the simulation of every single image, each input neuron may only spike once. The latency of each spike is determined by the intensity of the corresponding pixel. We evaluated three conversion rules, each with a different latency scale for the mapping of image to spike pattern. Spikes are generated within a duration of either 10 ms, 17 ms or 34 ms, chosen deliberately in that 10 ms is about half of the time constant of classical STDP, 17 ms is the time constant of classical STDP, and 34 ms is twice of the time constant of classical STDP. Finally, spikes are propagated from the input neurons to the output layer. The number of neurons in the output layer equals the number of classes of the classification problem. The output layer is trained using supervised reinforcement learning with a teaching signal realized at the epilogue of each simulation (1 ms before the end of simulation). Weights are updated whenever a spike occurs. If a spiking output neuron is the target neuron, a reward is supplied to the network, and punishment is given otherwise. After weights adjust, normalization rules are applied to weights. Every individual weight is clipped to a range from $-0.3w_{\max}$ to $w_{\max}$, where $w_{\max}$ is the upper bound of the strength of every individual synapse. More detailed description of the whole process is presented in Section 2.3.1 to Section 2.3.3.

2.3.1. Encoding

Any natural (i.e., biologically implemented) spiking neural classifier - especially those receptive to visual information - should take advantage of the efficient coding employed by the mammalian brain. For example, humans typically have ≈ 4.6 million cone cells and ≈ 92 million rod cells, for a total of ≈ 96.6 million photoreceptors in each eye [24]. The output of the human eye typically has between 0.71 and 1.54 million retinal ganglion cells though this is highly variable across eyes surveyed [25]. This observation means there is an encoding process that reduces the dimensionality of the visual data by between 8 and 9 orders of magnitude before any neurons located in the brain perceive the visual signal. Visual information flows from retinal ganglion cells to V1, the mammalian primary visual cortex. V1 preprocesses the visual information for higher layers of processing by performing edge detection (and probably other computations) [26, 27].

We emulate this natural visual structure. As shown in Figure 3, the input image is passed through a receptive field layer, achieved by implementing Gabor filters of four orientations. (25) is a mathematical description of the Gabor filters we use.

$$g(x,y;\lambda,\theta,\psi,\sigma,\gamma) = \exp\left(-\frac{x'^2+\gamma^2 y'^2}{2\sigma^2}\right)\cos\left(2\pi\frac{x'}{\lambda}+\psi\right) \tag{25}$$

Here, x and y are coordinates of pixels and θ is the orientation applied. In this step, the original picture is projected to a higher dimension, and the edges of the four orientations are extracted. These receptive fields, in essence, project the images to a higher dimensional space, producing a more separable (and thus more tractable) classification problem in a manner similar to kernel tricks commonly used to preprocess data for classification with support vector machines [28]. Furthermore, this edge detection process emulates the functionality of the receptive fields in human V1.

The second class of encoding methods considered consists of sequential max-pooling and winner-take-all. In this procedure, the input is passed through a 2×2 max-pooling operation with stride 2. Subsequently, a winner-takes-all (WTA) rule is applied to the max-pooled image. Algorithm 1 shows the flow of these operations. As stated above, orientations are of four values, 0, 45, 90, and 135 degrees. The "max orientation" function at line 1 in Algorithm 1 returns the orientation corresponding to the largest value of the pixel at position (x,y). The pixel value at (x,y) in orientation j is left unchanged and $p(x,y)$ at other orientations are set to 0. Consider a pixel with coordinates (x_0, y_0) in orientation i, denoted by $p_i(x_0, y_0)$. If pixel $p_1(x_0, y_0)$ has the largest intensity, then the value of pixel $p_2(x_0, y_0)$, $p_3(x_0, y_0)$, $p_4(x_0, y_0)$ will be set to 0, while $p_1(x_0, y_0)$ maintains the original value. After max-pooling and WTA, the dimension of input is reduced from 4096 to 1024, a procedure that also emulates the dimensionality reduction in the human visual cortex. In addition, this operation significantly reduces required learning time.

Algorithm 1 WTA Encoding

Input: Orientation i; Pixel Position x,y; Pixel value $p_i(x, y)$
Output: Updated pixel value, $p'_i(x, y)$

```
j = max orientation(p_i(x, y))
for i in orientations do
    if i ≠ j then
        p_i(x, y) = 0
    end if
end for
```

The final stage of encoding converts preprocessed pixel intensity to spike time. Reference [9] proposed that the latency of encoded spike should be inversely proportional to its intensity; however, their work did not produce a convenient mathematical expression. We propose to follow this idea about the proportionality, but provide such an expression. Our proposed encoding scheme, which transforms pixels values into latencies, is given by

$$t_{x,y} = \frac{(D-3)}{I_{x,y}} - D + 4, \text{ if } I_{x,y} > 0.5 \tag{26}$$

where $I_{x,y}$ is the intensity of pixel at position (x, y), taking a value between 0 and 1, and D is the simulation duration. Under this rule, pixels with intensities below 0.5 are discarded and pixels with larger intensities are converted to spikes with

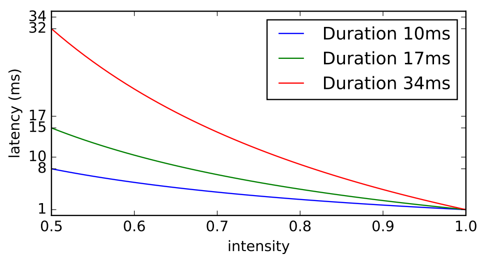


Figure 4: Encoding Duration and Intensity

smaller latencies. All spikes are mapped to a latency between 1 ms and $D-2$ ms, 1 ms before the teaching signal is presented. To study the interactions between the STDP learning window and simulation duration, we consider three encoding durations: 10 ms, 17 ms and 34 ms.

2.3.2. Decoding

We evaluated two decoding schemes, first-spike decoding and count-decoding [9]. We choose the number of output neurons to be the same as the number of classes in the classification problem. First-spike decoding selects the class with the output neuron that is the first to emit a spike and count-decoding examines the number of spikes at each output neuron and returns the label of the neuron with the most spikes.

2.3.3. Learning Process and Weight Update

The network utilizes the proposed weight normalization approach paired with a rewarded STDP mechanism, a supervised learning algorithm that is used in [9] (see Algorithm 2). This algorithm takes in X_{train}, a list of images, and Y_{train}, a list of targets (i.e. the label of each training image). A single image from the list is converted to spikes, X, then fed into the network by inducing spikes in input neurons according to the encoded image. At time step t denote these spikes by X_t.

The spikes emitted by the SNN classifier, S_t, are then determined by simulating the propagation of X_t through the network (see line 6). At the epilogue of the previous step, a teaching signal is provided to the network: the induction of a single post-synaptic spike on the target neuron. Then, S_t is compared to Y to determine reward or punishment and STDP rules are applied to the network to update the synaptic weights (see lines 9 and 12, respectively). After the STDP update, we apply our proposed weight normalization rule to the network. There are two types of normalization rules considered in our simulations: (1) normalization of each individual synapse; and (2) normalization of a cluster of synapses. Normalization for an individual synapse is implemented by constraining strength to the range $[-0.3 \cdot w_{\max}, w_{\max}]$. Cluster normalization is implemented by the rules in Section 2.2.3 (see line 14 in algorithm 2).

The training process for the first-spike decoding SNN classifier follows a similar approach to that of the count decoding classifier (see Algorithm 3). However in the first-spike decoding scheme, decoding is complete when the first post-synaptic spike occurs. This is detected by the spike flag, F_s, which is set to 1 when a spike in the output layer occurs.

3. Experimental Results

3.1. Performance Testing Methods and Data

We present an empirical analysis of the impacts of varying encoding/decoding schemes and normalization of the synaptic weights on SNN classification performance. We benchmarked the performance of our SNN classifiers on three datasets: MNIST [29], NIST [30], and ETH80-Contour [31]. MNIST contains $70,000$ hand-written digits ($60,000$ training and $10,000$ testing samples) belonging to 10

Algorithm 2 Learning Process in Training Session (Count Decoding)

Input: A list of images, X_{train}; A list of targets, Y_{train}, A Network N

Output: A trained network, N_c

```
for each image, X ∈ X_train, Y ∈ Y_train do
    for t in T_s do
        if t = D − 1 then
            N ←Teach(N,Y)
        end if
        S_t =Update(N,X_t)
        if S_t ≠ Y_t then
            reward = −1
            N ←STDP(N,reward)
        else if S_t = Y_t then
            reward = 1
            N ←STDP(N,reward)
        end if
        N ←Norm(N)
    end for
    N ←Reset(N)
end for
N_c ← N
Return N_c
```

Algorithm 3 Learning Process in Training Session (First-Spike Decoding)

Input: A list of images, X_{train}; A list of targets, Y_{train}, A Network N
Output: A trained network, N_c

for each image, $X \in X_{\text{train}}$, $Y \in Y_{\text{train}}$ **do**
    $F_s = 0$
    while t in T_s and $F_s == 0$ **do**
        if $t = D - 1$ **then**
            $N \leftarrow$Teach(N,Y)
        end if
        $S_t =$Update(N,X_t)
        if $S_t \neq Y$ **then**
            $reward = -1$
            $N \leftarrow$STDP(N,$reward$)
            $F_s = 1$
        else if $S_t = Y$ **then**
            $reward = 1$
            $N \leftarrow$STDP(N,$reward$)
            $F_s = 1$
        end if
        $N \leftarrow$Norm(N)
    end while
    $N \leftarrow$Reset(N)
end for
$N_c \leftarrow N$
return N_c

classes. In our experiment we randomly pick 5000 training samples and 900 testing samples from each class of the MNIST dataset. NIST is a hand-written character and digit dataset. We choose 10 classes from the original NIST dataset to evaluate the performance of our proposed normalization scheme. ETH80-Contour contains eight classes of objects that include fruits, animals and cars, and this dataset is often used for 3-D image reconstruction. We build an individual classifier for each dataset. On that particular dataset, the classifier is trained consecutively on each batch, then the classifier is evaluated on testing samples of the corresponding dataset and the accuracy is reported. The samples within each dataset are shuffled before encoding and simulation. The same training protocols are employed on MNIST, NIST as well as ETH80-Contour, except that the batch size, number of batches and testing size are necessarily different.

3.2. No Normalization Experiment

In this section, we conducted a series of experiments without the use of normalization rules. We studied the influence of STDP learning window and encoding/decoding on the performance of the proposed SNN classifier. The STDP learning windows implemented are formulated in Section 2.2.2. Simulation durations are chosen to be 10 ms, 17 ms, and 34 ms. Under each simulation duration, the latency of each input spike is scaled to range from 1 ms to $(D-2)$ ms, as stated in Section 2.3.1.

3.2.1. Accuracy Comparison w.r.t. STDP Kernel and Decoding Scheme

To compare the performance of classifiers under count decoding and first-spike decoding schemes, we plot the mean accuracy and error bar for each STDP learning window on each dataset under different decoding schemes in Figure 5. The x-axis of Figure 5 lists the datasets combined with the decoding scheme used (C denotes count decoding and F denotes first-spike decoding). Different STDP learning windows are tested, where STDP-C denotes the classical learning window, and STDP-I means STDP variant I. Each single bar in Figure 5 is computed as $p_m^{d,s} = (\sum_{j=1}^{N_E} \sum_{i=1}^{N_B} p_{i,j}^{d,s})/(N_E \cdot N_B)$, where $p_m^{d,s}$ is the mean accuracy using STDP learning window s on dataset d, N_E is the number of choices of encoding durations, N_B is the number of batches under each encoding duration, and $p_{i,j}^{d,s}$ is the testing accuracy of batch i under encoding option j using STDP s on dataset d. For example, for STDP-I on MNIST tasks, we have 50 batches, and the encoding durations are 10 ms, 17 ms and 34 ms, so that $d = \text{MNIST}$, $s = \text{STDP-I}$, $N_B = 50$ and $N_E = 3$. In addition, we also compute the standard deviation and plot the error bar in the figure. Standard deviation is given by: $s^{d,s} = \sqrt{[\sum_{j=1}^{N_E} \sum_{i=1}^{N_B} (p_{i,j}^{d,s} - p_m^{d,s})^2]/(N_B \cdot N_E)}$, and the error bar corresponding to 95% confidence intervals is computed as $err = \pm 1.96 \frac{s^{d,s}}{(N_B \cdot N_E)^{0.5}}$. Results in Figure 5 show that under count decoding scheme, classical STDP performs significantly better than other STDP variants on MNIST dataset, and it performs significantly better than STDP-II & III on NIST dataset. However, this phenomenon does not appear on ETH-80 Contour dataset. As for first-spike decoding, since decoding stops as soon as the first post-synaptic spike is emitted, the depression window of STDP never takes effect. Thus only the shape of the potentiation window affects classification performance. Since STDP-C and STDP-I have the same potentiation window, their performances are identical under first-spike decoding scheme. The shape of potentiation window of STDP-III is very similar to that of STDP-C and STDP-I, except that the time constant of potentiation window of STDP-III is about twice as that of the other two, resulting in a slight performance difference from that of STDP-C and STDP-I. STDP-II performs

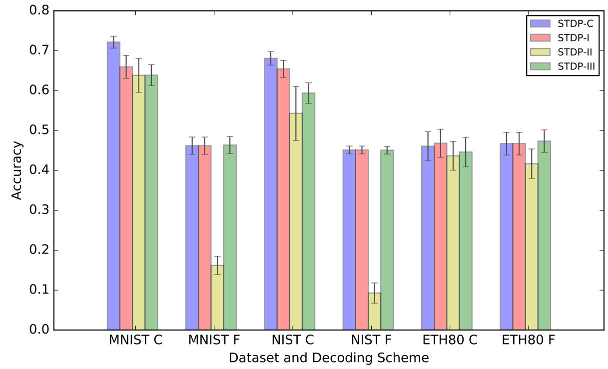


Figure 5: Average Accuracy of STDP without Normalization

poorly on MNIST and NIST dataset under the first-decoding scheme. Figure 5 also shows that on MNIST and NIST datasets, count decoding performs significantly better than first-spike decoding.

3.2.2. Comparison of Simulated Time

Testing accuracy of the classifier using count decoding is significantly better than that of first-spike decoding, an improvement achieved at the expense of decoding time. Figure 6 shows the simulated time per image under count decoding scheme and first-spike decoding scheme. Simulated time is not the execution time of the program. Instead, it refers to the duration of simulated neural activity of our classifier. For example, if we use 34 ms encoding duration scheme, the simulated neural activity for classifying each image is from 0 ms to 33.8 ms. Currently, we use a time step of 0.2 ms, so that there are 170 steps to simulate. After 170 steps, the classifier's state is reset to initial state (except the weight), and next simulation will start. The top three panels of Figure 6 show the simulated time per image for classifiers using different encoding durations. The simulated time for both training

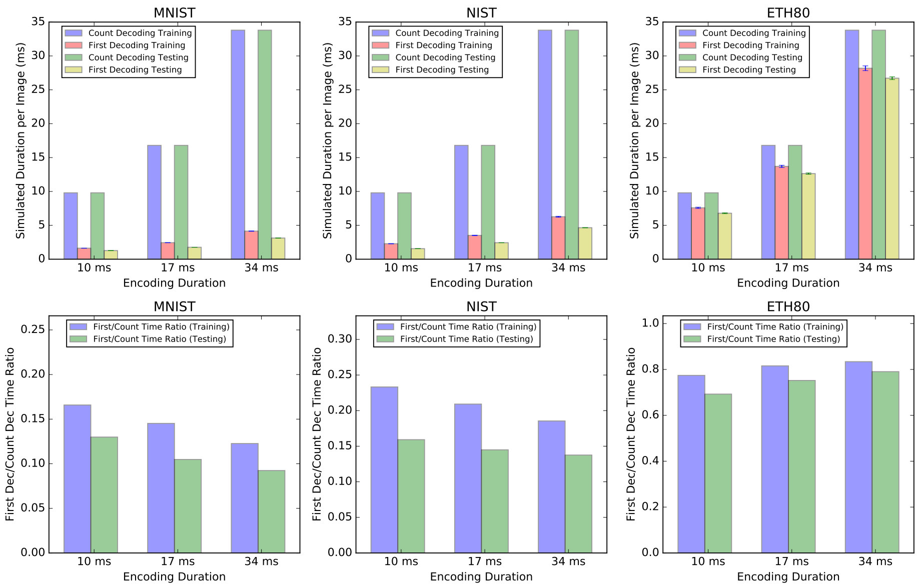


Figure 6: Simulation Time using Count Decoding and First Decoding Scheme

session and testing sessions are recorded. Under count decoding, the simulated time for processing each image is constant. If we use D to denote the encoding duration, then the simulated time for processing each image is $D-0.2$ ms. These results show that on MNIST and NIST datasets, the simulated time for both training and testing sessions under first-spike decoding is significantly less than that using count-decoding. On ETH80 dataset, this phenomenon is not as apparent as on the other two datasets. The bottom three panels of Figure 6 show the ratio of simulated time per image using first-spike decoding and count decoding (Assume the time using first-spike decoding is T_F, and the time using count decoding is T_C, then the ratio is calculated as $R = T_F/T_C$). On MNIST and NIST, the ratio is less than 25%. As the encoding duration increases from 10 to 34 ms, the ratio decreases. Notably, this ratio is even lower in testing session than in training session. Figure 5 and

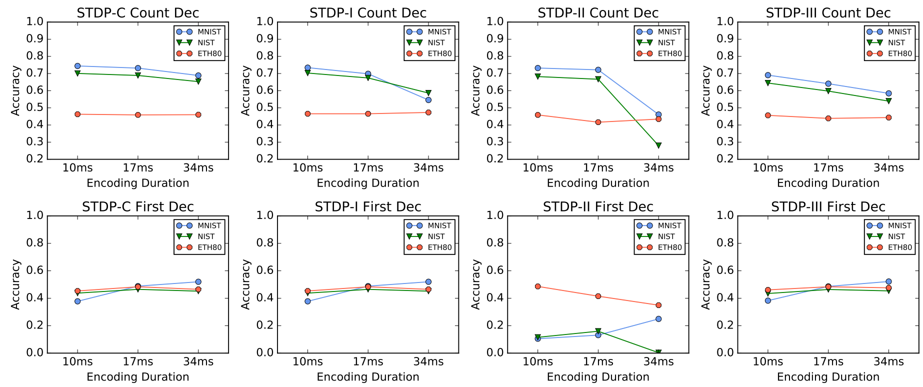


Figure 7: The Effect of Encoding Duration on Classification Accuracy

Figure 6 together show that although using the first-spike decoding may produce a decrease in classification accuracy, the reduction in simulated time is significant. The neural classifier using the first-spike decoding scheme needs much less time to train and test.

3.2.3. Accuracy Comparison w.r.t. Encoding Duration

Figure 7 shows the testing accuracy of the classifier with different encoding schemes. The four sub-figures in the first row of Figure 7 show the performance of four STDP learning windows under the count decoding scheme. The bottom four figures show the performance of STDP learning windows under first-spike decoding. Each point in the figure denotes the average testing accuracy under particular encoding duration and decoding scheme. Under count-decoding scheme on MNIST and NIST dataset, the accuracy drops as the encoding duration increases from 10 ms to 34 ms. This trend is particularly obvious for STDP-II. The accuracy drops to as low as 30% using an encoding duration of 34 ms. Under the first-spike decoding scheme, there is not such a consistent trend across all STDP learning windows. Since STDP-C, STDP-I, and STDP-III have very similar potentiation windows, it is not surprising that they have very similar performance. For these three learning windows, the increase in encoding duration will lead to an increase in classification accuracy on MNIST dataset. However, this is not the case for NIST and ETH80 dataset. There is a peak of classification accuracy which emerges when the encoding duration is 17 ms. For STDP-II, the trend is a little different. Again, classification accuracy on MNIST dataset increases as the encoding duration increases, and there is a peak on NIST dataset. However, there is no peak in classification accuracy for ETH80 task. Instead, classification accuracy only decreases as the encoding duration increases.

3.3. Normalization Experiment

In this section, we evaluate the performance of the proposed normalization rules. For these experiments, the encoding duration of the SNN classifier is fixed to 10 ms, and there are several reasons for this choice. The experiments using count-decoding scheme in Section 3.2 demonstrated that 10 ms encoding typically provides the best accuracies compared to the other encoding durations. Second, the influence of encoding duration on the classifier's performance under first-spike decoding scheme is not obvious, but to compare the performance of classifier under different decoding schemes we need to set the encoding duration of all SNN classifiers to be the same.

3.3.1. Performance Enhancement using Output Normalization

Figure 8 shows the performance of classical STDP combined with different normalization rules as introduced in Section 2. For ETH80 dataset, normalization rules do not make significant difference either using count decoding scheme or using first-spike decoding scheme. Input normalization improves performance in most cases, but a larger improvement of accuracy is achieved with output normalization. Under count decoding, the output normalization rule undermines the classification accuracy. Whereas under the first-spike decoding, the output normalization rule can significantly enhance the performance. When no normalization rule is applied, the average classification accuracy of classical STDP under count decoding on MNIST and NIST reaches $\approx 70\%$, while the accuracy of classical STDP under first-spike decoding on these two datasets are only around 40%. However, if our output normalization rule is applied, the classification accuracy reaches around 60%, just 10% shy of that achieved with count decoding. Meanwhile, first-spike decoding outperforms count decoding in required simulation time. The right bottom panel of Figure 8 shows the comparison of time consumption. On MNIST dataset, the classifier with first-spike decoding scheme consumes only about 15% of the time consumed under count decoding. On NIST dataset, first-spike decoding classifier takes about 20% of the time required by count decoding. Although the first-spike decoding scheme with normalization takes slightly more time than merely using first-spike decoding without normalization, it maintains a small required simulation time.

3.3.2. Early Exit

To investigate how output normalization boosts the performance of our proposed classifier, we record and plot the training and testing accuracy of each batch on all three datasets with and without output normalization. Figure 9 shows a plot of training and testing accuracy for each configuration. Training accuracy is obtained by testing the classifier using all samples in current training batch. The three figures in the first row show the performance of classical STDP/STDP Variant I on the three datasets. Since classical STDP and STDP Variant I perform

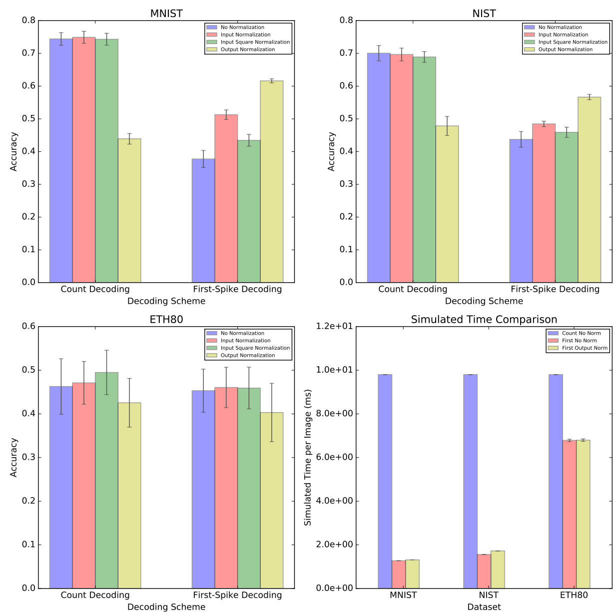


Figure 8: Performance of STDP with Normalization Rules

identically under the first-spike decoding scheme, we merged their figures into one. On MNIST dataset and NIST dataset, no-normalization classifier performs well in the first few batches, which means it could reach an accuracy very close to that using output normalization rule. However, after the first few batches, both training and testing accuracy of no-normalization classifier decreases until the end of training session. The same phenomenon appears in the experiments with STDP Variant III. This shows that although first-spike decoding classifier without normalization can successfully extract some image features at the beginning of training session, it fails to maintain what it learns. For STDP Variant II, this trend shows itself only subtly on NIST dataset but is absent on the MNIST dataset. In fact, this trend does not appear for any STDP learning windows on the ETH80 dataset, regardless of normalization rule (or lack thereof).

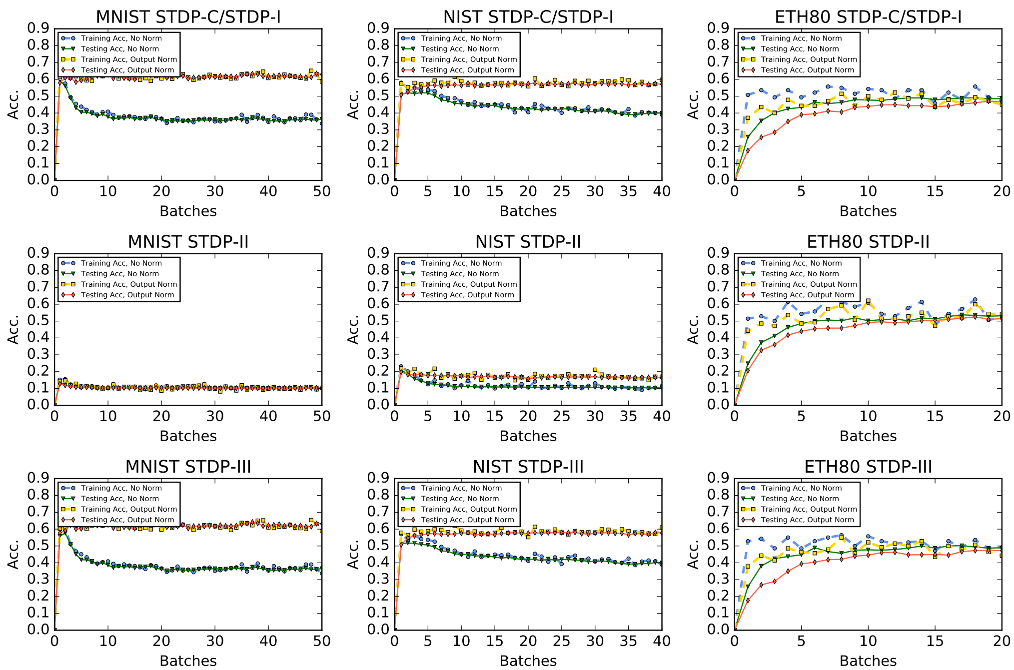


Figure 9: Training and Testing Accuracy

To further investigate how an output normalization rule can prevent decreases in accuracy in later batches of the training process, we plot the time consumption of classical STDP with and without normalization rule in each training batch. Figure 10 shows time consumption as a function of the batch number. On MNIST dataset, the training time of the first batch is about 2.6 s for both the no-normalization classifier and output-normalization classifier. Since there are 1000 training images in each batch in MNIST dataset, we infer that each image takes about 2.6 ms to train. As the training process continues, the training time for both no-normalization classifier and output-normalization classifier decreases. At the end of the training process, each batch takes about 1.5 s for the no-normalization classifier and 1.8 s for the output-normalization classifier to train. Similar trends appear on the NIST dataset. Both training and testing time for output-normalization classifier are larger than those of no-normalization classifier. Combined with the results from previous experiments on MNIST and NIST, we have the following observations. Recall that in previous experiments, we showed that output-normalized classifier under first-spike decoding scheme performs better than a no-normalization classifier on MNIST and NIST dataset, especially in the later phase of the training process. Meanwhile, a no-normalization classifier consumes less time than a comparable output-normalized classifier, especially in the later phase of the training process. This means that although no-normalization classifiers make their judgments earlier, this selection is more likely to include error. In contrast, output-normalized classifier tends to make its judgment at a later stage of the simulation for each image boosting the performance of the classifier with this delay of judgment. This hypothesis also fits our results when we consider the decision process of the classifiers. Under first-spike decoding scheme, the classification result is given by

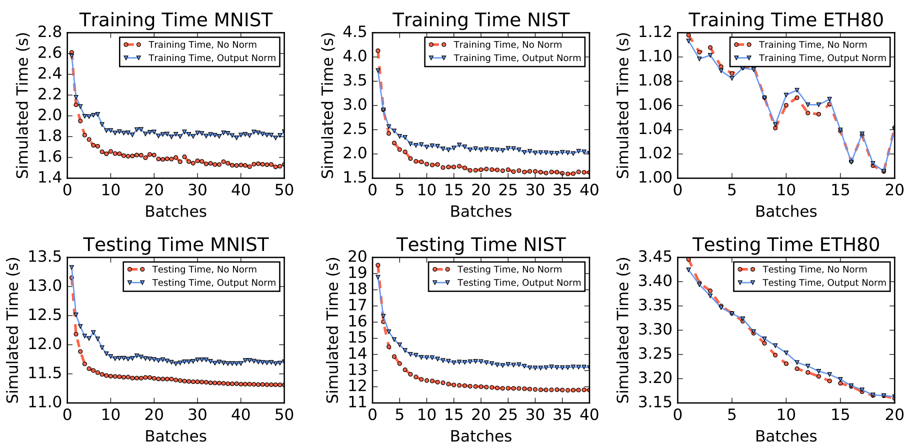


Figure 10: Time Consumption in Each Batch

the post-synaptic neuron which emits the first post-synaptic spike. For example, according to Figure 10, on MNIST, it takes about 1.5 ms for the no-normalization classifier to make the judgment on average in the last few batches. Recall that under 10 ms encoding duration, the input spikes are encoded to a latency ranging from 1 ms to 8 ms. Thus the first-spike of each image cannot arrive earlier than $t = 1$ ms. This means that the classifier tends to make its judgment using the spikes whose latencies range from 1 ms to 1.5 ms. This implies that the classifier tends to classify the image using only the most salient features of each image, as other features are encoded with latencies ranging from 1.5 ms to 8 ms. In contrast, an output-normalized classifier tends to require more time to settle on a decision, using this time to process more features of each image, thus reducing misclassification. On the ETH80 dataset, the differences between time consumption of the no-normalization classifier and output-normalized classifier are subtle if they exist. Similarly, the performance difference between the no-normalization classifier and output-normalized classifier is also not apparent on ETH80 dataset. Finally,

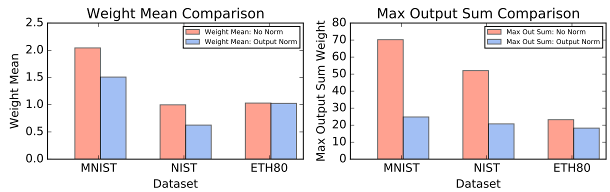


Figure 11: Weight Comparison

we investigated the influence of output normalization on the classifier's weight. Mean weight is calculated using the weight after the last batch of training process. The left panel of Figure 11 shows that after using an output-normalization rule, the mean weight on MNIST and NIST tasks is less than that of no-normalization classifier. In the right panel of Figure 11, the max output sum is reported using the following expression:

$$W_{\text{max,S}} = \text{Max}(W_i) = \text{Max}(\sum_{j=1}^{J} w_{i,j}) \tag{27}$$

where $w_{i,j}$ is the synapse weight from neuron i to neuron j. When no output normalization rule is configured, there is only our imposed minimum and maximum bounding of the strength of each individual synapse. The maximum strength of any synapse w_{max} is chosen to be 20. Since there are 10 classes in MNIST/NIST dataset, each pre-synaptic neuron has 10 output synapses. Thus the maximum value of the summed output strength is $20 \cdot 10 = 200$, which occurs when each of the output synapses departing from one neuron reaches $w_{\text{max}} = 20$. When an output normalization rule is applied, the output sum of strength is constrained to be less than or equal to 30. Figure 11 illustrates this comparison. On both MNIST and NIST datasets, both the mean weight and the maximum output sum of weights are larger when no normalization rule is applied.

4. Discussion

Several interesting conclusions can be drawn from the sequence of experiments performed and the corresponding results. One unexpected outcome is that although the results from the MNIST and NIST are mostly consistent, these results are not always consistent with those obtained from ETH80-Contour dataset. In the no-normalization experiments presented in Section 3.2, we compared the performance of multiple STDP learning windows with different encoding and decoding schemes without normalization rule. On MNIST and NIST dataset, the performances of different STDPs are discernible. However, we did not observe such a difference with the ETH80-Contour dataset. Secondly, the encoding duration of the SNN has a different influence on ETH80 in comparison with MNIST and NIST, especially with count-based decoding.

The output normalization rule can significantly boost the performance of the classifier under first-spike decoding scheme on MNIST and NIST dataset, whereas this trend does not appear on the ETH80-Contour dataset. Combined with the fact that ETH80 experiments showed a different trend in the no-normalization experiment as well, we propose several explanations to account for this outcome. First, this deviation could result from the relatively small number of samples in ETH80 dataset. It is possible that the number of samples in ETH80 is insufficient for the classifier to learn a trend obtainable from a larger dataset. Secondly, this could be a result of the differences of the dimensionality of MNIST/NIST and ETH80 data. MNIST/NIST consist of images that are converted to a size of 16×16 pixels after convolution and max-pooling, whereas the input dimension of ETH80 is 32×32 after these operations. This difference is because of the initial size of the images in ETH80, whose image sizes range from 377×377 to 825×825 . Finally, the differences could be due to the innate characteristics of the images in ETH80 dataset. More experiments can be executed to investigate the dependence of the classifier's performance on different datasets.

A complex system, the neural classifier proposed here has many hyper-parameters, including the time constants of neuronal membrane potential, the time constants associated with post-synaptic current, the maximum strength for each synapse, the overall maximum strength of the synapses departing from one neuron, etc. These parameters affect the dynamics of the whole system and are thus likely to influence the performance of classifier as they are varied. As a result, the optimization of these hyper-parameters is to be the subject of future research. Furthermore, more experiments would to study the effectiveness of the normalization rules on networks with different neuron models is of great interest. Here, we implemented the network based on Leaky Integrate and Fire (LIF) model of a neuron's activity. However, this is a highly simplified model that mimics real dynamics very coarsely. In later experiments, the impacts of including more realistic neuron models can be investigated (e.g. Izhikevich's model in [32] is popular in neural modeling for the compromise between complexity and realism it affords).

Another interesting avenue for further research is analysis of different decoding mechanisms. As stated in our work, first-spike decoding tends to consume much less time than count decoding classifier. On the other hand, first-spike decoding attains a lower classification accuracy. However, this does not imply that first-spike decoding is inferior to count decoding. Instead, classifiers with different decoding rules may compensate for each other's shortcomings. In scenarios where classification speed is extremely important, first-spike decoding may be beneficial. In other situations where classification accuracy is important, a count decoding scheme may prevail. Furthermore, as reported in this work, normalization can collude differently with different decoding rules, significantly improving accuracy of first-spike decoding.

5. Conclusion

In this article we build an SNN classifier trained with STDP for image classification problems. The proposed classifier is tested on the ubiquitous MNIST, NIST and ETH80 datasets. A spike latency encoding system is applied to the images and several configurations of the proposed classifiers are tested, including different STDP learning windows, decoding schemes, encoding durations and normalization rules.

In the no-normalization experiments, we test the performance of the SNN classifier using four different STDP learning windows, three encoding durations, and two decoding schemes. Results show that on MNIST and NIST, the classification accuracy using count decoding is significantly improved over first-spike decoding. Further, under the count decoding scheme, classical STDP achieves the highest average accuracy on both MNIST and NIST dataset. Although first-spike decoding scheme tends to have a lower classification accuracy, it tends to use much less time than count decoding. On MNIST and NIST dataset, the time consumption of first-spike decoding is no more than 25% of that of count decoding. Finally, under the count decoding scheme, as the encoding duration increases from 10 ms to 34 ms, the classification accuracy decreases (on MNIST and NIST data, but notably not ETH80).

In the normalization experiments, we test performance of our SNN classifiers using four different STDP learning windows, three normalization rules, and two decoding schemes. These results show that normalization rules could not significantly enhance the classifier's accuracy under the count decoding scheme. However, under a first-spike decoding scheme, output normalization can significantly improve the classifier's accuracy. Further investigation shows that this is because output normalization prevents the classifier from spiking too early. This result holds for MNIST and NIST datasets.

Acknowledgements. This work is supported in part by NSF award IIS-1464349.

References

[1] W. Maass, "Networks of spiking neurons: the third generation of neural network models," *Neural networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[2] F. Ponulak and A. Kasinski, "Introduction to spiking neural networks: Information processing, learning and applications," *Acta neurobiologiae experimentalis*, vol. 4, no. 71, 2011.

[3] H. Markram, W. Gerstner, and P. J. Sjöström, "A history of spike-timing-dependent plasticity," *Frontiers in synaptic neuroscience*, vol. 3, 2011.

[4] G.-q. Bi and M.-m. Poo, "Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type," *Journal of neuroscience*, vol. 18, no. 24, pp. 10464–10472, 1998.

[5] R. K. Mishra, S. Kim, S. J. Guzman, and P. Jonas, "Symmetric spike timing-dependent plasticity at ca3–ca3 synapses optimizes storage and recall in autoassociative networks," *Nature communications*, vol. 7, 2016.

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

[7] F. Ponulak and A. Kasiński, "Supervised learning in spiking neural networks with resume: sequence learning, classification, and spike shifting," *Neural Computation*, vol. 22, no. 2, pp. 467–510, 2010.

[8] T. Masquelier and S. J. Thorpe, "Unsupervised learning of visual features through spike timing dependent plasticity," *PLoS computational biology*, vol. 3, no. 2, p. e31, 2007.

[9] M. Mozafari, S. R. Kheradpisheh, T. Masquelier, A. Nowzari-Dalini, and M. Ganjtabesh, "First-spike based visual categorization using reward-modulated stdp," *arXiv preprint arXiv:1705.09132*, 2017.

[10] E. M. Izhikevich, “Solving the distal reward problem through linkage of stdp and dopamine signaling,” *Cerebral cortex*, vol. 17, no. 10, pp. 2443–2452, 2007.

[11] N. Caporale and Y. Dan, “Spike timing–dependent plasticity: a hebbian learning rule,” *Annu. Rev. Neurosci.*, vol. 31, pp. 25–46, 2008.

[12] W. Gerstner and W. M. Kistler, “Mathematical formulations of hebbian learning,” *Biological cybernetics*, vol. 87, no. 5-6, pp. 404–415, 2002.

[13] E. M. Izhikevich, “Which model to use for cortical spiking neurons?,” *IEEE transactions on neural networks*, vol. 15, no. 5, pp. 1063–1070, 2004.

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

[15] G.-q. Bi and M.-m. Poo, “Synaptic modification by correlated activity: Hebb’s postulate revisited,” *Annual review of neuroscience*, vol. 24, no. 1, pp. 139–166, 2001.

[16] H. D. Abarbanel, R. Huerta, and M. Rabinovich, “Dynamical model of long-term synaptic plasticity,” *Proceedings of the National Academy of Sciences*, vol. 99, no. 15, pp. 10132–10137, 2002.

[17] H. Z. Shouval, S. S.-H. Wang, and G. M. Wittenberg, “Spike timing dependent plasticity: a consequence of more fundamental learning rules,” *Frontiers in Computational Neuroscience*, vol. 4, 2010.

[18] Y. Frégnac, M. Pananceau, A. René, N. Huguet, O. Marre, M. Levy, and D. E. Shulz, “A re-examination of hebbian-covariance rules and spike timing-dependent plasticity in cat visual cortex in vivo,” *Frontiers in synaptic neuroscience*, vol. 2, 2010.

[19] K. A. Buchanan and J. R. Mellor, “The activity requirements for spike timing-dependent plasticity in the hippocampus,” *Frontiers in synaptic neuroscience*, vol. 2, 2010.

[20] S. Scarpetta, A. De Candia, *et al.*, “Storage of phase-coded patterns via stdp in fully-connected and sparse network: a study of the network capacity,” *Frontiers in synaptic neuroscience*, vol. 2, p. 32, 2010.

[21] S. Song, K. D. Miller, and L. F. Abbott, “Competitive hebbian learning through spike-timing-dependent synaptic plasticity,” *Nature neuroscience*, vol. 3, no. 9, pp. 919–926, 2000.

[22] P. Dayan and L. Abbott, “Theoretical neuroscience: computational and mathematical modeling of neural systems,” *Journal of Cognitive Neuroscience*, vol. 15, no. 1, pp. 154–155, 2003.

[23] M. Stimberg, D. F. Goodman, V. Benichoux, and R. Brette, “Equation-oriented specification of neural models for simulations,” *Frontiers in Neuroinformatics*, vol. 8, 2014.

[24] C. A. Curcio, K. R. Sloan, R. E. Kalina, and A. E. Hendrickson, “Human photoreceptor topography,” *Journal of comparative neurology*, vol. 292, no. 4, pp. 497–523, 1990.

[25] A. B. Watson, "A formula for human retinal ganglion cell receptive field density as a function of visual field location," *Journal of Vision*, vol. 14, no. 7, pp. 15–15, 2014.
[26] H. Barlow and D. Tolhust, "Why do you have edge detectors," in *Optical society of America Technical Digest*, vol. 23, 1992.
[27] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of physiology*, vol. 160, no. 1, pp. 106–154, 1962.
[28] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their applications*, vol. 13, no. 4, pp. 18–28, 1998.
[29] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[30] P. J. Grother, "NIST special database 19. NIST handprinted forms and characters database," *World Wide Web-Internet and Web Information Systems*, 2016.
[31] "Max Plank Institut for Informatics. Analyzing Appearance and Contour Based Methods for Object Categorization, ETH80 dataset." [Online.] Available: `https://www.mpi-inf.mpg.de`. Accessed: 2017-08-20.
[32] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on neural networks*, vol. 14, no. 6, pp. 1569–1572, 2003.

Appendix

Choices of parameters

We choose $V_{\text{reset}} = -74\text{mV}$, $V_{\text{rest}} = -70\text{mV}$, $V_t = -55\text{mV}$, $\tau_m = 20\text{ms}$, $\tau_n = 10\text{ms}$, and $\alpha = 10\text{mv}$, $w_{\max} = 20$.