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

Zhengzhong Liang, David Schwartz, Gregory Ditzler, O. Ozan Koyluoglu†‡

Abstract

Spike-timing Dependent Plasticity (STDP) is considered an essential learning mechanism that can capture the causal relationship between events. STDP is considered a foundational element of memory and learning. Previous research efforts have made attempts to understand the learning window of STDP and its functionality in spiking neural networks (SNNs). Classical STDP learning windows have the capability of capturing the causal relationships among events. However, this capability has not been examined under different encoding/decoding schemes in a particular SNN. Moreover, the effect of normalization of weights is also not fully studied in a SNN classifier. Thus in this study, we investigate the interaction among different encoding/decoding schemes, STDP learning windows and normalization rules by running an SNN classifier on MNIST, NIST and ETH80-Contour datasets. Result shows several facts about the cooperation of spiking neural network. TODO: think about how to add result!

1 Introduction

Spiking Neural Networks (SNN) are the third generation of artificial neural networks that aim to emulate the biological activity of a human's and other species' neurons while maintaining a good level of computation ability [1]. Different from traditional Artificial Neural Networks (ANN), which encode data by real-valued numbers, an SNN encodes data in sequence of spikes [2], which is the impulse of neuron's membrane potential caused by stimulus. Signals can be encoded by the temporal sequence of spikes, the rate of spikes, the time interval between spikes, or other forms. Due to the way that SNN processes data, it is considered to have a greater computation ability than traditional ANN. Since the introduction of SNN, many machine learning tasks have been carried out on SNN, including speech recognition, motor control on classification.

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 of STDP is that the connection between two neurons can either be strengthened or weakened depending on the relative spike time of two neurons. If the pre-synaptic spike arrives before the post-synaptic spike, the connection is strengthened, which is long-term potentiation (LTP). However, if the post-synaptic spike arrives first then long-term depression (LTD) is induced, which means the connections is weakened. STDP dwells in a vast area of neural systems, including hippocampus, cerebral cortex, cerebellum-like structures and

*We should come up with a more informative title: was "Spiking Neural Network Classifier"

†Zhengzhong Liang, David Schwartz, and Gregory Ditzler are with University of Arizona, Tucson, AZ. O. Ozan Koyluoglu is with University of California, Berkeley, CA. Email: {zhengzhong.liang, dmschwar, ditzler, ozan}@email.arizona.edu.

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

retinotectal projection [4]. In these different areas, STDP shows different LTD/LTP properties. In recent studies, it is discovered that STDP may exist in absence of LTD [5].

The discovery and subsequent research on STDP prospers the research about SNN[c3]. Many of the works [c4]focusing on using STDP to train [c5]a SNN to finish various [c6]cognitive tasks or machine learning tasks. In these works, different STDP variants are proposed and applied to SNNs with different configurations. In [6], the authors applied classical STDP to an SNN where imaged are converted to poison spikes and fed into the network. An accuracy of 95% on MNIST is achieved using the proposed configuration[c7]. In addition, STDP may also be applied to a network where input signals are encoded using precise spike time. In [7], the author proposed a variant of STDP which enables the network to learn input patterns encoded with precise spike time. Besides, input signals that are encoded by first-spike can also be learned by STDP. In [8], an SNN emulating human's vision system is built tested for its ability of fast feature detection. The author was able to show that first-spike encoding and decoding scheme together with unsupervised STDP enable that network to detect visual features at a very fast speed. Later in [9], the network in [8] is modified and a supervised STDP regulated by reward is added to the network [10]. The modified network is then used for image classification. In addition to those computational trials of different STDP variants, multiple variants of STDP have been observed in biological experiments by scientists[11]. It is showed that the locations and types of synapses can largely influence the STDP learning windows. Finally, normalization mechanisms are proposed to account for the global property of synapse change [12].

In this work, we experiment with an SNN classifier to emulate the structure and coding scheme of a human's visual system, and test multiple STDP variants, including rewarded STDP, and normalization rules on our classifier. In section 2, we formalize the framework for learning an SNN classifier, including the neuron dynamics, synapse modelling, classifier configuration and performance evaluation. In section 3, we present the simulation result over MNIST, NIST and ETH80-Contour datasets. Several conclusions can be drawn from the experimental results. Firstly, although count-decoding scheme gains a higher classification accuracy, it consumes much more time than first-spike decoding scheme. Secondly, when normalization rules are applied, the accuracy of classifier under count-decoding scheme drops if the encoding duration is too long compared with the length of STDP learning window. Finally, choosing an appropriate normalization rule can improve the classification performance of first-spike decoding network, particularly the performance of last few batches.

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 properties of realistic neurons while maintaining a reasonable trade-off with

[c3] huh?

[c4] ~~have been done aiming at~~

[c5] ~~an~~

[c6] ~~coginitive~~

[c7] This is a tough thing to put in here because you're results are not 95%. We need to find a better why to phrase it.

computational complexity. We prescribe the dynamics of this model as governed by (1)[c1]

$$\begin{cases} \frac{\partial v}{\partial t} & = \frac{(V_{\text{rest}} - v + E)}{\tau_m} \\ v & = V_{\text{reset}}, \quad \text{if } v \geq V_t \end{cases} \tag{1}$$

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, 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).

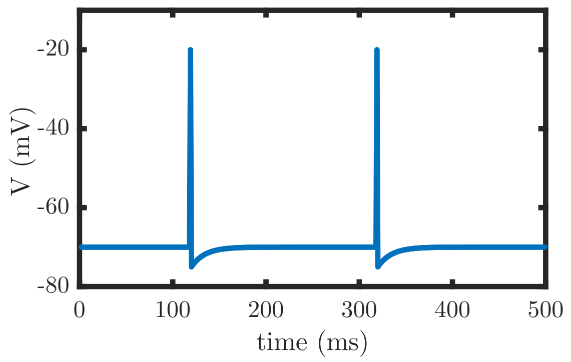


Figure 1: Depicted above is a trace of membrane potential (shown as a function of time) of a typical spiking neuron engaging in two spiking events

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 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 too small, even very small perturbations in E can produce spikes. Figure 1 demonstrates the desired spiking behavior emerges from our prescribed neuronal dynamics.

2.2 Synapse Dynamics and Plasticity

Synapse dynamics determines how the pre-synaptic affects the post-synaptic spike. In this context, we use the model called Spike Response Model (SRM) [13]. 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. Four normalization rules are also applied to our model, which is believed to add global stabilities to our network. Finally, STDP and the normalization rules are accompanied with reward learning which is inspired by the reinforcement learning found in the brain.

[c1]

2.2.1 Synapse Dynamics

Equation 2 describes our dynamical model of synaptic transmission (i.e., the effect on post-synaptic potential induced by incoming spikes)

$$\begin{cases} E_j & = E_j + \alpha \sum_{i=1}^{I} w_{i,j} s_i, \quad \text{if a pre-synaptic spike is received} \\ \frac{\partial E_j}{\partial t} & = -\frac{E_j}{\tau_n}, \qquad \text{otherwise} \end{cases} \tag{2}$$

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[c1], 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 a unitless quantity, included to incorporate the effect of synaptic resistance/conductance. If a post-synaptic spike occurs in the absence of pre-synaptic spikes then E decays exponentially.

2.2.2 STDP Learning Windows

The learning window is described as asymmetrical when STDP was observed for the first time [4][14]. Both LTD and LTP can be formulated by exponential equations with different time constants and amplitudes. Furthermore, recent work has reported that the learning window of STDP may show a symmetrical property [14]. Besides, the shape of learning window can be diversely variant. Experimental results have shown that the learning window is not only affected by the relative spiking time of pre-synaptic spike and post-synaptic spike, but also highly dependent on spike pair frequency, spike pair pattern and the type of synapse [15, 16, 17]. In this section we identified four learning windows.

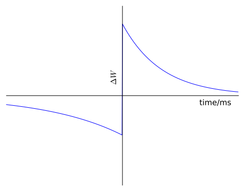
Classical STDP

$$\Delta w = \begin{cases} \mathrm{A}_{\text{pre}} \cdot \exp\left(-\frac{t_{\text{post}}-t_{\text{pre}}}{\tau_{s1}}\right), & t_{\text{post}} > t_{\text{pre}} \\ \mathrm{A}_{\text{post}} \cdot \exp\left(-\frac{t_{\text{pre}}-t_{\text{post}}}{\tau_{s2}}\right), & t_{\text{post}} < t_{\text{pre}} \end{cases} \tag{3}$$

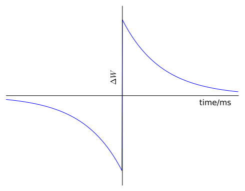
Equation (3) shows a classical STDP weight update rule, demonstrated in [14] to govern variation of synaptic weights as a function of relative spike times in vitro where t_{pre} is the time of the most recent pre-synaptic spike, t_{post}[c2] the time of the most recent post-synaptic spike[c3]and A_{pre} and A_{post} determine learning rate. We choose $\mathrm{A}_{\text{pre}} > 0$ and $\mathrm{A}_{\text{post}} < 0$ 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[c4]- and post[c5]-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 keep the ratio of $A_{\text{pre}} : A_{\text{post}}$ to be the same as the ratio in [14], which is 0.0096:-0.0053. τ_{s1} and τ_{s2} are chosen to be the same as that mentioned in paper, which are 16.8 ms and 33.7 ms.

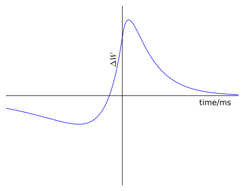
[c1] ~~,~~
[c2] ~~,~~
[c3] ~~,~~
[c4] *Text added.*
[c5] *Text added.*



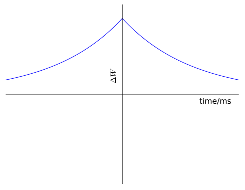
(a) Classical STDP



(b) STDP Variant I



(c) STDP Variant II



(d) STDP Variant III

Figure 2: Learning Windows of STDP Variants

STDP Variant I Shortly after the discovery of classical STDP, it was discovered that the STDP learning window may 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, meanwhile maintaining a symmetric depression window instead of an asymmetrical one. Equation (4) shows the mathematical expression of STDP Variant I, and panel (b) shows a graphical representation of it. In (4), A is chosen to be 0.0096, which is the same as the amplitude of potentiation window of Classical STDP introduced above. τ_s is chosen to be 16.8 ms.

$$\Delta w = \begin{cases} \text{A} \cdot \exp\left(-\frac{t_{\text{post}} - t_{\text{pre}}}{\tau_s}\right), & t_{\text{post}} > t_{\text{pre}} \\ \text{A} \cdot \exp\left(-\frac{t_{\text{pre}} - t_{\text{post}}}{\tau_s}\right), & t_{\text{post}} < t_{\text{pre}} \end{cases} \tag{4}$$

STDP Variant II Figure 2(c) shows a second variant of STDP. This learning window is firstly introduced in [18], aiming at fitting the STDP data collected in experiments. The depression window extends to the $t_{post} - t_{pre} < 0$ area, making it a little different from the classical STDP.

$$\Delta w = \begin{cases} E_N \cdot (A_p e^{-\Delta t/\tau_p} - A_d e^{\eta \Delta t/\tau_p}), \text{ if } \Delta t > 0 \\ E_N \cdot (A_p e^{-\eta \Delta t/\tau_d} - A_d e^{\Delta t/\tau_d}), \text{ if } \Delta t < 0 \end{cases} \tag{5}$$

where A_p and A_d are given by:

$$\begin{cases} A_p = \gamma[1/\tau_p + \eta/\tau_d]^{-1} \\ A_d = \gamma[\eta/\tau_p + 1/\tau_d]^{-1} \end{cases}$$

Equation (5) shows the mathematical expression of STDP Variant II. A special property of this learning window is that the integration form minus infinity to infinity is 0. In our experiment, the normalization coefficient E_N is multiplied in order to make the magnitude of STDP Variant II the same as that of Classical STDP.

STDP Variant III The prototype of the third variant of STDP learning window is found in [5]. The symmetrical variant STDP has no depression window. Besides, the two potentiation windows shows a symmetrical property. The time constant for the learning window in the original paper is too large, so that here we select the learning window to be larger than classical STDP, but still compatible with our classifier. The time constants is chosen to be 33.7 ms, which is the same as the time constants 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 equation 6 and shown in Figure 2(d).

$$\Delta w = Ae^{-|\Delta t|/\tau_s} \tag{6}$$

2.2.3 Weight Normalization

STDP is undoubtedly a powerful synapse plasticity tool being able to capture the causal relationships between events. However, this plasticity is a point-to-point mechanism, which means it is only determined by two directly-connected neurons. However, this plasticity is a point-to-point mechanism, which means it is only determined by two directly-connected neurons. However, we can avoid this restriction and allow plasticity to occur in a much wider range. For example, we can allow the plasticity to be incorporated based on the activity of a group of neurons instead of only two neurons. This is the basic idea of normalization. We hope to add some globally-desirable properties to our network by adding normalization. In this passage, we propose three normalization rules.

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

$$\text{if } \sum_i w_{i,j} > w_{\text{inCons}}, w_{i,j} \leftarrow w_{i,j} \frac{w_{\text{inCons}}}{\sum\limits_i w_{i,j}} \tag{7}$$

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 an upper bound w_{inCons}, and when $\sum\limits_i w_{i,j}^2 > w_{\text{inCons}}$, the weights are normalized as:

$$w_{i,j} \leftarrow w_{i,j} \sqrt{\frac{w_{\text{inCons}}}{\sum\limits_i w_{i,j}^2}}, \text{if } w_{i,j} > 0 \tag{8}$$

Output Normalization on Sum of the Weights

$$\begin{cases} \Delta w_{i,j} & = \mathrm{A}_{\text{pre}} \cdot \exp\left(\frac{t_i - t_j}{\tau_s}\right) \\ \Delta w_{i,k} & = -|w_{i,k}| \cdot \frac{\Delta w_{i,j}}{w_{\text{cons}} - w_{i,j}} \end{cases} \tag{9}$$

Under output normalization[c1], when the synapse $w_{i,j}$ is strengthened, the other connections from neuron i to neuron k are weakened. We implement this competitivity via normalization of output

[c1] Why is the equation above (9) not referenced?

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 strength of connections departing a neuron. Traditionally, as discussed in [12], synaptic competition is considered as implemented by normalization of weights. Our implementation of competitive spike time based learning differs from examples discussed in [12, 19] 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[c1] 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}. The choice of free parameters is discussed in Section 6.1.

We analyze the situation where all synapses have a non-negative strength. In this case where the weight is non-negative, we can remove the absolute value symbol for $w_{i,k}$ in (9). Then we reach equation 10 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. We know that

$$\sum_{k=1,k\neq j}^{K} \Delta w_{i,k} = \sum_{k=1,k\neq j}^{K} \left(-\frac{w_{i,k}\Delta w_{i,j}}{w_{\text{cons}} - w_{i,j}} \right), \tag{10}$$

where K is the number of synapses projected by the neuron in question. 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

$$w_{\text{cons}} = w_{i,j} + \sum_{k=1,k\neq j}^{K} w_{i,k} \tag{11}$$

Combinin (10) and (11) we have:

$$\begin{aligned} \sum_{k=1,k\neq j}^{K} \Delta w_{i,k} &= \sum_{k=1,k\neq j}^{K} \left(-\frac{w_{i,k}\Delta w_{i,j}}{\sum_{k=1,k\neq j}^{K} w_{i,k}} \right) && (12) \\ &= -\Delta w_{i,j} && (13) \end{aligned}$$

Equation (13) shows that when Ψ reaches w_{cons}, competition should impose depression in magnitude equal to that of the potentiation induced by the pair of spikes of neurons i and j[c1]. This should drive the network towards equilibrium and prevents epilleptic destabilization that results from run-away potentiation.

[c1]Neurotransmitters are molecules released at a synapse, and communicated to dendrites of post-synaptic neurons via diffusion across a gap[20].

[c1] Consider rephrasing this sentence

If Ψ remains much smaller than w_{cons} then

$$w_{\text{cons}} = B + w_{i,j} + \sum_{k=1,k\neq j}^{K} w_{i,k}, \tag{14}$$

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. Thus, $\frac{1}{B}$ can be thought of as a measure of competitivity. [c1] Combining (14) and (10) we have

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

$$= -\Delta w_{i,j} \left(\frac{\sum\limits_{k=1,k\neq j}^{K} w_{i,k}}{B + \sum\limits_{k=1,k\neq j}^{K} w_{i,k}} \right) \tag{16}$$

This latter expression shows use that When the neuron has lots of room in its budget of energy and neurotransmitter resources, B is relatively large, thus the decrease, $\Delta w_{i,k}$ is small. In this case, competition is mild. On the other hand, when the Ψ nears w_{cons}, B is very close to zero. Equation 16 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 $w_{i,k}$, so that (9) is obtained. Therefore, no matter $w_{i,k}$ is positive or negative, the change of direction of $w_{i,k}$ is always opposite to that of $w_{i,j}$.

2.2.4 Reward STDP

The weight change of reward STDP is not only relevant to the relative spiking time of pre-synaptic neuron and post-synaptic neurons but also affected 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 is the target then the reward given to network is a normal application of the STDP learning rule. However, if a classifier's output is different than the target then a punishment is applied that will reverse the weight change. In our model, we apply this rule when post-synaptic spike arrives later than the pre-synaptic spike. A detailed role of rewarded STDP will be discussed in section 2.3.

2.2.5 STDP Trace

The modification of synapse caused by STDP is expressed using (17), where pre means all pre-synaptic spikes, post means all post-synaptic spikes and K is the STDP learning window. Equation 17 shows that total modification caused by STDP is the summation of modifications of over all combination of pre-synaptic spikes and post-synaptic spikes. However, if we implement this relationship in software then we will need to record all previous spikes, which is computationally expensive. Therefore, we use an alternative way to represent this relationship by

$$\Delta W = \sum_{\text{pre}} \sum_{\text{post}} K(t_{\text{pre}} - t_{\text{post}}) \tag{17}$$

[c1] You need a better transition here.

Since in our classifier, each pre-synaptic neuron is allowed to emit only one spike during the simulation of each image then (17) can be simplified as (18). For the situation where t_{post} is larger than t_{pre}, this equation does not have to be re-formulated because there is only one pre-synaptic spike. However, for the situation when t_{post} is less than t_{pre}, (18) tells us that when there is a pre-synaptic spike, all post-synaptic spikes that happens before pre-one will influence the synapse[c2]. If we assume the depression window of STDP is an exponentially-decaying function and that $t_{\text{post}} > t_{\text{pre}}$ then (18) can be reformulated as (19).

$$\Delta W = \sum_{\text{post}} K(t_{\text{pre}} - t_{\text{post}}) \tag{18}$$

$$\Delta W = \sum_{\text{post}} A \cdot \exp\left(-\frac{t_{\text{pre}} - t_{\text{post}}}{\tau}\right) \tag{19}$$

$$= A_j \cdot \exp\left(-\frac{t_{\text{pre}} - t_j}{\tau}\right) \tag{20}$$

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. Here j means the latest post-synaptic spike, and $j-1$ is the post-synaptic spike arriving earlier than j[c1]. A_j is defined as the "trace" of STDP[c2]that is defined in [c3] (21) using [c4]a recursive formula[c5], which is given by

$$A_j = \begin{cases} A, & \text{if } j = 1 \\ A_{j-1} \cdot \exp\left(-\frac{t_j - t_{j-1}}{\tau}\right) + A, & \text{otherwise} \end{cases} \tag{21}$$

[c6]The expression in (?) accounts for the 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 use this trick for classical STDP, STDP Variant I and STDP Variant III, [c7]because [c8]it is assumed that the depression window of STDP is in a exponentially decaying manner [][c9]. [c10]However, this assumption of an exponentially decaying depression window does not hold. [c11]Therefore, we ignore the impact of previous post-synaptic spikes [c12]for STDP Variant II. [c13]Only the influence of the latest post-synaptic spike is considered when there is a pre-synaptic spike.

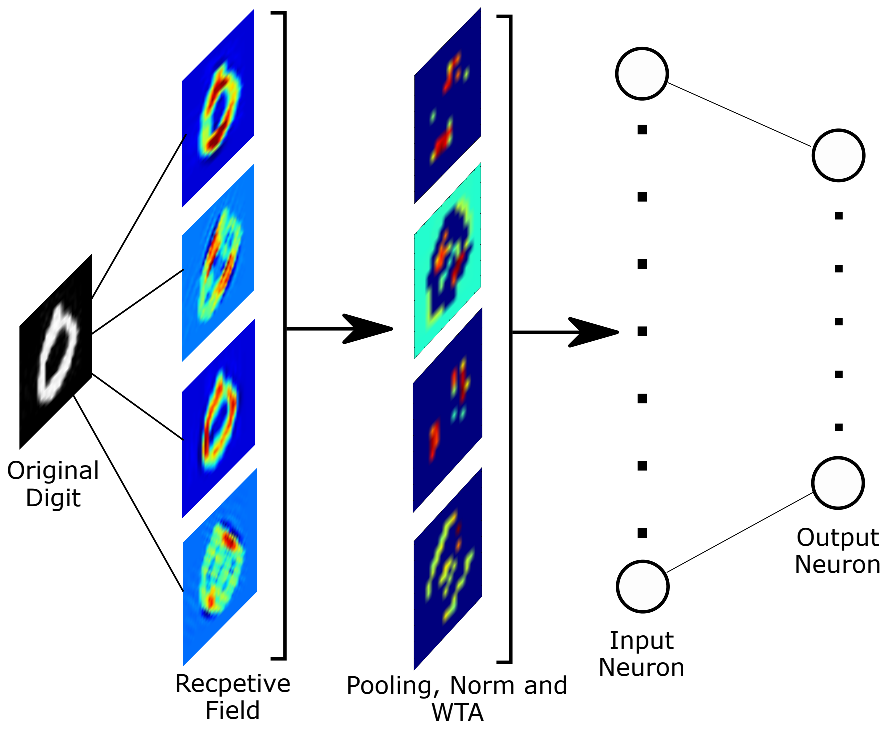


Figure 3: Structure of our SNN Classifier

2.3 Spiking Classifiers

Following the example of [9][c14], we implemented a network of leaky integrate and fire (LIF) neurons with plastic synapses with the dynamics described in 1 to construct a 4-layer feedforward SNN. Most of the image pre-process protocols and network properties are adopted from [9], whereas there are indeed some modifications. For example, the network used in [9] has additional hidden layer between the input neurons and output neurons, whereas we eliminated that layer in our network. The structure of our proposed network is illustrated in Figure 3. All other configurations are the same as those proposed in [9].

We now describe the preprocessing of the data. The images are fed into a receptive field layer of

[c2] This sentence needs some restructuring
[c1] I have a question about this expression. See me.
[c2] ~~, and it~~
[c3] ~~equation~~
[c4] ~~recurssion~~
[c5] ~~.~~
[c6] ~~In this way the impact of all previous post-synaptic spike can be recorded meanwhile a high computation efficiency of code can be maintained because we only have to store the value of A_j in order to compute the depression.~~
[c7] ~~becuase~~
[c8] ~~this trick replies on an assumption~~
[c9] add a reference
[c10] ~~However for STDP Variant II this assumption does not hold anymore~~
[c11] ~~Thus for STDP Variant II~~
[c12] *Text added.*
[c13] ~~Instead, when there is a pre-synaptic spike, only the influence of the latest post-synaptic spike will be considered.~~
[c14] *Text added.*

four orientations, which can be mathematically described as Gabor filters of 0 degree, 45 degrees, 90 degrees 135 degrees. After this operation the original image, whose size is 28×28 pixels, is projected to a higher dimension of 32×32 pixels by four orientations. The output of this first layer processing is then sent through max-pooling and winner-takes-all (WTA). This max-pooling and WTA operations impose a dimension reduction to the input, which is hopefully able to remove redundant information and speed up the simulation time. 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 instead of many of spikes, which means during the simulation of each single image each neuron can only spike once. The latency of that single spike encode pixel information. The latency of each spike is determined by the intensity of corresponding pixel. We evaluated three conversion rules such that in different conversion rules the spikes are mapped using different latency scales. The spikes are generated within a duration of either 10 ms, 17 ms or 34 ms, which are approximately half the length of STDP time constants and the length of STDP time constants and double length of STDP time constants [c1]. Spikes are finally propagated from the LIF neurons to the output layer. The number of neurons in output layer is the same as the number of classes of the classification problem. For example, MNIST dataset has 10 classes, as a result there are 10 neurons in the output layer when the network is used for MNIST 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). At each time step there is a post-synaptic spike and the weights are updated according to both the target and spiked neuron. If the spiked neuron is the target neuron, reward is given to the network, and punishment is given otherwise. After the weight change, normalization rules are applied to weights. And finally the weights are clipped to a range from -0.3 $w_{\max}$ to $w_{\max}$, where $w_{\max}$ is the upper bound of the strength of every individual synapse. More detailed description of the whole process are presented from section 2.3.1 to section 2.3.3.

2.3.1 Encoding

Any natural (i.e., biologically implemented) spiking neural classifier - especially ones 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 [21]. 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 [22]. 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 several other computations) [23, 24].

We emulate this natural visual structure in our model. As shown in Figure 3, the input image is firstly fed into a receptive field layer, which is achieved by implementing Gabor filters of four orientations. Equation 22 shows the Gabor filter. In the equation, x and y are the coordinates of pixels, θ is the orientation applied. The encoding procedure emulates the functionality of the receptive fields in human's visual system. Then the original picture is projected to a higher dimension and the edges of the four orientations are extracted. This receptive field in essence projects the original sample to a higher dimension, thus making the previously inseparable problem separable[c1],

[c1] Notice this sentence

[c1] do you mean nonlinearly separable problem linearly separable

which is [c2] similar to the kernel tricks in support vector machine [][c3].

$$g(x,y;\lambda,\theta,\psi,\sigma,\gamma) = \exp\left(-\frac{x'^2+\gamma^2 y'^2}{2\sigma^2}\right)\cos\left(2\pi\frac{x'}{\lambda}+\psi\right) \tag{22}$$

The second encoding procedure is max-pooling and winner-takes-all. In this procedure, the input is firstly applied to a 2 by 2 max-pooling operation with stride 2. Then the max-pooled image is applied with winner-takes-all(WTA) rule. Algorithm 1 shows the flow of a WTA operation. As stated above, orientations have 4 values, 0 degree, 45 degrees, 90 degrees and 135 degrees. The "return max orientation" function at line 1 in Algorithm 1 returns the orientation with the largest value at position (x, y). The pixel value at (x, y) in orientation j is maintained, while $p(x, y)$ in other orientations are set to 0. For example, pixel with coordinates (x_0, y_0) in orientation 1 is denoted by $p_1(x_0, y_0)$, the pixel with coordinates (x_0, y_0) in orientation 2 is denoted by $p_2(x_0, y_0)$, and the same notation applies until orientation 4. 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 $p_i(x_0, y_0)$ maintains the original value. After max-pooling and WTA, the dimension of input is reduced from 4096 to 1024, which emulates the dimension reduction in human's visual system. In addition, this operation can significantly reduce the time consumed in simulation.

Algorithm 1 WTA Encoding

```
Input: Orientation $i$; Pixel Position $x$,$y$; Pixel value $p_i(x, y)$
Output: Updated pixel value, $p'_i(x, y)$
1: $j$ = return max orientation($p_i(x, y)$)
2: for $i$ in orientations do
3:     if $i \neq j$ then
4:         $p_i(x, y) = 0$
5:     end if
6: end for
```

The final stage of encoding is to convert pixel intensity to spike time. Mozafari et al. proposed that the latency of encoded spike should be inversely proportional to its intensity; however, their work did not produce a formal mathematical expression. Therefore, we propose to use their idea about the proportionality, but provide a formal expression. Our proposed encoding scheme, which is able to encode pixel into three different encoding durations, is given by

$$\begin{cases} t_{x,y} = \frac{(D-3)}{I_{x,y}} - D + 4, & \text{if } I_{x,y} > 0.5 \\ t_{x,y} = None, \text{otherwise} \end{cases} \tag{23}$$

where $I_{x,y}$ is the intensity of pixel at position (x, y), whose value is between 0 and 1, D is the simulation duration. This rule pixels with intensity below 0.5 are discarded and pixel with a larger intensity is converted to a spike with a lower latency. All spikes are scaled to a latency range from 1 ms to $D - 2$ ms. The teach signal is then given at $D - 1$ ms. To study the collaboration between the STDP learning window and simulation duration, we hereby tested 3 different encoding durations: 10ms, 17ms and 34ms.[c1]

[c2] ~~somehow~~

[c3] add a reference here.

[c1] I change the previous two paragraphs quite a bit.

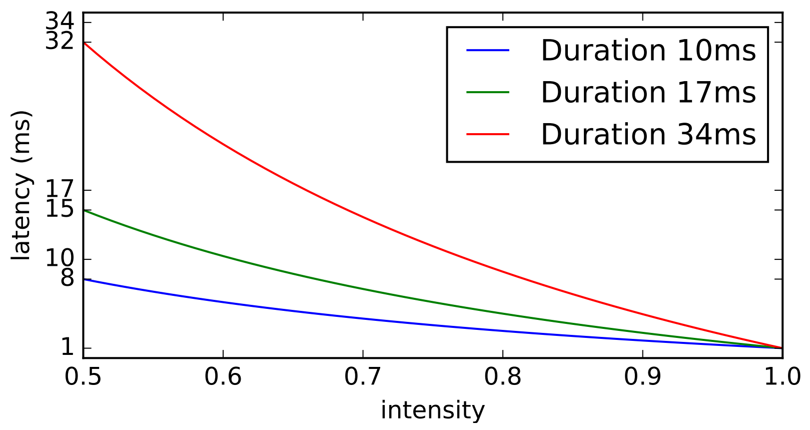


Figure 4: Encoding Duration and Intensity

2.3.2 Decoding

As is proposed in [9], we evaluated two decoding schemes, which are first-spike decoding and count-decoding. Our model choose the number of output neurons to be the same as the number of classes. First-spike decoding chooses the class with the output neuron that is the first to emits a spike and count-decoding examines the number of spikes at each output neuron and assign the class to the neuron with the most spikes.

2.3.3 Learning Process and Weight Update (I edited this section directly)

The network uses a rewarded STDP mechanism, which is a supervised learning algorithm that us used in [9] (see Algorithm 2). The algorithm takes in X_{train}, which is a list of images, and Y_{train}, which is the list of targets (i.e., the label of each image). A single image from the list is converted to spikes X then fed into the network at each round of learning at these spikes are denoted by X_t. The spikes emitted by the SNN classifier, S_t, are then determine by sending the X_t through the network (see line 6). Then S_t is compared to Y to determine whether reward or punishment should be given to the network. Then STDP rules are applied to the network with the reward/punishment to update the of weights of the network (see lines 9 and 12, respectively). After the STPD update we apply our proposed weight normalization rule to the network. There are two types of normalization rules in the simulation: (1) normalization of each individual synapse; and (2) normalization of a cluster of synapses. Normalization for an individual synapse is implemented by constraining the strength within a range from $[-0.3 \cdot w_{\max}, w_{\max}]$ and the cluster normalization is implemented using the rules in section 2.2.3 (see line 14). Finally, at epilogue of each single simulation, the teach signal is given. The teach signal is placed at the target neuron 1ms before the end of each single simulation, which induces a single post-synaptic spike on the target neuron.

The training process for first-spike decoding SNN classifier follows a similar approach as count decoding classifier (see Algorithm 3); however, the major difference is the end of the STDP simulation. First-spike decoding scheme the simulation ends when there is only one post-synaptic spike at an output. This is detected by spike flag F_s[c1]. Every time when there is a post-synaptic spike, the spike flag F_s is set to 1. This learning process is repeated for both count and first-spike decoding.

[c1] I am not sure you're using the term correctly

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

```
 1: for each image, $X \in X_{\text{train}}$, $Y \in Y_{\text{train}}$ do
 2:     for $t$ in $T_s$ do
 3:         if $t = D - 1$ then
 4:             $N \leftarrow$Teach($N$,$Y$)
 5:         end if
 6:         $S_t$ =update($N, X_t$)
 7:         if $S_t \neq Y_t$ then
 8:             $reward = -1$
 9:             $N \leftarrow$STDP($N$,$reward$)
10:         else if $S_t = Y_t$ then
11:             $reward = 1$
12:             $N \leftarrow$STDP($N$,$reward$)
13:         end if
14:         $N \leftarrow$Norm($N$)
15:     end for
16:     $N \leftarrow$Reset($N$)
17: end for
18: Return $N_c \leftarrow N$
```

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

```
 1: for each image, $X \in X_{\text{train}}$, $Y \in Y_{\text{train}}$ do
 2:     $F_s = 0$
 3:     while $t$ in $T_s$ and $F_s == 0$ do
 4:         if $t = D - 1$ then
 5:             $N \leftarrow$Teach($N$,$Y$)
 6:         end if
 7:         $S_t$ =update($N, X_t$)
 8:         if $S_t \neq Y$ then
 9:             $reward = -1$
10:             $N \leftarrow$STDP($N$,$reward$)
11:             $F_s = 1$
12:         else if $S_t = Y$ then
13:             $reward = 1$
14:             $N \leftarrow$STDP($N$,$reward$)
15:             $F_s = 1$
16:         end if
17:         $N \leftarrow$Norm($N$)
18:     end while
19:     $N \leftarrow$reset($N$)
20: end for
21: return $N_c \leftarrow N$
```

3 Experimental Results

3.1 Performance Testing Methods and Data

We now present an empirical analysis of the impacts of encoding/decoding schemes and normalization of the SNN weights. We benchmarked the performance of our SNN classifiers on three datasets: MNIST [], NIST [], and ETH80-Contour []. MNIST contains about 60,000 hand-written digits (50,000 training and 10,000 testing samples) belonging to 10 classes. NIST is a hand-written character and digit dataset we only use 10 class to evaluate the performance of our proposed normalization scheme. ETH80-Contour contains eight classes of objects that include fruits, animals and cars, and it is often used for 3-D image reconstruction. The training set of MNIST data is parsed into 50 non-overlapping batches. The classifier is trained consecutively on each batch then the classifier is evaluated on 9,000 (10,000?) testing samples and the accuracy is reported. The samples within each datasets are shuffled before fed into the classifier. The same protocols are employed to NIST as well as ETH80-Contour, except that the batch size, number of batches and testing size are different.

3.2 No Normalization Experiment

In this section we conducted a series of experiments without the useage of normalization rules. We studied the influence of STDP learning window and encoding/decoding on the influence of proposed SNN classifier. he 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 a range from 1 ms to (Duration-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 classifier under count decoding scheme and first-spike decoding scheme, 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. Character C means count decoding and F means first-spike decoding. Different STDP learning windows are tested, where STDP-C means 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 bathes, and the encoding durations are 10 ms, 17 ms and 34 ms, so that d = MNIST, s = STDP-I, $N_B = 50$ and $N_E = 3$. In addition, we also compute the standard deviation and plot the error bar in figure. Standard deviation is given by: $s^{d,s} = \sqrt{[\sum_{j=1}^{N_E}\sum_{i=1}^{N_B}(p_{i,j}^{d,s} - p_m^{d,s})^2]/(N_B \cdot N_E)}$. Finally, the error bar is computed as $err = \pm 1.96\frac{s^{d,s}}{(N_B \cdot N_E)^{0.5}}$. Result in figure 5 shows that under count decoding scheme, classical STDP performs significantly well than other STDP variants on MNIST dataset, and it performs significantly better than STDP-III on NIST dataset. However, this phenomenon does not show up on ETH-80 Contour dataset. As for first-spike decoding, since the simulation process stops as long as the first post-synaptic spike is emitted, the depression window of STDP will never take into effect. Thus only the shape of potentiation window can affect classification performance. Since STDP-C and STDP-I has the same potentiation window,

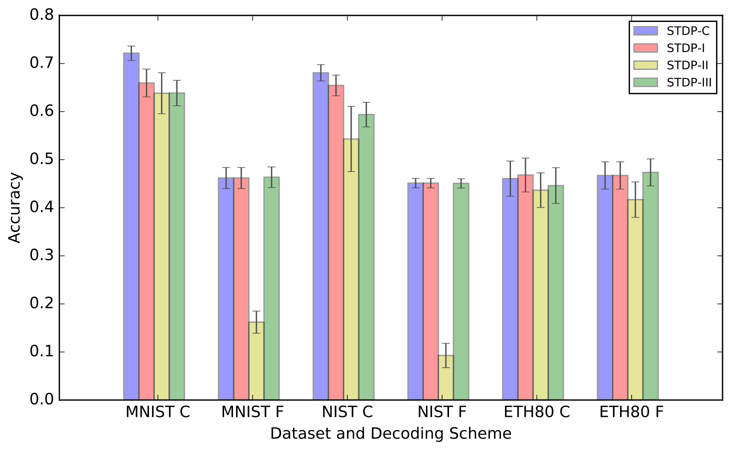


Figure 5: Average Accuracy of STDP W/O. Norm

the performances of them are exactly the same 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. So the performance of STDP-III is a little different from that of STDP-C and STDP-I. STDP-II performs poorly on MNIST and NIST dataset under 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

Even though the testing accuracy of classifier using count decoding is significantly better than that using first-spike decoding, this does not imply count decoding is superior to first-spike decoding. Because first-spike decoding uses much less time than count decoding. Figure 6 shows the simulated time per image under count decoding scheme and first-spike decoding scheme. Simulated time does no mean the execution time of program. Instead it means 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. So simulated time here is equivalent to the simulated steps. The top 3 panels of figure 6 shows the simulated time per image for classifier using different encoding duration. The simulated time for both training session and testing session are recorded. Under count decoding, the simulated time for processing each image is a constant. If we use D to denote the encoding duration, then the simulated time for processing each image is $D - 0.2$ ms. Result shows that on MNIST and NIST datasets, the simulated time for both training and testing session under first-spike decoding is significantly less than that using count-decoding. On ETH80 dataset, this phenomenon is not as significant as that on the other 2 datasets, but the gap is still not too subtle to distinguish. The bottom 3 panels of figure 6 shows 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 increase from 10 to 34 ms, the ratio continues dropping. In addition, this ratio is even lower in testing session than in

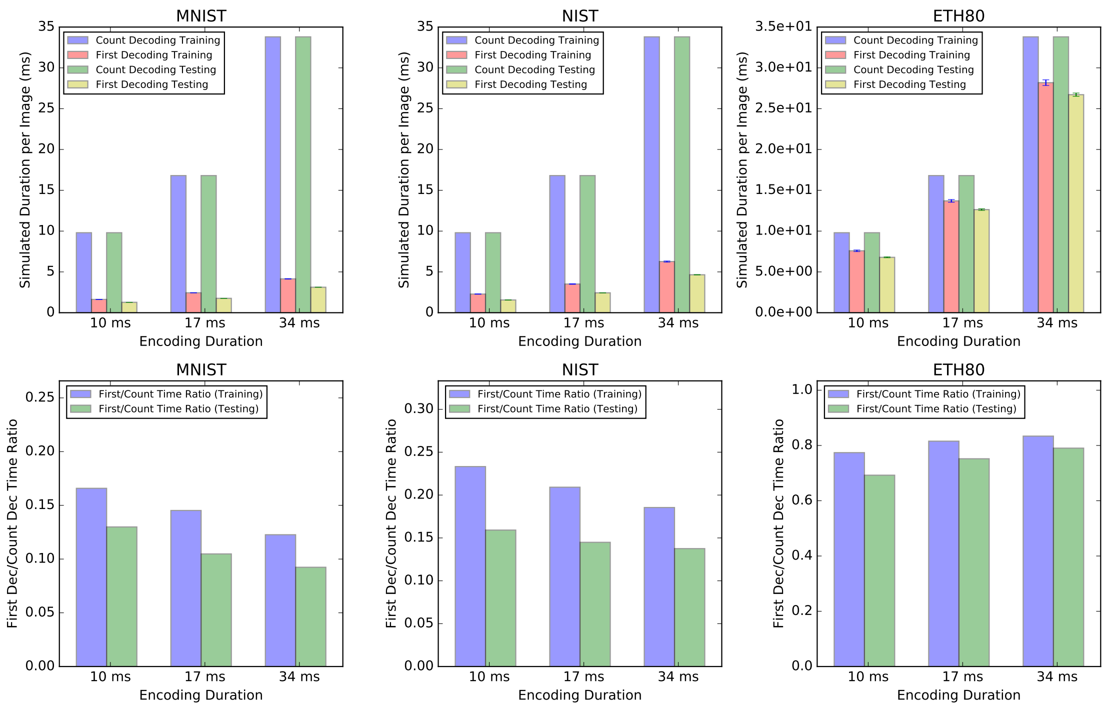


Figure 6: Simulation Time using Count Decoding and First Decoding Scheme

training session. Figure 5 and figure 6 combined show that although using first-spike decoding may experience a decrease in classification accuracy, the reduction in simulated time is also significant. The neural classifier using first-spike decoding scheme needs much less time to either train or test.

3.2.3 Accuracy Comparison w.r.t. Encoding Duration

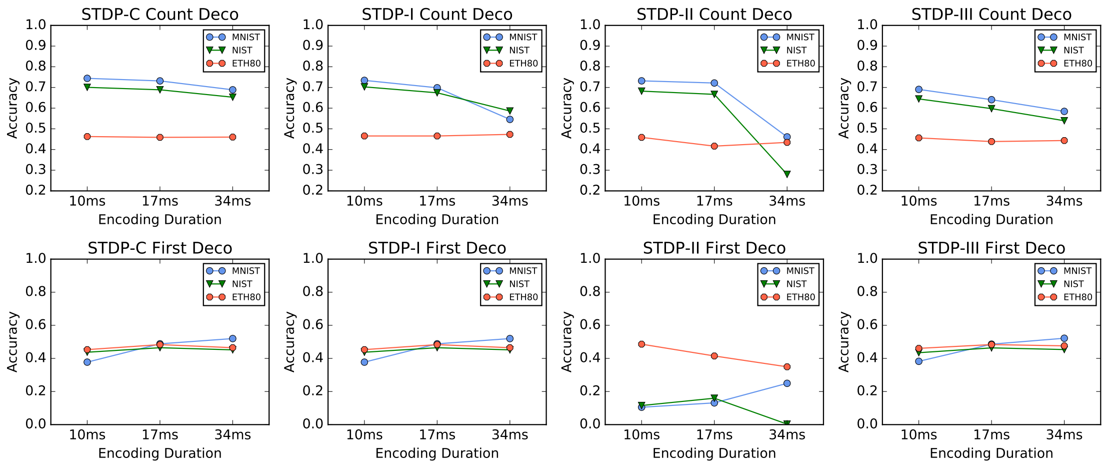


Figure 7: The Effect of Encoding Duration on Classification Accuracy

Figure 7 shows the testing accuracy of classifier under different encoding schemes. The 4 sub figures in the first row of figure 7 shows the performance of 4 STDP learning windows under count decoding scheme, whereas the bottom 4 figures shows the performance of STDP learning windows under first-spike decoding scheme. Each point in the figure means 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 increase 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 first-spike decoding scheme, there is not such an 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 3 learning windows, the increase in encoding duration will lead to an increase in classification accuracy on MNIST dataset. However that 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 the 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 the classification accuracy keeps dropping as the encoding duration increases.

3.3 Normalization Experiment

In this section, we [c1]evaluate the three normalization rules [c2]that we are proposing. [c3]For these experiments, the encoding duration of [c4]the SNN classifier is fixed to [c5] 10 ms, and there are several reasons for this choice. First[c6], the experiments using count-decoding scheme in section 3.2 [c7]demonstrated that 10 ms encoding [c8] [c9]in general provide the best accuracies compared to the other encoding durations. Second[c10], the influence of encoding duration on the classifier's performance under first-spike decoding scheme is not obvious[c11], but [c12] to compare the performance of classifier under different decoding schemes[c13] we [c14]need to set the encoding duration of all [c15]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 seem to make no difference either using count decoding scheme or using first-spike decoding scheme. For MNIST and NIST dataset, the influence of output normalization rule make a difference. Under count decoding, the output

[c1] ~~applied 3~~
[c2] ~~to the neural classifier~~
[c3] ~~T~~
[c4] *Text added.*
[c5] ~~be~~
[c6] ~~ly~~
[c7] ~~shows~~
[c8] ~~duration~~
[c9] ~~enables the network to obtain a better performance in most cases~~
[c10] ~~ly~~
[c11] ~~. B~~
[c12] ~~in order~~
[c13] ~~,~~
[c14] *Text added.*
[c15] *Text added.*

normalization rule undermines the classifier's performance. Whereas under first-spike decoding, the output normalization rule can significantly enhance the performance. On MNIST and NIST dataset, the average classification accuracy of classical STDP under count decoding reaches around 70%, whereas the accuracy of classical STDP under first-spike decoding on these 2 datasets are only around 40%. However, if output normalization rule is applied, the classification accuracy reaches around 60%, which is only about 10% lower than count decoding. Meanwhile first-spike decoding consumes much less time than count decoding. 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 20% of the time consumed under count decoding. On NIST dataset, first-spike decoding classifier takes about 30% of the time under count decoding. Although first-spike decoding scheme takes slightly more time than merely using first-spike decoding, it still stays at a low time consumption level. Thus first-spike decoding scheme together with output normalization rule provides an alternative option in addition to count-decoding scheme. The combination of first-spike decoding scheme provides comparable classification performance in comparison with that of count decoding scheme, meanwhile maintaining a low level of time consumption.

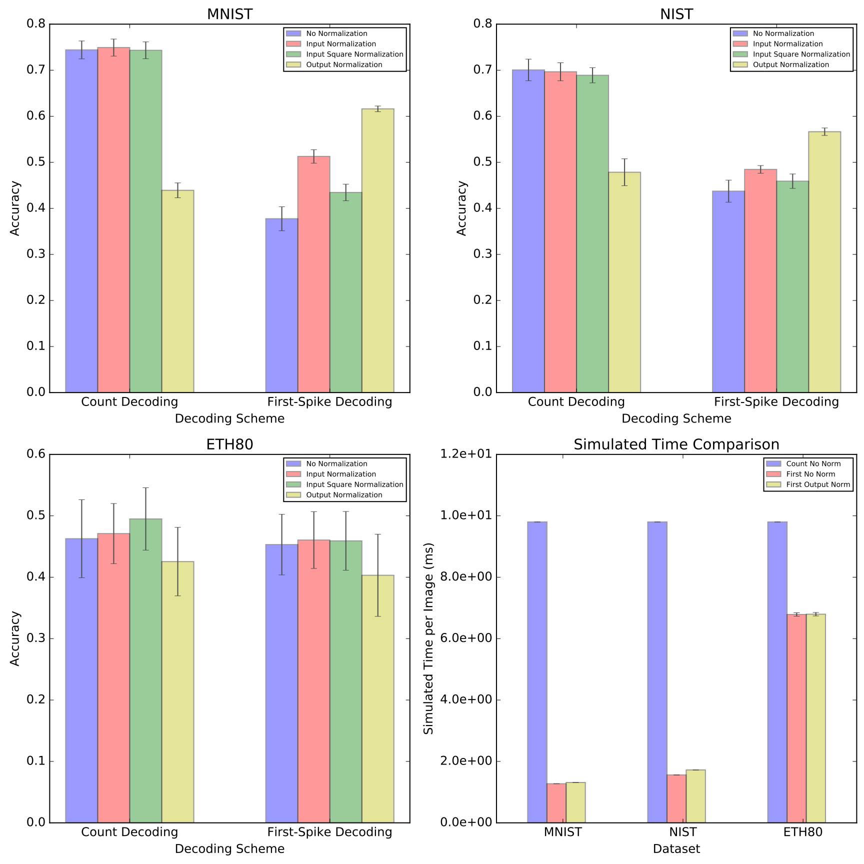


Figure 8: Performance of STDP with Normalization Rules

3.3.2 Early Exit

To investigate how output normalization rule actually boosts the performance of our proposed classifier, we record and plot the training and testing accuracy of each batch on all 3 datasets with and without output normalization rule. Figure 9 shows the plot of training and testing accuracy. Training accuracy is obtained by testing the classifier using all samples in current training batch. The 3 figures in the first row shows the performance of classical STDP/STDP Variant I on the 3 datasets. Since classical STDP and STDP Variant I have the same performance under first-spike decoding scheme, we merged the figures together into 1 group. 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 start to drop, until the end of training session. The same result shows up in experiment of STDP Variant III. This result shows that although first-spike decoding classifier without normalization rule can successfully extract some image feature in the beginning of training session, it fails to maintain these features. After a few batches, the classifier fails to extract those important features from image. By comparing training and testing accuracy, we can conclude that the drop in accuracy in later batches of training session is not because of over-fitting. Because when over-fitting happens, training accuracy should keep going up, meanwhile testing accuracy should have a bell-shape curve. However, in our experiment, there is no such relationship between training and testing accuracy. Instead, training and testing accuracy are very close from the beginning of training process until the end of training process. But there are exceptions in our experiment. For STDP Variant II, the trend reported above only slightly shows up on NIST dataset but not shows up on MNIST dataset. For all STDP learning windows on ETH80 dataset, no such trend shows up. There is no “dropping" in later batches training process even without normalization rule.

To further investigate why output normalization rule can prevent accuracy from dropping in later batches of training process, we plot the time consumption of classical STDP in each training batch. Figure 10 shows the trend of time consumption along with batch. On MNIST dataset, the training time of the first batch is about 2.6 s for both no-normalization classifier and output-normalization classifier. Since there are 1000 training images in each batch in MNIST dataset, it means each image takes about 2.6 ms to train on average. As the training process continues, the training time for both no-normalization classifier and output-normalization classifier drops. At the end of 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 show up on NIST dataset. Both training and testing time for output-normalization classifier are larger compared with those of no-normalization classifier. Combined with the results from previous experiments on MNIST and NIST, some inference can be made here. Recall that in previous experiments we showed that output-normalized classifier under first-spike decoding scheme performs better than no-normalization classifier on MNIST and NIST dataset, especially in later phase of training process. Meanwhile no-normalization classifier consumes less time than output-normalized classifier, especially in later phase of training process. It means that although no-normalization classifier makes the judgement of classification at an earlier stage of the simulation of each image, this judgement is more likely to be mis-judgement. In comparison, output-normalized classifier tends to make its judgement at a later stage of the simulation of each image. And this delay of judgement seems to boost the performance of classifier. This hypothesis also makes sense if we analyze the judgement process of classifier. Under first-spike decoding scheme, the classification result is given by 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 judgement on average in the last

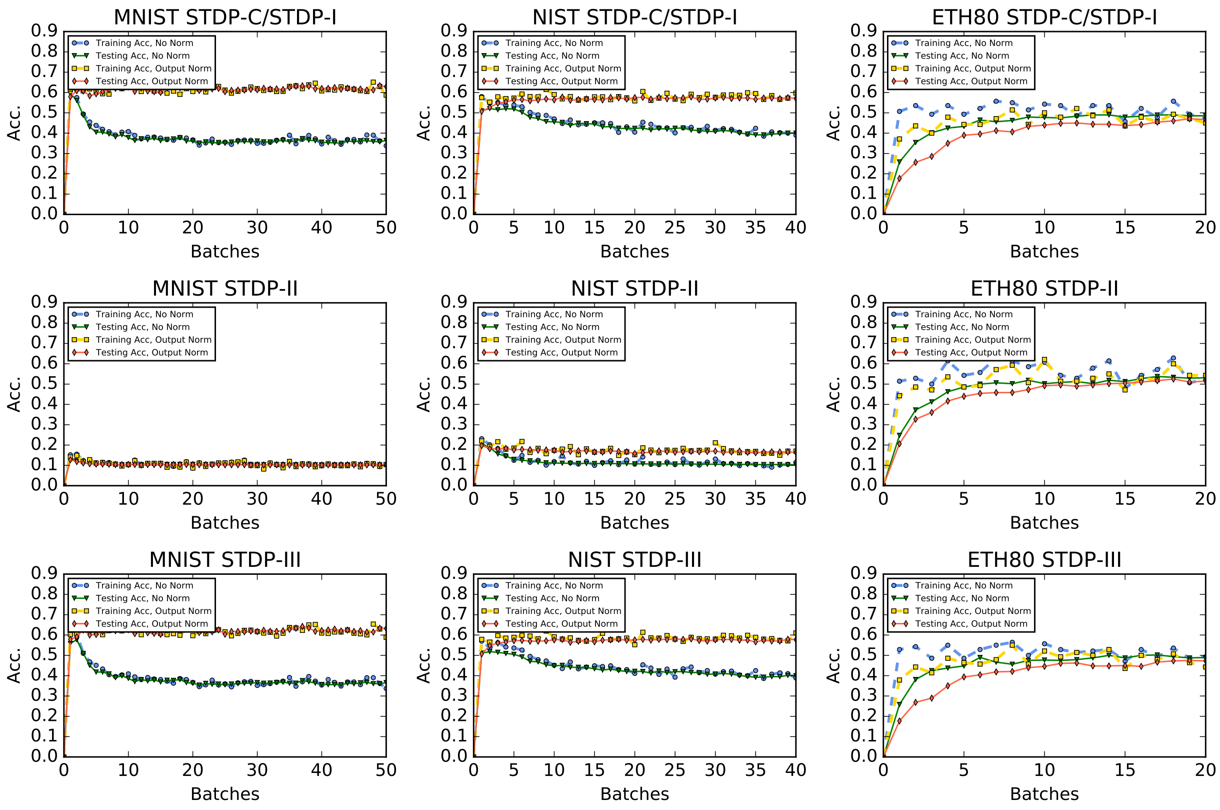


Figure 9: Training and Testing Accuracy

few batches. Meanwhile, recall that under 10 ms encoding duration, the input spikes are encoded to a latency ranging from 1 ms to 8 ms. So the first spike of each image could not be earlier than 1 ms. It means that the classifier tends to make the judgement only using the spikes whose latency are from 1 ms to 1.5 ms. This implies that the classifier tends to classify the image using very few features of each image because many other features are encoded with latency ranging from 1.5 ms to 8 ms. In contrast, output-normalized classifier tends to make judgement using a longer time, which enables the classifier to utilize more features of each image, thus reducing mis-classification. On ETH80 dataset, the difference of time consumption between no-normalization classifier and output-normalized classifier is not obvious. This is in accordance with the fact that the performance difference between no-normalization classifier and output-normalized classifier is not obvious on ETH80 dataset.

Finally we investigated the influence of output normalization on the classifier's weight. Weight mean is calculated using the weight after the last batch of training process. The figure shows that after using output-normalization rule, the weight mean on MNIST and NIST tasks is less than that of no-normalization classifier. In the right figure, the max output sum is calculated using:

$$W_{\max,\mathrm{S}} = \mathrm{Max}(W_i) \tag{24}$$

$$= \mathrm{Max}(\sum_{j=1}^{J} w_{i,j}) \tag{25}$$

When no output normalization rule is set up, there is only upper and lower bound for the strength

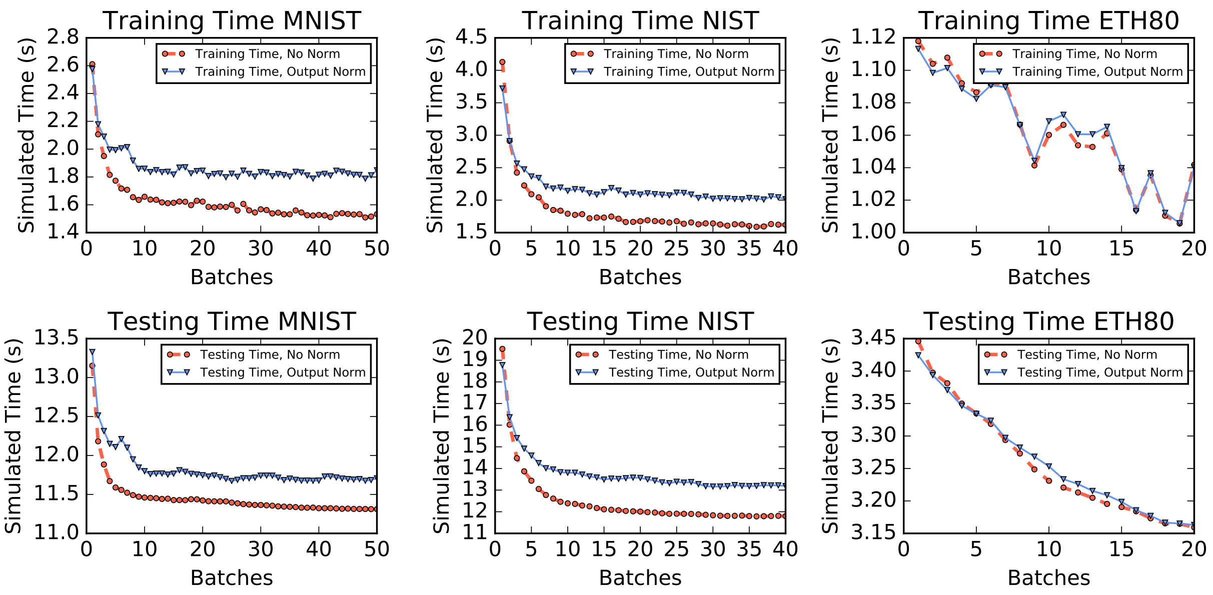


Figure 10: Time Consumption in Each Batch

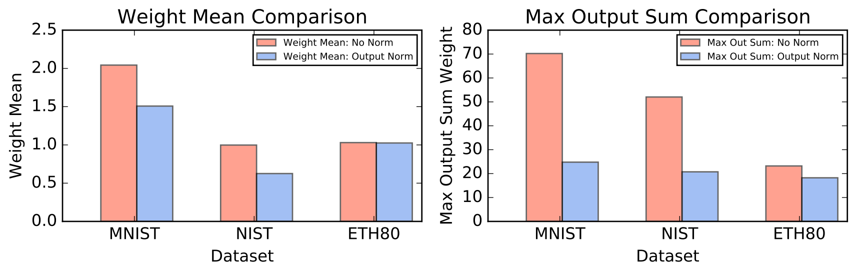


Figure 11: Weight Comparison

of each individual synapse. The upper bound for each individual synapse $w_{\max}$ is set 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 1 neuron reaches 20. When output normalization rule is applied, the output sum of strength is constrained to be less than or equal to 30. Figure 11 shows this comparison. On MNIST and NIST, both the mean weight and the maximum output sum weight are larger when no normalization rule is applied.

4 Discussion

[c1]Several interesting conclusions can be drawn from the sequence of experiments that were performed. [c2]First, one unexpected outcome is that although the results [c3] from [c4]the MNIST and NIST are [c5]mostly consistent[c6]; however, these results are not [c7]always consistent with those obtained from ETH80-Contour dataset. In the first experiment[c8], we compared the performance of [c9]multiple STDP learning windows [c10]with different encoding [c11] and decoding schemes without normalization rule. On MNIST and NIST dataset, the performance of different STDPs is distinguishable[c12]; however, we did not observe such difference with the ETH80-Contour dataset[c13]. Secondly, the encoding duration of [c14]the SNN has a different influence on ETH80 in comparison with MNIST and NIST, especially [c15]with count-based decoding[c16].[c17]

The output normalization rule can significantly boost the performance of classifier under first-spike decoding scheme on MNIST and NIST dataset, whereas this trend does not show up on ETH80-Contour dataset. Some reasons may account for this outcome[c1], such as there are much fewer samples in ETH80 datasets than in MNIST dataset and NIST[c2]. It is possible that the number of samples in ETH80 is not enough for the classifier to demonstrate a trend obtained from a larger dataset. [c3]This [c4]could [c5]caused as a result of the difference of the input dimension [c6]of MNIST/NIST and ETH80. [c7] MNIST/NIST [c8] [c9]have input image[c10]s [c11]that are converted to a size of [c12]32×32 pixels after convolution and max-pooling[c13], whereas of [c14]16×16 [c15] with the

[c1] *Text added.*
[c2] ~~One~~
[c3] ~~obtained~~
[c4] *Text added.*
[c5] ~~mainly~~
[c6] ~~,~~
[c7] *Text added.*
[c8] tell the reader which experiment or what section they need to be looking at.
[c9] ~~different~~
[c10] ~~under~~
[c11] ~~durations~~
[c12] ~~. But on~~
[c13] ~~there is no such difference~~
[c14] ~~classifier~~
[c15] ~~under count~~
[c16] ~~scheme~~
[c17] This observation is great, but the question becomes, why is the ETH80-Contour causing these different results. I seem to remember us having a conversation about how the ETH80 was extremely easy to classify and may not be the best benchmark; however, you felt that it was important to include it. I think there needs to be a discussion about why these results are observed.
[c1] ~~. Firstly,~~
[c2] ~~dataset~~
[c3] ~~Secondly, t~~
[c4] ~~difference~~
[c5] ~~be because of~~
[c6] ~~between~~
[c7] ~~For~~
[c8] ~~data,~~
[c9] ~~each~~
[c10] *Text added.*
[c11] ~~is~~
[c12] ~~32*32~~
[c13] ~~instead~~
[c14] ~~16*16~~
[c15] ~~as in MNIST/NIST~~

ETH80 data set. This [c16]difference is because the initial size of the images in ETH80[c17]. [c18]Finally, this could be because [c19]of the innate characteristics of the images in ETH80[c20] dataset. More experiments [c21]can be carried out to investigate the dependence of the classifier's performance on datasets.

As a complex system, the neural classifier proposed here has many hyper-parameters, including the time constant of neurons, the time constant for post-synaptic current, the maximum strength for each individual synapse, the overall maximum strength of the synapses departing from one neuron and many other parameters. These parameters model the dynamics of the whole system thus being very likely to influence the performance of classifier if we change them. As a result, the optimization of these hyper-parameters can be studies in future research. Furthermore, more experiments would be desirable to study the effectiveness of the normalization rules on networks with different neuron models. Currently we are implementing the network based on Leaky Integrate and Fire (LIF) model. However this is a highly simplified neuron model which mimics the neuron dynamics very coarsely. In later experiments some more realistic neuron models can be utilized. For example, Izhikevich model is know for the ability of mimic the behaviours of multiple neurons meanwhile maintaining a low level of computation complexity. In later experiment, the LIF neurons can be replaced with Izhikevich neurons so that we can study the influence of neuron behaviours on the classifier's performance.

Another interesting topic would be the combination of different decoding classifiers. As stated in our work, first-spike decoding classifier tends to use much less time than count decoding classifier. On the other hand first-spike decoding classifier also gains a lower classification accuracy. However, this does not imply that first-spike decoding classifier is inferior to count decoding classifier. Instead, classifiers with different decoding manners may compensate each other's shortcoming. In some situation where classification speed is extremely important, first-spike decoding may take into effect. In other situations where classification accuracy is important, count decoding scheme may prevail. The result in our work may imply that multiple decoding schemes are in human's visual system. And how are these different decoding schemes cooperate together may be an interesting topic.

5 Conclusion

[c1]To be written.

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.

[c16] *Text added.*

[c17] ~~is much larger than the size of image in MNIST/NIST~~

[c18] ~~Thirdly~~

[c19] *Text added.*

[c20] ~~Contour~~

[c21] ~~should~~

[c1] *Text added.*

[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] A. M. Andrew, "Spiking neuron models: Single neurons, populations, plasticity," *Kybernetes*, vol. 32, no. 7/8, 2003.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] H. Barlow and D. Tolhust, "Why do you have edge detectors," in *Optical society of America Technical Digest*, vol. 23, 1992.

[24] 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.

6 Appendix

6.1 Choices of parameters

In the classical STDP framework, we choose $V_{\text{reset}} = -75\text{mV}$, $V_{\text{rest}} = -70\text{mV}$, $V_t = -54\text{mV}$, $\tau_m = 5\text{ms}$, $\tau_n = 10\text{ms}$, $\tau_s = 50\text{ms}$, and $\alpha = 10\text{mv}$.
In the competitive framework, we choose $V_{\text{reset}} = -74\text{mV}$, $V_{\text{rest}} = -70\text{mV}$, $V_t = -55\text{mV}$, $\tau_m = 150\text{ms}$, $\tau_n = 40\text{ms}$, $\tau_s = 200\text{ms}$, and $\alpha = 10\text{mv}$.